

# Decision support software for resolving a security incident

Lukáš Sadlek, Michal Javorník, Martin Husák

2020

## Abstract

This document describes the output of the project of the same name ***"Research of tools for assessing the cyber situation and supporting decision-making of CSIRT teams in the protection of critical infrastructure"*** (VI20172020070) addressed in ***the Security Research program of the Czech Republic*** in the years 2017-2020 at Masaryk University. The document contains a description of the result, installation instructions, user documentation and programming documentation.

# Content

<b>Introduction</b>	<b>4</b>
Technical parameters of the result	5
Economic parameters of the result	5
<b>Description of the result</b>	<b>6</b>
Methods used	7
Attack graph	7
Bayesian network	7
An optimization problem	8
Mission modeling	8
Mission decomposition model	8
An example of mission decomposition	9
Analytical process	10
Decision support	12
Choosing the most durable configuration	12
The final decision	13
<b>Installation instructions</b>	<b>14</b>
Prerequisites	14
Installation	15
Settings	16
<b>User documentation</b>	<b>17</b>
Insert mission description	17
Using the tool using the command line	18
Tool control via the control panel	19
List of missions and configurations	19
Security threshold	20
Mission description visualization	21
<b>Programming documentation</b>	<b>22</b>
Structure of SW Decide phase	22
Bayes module	22
Generator module	23
The Process module	23
The Components module	23
The Run_Mulval module	23
Directory test	23
Data directory	23
Input data	23
Implementation of the analytical process	25

General description of the algorithm	25
Discovery of configurations	26
Generating the input file	26
Generating an attack graph	27
Bayesian attack graph construction and probability inference	27
Combination of probabilities and output for visualization	28
Component output	29
The return value	29
Format of the results in the database	29
Tests	31
<b>Thanks</b>	<b>32</b>

## Introduction

This document contains documentation on the result **"Software for decision support in solving a security incident"** (hereafter Decide), which is the output of the project **"Research of tools for assessing the cyber situation and supporting decision-making of CSIRT teams in the protection of critical infrastructure"** (VI20172020070, hereinafter referred to as the CRUSOE project ) addressed in **the Security Research program of the Czech Republic** in the years 2017-2020 at Masaryk University.

The aim of the aforementioned project was to create tools that help specialists in the cyber security team to map and navigate the current cyber security situation in the computer network, quickly and well decide on the procedure for solving ongoing incidents and implement the proposed solution. The set of created tools reflects the thought concept of the so-called OODA cycle, which consists of four phases - Observe, Orient, Decide and Act. The OODA cycle was designed and described as a general procedure for information gathering and decision support.

The user of the OODA cycle iterates through individual phases, thanks to which he formalizes his thinking and is thus able to make decisions and act effectively and quickly without unnecessary mistakes. First, it is necessary to collect information about the environment (Observe phase), then to orient oneself in it (Orient phase), then to decide on a suitable next course of action (Decide phase) and finally to implement this procedure (Act phase). As part of the CRUSOE project, a tool was created for each phase of the OODA cycle, which allows users to implement the objectives of the given phase. **"Software for decision support in solving a security incident"** implements the Decide phase, i.e. the decision support phase.

The Decide software is used to select the most durable configuration of an information system or computer network, taking into account security requirements and requirements to ensure critical processes of the organization that operates the information system or computer network. The software takes into account the current situation in the computer network, including the vulnerabilities of the systems in the network and the position of the attacker. The Decide software regularly checks information about the computer network or information system that is the subject of the cyber security team's protection. In the input data, it expects a list of devices including the services and software operated, an overview of vulnerabilities and the position of the attacker (a list of devices and authorization levels that can be in the hands of the attacker) and a formal description of the requirements for the processes of the given organization. The formal description of the requirements contains a list of critical processes within the organization, a list of elements of the critical infrastructure of the given organization and their mutual mapping. The software processes these inputs and compares possible configurations of critical infrastructure elements that allow all critical processes within the organization to be fully operational. A score is added to the individual configurations, which determines the probability of a successful attack that would endanger the processes in the organization. The configuration with the lowest attack probability is selected as the recommended one. The output of the Decide software is therefore a recommendation to (re)configure the computer network or information system so that the probability of a successful attack is as low as possible. Reconfiguration of the given network or system is not part of the implementation and always depends on the final decision of the user whether to follow the recommendation. However, the outputs of the Decide software are usable by the tool implementing the Act phase.

This document consists of five parts. An introduction containing a summary of the technical and economic parameters of the result is followed by a description of the result explaining the principles on which the software is built. Installation instructions, user documentation and programming documentation are presented in other sections.

## Technical parameters of the result

The software implements a set of algorithms to support the decision-making of the cyber security team when protecting a computer network or information system. The software processes a list of devices, services and vulnerabilities, and devices and permissions that may be in the hands of an attacker. Another input is the formal representation of services that provide critical processes for the organization operating a given computer network or information system, and their mapping to devices and services in the network. Based on the given information, the software creates a list of possible configurations that enable critical processes to function. A score is added to each configuration, which determines the probability of a successful attack. The output of the software is a recommendation of the most robust configuration.

The software is distributed as open-source under the MIT license, the owner of the result is Masaryk University, IČO 00216224.

Contact person: RNDr. Martin Husák, Ph.D. Institute of Computer

Technology Masaryk University

Šumavská 416/15 Brno 602 00 e-

mail: husakm@ics.muni.cz

phone: +420 549

49 4857

## Economic parameters of the result

The market segment is mainly represented by organizations operating critical information infrastructure or other infrastructure with high requirements for confidentiality, availability and integrity.

The result allows users (operators of networks and services) to choose the most resistant configuration of computing infrastructure at a given time under given conditions, which makes it possible to eliminate cyber security threats and prevent attacks with the most serious impacts on the infrastructure. The use of the result thus primarily has preventive effects against damage caused by cyber attacks. The effectiveness of the created software depends on the quality of the input data provided by the user and whether and how the user adjusts the infrastructure according to the recommendations that are the output of the software. Use of the result is licensed free of charge.

## Description of the result

As the complexity and scale of the IT infrastructure increases, it becomes more and more complicated to effectively protect it as a whole. Defense activities need to be targeted at critical activities that take place on the infrastructure. In other words, it is necessary to map in detail the individual missions, their supporting processes, priorities and especially the security requirements imposed on individual processes in terms of confidentiality, integrity and availability. The key question is how to assess the security situation of the mission, or how to quantify (derive, calculate) the resilience of the mission in terms of the probability of breaching the security requirements of any of the processes forming the mission.

The decision-making process is based on the following assumptions and characteristics:

- The principle of the decision-making process assumes that it leads to the fulfillment of the objectives of the monitored mission several different variants of the configuration of the supporting infrastructure.
- The goal of the decision-making process is to find such a configuration of the supporting infrastructure (such measures against an attack) that will keep the critical processes of the monitored mission as long as possible in a state where security requirements have not been violated.
- The decision support process integrates internal specific information about permissible mission configurations and external information about vulnerabilities, attack vectors and security metrics from publicly available sources.
- The primary application is in a situation where the infrastructure faces an attack. The computational algorithms used in the decision-making process can also be used proactively, i.e., for example, when simulating potential attack variants with the aim of revealing vulnerabilities in infrastructure components that have a critical impact on the monitored mission.

- The mission decomposition model plays a key role in the decision-making process. The hierarchical model makes it possible to express clearly and in context the consequences of individual decision variants, thereby increasing the transparency of the decision-making process.
- The principle of the process is based on the division of roles between the team ensuring cyber security and the institution's management, which, based on knowledge of a specific problem area (domain), makes decisions at the level of managing the processes that make up the monitored mission.
- The decision-making process enables the security situation to be quantified, i.e. enables

quantitative comparison of the safety of available mission configurations and, in particular, identification of the safest variant.

- The algorithmic principle of the method,

which is described in detail in individual chapters of the text, is as follows. The proposed method is in principle an optimization task, the goal of which is to find the optimal solution of the optimization (penalization) function, which is defined on the set of all admissible solutions of the given problem. The set of admissible solutions is formally determined by the decomposition model of the given mission. The model contains all relevant mission support entities, their interrelationships and security requirements.

The input of the method is a mission decomposition model, together with abstract knowledge obtained from publicly available sources. This input information and the available algorithms will make it possible to construct a Bayesian network based on the attack graph. By using inference mechanisms over the Bayesian network, we can quantify individual decision variants and solve the given optimization task.

## Methods used

The decision process is based on three main theoretical methods, attack graph, Bayesian network and optimization problem. Following is their brief introduction.

### Attack graph

An attack graph can be formally viewed as a set of system states and a set of their transition relations. The initial state of the system represents the state before the attack was launched. Transition relations represent the possible steps of an attacker. Transition relations can be supplemented with information about the probability that the attacker will choose a particular step (for example, they can represent the amount of effort required to implement a given step). If the attacker successfully takes all the steps leading to the transition from the initial state of the system to one of the target states, we consider the attack to be successful. A target state means a system has been compromised in the sense of compromising its confidentiality, integrity or availability.

A spectrum of available algorithms can be used to generate the attack graph. As part of the project solution, we chose the MuVAL tool, a tool for network vulnerability analysis. It consists of scanners that are installed on each computer in the network and an analyzer that then processes the results and information about the vulnerability. Among the main advantages of this tool is that it can automatically process vulnerability specifications from publicly available sources, and then especially the polynomial complexity of the algorithm for constructing the attack graph.

Information about the consequences of vulnerabilities was taken from the NVD database and CVSS score values of the given vulnerability. The impact of each vulnerability can be divided into the following four categories: loss of confidentiality, loss of availability, loss of integrity, and privilege escalation. These vulnerabilities are detected using vulnerability scanners, such as the tools described in the CRUSOE project "Software for the detection of vulnerabilities in a computer network" outcome.

Attack graphs enable a comprehensive description of possible attack scenarios, i.e. they provide quantitative information about the security status of the system. Attack graphs do not allow a formal expression of the uncertainty associated with the attacker's behavior, i.e. the possibility to quantify the security state of the system. By supplementing data on local probabilities, or local conditional probabilities of the attacker's potential steps, the attack graph can then be transformed into a Bayesian network structure. The missing quantitative information about the security of the system can subsequently be derived from this structure.

### Bayesian network

A Bayesian network is a structure belonging to the group of probabilistic graph models.

A Bayesian network can be described as a directed acyclic graph, where individual nodes of the graph represent random variables and directed edges represent causal relationships between these random variables. Each variable to which at least one oriented edge leads is assigned a table which, in the form of local conditional probabilities, quantifies the relationship between this variable and the random variables from which the oriented edges originate.

To quantify the local security risk, relevant metrics defined within the CVSS and transformed into conditional probabilities can be used.

In this way, the attack graph can be expanded with data on probabilities quantifying the relationships between the individual nodes of the attack graph, i.e. the relationships between the related positions of the attacker. The proposed process dynamically calculates the current risk and compares permissible alternatives of possible measures leading to an increase in the system's resistance to attack.

### An optimization problem

We are looking for a solution to an optimization task, in this case a stochastic problem, i.e. a solution (mission configuration) with the greatest probability of preserving the mission's safety requirements within explicitly formulated mission configuration alternatives (preserving functional requirements). The structure of the Bayesian network allows us to quantify the (marginal) probability of a successful breach of security (requirements placed on a given configuration of mission processes) by exploiting the vulnerabilities of the software components forming the mission. We therefore have a real optimization function defined on the set of all admissible solutions to the problem. The computation can rank the available configurations based on the attack resistance criterion with respect to the attacker's current position. In particular, the calculation will determine the most advantageous (most durable configuration) countermeasure. The final decision on the choice of countermeasures and system reconfiguration will be made by the management of the institution.

### Mission modeling

The mission model within the organization is part of the inputs to the decision support software. The mission needs to be identified, decomposed and formally described so that it can be considered in decision support.

### Mission decomposition model

We assume that a mission is made up of one or more processes. We treat these processes as key assets. Individual support processes can then be mapped to the necessary support IT services, and these can then be mapped to support software components and their specific interactions.

This concept assumes setting requirements for confidentiality, integrity and availability at the level of individual processes and then mapping these requirements to relevant software components and their interactions.

We assume that mission objectives (compliance with functional requirements) can be achieved through multiple configuration alternatives for supporting processes, IT services, and software components. Allowable mission configurations are defined using an oriented graph.

The rules are expressed by special AND/OR nodes and corresponding edges in the graph.

We look at the mission from four different perspectives. We distinguish mission functional requirements, mission security requirements, permissible mission configuration and mission resilience.

**Mission functional requirements.** A mission can be formally described as a system of permissible combinations of functional requirements imposed on its supporting processes, IT services, software components and their interactions.



**Mission security requirements.** Mission security requirements can be formally described as a system of security requirements (expressed as the required level of confidentiality, integrity and availability) imposed on individual mission processes.

**Allowable mission configuration.** The permissible mission configuration must always be in accordance with the defined functional requirements of the mission. Formally, it is expressed as a specific configuration of all supporting entities and their interactions. Mission configuration alternatives are expressed in graphical notation using special AND/OR nodes, i.e. they represent AND/OR logic.

The AND node in a specific mission configuration requires the presence of all mentioned nodes representing supporting IT services or computer components. OR nodes in the mission configuration require the presence of at least one of the mentioned nodes representing supporting IT services or computer components.

**Mission endurance.** The resilience of a mission is its ability to continue operating, or the ability to fulfill the required mission objectives (defined functional requirements) in the given configuration and taking into account the current position of the attacker. We quantify the resilience of the mission in terms of the probability of breaching the security requirements of any of the processes that make up the mission.

#### **An example of mission decomposition**

The following case study of a model medical workplace in the field of medical image information processing illustrates the proposed method. First, we briefly describe the problem domain, followed by the mission decomposition model.

The processing of medical image information in the medical workplace represents a wide spectrum of interconnected activities over the spectrum of supporting IT processes of many healthcare and IT service providers. The IT application environment consists of domain-specific applications running on specially configured computer networks.

The mission decomposition model is made up of predefined rules describing all admissible constellations of sub-components that meet the functional requirements of the mission. Security requirements placed on individual support processes reflect legal, ethical, contractual or other specific requirements. Collectively, these can be expressed as desired levels of confidentiality, integrity and availability.

The key processes of the mission are, in particular, the processes enabling the examination of the patient using the acquisition modality (CT, MRI, X-ray, mammographic screening, etc.), basic diagnostics, professional consultations, or other specific processing of image information.

Key IT services include, in particular, support for the acquisition of image data, operation of an institutional or regional archiving and communication system (PACS), services for the exchange and sharing of image data between individual hospital departments and other remote healthcare institutions, and support for the diagnosis of image examinations.

The subsystem of supporting software components consists of a wide range of products: specific PACS implementations, software components for acquisition management, software for special diagnostics, etc. Individual components communicate through domain-specific communication protocols.

Assume that the model workplace is equipped with two computed tomography (CT) machines. Diagnostics of image data is mainly carried out locally, more complicated situations are additionally consulted with experts from distant specialized institutions.

The workplace ensures two critical processes. The first is the process of acquisition (Acquisition), i.e. the realization of one's own CT examination and the subsequent making available of image data through archiving or of the PACS communication system for further processing. The second process is the diagnostic process (Diagnostics), i.e. obtaining an image examination from the archival or of the PACS communication system and own implementation of the required diagnostics.

The processes use the following supporting IT services: acquisition of medical image data (PrimaryCT, respectively SecondaryCT), making image data available within the workplace or between remote healthcare institutions of the region (LocalPACS, respectively RegionalPACS) and local and external diagnostic CT examination services (Local-Diagnostics, or External-Diagnostics).

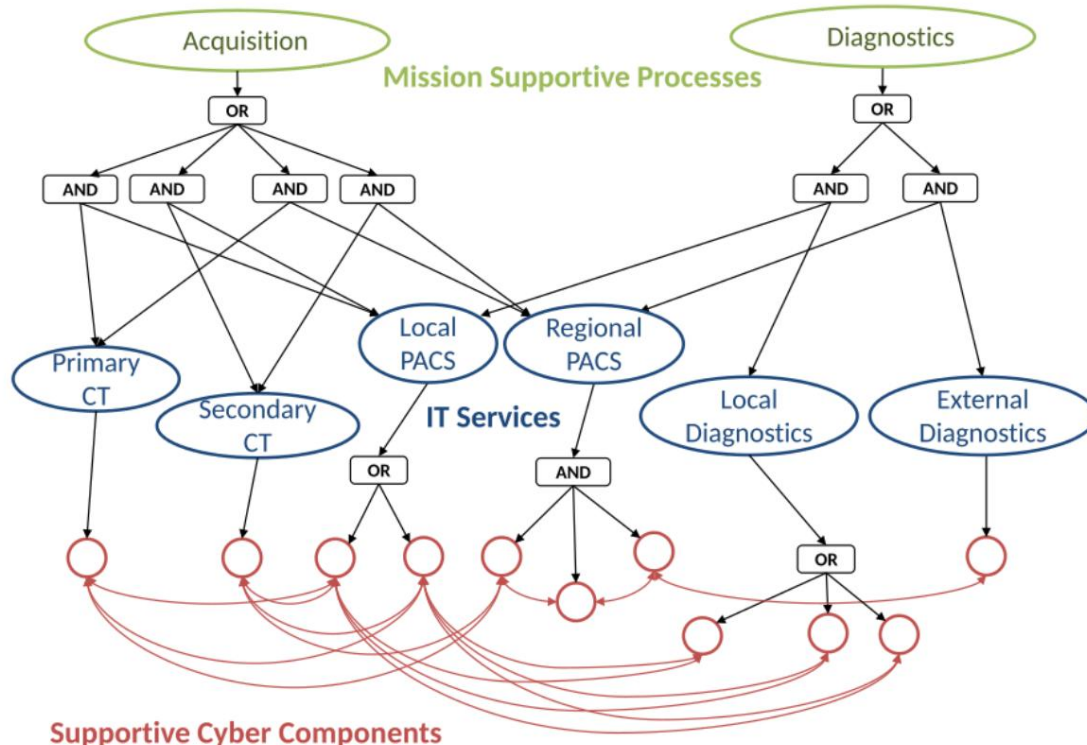


Figure 1: An example of an organization's mission model and their dependencies.

## Analytical process

The analytical process for decision support is based on the mission decomposition model and uses the mathematical methods described above. The initial step of the process is the determination of a set of limiting conditions defining admissible solutions to the decision problem and the definition of a utility (penalty) real function that quantifies the individual admissible solutions. The goal of the decision-making process is not only to find the optimal solution to a problem defined in this way,

i.e. finding the most durable configuration within the set of permissible mission configurations, but also providing understandable justification to all participants in the decision-making process, i.e. especially to persons responsible for network security and persons responsible for decision-making at the management level of the given application domain.

The decision support process consists of three phases:

1. The goal of the first phase is the construction of a static model of the mission decomposition. The decomposition process is described in the previous chapter. This step is performed only once. The output of this phase is the mission decomposition model and a formal notation of limiting conditions derived from the model, e.g. in the form of a logical expression, which can then be processed algorithmically. The formal notation of suitable mission configurations using the expressive means of mathematical logic is the basis for the next stages of the decision-making process.
2. The goal of the second phase of the process is to quantify the security situation. It uses a mission decomposition model together with mathematically expressed constraint conditions that define allowable mission configurations. This phase is built on the concept of attack graphs and Bayesian networks, and then on general knowledge about cyber threats and, in particular, information about the current position of the attacker. The result is a Bayesian graph of attacks, which represents the dynamics of the situation, i.e. it makes it possible to derive (quantify) the probabilities of variants of its further possible development.
3. The third stage integrates the static output of the first stage and the dynamic output of the second stage. Based on the calculation, it identifies the resulting optimal configuration of the mission, and corresponding countermeasures optimally reflecting the current cyber threat, i.e. reconfiguration of active network elements, or changes directly in the configurations of individual application components.

The output of the entire process is a recommendation of the most resistant system configuration, which can maximally reduce the effectiveness of the attack under the described security situation. The recommended configuration needs to be set manually or automatically. If the attacker's position or other information entering the calculation algorithm subsequently changes, the second and third phases of the process need to be repeated cyclically.

In order to construct a Bayesian attack graph, in addition to the description of the active software components of the mission and their allowed interactions, current information about the relevant attack vectors and information about the current position of the attacker, i.e. information that we assume will change dynamically over time, is also needed. Cooperation with attack detection systems, including knowledge of the sequence of activities that the attacker has already successfully carried out, is essential here.

General information about vulnerabilities can be obtained from various sources maintained by software manufacturers or from sources maintained by security specialists. In the proposed algorithms, we use information from publicly available sources. NVD (National Vulnerability Database) represents the de facto standard library of vulnerabilities described in CVE format.

Vulnerabilities in the CVE format are supplemented by the CVSS (Common Vulnerability Scoring System) scoring system, which in the form of specialized metrics characterizes, for example, the complexity of the attack or its impact in terms of breach of confidentiality, integrity and availability. The CVSS database, as a publicly accessible reliable source of information, provides us with the necessary local quantitative characteristics necessary to construct a Bayesian attack graph.

The CVSS Attack Complexity security metric, which after a simple transformation can be interpreted as the conditional probability of successfully exploiting a given vulnerability, supplies initial information for the final calculation of the marginal probability of a successful multi-stage attack by exploiting a sequence of vulnerabilities critical to the monitored mission.

A security metric referring to the potential impact of successfully exploiting a given vulnerability (with respect to the confidentiality, integrity, and availability of a given cyber component) specifies the privileges an attacker would gain as a result of successfully exploiting the vulnerability.

The sequence of specific steps in the security situation quantification process is as follows:

- identification of system vulnerabilities and weaknesses using NVD database and model mission decomposition, which provides a list of all IT components used and their interactions,
- obtaining the current position of the attacker by using an intrusion detection system or by using a similar tool that reports the exploitation of a vulnerability or similar events, • generating an attack graph using one of the available algorithms, for example MulVal,
- transformation of the attack graph into a Bayesian attack graph by supplementing the parameters in the form of local probabilities with an appropriate interpretation of the CVSS security metrics, • calculation of the marginal probability of a successful multi-stage attack aimed at obtaining the attacker's authorization, which means a violation of the confidentiality, integrity or availability of the system.

## Decision support

The output of the analysis process is a list of calculated resiliencies of each configuration of devices in the network that fully support the given mission. Decision support consists in choosing the most resilient IT infrastructure configuration. In the decision support phase, the most durable configuration is first selected and then the final decision can be made.

### Choosing the most durable configuration

The selection of the most robust configuration is straightforward, however in practice it may be advantageous to subject the selection to other conditions in order to avoid recommendations that would lead to unnecessary or too frequent changes to the configuration of the IT infrastructure. For each configuration that fully supports the given mission, the probability of endangering the given mission is calculated. The calculation is based on the impact of the vulnerabilities, the effort the attacker must put in, and the severity of the impact on the mission. The configuration with the lowest probability of compromising the mission is selected as the most resilient. If no further control is introduced, this result is accepted.

In practice, a situation may arise where a configuration change based on a recommendation would occur too often or the rationale for the recommendation would be too weak to justify such a change. An additional check may consist of comparing the mission disruption probabilities between the current and recommended configurations. If the difference is small, for example less than one percent, there is no need to change the configuration of the IT infrastructure. The impact on mission security would also be negligible. Additionally, changing the configuration can be a non-trivial task that takes a lot

time and limit the comfort of IT infrastructure users. Therefore, it is advisable to change the configuration only when the recommended configuration would significantly increase the resilience of the infrastructure.

### **The final decision**

Selecting the most robust configuration completes the decision support process. As described above, the recommendation may or may not be accepted. The final decision is the responsibility of the relevant administrators, managers, security experts or other interested persons.

The implementation of the given decision, i.e. changing the configuration of the IT infrastructure, is then fully in the hands of the infrastructure operators.

When making a final decision, it must be remembered that the resistance values of the given configurations are only relative and must be taken in the context of other factors.

The most important are:

- impacts on the economic efficiency of the organization,
- impacts on the redundancy of critical components,
- impacts on the work habits of IT infrastructure users.

It is difficult to describe the specific impacts without mentioning specific cases, or it would deviate from the issue of supporting the decision-making of the cybersecurity team. Likewise, automating infrastructure configuration changes is beyond the scope of this topic. Human participation in the decision-making process is still required in this case, at least in the form of supervision.

## Installation instructions

The instructions for installing the decision support software for solving a security incident consist of three parts. First, the installation of the prerequisites is described, then the installation of the software itself is described, and finally, the configuration procedure before the first launch is described.

The installation guide has been verified on Ubuntu 18.04 LTS and Debian 10 operating systems.

## Prerequisites

Security incident resolution decision support software requires the MulVAL and XSB software tools to be installed on the system. First, you need to choose where these tools will be installed. We recommend placing them in the */opt directory*.

Note: some of the following commands may require **root** privileges or **sudo to run**.

First, download and extract the MulVAL installation package using the commands: `$ wget -P /opt https://people.cs.ksu.edu/~xou/argus/software/mulval/mulval_1_1.tar.gz` `$ cd /opt` `$ tar -xzf mulval_1_1.tar.gz` `$ rm mulval_1_1.tar.gz`

A prerequisite for MulVAL is the XSB tool, whose installation package must be downloaded and unpacked using the commands: `$ wget -P /opt http://xsb.sourceforge.net/downloads/XSB.tar.gz` `$ tar -xzf XSB.tar.gz` `$ rm XSB.tar.gz`

Before installation, environment variables specifying the location of the MulVAL tools must be set and XSB:

```
$ export MULVALROOT=/opt/mulval $ export  
XSBROOT=/opt/XSB $ export
```

```
PATH=$PATH:$MULVALROOT/bin:$MULVALROOT/utls:$XSBROOT/bin:$XSBROOT/build
```

In addition, tools for compiling source codes in C++ and Java languages, as well as **bison** and **flex** tools, must be installed on the system using the following commands:

```
$ apt-get install build-essentials $ apt-get install  
bison flex
```

The XSB tool compiles after entering the commands:

```
$ cd /opt/XSB/build $ ./  
configure
```

```
$ ./makexsb
```

The XSB tool can now be started with the command:

```
$ /opt/XSB/bin/xsb
```

and exit by entering the **halt command**. (with a period at the end).

MulVAL compiles after entering the commands:

```
$ cd /opt/mulval $ make
```

If the output of the **make** command contains **javac: Command not found**, it means that Java is not installed on the system or is not in the **PATH environment variable**. In this case, the JDK must be installed on the system, the **PATH** environment variable must then contain the directory where **the javac command is located**.

If using OpenJDK, you will probably need to install **the dom4j.jar** package as well.

First, you need to go to the directory for external java packages, which is usually of the stable form **/usr/lib/jvm/<name of java>/jre/lib/ext**, and then download the package itself with:

```
$ wget -P . https://github.com/dom4j/dom4j/releases/download/dom4j_1_6_1/dom4j-1.6.1.jar
```

If the compilation was successful, MulVAL can be tested using the commands (in the directory **MULVALROOT**, typically **/opt/mulval**):

```
$ cd testcases/3host/ $  
graph_gen.sh input.P -l -p
```

## Installation

After downloading or decompressing, the software for the Decide phase (under the internal name **attack graph component**) can be installed using **the pip install command**:

```
$ pip install . -r requirements.txt
```

The command in this form is used if we are in the directory that contains the **setup.py** and **requirements.txt** files. A dot after the **install** keyword indicates that it is installing from the current directory. The current directory can be replaced with any path in the file system. Furthermore, **the pip install** command similarly uses the option to specify a requirements file using the **-r** switch followed by the path to the requirements file.

## Settings

Before starting the software, initial settings must be made. In the ***example\_data*** directory there is a ***conf.ini*** file in which the following variables must be set:

- ***The mulval\_root*** and ***xs\_b\_root*** variables must be set to the same value as ***the MULVALROOT*** and ***XSBROOT*** environment variables used during the prerequisite installation, typically ***/opt/mulval*** and ***/opt/xsb***. • The ***mulval\_dir*** variable refers to the directory where the files will be created required for generating attack graphs.
- ***The interaction\_rules\_file*** variable refers to the ***crusoe\_rules.P*** file in the ***example\_data directory***. This variable only needs to be reset if the file in question is renamed.



# User documentation

The user documentation of the decision support software for solving a security incident consists of three parts. First, the technical procedure of mission modeling and their insertion into the database is described. The general procedures for creating the mission model are described in the description of the result, there are only user instructions for introducing the mission description into the system. Subsequently, the use of the decision support tool using the command line is described. The last part is a brief description of the use of the tool through the control panel, which is part of the result "Web application for visualizing the security situation in a computer network" from the same project.

## Insert mission description

Within *neo4j-rest*, endpoint *missions* are implemented, POST to this endpoint will create a new node of type :Mission in the database as well as all nodes of other types (i.e., :Component, :Host and :IP) found in the input JSON file, for assuming they did not previously exist in the database. This JSON file contains *a constrained AND-OR tree* for a given mission and must contain data in the format specified in other parts of this technical report ( Input *data* section in *the Programming documentation chapter*).

To insert the mission into the database itself, the following command can be used (or modified as needed): `$ curl --header`

`"Content-Type: application/json" --`

```
data '{"nodes": {"missions": [{"id": 1, "name": "Web mission", "criticality": 7, "description": "Mission is responsible
for proper functionality of a web server supported by a database server."}], "services": [{"id": 11, "name":
"Webserver service"}, {"id": 12, "name": "DB Server service"}], "aggregations": {"or": [ 3 ], "and": [ 2 ]}, "hosts": [{"id": 20,
"hostname": "host2.domain.cz", "ip": "128.228.250.67"}, {"id": 21, "hostname": "host1.domain.cz", "ip": "128.228.251.133"},
{"id": 22, "hostname": "host3.domain.cz", "ip": "128.228.123.47" }], "relationships": {"one_way": [{"from":
1, "to": 2 }, { "from": 2, "to": 11 }, { "from": 2, "to": 12 }, {
```

```
"from": 11, "to": 20 }, { "from": 12, "to": 3 }, { "from": 3, "to": 21 }, {
"from": 3, "to": 22 }], "two_way": [], "supports": [{"from": "Web mission", "to": "Webserver service"}, {"from": "Web
mission", "to": "DB Server
service"}], "has_identity": [{"from": "Webserver service", "to": "host2.domain.cz"}, {"from": "DB Server
service", "to": "host1 .domain.cz"},
{"from": "DB Server service", "to": "host3.domain.cz"}]}' http://localhost:8000/rest/missions
```

In this *curl* command, the type of sent data (*Content-Type*) is first specified as JSON, and the data itself is passed using the *--data switch*. The given command contains a minimalistic example of a passed JSON file corresponding to the format described in other parts of this technical report. Since the JSON file contains quotation marks ("), the data itself must be delimited in the *--data* section, for example, using single quotation marks (').

Additionally, the command must be passed the correct URL path to the mission creation endpoint, which corresponds to the specific REST API settings in the production environment and does not have to match the URL path shown in the previous example. If the REST API is secured with a username and password, you need to authenticate, e.g. by using the **--user switch**.

However, the most user-friendly way is to pass the JSON through a web browser.

The Django REST framework provides a so-called **Web browsable API**, which simply means that when you enter the URL address of **the missions** endpoint, a form is displayed, in which the content of the JSON file is copied into the **Content** field, and it is forwarded to the REST API during a POST operation in a similar way to using curl tools.

The mission node created in the database will then contain the attributes **name, criticality, description**, and the content of the file will be located in the parameter named **structure**. Furthermore, nodes of other types and their relations will be created in the form: *(:Mission)<-[:SUPPORTS]-[:Component)-[:PROVIDED\_BY]->(:Host)<-[:IS\_A ]-([:Node)-[:HAS\_ASSIGNED]->(:IP).*

## Using the tool using the command line

For the correct functioning of the component, the software tools MulVAL and XSB must be installed based on the instructions found in the previous chapter or in the README.md file.

Then it is possible to proceed according to the following instructions.

The input method of the entire analytical process is the `analytical_process()` method in the **process.py file**. This function expects two arguments on input, the password to the Neo4j database and the address where the **bolt database connector is running**.

```
>>> from attack_graph_component import process >>>
process.analytical_process("password", "bolt://localhost:7687")
```

In order for the component to function correctly after startup, the data in the database must match the CRUSOE data model and all variables in the **conf.ini** configuration file must be set correctly. In addition, the process needs to have sufficient permissions to create files in the **mulval\_dir** folder set in the configuration file.

It is possible to pass your own logger to the analytical process function as the third parameter using the keyword **logger**:

```
>>> process.analytical_process("password", "bolt://localhost:7687", logger=own_logger.get_logger())
```

By default, the component uses the python package structlog as a logging tool .

The list of missions and configurations contains a list of all missions that the Decide tool works with. A list of possible configurations is also given for each mission. The mission and configuration list panel is shown in Figure 3.

### Decide Configurations List

**Network Monitoring (No config selected)**

	Name	Integrity	Confidentiality	Availability
<input type="checkbox"/>	Config 1	0%	0%	0%
<input type="checkbox"/>	Config 2	0%	0%	0%
<input type="checkbox"/>	Config 3	0%	0%	0%

**Incident Handling (No config selected)**

	Name	Integrity	Confidentiality	Availability
<input type="checkbox"/>	Config 1	0%	0%	0%
<input type="checkbox"/>	Config 2	0%	0%	0%
<input type="checkbox"/>	Config 3	0%	0%	0%

Apply selected configs

Figure 3: List of missions and their configurations.

Under the name of the mission, a list of possible configurations is always displayed along with the degree of threat for the three monitored parameters - integrity, confidentiality, and availability. By clicking on the name of the mission, a description of the given mission and its criticality value, i.e. the degree of importance for the organization, will be displayed. By clicking on the name of the configuration, a map is displayed on which the machines that the configuration requires to be blocked are marked in red. The checkbox for each configuration allows you to select this configuration and use the Apply selected configs button to block and unblock active defense elements.


## Security threshold

Each machine located in the network protected by the CRUSOE system receives an evaluation in three monitored parameters - the degree of threat to integrity, confidentiality, and availability. More about these parameters can be found in the description of the result and is related to filtering the results after selecting the most resistant configuration before recommending it. The Act phase software works with the aggregate machine threat level, which is calculated as the arithmetic average of the threat level of integrity, confidentiality, and availability of the given machine. It therefore reaches values from the interval  $<0, 100>$  and its application is detailed in the documentation of the Act tool. An example is shown in Figure 4.

### Security threshold

Threshold value

51



%

Update threshold

Figure 4: Setting the Security threshold value.

## Mission description visualization

The lower part of the panel is reserved for a clear visualization of the description of the missions that are recorded in the database. Missions are drawn in a hierarchical graph structure corresponding to the formal description of the mission model. In the relevant part of the panel, it is possible to zoom in and out of the mission visualization, which enables a clear view of the entire mission model and its individual details. The relevant part of the panel can be viewed in Figure 5.

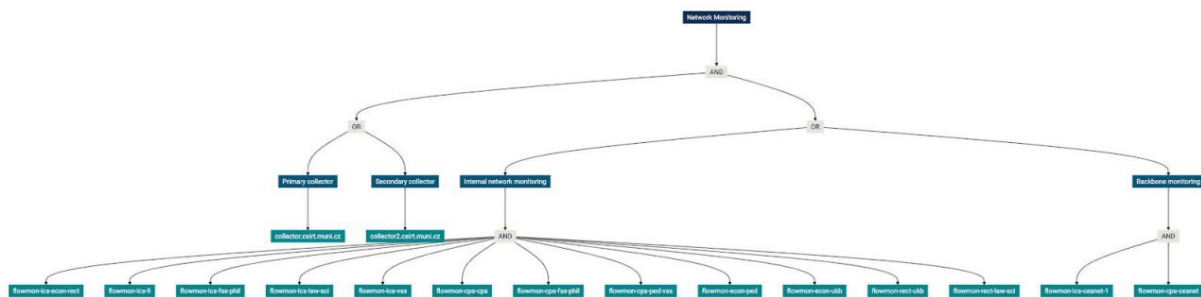


Figure 5: Visualization of the mission description.

# Programming documentation

This section describes the programming documentation. First, the structure of the package is presented, which clarifies the division of partial functionalities of the entire process into individual modules. The detailed description of the implementation of the analytical process for decision support is supplemented by subsections dedicated to the input data that the component works with and the outputs that the component produces either in the form of a program return value or as a record in a graph database. The last subsection is devoted to tests.

## Structure of SW Decide phase

The content of the component can be divided into three categories. The first consists of files that implement the analysis process (***bayes.py***, ***components.py***, ***generator.py***, ***process.py***, ***run\_mulval.py*** and the data file ***crusoe\_rules.P***). Furthermore, test files are located in the ***test directory***. The last group consists of the remaining files for the necessary formal requirements - the configuration file (***conf.ini***), the Python files required for installation (***\_\_init\_\_.py***, ***setup.py***, ***requirements.txt***) and the description file (***README.md***).

attack-graph-component

```

├── attack_graph_component
│   ├── data
│   │   ├── conf.ini
│   │   └── crusoe_rules.P
│   └── test
│       ├── test_data
│       │   ├── attack_graph_test.py
│       │   ├── config_decomposition_test.py
│       │   └── neo4j_test_client.py
│       ├── bayes.py
│       ├── components.py
│       ├── generator.py
│       ├── process.py
│       └── run_mulval.py
├── __init__.py
├── requirements.txt
├── setup.py
└── README.md

```

## Bayes module

This module contains functionality related to the conversion of the created attack graph into a Bayesian attack graph, which assigns a conditional probability distribution table to each vertex. Based on the created Bayesian graph of attacks, the inference of the resulting unconditional probability is performed for each of the attack targets located in the graph.

### Generator module

This module contains methods that create an input file for the MulVAL tool from the input data found in the graph database and corresponding to the CRUSOE data model. This input file contains the facts (a logic programming term) that will be processed by MulVAL.

### The Process module

The module contains the main function for the analytical process and the functionality needed to process individual mission configurations located in the database. Furthermore, it ensures the combination and selection of the resulting probabilities and the creation of results for individual configurations in the graph database.

### The Components module

This module contains functionality for finding out the available configurations of a given mission.

### The Run\_Mulval module

This module mediates the command line call through which the attack graph is generated by the MulVAL tool.

### Directory test

The test directory contains test files that are discussed in one of the following subsections.

### Data directory

The data directory contains the **conf.ini** and **crusoe\_rules.P files**. The first file - **conf.ini** - is a configuration file in which several variables need to be set: • the location of the MulVAL and XSB tools (**mulval\_root / xsb\_root**), • the path to the folder for creating the attack graph (**mulval\_dir**), • the default name of the file containing the custom rules for generating an attack graph (**interaction\_rules\_file**),

The second file - **crusoe\_rules.P** - contains rules for generating the attack graph. These rules will be described in the subsection on the implementation of the analytical process.

## Input data

All inputs are loaded by the decide component from the Neo4j database, which is organized according to the CRUSOE data model. Specifically, this is information regarding:

- vulnerabilities in CVE format (machine, software, remote exploitability, impact, CVSS), • the mission that the organization secures,

---

<sup>1</sup> OU, Xinming; GOVINDAVAJHALA, Sudhakar; APPEL, Andrew W. MulVAL: A Logic-based Network Security Analyzer. In: **USENIX security symposium**. 2005. pp. 113-128.

<sup>2</sup> KOMÁRKOVÁ, Jana, Martin HUSÁK, Martin LAŠTOVICKA and Daniel TOVARÝÁK. CRUSOE: A Data Model for Cyber Situation Awareness. In Proceedings of the 13th International Conference on Availability, Reliability and Security. Hamburg: ACM, 2018. pp. "36:1"-"36:10".

- network status - IP addresses of machines in the network, open ports, services and software products running on that machine and others.

Vulnerability and network health data were added to the database by other components during the previous phases of the OODA cycle. In a specific way, an input JSON is added to the database describing the requirements that the mission imposes on the infrastructure. For this, you need to use the REST API ( *neo4j-rest package* ), through which the given JSON is uploaded. The description contains **a constrained AND/OR tree**, in which the relations between the guests / components and the services required by the mission are captured using AND and OR logical formulas. A graphical representation of this tree is shown in Fig. 1.

The JSON itself then contains nodes ("**nodes**") and relationships between them ("**relationships**").

The vertices can be divided into four groups (see Fig. 1) - they are either missions (green in the picture), services (blue in the picture), AND/OR nodes or system components (red in the picture). All vertices within a single JSON must have a unique ID.

An important section is "**aggregations**", where AND and OR relationships are summarized in such a way that each AND / OR vertex (see Fig. 1) is assigned an ID and this ID is subsequently used within **one\_way** relationships.

Relationships describe graph edges represented by one-way **or two** -way arrows . The format defines the **two\_way section**, but the implementation uses instead of this manually specified data dynamic information from network monitoring, which is added to the database by components of the Observe phase. The "**supports**" and "**has\_identity**" sections are helpful for distinguishing what relationship there will be between vertices in the graph database ("**SUPPORTS**" between mission and services, respectively

**"HAS\_IDENTITY"** between service and components).

```

{ "nodes":
  { "missions":
    [ { "id":
      1, "name": "Network Monitoring",
      "criticality":
        8 },
      { "id": 2, "name": "Incident
        Handling",
        "criticality": 5 } ],
    "services":
      [ { "id":
        11, "name": "Internal network monitoring" },
        { "id": 12,
          "name": "Backbone monitoring" },
          { "id": 13,
            "name": "Primary collector" },
            { "id": 14,
              "name": "Secondary
                collector" }, ... ], "
            aggregations": { "or":
              [ 4, 5, 7, 23, 26 ], "and": [ 3,
                6, 8, 29,
                40, 50 ] },
              "hosts": [ { "id": 20 , "hostname":
                "collector.csirt.muni.cz",
                "ip": "147.251.14.52" },

```



```

{ "id": 21,
  "hostname": "collector2.csirt.muni.cz",
  "ip": "147.251.14.53" },

{ "id": 22,
  "hostname": "rt.csirt.muni.cz",
  "ip":

"147.251.14.49" }, ] ...],
  "relationships":
  { "one_way":
    [ { "from": 1, "to": 3 },
      { "from": 3, "to": 4 },
      { "from": 3, "to": 5 }, { "from": 4, "to": 11 }, { "from": 4, "to": 12 }, { "from": 5, "to": 13 },
        ....
    ],
    "two_way": [
      ], "supports":
        [ {"from": "Network Monitoring", "to":
          "Internal network monitoring"},
          {"from": "Network Monitoring", "to":
            "Backbone monitoring"},
            {"from": "Network Monitoring", "to":
              "Primary collector"},
              ...

        ], "has_identity":
          [ {"from": "Internal network monitoring", "to":
            "flowmon-ics-econ-rect"},
            {"from": "Internal network monitoring", "to":
              "flowmon-ics-fi"},
              {"from": "Internal network monitoring", "to":
                "flowmon-ics-fss-phil"},
                ...
          ]
        ]
      ]
    ]
  }

```

## Implementation of the analytical process

In this subsection, the analytical process is described in more detail, first in rough outlines, then the individual parts of the process are described in detail.

### General description of the algorithm

A rough outline of the analytical process looks like this. The algorithm first finds out which missions are in the database. For each of these missions, it loads the input JSON with a description of the mission configuration, which is located in the :Mission type node in the :structure **parameter**. Based on the input JSON structure, available configurations are detected for each mission. A configuration is a list of network devices or software components that provide the required functionality for the network to ensure the smooth operation of the mission. Each of these devices or software components is represented using the information provided in the **hosts** section of the input JSON. Since each configuration contains multiple devices or components that can be targeted for attack, a decision support analysis process will be run for each of them, creating a Bayesian attack graph and determining the resulting probability for each attack target separately. These probabilities are finally combined and the configuration that is most robust is chosen for each mission. The selection of the most robust mission configuration is the output of the analytical process. The component also stores all partial results for individual configurations in the database.

The processing of one configuration (ie one iteration of the analysis process) consists of three steps:

1. First, the input file for the MULVAL tool is generated.
2. Subsequently, an attack graph is generated using the MULVAL tool.
3. The attack graph is converted to a Bayesian attack graph and the resulting unconditional probability is determined (the so-called inference of the resulting probability).

### Discovery of configurations

The requirements placed on the mission are represented using the so-called **constrained AND/OR tree**, a tree of limiting conditions with logical links. This tree can be written in JSON format, the structure of which was described in the previous subsection. In the **components.py** module there is a **get\_mission\_components()** function that traverses the tree to determine which configurations are available for a given mission. The procedure is as follows: 1. The tree that was specified at the input is traversed

from the root of the tree to the leaves in the direction  
edges.

2. When passing through an AND node, all vertices to which the given node refers are added to the configuration in the intermediate result.
3. When passing through an OR node, configuration variants are created in the intermediate result for each vertex to which the given OR node refers. Each variant of the configuration contains exactly one of the vertices.
4. The traversal of the tree is stopped if only components (**Supportive Cyber Components**, shown in red in the sample image 1) are found in the given configuration .

### Generating the input file

Since we use the MulVAL tool in the component, the command to generate the attack graph must be entered in a way supported by this tool, i.e. using an input file and possibly also a file with the rules based on which the attack graph is generated. However, the input file must contain facts that preserve the syntax defined in the rules file. The functionality itself, located in the **generator.py file**, takes data from the database (created according to the CRUSOE data schema) and writes information about open ports, vulnerabilities, network services on machines, and network status using predicates defined in the **crusoe\_rules.P file**.

The default rules are available directly in the MulVAL tool, where they were entered by the creators of the tool based on the experience of experts in the field of cyber security. The rules for generating the attack graph were taken from the set of default rules for MulVAL, from which the rules relevant to the investigated domain were selected and possibly adapted. The customized ruleset is located in the **crusoe\_rules.P file**. Furthermore, predicates were added describing the loss of confidentiality, integrity and availability of the application or the entire system.

Specifically, the following changes have occurred:

- The original rules contained only the result of type **privEscalation** (privilege escalation) in **the vulExists** (vulnerability existence) predicate. Instead of privEscalation was

used the taxonomy of impacts proposed in a previously published article. In its current<sup>3</sup> form, we distinguish whether the vulnerability results in the execution of code with the authority of a regular user or administrator or the escalation of authority, or loss of confidentiality, availability or integrity in the system or in the application.

- The MulVAL tool works with only one possible target, **execCode** (code execution), when generating the attack graph, which did not meet the project's requirements to consider loss of confidentiality, availability, and integrity. For that reason, the predicates **appConfLoss()**, **appIntegLoss()**, and **appAvailLoss()** were added for application vulnerabilities and **sysConfLoss()**, **sysIntegLoss()**, and **sysAvailLoss()** for system vulnerabilities. • Other changes include the creation of the **inSubset** predicate (belonging to a subnet) and new rules for predicates related to threats to confidentiality, availability, and integrity.

## Generating an attack graph

In order to generate the graph, the MULVALROOT and XSBROOT system variables that describe the location of the MulVAL and XSB tools must be set and added to the PATH system variable. An attack graph is generated based on the generated input file and the rule file supplied by MulVAL. In the tool, we do not use the option to generate a PDF file with a graph, but only generate the graph into a TXT file.

The vertices of the graph are saved in the file VERTICES.CSV and the edges of the graph in the file ARCS.CSV. The files are stored in the **mulval\_dir directory**, which is defined in the configuration file.

## Bayesian attack graph construction and probability inference

From the files VERTICES.CSV and ARCS.CSV, which were created in the previous step, the vertices and edges of the attack graph, from which the Bayesian network is created, are loaded. However, when creating a Bayesian network, a so-called conditional probability distribution in the form of a table must be entered for each peak (the so-called **Tabular Conditional Probability Distribution**). In solving the problem, four general cases of what the table might look like were identified.

These cases are based on the logic of the MulVAL tool, which divides the graph nodes into AND, OR and LEAF nodes. Vertices of the LEAF type are vertices that are located in the position of leaves in the graph, i.e. no edge enters them, only an edge comes out of them. Vertices of the AND type are vertices that correspond to the implementation of a rule. In MulVAL output, they can be recognized by their name beginning with the keyword "RULE", e.g. "RULE 33" indicates that rule #33 was applied. OR vertices are simply those vertices that are not leaves either nor AND vertices.

The above distribution of conditional probability in the form of a table is as follows in individual cases: • For LEAF-type peaks • A LEAF-type peak

that does not correspond to the fact of

the existence of a vulnerability will be assigned a probability equal to 1.0 that the attacker will reach the exploit, if the phenomenon exists.

---

3 KOMÁRKOVÁ, Jana, Lukáš SADLEK and Martin LAŠTOVICKA. Community Based Platform for Vulnerability Categorization. In NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium. Taipei, Taiwan: IEEE, 2018., 2 pp. ISBN 978-1-5386-3416-5.

- A vertex of type LEAF that corresponds to the fact of the existence of a vulnerability with the assignee CVE gets the probability that an attacker will achieve an exploit if the event exists, the so-called **exploitability score** from CVSS version 3 can be <sup>4</sup>, which is possible obtained for individual CVEs from the data found in the NVD database. • For an <sup>5</sup>

AND type vertex with  $n$  parents:

- The probability that the attacker succeeds in taking the next step is zero in all  $(2^n)-1$  cases where at least one of the parents has an evaluation of FALSE. • If all parents have an evaluation of TRUE, the probability of success is equal to 0.8. This probability was taken from the default setting of the MulVAL tool, where it was determined based on expert experience.
- For an OR vertex with  $n$  parents, the following applies:
  - The probability that the attacker succeeds in taking the next step is zero if all parents have an evaluation of FALSE.
  - In all  $(2^n)-1$  cases where at least one of the parents evaluates to TRUE, the probability of success is 1.0, this is because there is no implementation of the rule at that vertex, the probability is only propagated towards the target of the attack.

To calculate the resulting likelihood inference, the pgmpy package is used, which provides an efficient implementation<sup>6</sup> of operations for working with Bayesian networks. The inference is calculated separately for each component from the confidentiality, availability, integrity triad and the result is a triad of probabilities.

## Combination of probabilities and output for visualization

Using the procedure described in the previous subsection, we obtain a trio of probabilities for threats to confidentiality, integrity and availability for one target component. As the resulting trio of probabilities for the mission configurations, the worst variant among the intermediate results that were calculated for the individual components in the configuration is selected.

If the probability for each mission configuration is obtained, the most robust configuration is selected. The configuration with the best probability score is selected as the most robust configuration, and the list of components of this configuration is returned in the return value.

As the previous two paragraphs suggest, comparing triples of probabilities can be confusing. For example, it is not generally possible to determine the worse variant when comparing triplets such as (0.7, 0.6, 0.5) and (0.9, 0.2, 0.2). In such a case, the first triplet has a higher integrity and availability violation probability value, while the second has a higher confidentiality violation probability value. This problem is solved by the so-called **utility function**, which makes it possible to convert triplets of values into one comparable value. Simple implementations **of the utility function** can be the selection of the maximum or average value. As part of the implementation of the component, the sum of the values in the triplet is used as a utility function, which effectively corresponds to an unweighted average.

---

<sup>4</sup> <https://www.first.org/cvss/specification-document>

<sup>5</sup> <https://nvd.nist.gov/general>

<sup>6</sup> <https://pgmpy.org/>

However, it can often be more advantageous to assign weights to individual components (confidentiality, integrity, availability) and calculate a weighted average depending on the specific requirements of the monitored infrastructure and the priorities of their administrators.

To allow domain experts, such as administrators of critical information infrastructure, to bring their own opinion and experience to the decision-making process, all intermediate results (triples of probabilities) that the component has arrived at are written to the individual configurations in the database. The user of the component thus has a chance to look at the intermediate results and adjust the final decision according to his experience and priorities.

## Component output

The output of the Decide phase component has two forms, the return value of the program and the data created in the graph database that correspond to the intermediate results for the individual configurations.

### The return value

For example, the return value might look like this:

```
{'Network Monitoring': {'configuration': {58, 51, 20, 53, 54, 52, 55, 57, 56, 59, 60, 61, 62}, 'probability': (0, 0, 0)}, 'Incident Handling': {'configuration': {32, 20, 22, 24, 27, 30, 31}, 'probability': (0.7474, 0.7474, 0.7781)}}
```

This return value contains a data structure describing two missions, **Network Monitoring** and **Incident Handling**. For the first mission, **Network Monitoring**, there is a configuration with a probability of compromise of (0, 0, 0), which means that no attack graph has been generated for that configuration based on the acquired data about vulnerabilities, about the services running on the given devices, and about the open ports, which would describe an attacker's path leading to the compromise of a device. It should be remembered that the given output does not mean that the given devices cannot be compromised, it only means that, based on the data obtained, it was not possible to find any procedure by which an attacker could compromise the given device.

For the second mission, **Incident Handling**, the configuration determined by the combination of the IDs of the individual devices that make up the given configuration was selected as the most resistant configuration. Complete information can be found by ID). Based on the data collected in the database by the tools of the Observe phase, we assigned this configuration a probability of misuse of 0.7474 for confidentiality, 0.7474 for integrity, and 0.7781 for availability. All probabilities range from 0 to 1.

### Format of results in the database

As part of the analysis process, the decision support software calculates partial results for each possible configuration of each mission. The calculation results for each configuration are based on the calculation results for each component in that configuration, such as each device or network service. The results in all cases contain

probability of compromise of the given component from the point of view of confidentiality, availability and integrity.

The relevant :Mission node is linked in the database by a :HAS session to the :Configuration **node**. A :Configuration node is connected by a :CONTAINS session to the device on which it is hosted. The resulting path is **(:Mission)-[:HAS]->(:Configuration)-[:CONTAINS]->(:Host)**. The node **(:Configuration)** contains the results (probability of compromise from the point of view of confidentiality, availability and integrity), the configuration ID and the timestamp of the last evaluation. The results for each device are stored on the :CONTAINS **edge**.

Figures 6 and 7 show a specific representation of the data in the database for one sample mission configuration called **Incident Handling**. Figure 6 shows the evaluation of the mission configuration, and Figure 7 shows the evaluation of the devices in the network that support the given mission.

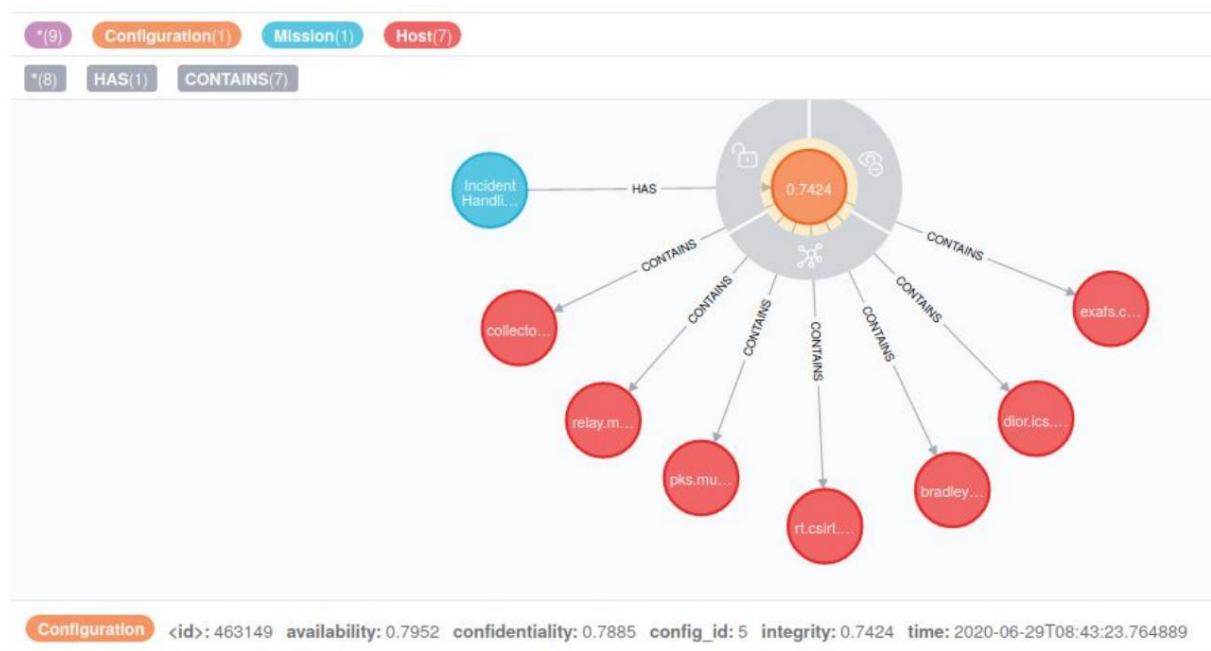


Figure 6: Data format in a graph database (configuration rating).

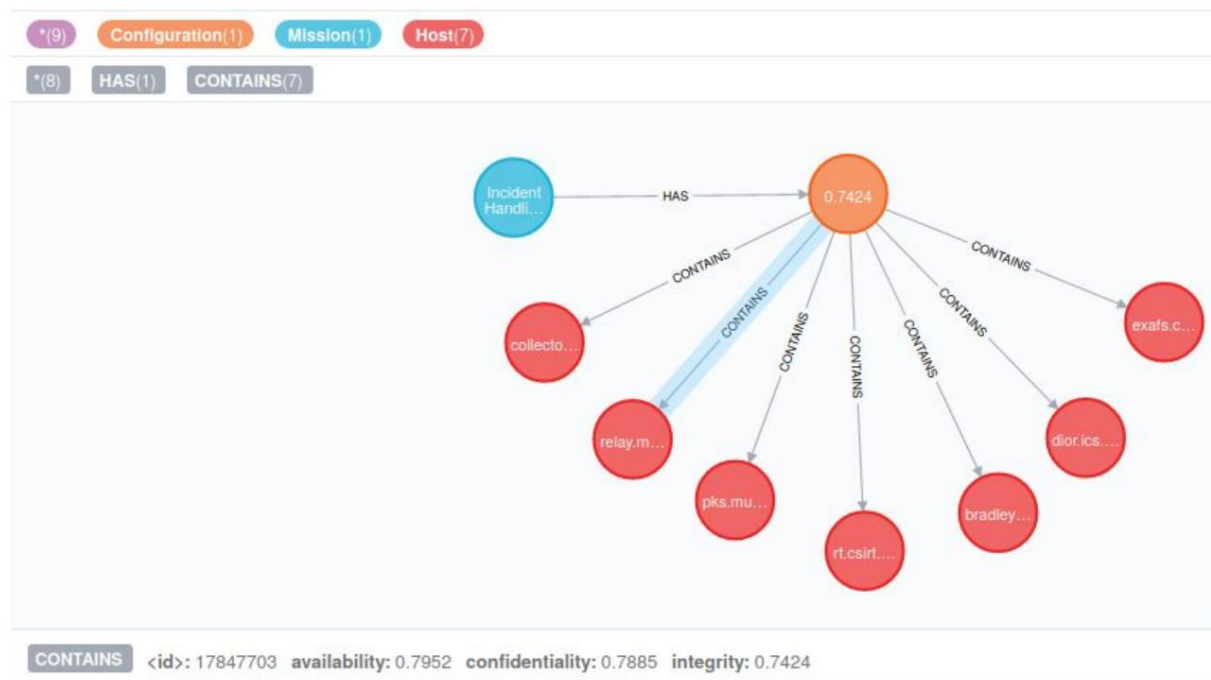


Figure 7: Data format in graph database (equipment rating).

## Tests

The tests presented in the file **attack\_graph\_test.py** test the evaluation of the security situation of some basic modeled cases. To model these cases, the functions **create\_real\_vulnerabilities()** and **create\_fake\_vulnerabilities()** are used, which store real or fictitious vulnerabilities with a given impact in the database. The other data needed to generate the AG, namely IP addresses, services running on the given machines and open ports, are also inserted into the database via **the create\_data()** function. For each of the test model examples, the selection of the correct configuration with the correct probability is checked within the tests.

The test in the **config\_decomposition\_test.py** file checks whether the functionality for detecting available configurations for a given mission and given sample files (**missions.json** or **constraint.json**) finds all configurations in the correct composition.

The **neo4j\_test\_client.py** file contains functions that are used in creating data for model test examples according to the CRUSOE data model in the Neo4j graph database.

# Thanks

The work on the software was supported by the project ***Research of tools for assessing the cyber situation and supporting decision-making of CSIRT teams in the protection of critical infrastructure*** (VI20172020070) solved in ***the Security Research of the Czech Republic*** program in the years 2017-2020 at Masaryk University. The authors of the software are Lukáš Sadlek, Michal Javorník and Martin Husák.