Horizon 2020 Programme
Instrument: Innovation Action

Proactive Risk Management through Improved Cyber Situational Awareness

PROTECTIVE
PROACTIVE RISK MANAGEMENT

Start Date of Project: 2016-09-01

Duration: 36 months

D4.4 Context Awareness Platform v3

Deliverable Details	
Deliverable Number	**D4.4**
Revision Number	**F**
Author(s)	**AIT**
Due Date	**M25**
Delivered Date	**23/10/2018**
Reviewed by	**PSNC/TUDA**
Dissemination Level	**PU**
Contact Person EC	**Alina-Maria Bercea**

The research leading to these results has received funding from the European Union's Horizon 2020 Research and Innovation Programme, under grant agreement no 700071.

Partner Roles

Contributing Partners	
1.	**AIT (contributor)**
2.	**ITTI (contributor)**
3.	**TUDA (reviewer)**
4.	**PSNC (reviewer)**

Revision History

Example

Revision	By	Date	Changes
F	AIT	25/10/2018	Dissemination level set to PU
E	AIT	22/10/2018	Released to Portal
B1	TUDA, PSNC	22/10/2018	General review of the document
A2	ITTI	21/10/2018	Added ASM material
A1	AIT	21/10/2018	Version for internal review

Abbreviation's List

AS | ASM	Asset State Management
BP	Business process
CA	Context Awareness
CDG	Context Dependency Graph
CI	Critical Infrastructures
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CPE	Common Platform Enumeration
CSIRT	Computer Security Incident Response Team
MAIR	Mission and Asset Information Repository
MCDA	Multi Criteria decision Analysis
MIM	Mission Impact Management
NIST	National Institute of Standards and Technology
NVD	National Vulnerability Database
NREN	National Research and Education Network
SCAT	Security Content Automation Protocol
SME	Small to Mid-size Enterprise
SOTA	State of the art analysis
TI	Threat Intelligence
WFN	Well Formed CPE Name

Executive Summary

This deliverable, named Context Awareness (CA) Implementation Platform v3, documents the contents of the CA subsystem delivered for phase 1 of Pilot 2. For the Mission Impact Management (CA-MIM), the software in this delivery is focused on enabling the definition and manipulation of Context Dependency Graphs (CDG) and the handling of Asset State Management (ASM). Specifically, this deliverable:

- Defines a schema to model and represent different forms of CDG;
- Displays the capability to upload and store one or a number of CDG;
- Demonstrates the capability to view and manipulate CDG's
- Implements AssetRanking.
- Describes and implements the process of relating inventory information to asset vulnerabilities using Common Platform Enumeration (CPEs).

It is important to also note that the direction of Mission Impact Management has been changed somewhat to take on feedback arising from the mid-term review to make the development more *scalable* i.e. reachable to a wider audience of users and applications and to deal with dynamic dependency generation. Our focus has been on the first of these goals i.e. to enable the MIM to be used for different dependency applications and to support the dynamic collection of dependency graphs. The collection of dependency information tends to rely on application specific procedures and the number of different applications is large[1]. Therefore, the definition and development of such mechanisms is outside the scope of PROTECTIVE.

Chapter 2 updates the status of the scenario and requirements addressed by this delivery. Chapter 3 updates the description of the main building blocks, interfaces and functional behaviour of the system. Chapter 4 provides more information on the implementation, including user interfaces and external interfaces. Chapters 5 and 6 deal with IPR management and ethical aspects.

[1] Indeed even amongst the NRENS in PROTECTIVE there are different views of what kind of models to use and a variety of mechanisms to collect dependency data.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Scope of this delivery

The software in this delivery is focused on enabling the definition and manipulation of Context Dependency Graphs (CDG's) for the Mission Impact Modelling and the handling of Asset State Management (ASM), specifically, it:

- Extends the schema to model and represent different forms of CDG defined initially in D4.2;
- Demonstrates the capability to view and manipulate CDG's
- Implements the AssetRanking algorithm [1] to map mission priorities to computing assets
- Describes and implements the process of relating CA-MAIR / Inventory Management system information to asset vulnerabilities using CPEs and vulnerability database.

1.2 CA Overview

This document describes a second implementation of the Context Awareness (CA) platform – see Figure 1. As described in D4.1 and D4.2, the CA system is composed of several complementary subsystems

- Mission Impact Management (MIM) subsystem;
- Asset State Management (ASM) subsystem;
- Mission and Asset Information Repository (MAIR).

The high-level architecture of the CA subsystem, as described in D4.1, is shown in Figure 1

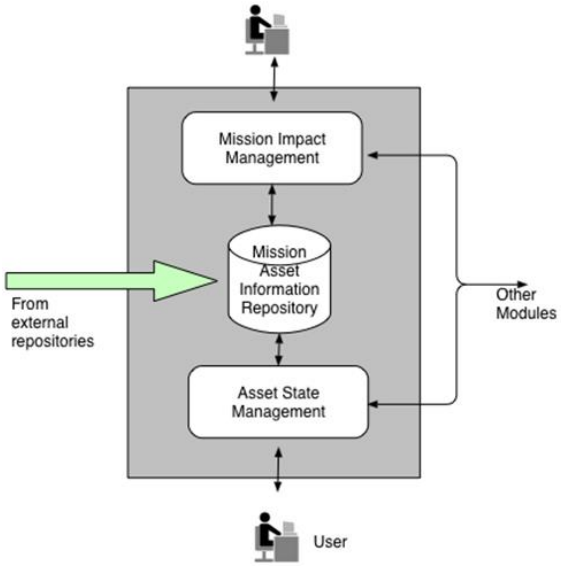

Figure 1: CA System Architecture

D4.1 described the initial CA meta-model and outlined a methodology to capture and document dependencies between entities in that model. D4.2 described the initial software implementation to ingest and store Information Technology (IT) *infrastructure entities* e.g. assets and services. In this document, we focus on how to *model relationships between these entities* and *how to use these relationship models to calculate node and vertex importance*.

1.3 Context Dependency Modelling

A dependency relationship between two objects *A* and *B* means that object *A* needs or uses the services or resources of object *B* in order to fulfill its mission[2]. Conversely, any damage or impairment to object *B* is likely to have an impact on the ability of object *A* to achieve its goal. Dependency relationships, expressed as CDG's, have multiple uses in computing including capturing reliability, robustness, fault tracking, root cause analysis, deployment etc. They are also used in IT security primarily as part of risk assessment in order to show impact relationships between different IT components. We can characterise such use of CDG's in a number of dimensions:

- **Purpose** for which the CDG is created, including[3]
  - Tracing the path of a *threat* to a particular asset through the CDG to help determine mission risk/situation awareness [4] [2,3].
  - Tracing the path of a particular *vulnerability* through the CDG to determine asset risk, [4,5].
  - Determining asset criticality by tracing the dependency from higher level mission objectives to individual assets, [6,7].
  - Assessing the potential effect of an Intrusion Response System action to react to a particular threat on system availability.
- **Layering** of the CDG i.e. what are the *types* of objects between the dependencies that exist and what the nature of these dependencies is.
- **Direction** of dependency propagation e.g. up or down the CDG as well as between layers, within layers or both.
- **Acquisition** of dependency information i.e. is it done statically/manually or dynamically? This is perhaps the most open and difficult question in the use of CDG's because of the massively heterogeneous nature, number and granularity of IT object types.
- **Weighting** of dependencies – how are weightings determined? Weights are used to represent the *degree* of dependency e.g. 0 = no dependency, 1 = minor dependency.
- **Aggregation** of dependencies, see [2] [3] [6] [8] ,, i.e. how to calculate the impact on object *A* from multiple dependent child nodes, each with perhaps different dependency weightings?

We can make a number of observations based on the above list:

1. The constituency of users for dependency modelling is diverse and has many different requirements with respect to the scope and nature of what they wish to model. For example, an NREN CSIRT may wish to model a backbone network at a very coarse granularity and to model the end-user services – e.g. video-conferencing in more detail. An enterprise CSIRT may be focused on software application modelling and may wish to model just the software application and dependencies on a single server. Both of these models may be similar size but have totally different granularity.

[2] The relationship currently being identified is unidirectional whereas dependency relationships can be bidirectional

[3] Given the greater diversity of use we adopt the term Context Dependency Graph (CDG) as a general usage rather than the Mission Dependency Graph (MDG) of D4.1, D4.2

[4] Note we do not mean *attack graphs* here. CDG nodes are assets, services etc i.e. elements that define the *context* of the security system.

2. As follow-on from the last point, the heterogeneity of model types and granularities, there is an equal heterogeneity of dependency capture mechanisms and methods. Some are manual and some – the bulk – are automated. In D4.1, we described a manual method to capture dependencies which can be used by NREN's to capture dependency information for small scale, high value, services. Manual dependency capture is relevant and useful to capture business and mission dependencies but, in practise for large scale infrastructures, an automated approach is required for entity dependency capture. We will not attempt to address the issue of automatic capture within PROTECTIVE – the scope is simply too large. Dependency capture mechanisms tend to be specific to each problem area and generic solutions are not applicable. Even within the NRENS there is not a common approach to capture dependency information.
3. The methods used to calculate dependencies importance also varies greatly. Different aggregation methods (AND vs. OR) may be used. Also, many researchers have used various graph ranking algorithms e.g. PageRank, HITS etc.

The net combination of the above factors encourages us to define a CA system that will enable the capture of many different dependency types of– e.g. mission, threat, vulnerability dependencies – and which will allow system designers to experiment with different types of modelling algorithms in order to choose the most appropriate for their particular use. **Thus, one significant change from our earlier work is that we focus on how to represent and reason over the dependency models rather than how to capture the dependencies. This lead to a more scalable solution approach as the CA can be applied across many different scenarios, including those defined in D4.1.**

2 Scenarios & Requirements

Scenarios, requirements and the architecture are documented in chapter three of the deliverable "D2.1 Requirements Capture, Specification, Architectural Design and Model". In this chapter, those scenarios and requirements, which have been fulfilled or partially fulfilled by this deliverable, are documented. It also includes fulfilments from earlier deliveries.

2.1 Scenarios

The scenarios for PROTECTIVE are shown in Figure 2. Scenario 7 – Context Awareness Development – shaded in yellow, has been partially completed and is additive to the work completed earlier and described in D4.2.

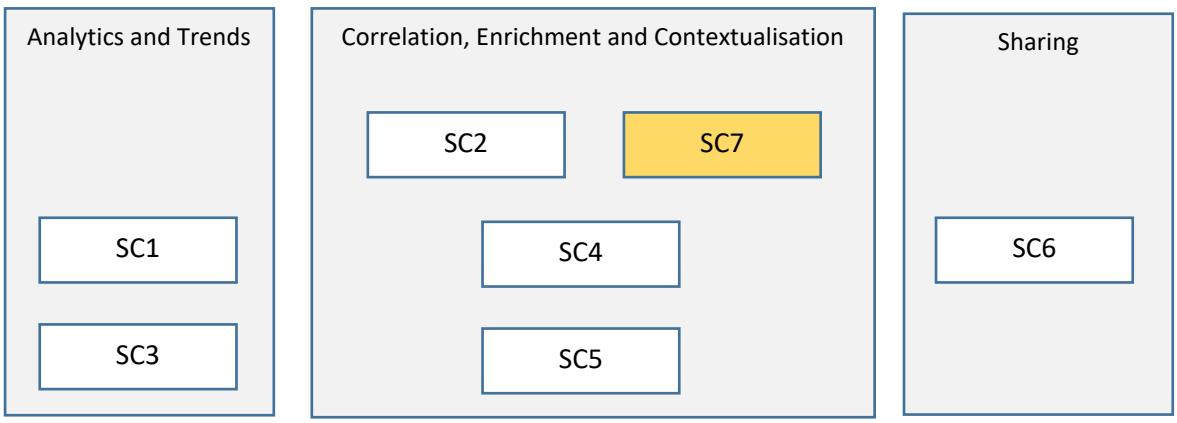

Figure 2: Scenario Status

2.2 Requirements

The following requirements have been fulfilled (“Done”) or partially fulfilled (“Partially”)) by this deliverable.

2.2.1.1 Context Awareness

ID	CA-01	Done
Type	FR	
Slogan	PROTECTIVE must support import of computer and network inventories	
Rationale	To define the relevance of a potential security threat, the system must be able to identify if the affected node or service exists in infrastructure of the operator	
Related Scenarios	SC-7	
Related Requirements	PR-01	
ID	CA-02	Done
Type	FR	
Slogan	PROTECTIVE must support import of asset vulnerability	
Rationale	To inform the operator regarding possible resolutions to the received information the system must be able to look up additional, relevant information	
Related Scenarios	SC-7	
Related Requirements	PR-01	
ID	CA-03	Done
Type	FR	
Slogan	PROTECTIVE must support the modelling of services	
Rationale	This includes business service, IT services and network services	
Related Scenarios	SC-7	
Related Requirements	PR-01	
ID	CA-04	Done
Type	FR	
Slogan	PROTECTIVE must support modelling of users and organisations to support external constituencies.	
Rationale		
Related Scenarios	SC-7	
Related Requirements	PR-01	
ID	CA-05	Done
Type	FR	
Slogan	PROTECTIVE must support asset and service aggregation modelling	
Rationale	Via e.g. network and service groups to capture that capture redundancy such as load balancing.	
Related Scenarios	SC-7	

Related Requirements	PR-01	
ID	CA-06	Done
Type	FR	
Slogan	PROTECTIVE must support the creation of relationships between assets and services.	
Rationale	This is required in order to capture dependencies. It will specify how dependency relationships between the assets and mission functions are modelled between the entities in the taxonomy	
Related Scenarios	SC-7	
Related Requirements	PR-01	
ID	CA-07	Done
Type	FR	
Slogan	PROTECTIVE must support the definition of top level mission/business risk assessment criteria	
Rationale	This is needed for PROTECTIVE to be capable of assessing the priority of a particular threat to a specific asset	
Related Scenarios	SC-7	
Related Requirements	PR-01	

Table 1: CA Requirements

2.2.1.2 User Interfaces

ID	UI-01	Partially
Type	FR	
Slogan	PROTECTIVE MUST allow administrators and operators to manage and interact with the systems	
Rationale	A key part of improving the workflow of the operator is to provide a intuitive and easy to use interface to manage the PROTECTIVE system and to remain constantly informed	
Related Scenarios	SC-1, SC-2, SC-3, SC-4, SC-5, SC-6, SC-7	
Related Requirements	N/A	
ID	UI-05	Done/Partially
Type	FR	
Slogan	PROTECTIVE MUST support referencing between stored objects	
Rationale	To allow the operator to navigate through the vast amounts of stored information, the relevant objects need to be linked together; e.g., while investigating a meta alert, the operator should be able to get additional information about the system that may be impacted, the attackers IP address, etc.	
Related Scenarios	SC-1, SC-2, SC-3, SC-4, SC-5, SC-6, SC-7	
Related Requirements	N/A	

Table 2: UI Requirements

3 Architecture & Design

In this chapter, we describe the main building blocks, interfaces and functional behaviour of the system. The CA subsystem is part of the Enrichment subsystem of the PROTECTIVE node – see Figure 3.

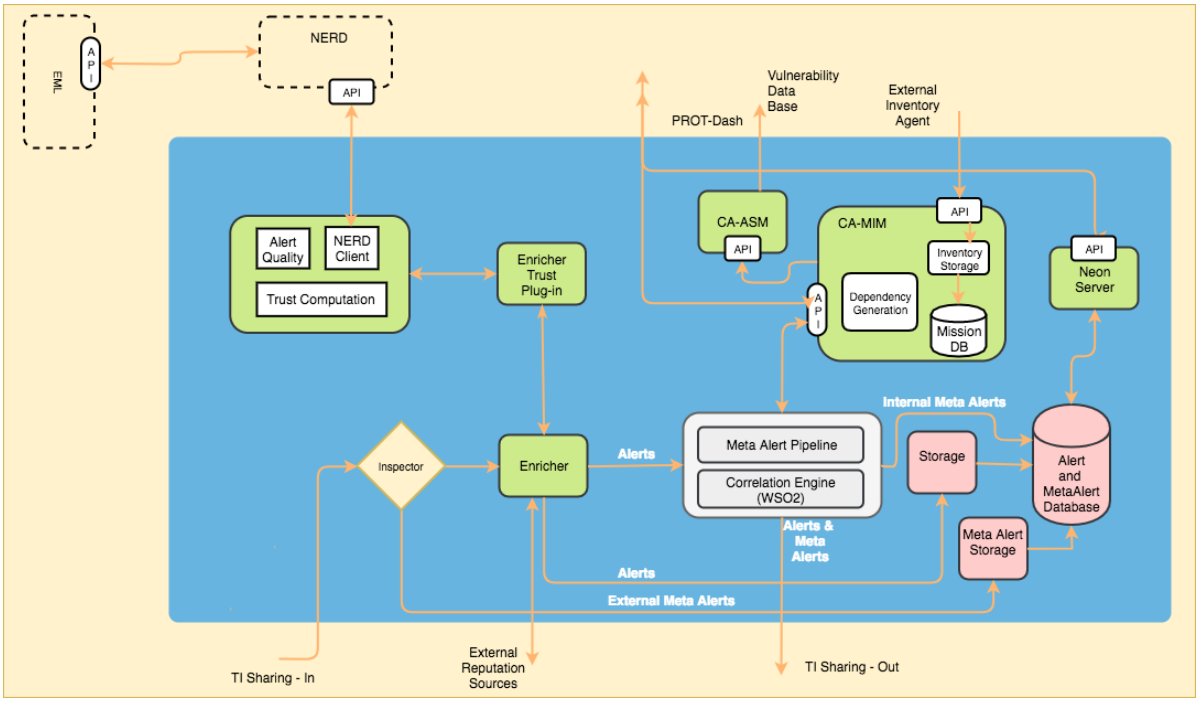

Figure 3: Alert Flow Processing

The CA subsystems, CA-ASM and CA-MIM, can be seen at the top right-hand corner of the diagram.

3.1 CA Meta-model

One consequence of the decision to widen the scope from MDG to the more general CDG has been a need to revisit the CA meta-model (previously defined in D4.1).

To recall terminology, in D4.1 CA we define a *mission* as the combination of a *goal or objective* and the *tasks* required to achieve it. We now organise the meta-model in a more structured form to categorise the elements of the model in successively dependent *layers* in order to more clearly define the types of dependencies. This is in turn will allow us to describe the individual dependencies in detail and define schemes to express and reason about them. With respect to the CDG approaches referenced in Chapter 1.3, and [8, 9, 10] we can identify the following representative dependency layers:

- Hardware (on a device),
- Network (physical connectivity),
- IT services,
- Applications,
- Information Containers (e.g. databases),
- Information Assets (logical),
- Software (physical image, library etc.),

- System artefact (process, file, network socket),
- Business Process,
- Operational Tasks,
- Business Objectives,
- Mission.

Some of these are similar e.g. Business Process/Operational task or IT/Services/ Applications etc. As a concrete example of layering see that of RisMAP [6] in Figure 4.

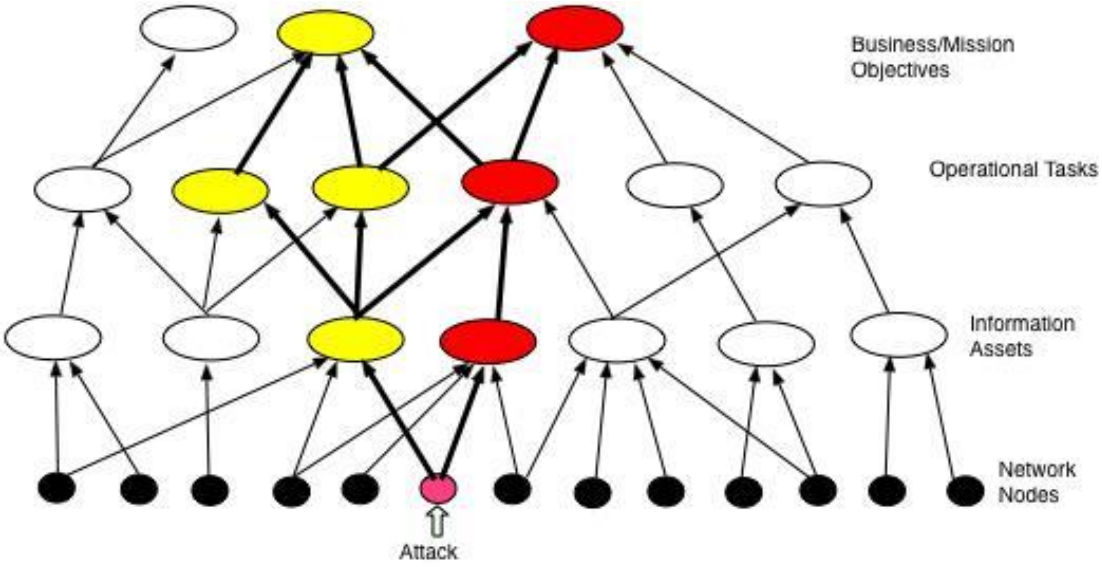

Figure 4: RiskMAP Dependency Model [after Watters, 2009]

Based on a detailed consideration of these various approaches and the likely needs of PROTECTIVE end-users (including those NREN's and SME's within the project but also extrapolating to more general use) we reshape (and simplify) the meta-model[5] from D4.1 into a number of layers – see Figure 5.

[5] This version does not yet include modelling of vulnerabilities

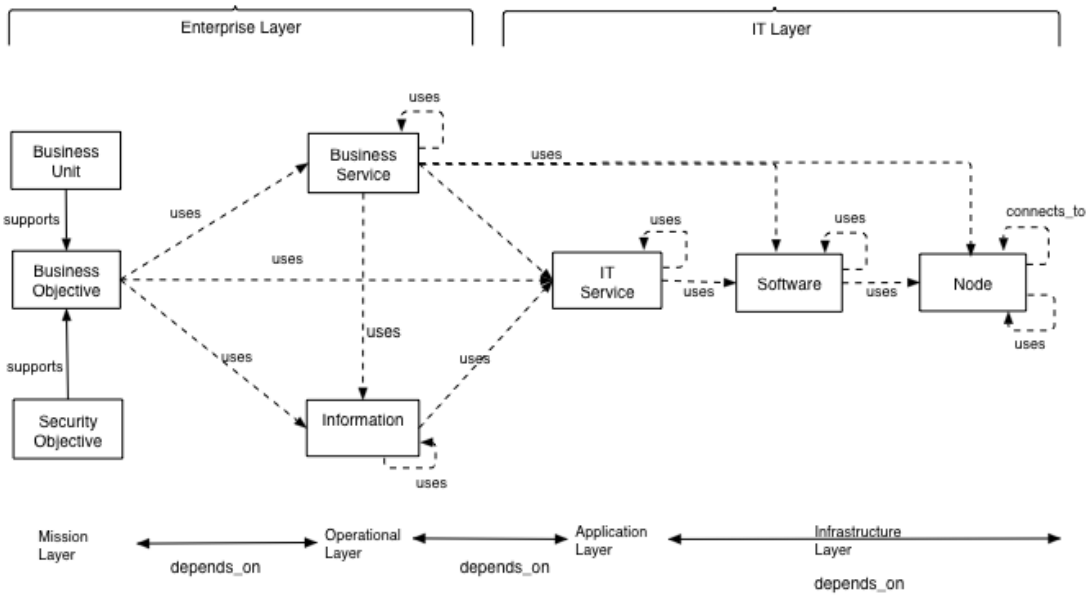

Figure 5: CA Enterprise Dependency Model

Dependencies are indicated by dashed lines and we can see there are a variety of them. Using the above model we identify the following dependency types:

1. Inter layer Mission to Operational layer – this involves relationship between the Mission entity and the Business Process (BP), Information entities.
2. Intra layer Operational layer between Business Process (BP) entities, Information entities and BP to Information entity
3. Inter layer Mission to Application layer i.e. Mission to IT Service
4. Intra layer Applications layer i.e. IT Service to IT Service
5. Inter layer Application to Asset layer i.e. IT Service to Software
6. Intra layer Asset layer i.e. SW to SW, SW to Node and Node to Node.
7. Inter layer Operational layer to Infrastructure layer. The purpose of this “two-layer jump” is to allow abstract modelling of dependencies from business entities to network or node entities directly. Not every Operational layer entity depends on an IT Service, though, all must are realised as Software. The CDG can be simplified further by omitting the Software altogether and showing the Operational Layer dependency directly on the hardware. These features make it possible for example to model RiskMAP type relationships.

The classes in Figure 5 are, for the most part, parent classes that can be specialised as shown in Figure 6.

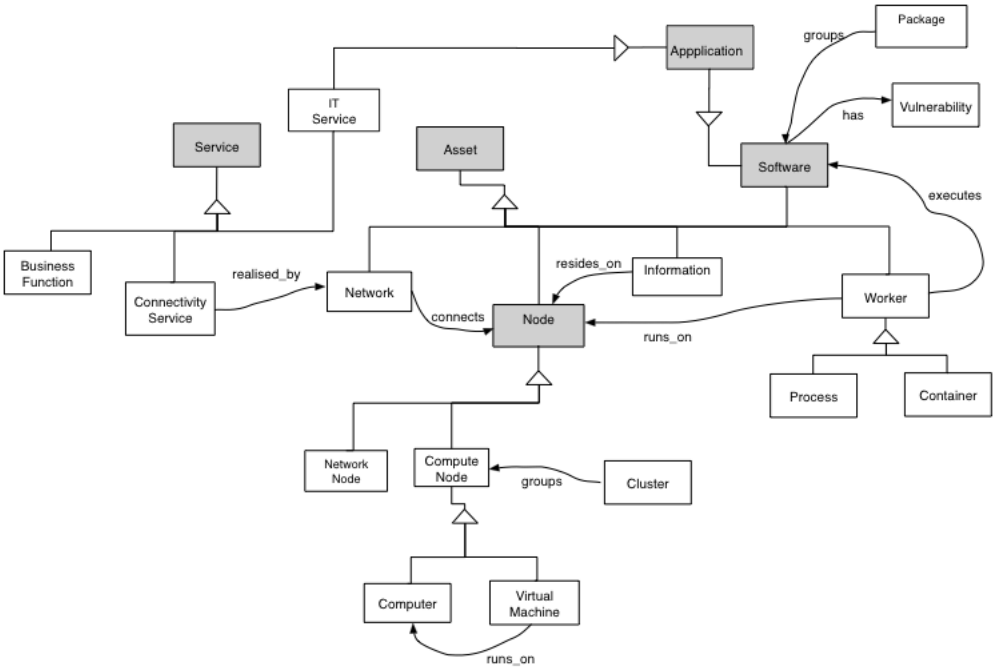

Figure 6: CA Meta-model Inheritance Hierarchy

This inheritance capability enables the CA-MIM meta-model to be also used to model dependencies between virtual nodes. Node to node includes execution dependencies e.g. a server may hosts VM that in turn hosts a Docker container as well as physical connectivity dependencies via network nodes. Furthermore, by including the "Worker" concept, the meta-model may easily be extended to capture Operating System artefact (process, file, network socket) type dependencies. However, this is not shown at present in Figure 5.

The CA-MIM CDG thus has the capability to flexibly model dependencies and can be used for a wide range of asset dependency scenarios. Thus it could be used to implement any or all of the dependency modelling approaches described in the literature:

- All layer (inter-layer) RiskMAP Mission type modelling approach as described in Figure 4 or similar described by [7, 9, 11],
- Intra layer System artefact modelling [10, 12],
- Intra layer modelling for Network Service and Infrastructure layers as in [5, 8]
- Interlayer between Infrastructure, Application and Operation layers as described by [2, 3] – these authors are concerned to propagate impact upwards rather than risk downwards; however these semantics can also be captured.
- The meta-model is also capable to model dependencies between virtualised network layers that are becoming ever more complex as the use of Software Defined Networking (SDN) techniques. This capability is captured by distinguishing between the expression of intra-layer and inter-layer dependencies when capturing the CDG. This design is based on the network topologies modelling approach described in RFC 8345 [13].

3.2 Model ranking

Most related work is concerned with determining the relative weighting or *ranking*[6] of individual nodes. There are a number of considerations to be taken into account to rank including the choice of algorithm, determining how to combine layers, choosing how to weight dependencies and selecting how to propagate dependencies.

The majority related work uses one or more graph ranking algorithms, with PageRank being the most widely used [4, 5, 14, 15] while HITS [16][7] and a variant of PageRank adopted for ranking assets called AssetRank is proposed by Sawilla [1]. However, as suggested in [4] and [5], and demonstrated in [15] and [16], different ranking algorithms can flexibly be applied to the same problem once the ranking graph is obtained. Jakobsson [2] uses constraint programming to propagate dependency information while other authors suggest Multi Criteria Decision Analysis (MCDA) [7, 11] (as we also did in D4.1) or a combination of MCDA and other methods [6].

In addition to choosing which ranking algorithm to use, researchers must determine how to combine the different layers. The authors of RiskMAP [6] only focus on propagating inter layer dependencies while ignoring intra layer dependencies. Some authors consider only a single layer as, e.g. the network services /IT services layer [14, 15, 16], – in which case inter layer dependency is not an issue. Although they include layering, neither [2] nor [3] distinguish between intra layer and inter layer dependency propagation. On the other hand, Zhuang [5] creates three separate graphs of (intra-layer) software dependencies on a single node, (intra-layer) network service dependency and (inter-layer) software-hardware node dependency. The author then applies a ranking (PageRank) to each graph and combines the results in a weighted fashion to derive a final ranking. Jiang [4] does not use layers and so in concerned with only a single uniform graph.

A third issue to consider is weighting dependencies between nodes and the related issue of choosing initial node values. Indeed, in many cases [2, 3, 6, 7, 8, 14], researchers annotate a sub-set of nodes with specific weights to reflect the relative importance of these nodes and the rest of the nodes with default weights. The ranking algorithms are then used to propagate the weights through the graph. In general, PageRank assigns uniform weights to all nodes and arcs. AssetRank [1] modifies PageRank to vary the weighting on nodes. Research using MCDA [6, 7] also vary node weightings. Some researchers [1, 6, 7] assign weights to arcs to reflect the degree of dependence e.g. 0 = no dependence, 1 = minor, 2 = major, 3 = critical. Some researchers do not assign specific weights and work only with the ranking algorithm defaults, [4, 5, 16. In general, the issue of how to assign weights in dynamic graphs is a difficult and largely open research question.

Finally, researchers have to consider how to aggregate dependencies i.e. how combine the values of a number of incoming dependencies to calculate the node value. The two most common aggregators are AND and OR. In the former case, the updated value reflects the combination values of *all* of the child dependencies while the latter reflects the value of *any one* of the dependencies. In practise, an AND operation results in the minimum value being selected while OR reflects the maximum value being selected – see [2, 3] for a further discussion. RiskMAP [6] also discusses the issue for mission priority propagation introducing SUM and MAX operators. How to incorporate AND/OR semantics in

[6] In this chapter we outline the prior work carried out on this topic. The implementation of ranking for CA is not included in this this delivery but the background is documented for use in the next deliverable.

[7] This is repeated [15], applying HITS rather than PageRank.

to a dynamic graph is addressed only by AssetRank [1], although [14] does try to determine AND/OR type dependencies at run-time though they do not specifically consider aggregation issues.

3.3 CA Ranking

In D4.1, we described a manual scheme for calculating and propagating Mission priorities based on MCDA. This approach serves as a default ranking scheme for PROTECTIVE and does not require the use of the CA-MIM to do the ranking. The manually ranked model can be stored in CA-MAIR. However, this manual ranking does not scale very well and does not make best use of the capabilities of the CA system. Therefore, for later versions of the CA system, we have expanded upon this original scoring approach. We consider implementing two ranking schemes in CA-MIM.

The first scheme, which we designate **MAIR-Rank**, is based on a combination of RiskMAP for inter-layer propagation and one, or more, graph algorithms for intra-layer ranking. MAIR-Rank is not yet implemented and so some of the parameter choices remain open in the description below. The scheme will proceed as follows:

1. Firstly the mission nodes are prioritised relative to each other. This can use MCDA as in D4.1 or some variation as in [6, 7, 11]. Alternatively, it could be implemented as in the OCTAVE Allegro [17] risk assessment methodology, where a simple ordering of priority is made without taking any weighting into account. In any event, the mission nodes are assigned numerical values that reflect their relative priorities. These values are most likely calculated manually after consultations with subject matter experts.
2. The edges between the nodes in the Mission layer and the Operational layer are assigned a weighting with a numerical range to reflect the degree of dependency. Various examples of such weightings are to be found in D4.1 and [1, 6, 7, 8, 18]. The value of each mission nodes is multiplied by the edge value and the result is transferred to the depended node in the lower layer where it is aggregated using a method as described above. Each asset in the Operations layer must also be assigned an initial value – this value will depend on the ranking method to be used for the layer. Also each edge in the Operations intra-layer graph must be initialised with a value – these also may depend on the ranking algorithm to be applied or they may have been assigned manually.
3. Once the mission priorities have been transferred to the nodes in the Operational layer, the graph-ranking algorithm can be run to calculate and propagate intra-layer dependencies in the Operations layer. We consider AssetRank to be the default algorithm for CA-MIM but other algorithms could be used.
4. Step 2 and 3 are now repeated to transfer nodes values from the Operational layer, calculate the intra-layer dependencies and so on to the Infrastructure layer.
5. This results in each node being assigned a criticality value.

MAIR-Rank is intended to give flexibility to the Risk Manager to choose and combine the most valid criticality ranking algorithms and to allow it to experiment with different node and edge weighting in order to get the best possible result. As noted, MAIR-Rank is a powerful scheme and not all options have yet been implemented in this version of the system.

The second ranking scheme which we consider is to use AssetRank across the whole CDG graph i.e. we do not distinguish between layers in the CDG. Instead, we:

1. initialise the Mission nodes using a scheme such as Step 1 above for MAIR-Rank, and
2. run the AssetRank algorithm for the whole CDG.

We closely follow the scheme for AssetRank proposed by Sawilla in [1, 21] to assign initial values to the nodes and edges and to calculate dependency values. The reader is referred to these publications for more details on the AssetRank algorithm.

AssetRank is implemented in this version of the software and is the default ranking method for CA-MIM. AssetRank is chosen for ranking primarily because of the well demonstrated value of its parent, PageRank, to calculate the value of assets in a dependency graph. AssetRank therefore has both a sound theoretical inherited and a well demonstrated real world application. Asset Rank however allows the inclusion of variable node and edge weightings which PageRank does not allow. Furthermore, AssetRank allows an AND or OR semantics for node dependency aggregation while PageRank allows only OR aggregations.

3.4 Asset State

The main goal of PROTECTIVE Context Awareness Asset State (AS or ASM) is to enrich CA data with information related to potential vulnerabilities of assets. According to the literature reviewed in D4.1, the most common approach in research is to use CVE/NVD databases and the CVSS framework to get vulnerability scores of assets in a network. The main limitation of CVE/CVSS listings is that they may not provide a comprehensive view of the network as a whole. This can result in disinformation for decision-makers while they analyse CVSS scores without other context information. CA as a whole attempts to fill-in parts of this information void. CA gathers and merges lower-level information (through inventory systems e.g. FI), high-level information through defined methodology, and also enriches this information by correlating assets with their potential vulnerabilities. Therefore, CA can provide more insights into the network as a whole, including various elements' interdependencies as well as their criticality to the mission at hand.

Considering CVSS score alone, one of the most common topics in research is to adjust CVSS (CVSS sub-metrics) or to compute new scores based on CVSS. Although these efforts are made in order to potentially provide a "better CVSS", there is an inherent problem to this approach. While CVSS is often imperfect, it is widely known and recognized. Therefore professionals tend to trust CVSS score more than other, unknown – and usually also arbitrary – metric. Therefore CA/AS takes the approach of presenting raw vulnerability CVSS score. The main difference is that CVSS supplements (enriches) information, rather than being a central decision maker.

Furthermore, as part of PROTECTIVE exploitation efforts, we have analysed some of the needs of CI (Critical Infrastructures). It seems that in CI (quasi) passive vulnerability assessment, e.g. based on network configuration, is preferable over active scanning. With proprietary equipment and the "security through obscurity" vendors' approach, active scanning could result in unforeseen network devices operation, a risk which cannot be accepted when dealing with CIs.

D4.1 concluded that the crucial step of gathering the data about network under analysis – e.g. its topology, present software or feeding model with knowledge – is not the primary subject in scientific considerations. Even if the data-gathering aspect is given more focus, resulting tools are usually not made available and their performance cannot be verified. As such, this aspect is often omitted and solutions are often tested with artificial or "handcrafted" datasets. This may be the result of the fact that while researchers are aware of the data provided in NVD[8], they are often not aware of the obstacles present when trying to relate network elements to the NVD data.

[8] National Vulnerability Database maintained by NIST (National Institute of Standards and Technology)

Figure 7 presents a general approach to relate assets (software) to vulnerabilities in NVD applied in Context Awareness Asset State (CA-AS or AS). AS allows searching for critical vulnerabilities in software or lists of software. Solutions use CPE dictionary definitions and vulnerabilities data provided in the National Vulnerability Database (NVD) maintained by NIST. AS can also use pre-defined (labelled) data to mitigate some of the problems associated with automatic CPE-based vulnerability reporting, as described in Table 1.

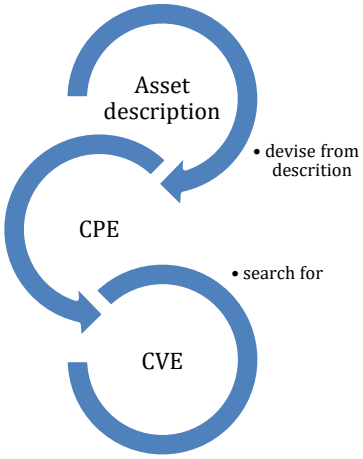

Figure 7: Overall approach to CPE-based vulnerability reporting

Process of vulnerability searching starts after receiving requests via REST API. Based on provided information about the product (name, vendor and version), CPE candidates are constructed, compared, selected, and finally the NVD data-sets are searched. As a result, currently the information about asset highest CVSS score is returned. For performance, matching results can also be cached in anonymised way[9] in the database. This approach allows for significant reduction of processing time for subsequent calls of the same query. In the case of the query is requested first time and it's also not found in the labelled datasets, two main phases can be distinguished:

- CPE-matching phase
- CVE-search - vulnerability searching phase (maximum CVSS selection)

This process may seem straightforward at the first glance. In order to perform this process, data between different databases must first be correlated. Unfortunately, given vast number of vendors, multiple inconsistencies in naming conventions across datasets, poor (or non-existent) synchronization between databases, as well as other problems, the process of relating assets/software becomes a challenging topic.

In order to automate the vulnerability search process, we first use information from Inventory System (ingested by MAIR) to devise proper Common Platform Enumeration (CPEs) of the assets' software instances. CPE is defined as "*a standardized method of describing and identifying classes of applications, operating systems, and hardware devices present among an enterprise's computing assets*"[10]. CPE is a part of SCAP[11] standard proposed by NIST. CPE consists of a number of specifications. Combined, these modular specifications perform various functions – see Figure 8.

[9] Meaning that it is impossible for AS to track the cached inventory items back to a specific asset
[10] https://csrc.nist.gov/projects/security-content-automation-protocol/specifications/cpe/
[11] Security Content Automation Protocol

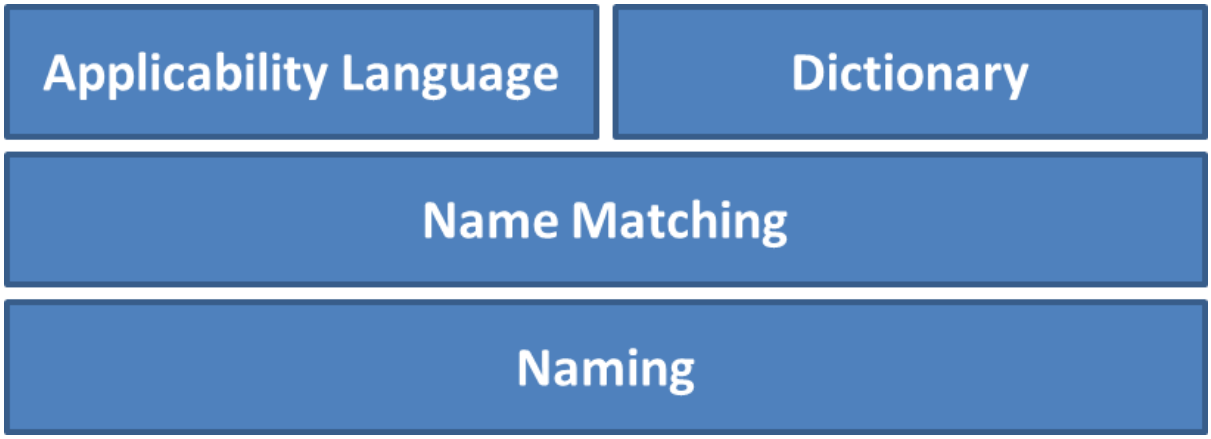

Figure 8: CPE layers - each layer is defined by separate specification[12]

In general, CPE naming scheme uses the so-called Well-Formed-CPE Name (WFN). WFN can be used to construct a "formatted string"[13] – a process referred to as CPE binding – that can then, in theory, be used by e.g. security automation software. An example of WFN and CPE string and URI is presented in Figure 9. ASM follows CPE 2.3 specification format, although the notation of 2.2 can be supported for backward compatibility.

WFN:

```
wfn:[part="o",vendor="microsoft",product="windows_vista",version="6\.0",
update="sp1",edition=NA,language=NA,sw_edition="home_premium",
target_sw=NA,target_hw="x64",other=NA]
```

WFN bound to a URI:

```
cpe:/o:microsoft:windows_vista:6.0:sp1:~-~home_premium~-~x64~-
```

WFN bound to a formatted string:

```
cpe:2.3:o:microsoft:windows_vista:6.0:sp1:-:-:home_premium:-:x64:-
```

Figure 9: Example WFN and its binding to CPE2.2 and 2.3 formats[14]

CPE specifies also a CPE-dictionary format. Although the specification can be used by anyone that wishes to include their custom CPE descriptions, NIST hosts the Official CPE Dictionary, which is the *"authoritative repository of identifier names*". AS uses the Official CPE dictionary to perform CPE-matching. Thus, in the remainder of this section, whenever we refer to CPE-dictionary, we in fact refer to Official CPE Dictionary, which is a specific instance of the CPE-dictionary.

In the first step, AS constructs WFN based on data from Inventory System. Next WFN binding is performed, and different combinations of product, vendor and version are created. Next, generated combinations are combined into a new set of CPE identifiers. Then CPE identifiers containing any of the generated combinations are selected from the dictionary. In general, this step is adhering to the specification defined in SCAP, although modifications are made to account for different possible formatted string combinations. AS can assign weights to specific WFN attributes. Empirical tests performed with AS for common system set-ups in Windows environment[15] have shown that these weights do not seem to have a significant impact on CPE-matching performance. The values may be

[12] https://csrc.nist.gov/projects/security-content-automation-protocol/specifications/cpe/
[13] Formatted string is used in version 2.3 of the CPE specification, while version 2.2 uses URIs to describe CPEs
[14] https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir7697.pdf
[15] Please refer to Annex C for more details on configuration used

adjusted in the future based on data obtained in the pre-pilot integration phase. This is to mitigate some of the possible obstacles identified in Table 1. Finally, if a similarity threshold is crossed – this is in order to avoid matching software that does not exist in CPE dictionary or NVD feeds – an identifier with the most similar version is assigned.

Once AS obtains a CPE identifier, we can use it to correlate specific vulnerability data from NVD and select vulnerabilities with high CVSS. Though, this presents some further obstacles related to inconsistencies in NVD/CPE entries. Researchers[16] as well as other sources[17] present that the challenges in providing CPE-based vulnerability reports are related to the aspects shown in Table 1.

Table 1: Common challenges related to CPE-based vulnerability reporting and mitigation strategies

CVE entries without CPE entries
There are hundreds of CVE entries that leave the CPE filed blank; sometimes the field is left blank even though, for example, software with the described vulnerability has CPE assigned so there would be no logical obstacle to fill-in the CPE field. This problem can be fixed by manually labelling the data or – in some cases – searching CVE entries description for references of CPEs. As the searching for references in CVE descriptions would have a negative impact on performance while providing value in few instances, labelling data seems the most appropriate approach.
CVE and CPE-dictionary synchronization problems
Products without assigned CPEs: there are entries in CVE that refer to CPE's that are not defined (do not exist) in CPE dictionary. Therefore when constructing CPEs based on WFNs, CPE-matching will fail to assign proper identifier in such situation. This can overcome by either extracting CPEs from CVE entries (or labelling data) and feeding them into separate DB or by searching through CVE entries while performing CPE-matching. For performance reasons we choose extracting CPEs from CVEs as the preferred solution.
Errors in CPE-binding: in CVE entries, tens of thousands of CPEs can be found that do not exist in CPE dictionary. These are largely due to the fact, that the CPEs contained in CVE entries often use the WFNs in a non-standard (or erroneous) way by e.g. joining fields instead of following "standard" CPE-binding procedure. This results in CPEs that are constructed based on the same WFNs, but are expressed as different CPEs
CPE dictionary deprecation
CPE dictionary is not only updated with new information, but the information already present in it can be changed and removed. There are thousands of entries that have either been removed or changed. This is sometimes due to trivial reasons such as human-made typos in CPEs. Although these may be fixed, it is hard to estimate how many mistakes go unnoticed in CPE-dictionary. This again would lead to improper CVE-matching results. Authors in [19] point-out that this can be to an extent fixed by using e.g. string similarity algorithms such as Levenshtein edit distance[18]. For performance reasons, we currently are not using similarity algorithms with CPE-matching. The problem seems to be to an extent mitigated by the fact that this type of error does not occur often with popular software, and if it does, it usually is spotted and fixed within reasonable time.
Inconsistencies between "vulnerability conditions" and "vulnerable software list" in CVE
In theory the "vulnerability conditions" is meant for describing specific system configuration conditions that need to be met in order for the vulnerability to appear. Unfortunately, this filed sometimes provides little to no information as it e.g. sometimes only repeats information from the "vulnerable software" section. The usual practice, also employed by ASM – also for

[16] https://arxiv.org/pdf/1705.05347.pdf
[17] https://www.cvedetails.com/how-does-it-work.php
[18] https://en.wikipedia.org/wiki/Levenshtein_distance

performance reasons – is to apply the "better safe than sorry" approach and consider the vulnerable software field regardless of the configuration field.
CPE names inconsistencies
One software product is sometimes referred to in a different manner in CPE-dictionary, NVD, sometimes meaning, there are e.g. a couple of different CPEs in the dictionary itself. As mentioned before, this problem is magnified by the lack of proper synchronization between CPE and NVD, thus creating additional confusions. An example is Internet Explorer, which can be referenced as "IE" or "Internet Explorer" in NVD. This can be solved by labelling data manually.[19]

Although CPE specification was published as part of SCAP – whose aim was to automate security processing – the above challenges make automatic CPEs processing a hard and often infeasible approach. This is confirmed in publications such as [19] or [20]. As completely automatic CPE – CVE assignment process is error-prone, CPE-based solutions need to either:

1) use manual user supervision over the assignment process, or
2) use labelled datasets.

Therefore, AS allows for including labelled reference datasets / manual definition of translations for CPE-matching phase representations (e.g. for the most popular software). We plan to use opportunity presented by the integration phase and Pilot 2 phase 1 to calibrate/optimize a labelled dataset that would aid automatic vulnerability assignment. Especially correctly labelling "product version" may be important as there are many discrepancies in software versions in Inventory / CPE / CVE. This may lead to finding too many CVSS with score 10 - especially if all software versions need to be included in report due to CPE-CVE discrepancies. If the translation is not defined, CPE matching phase is done automatically, employing mitigation strategies as described in Table 1. The high-level block diagram of the AS algorithm is shown in Figure 10.

[19] Moreover some of the software items are miss-categorized in NVD, which does not impact CPE matching in a distinct manner, but can be confusing for anyone using websites such as e.g. "cvedetails".

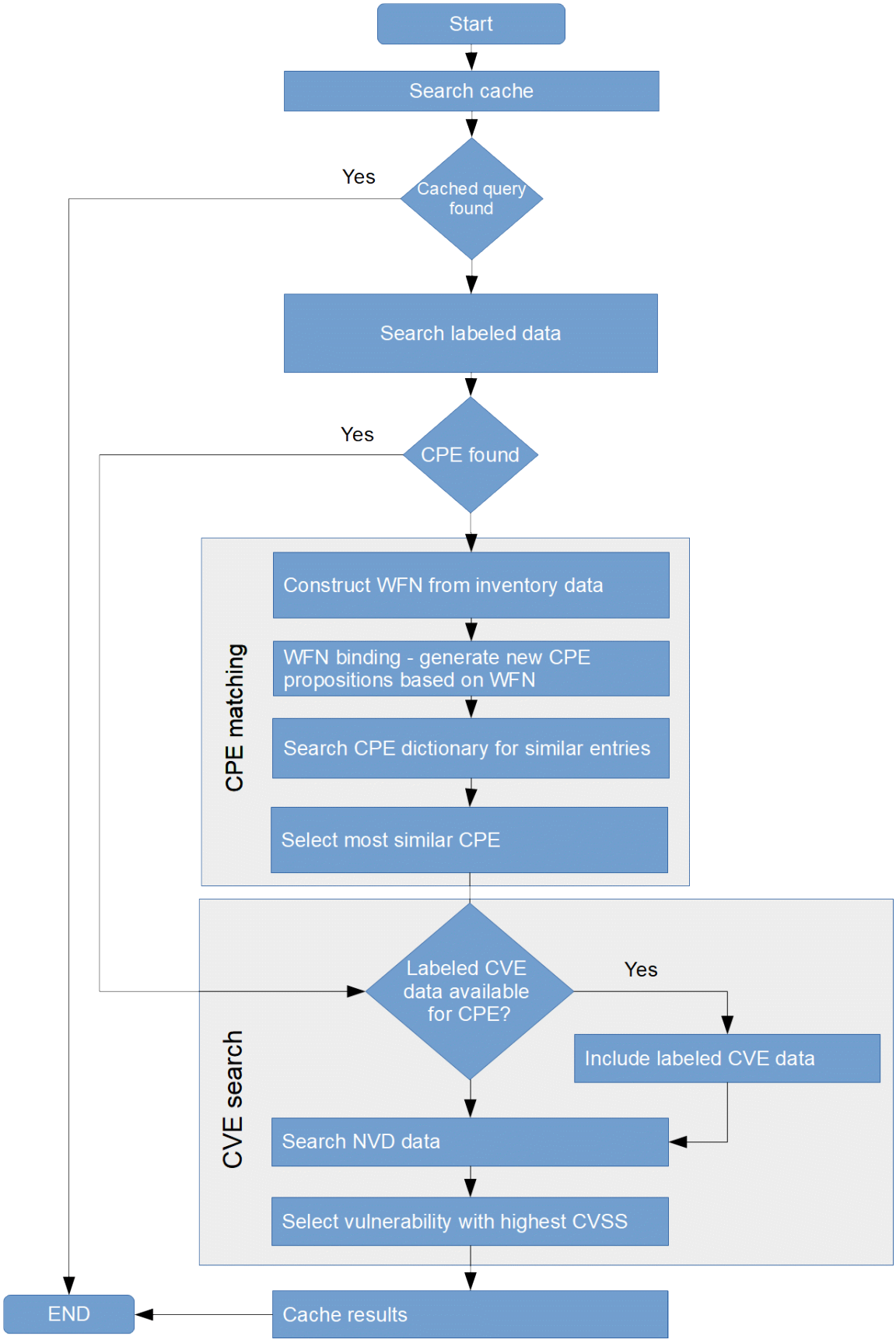

Figure 10: high-level AS algorithm description

3.5 Use Cases

The CA system has the following use cases.

3.5.1 Inventory Upload

CA provides an interface that enables remote inventory agents to upload inventory information from computers to the MAIR. While the interface is usable by any agent, CA comes with built in support for the open source FusionInventory asset management agent [20]. CA-MIM retrieves asset state information for each SW asset from CA-ASM at upload time and stores this information in the MAIR. The sequence of events is shown below.

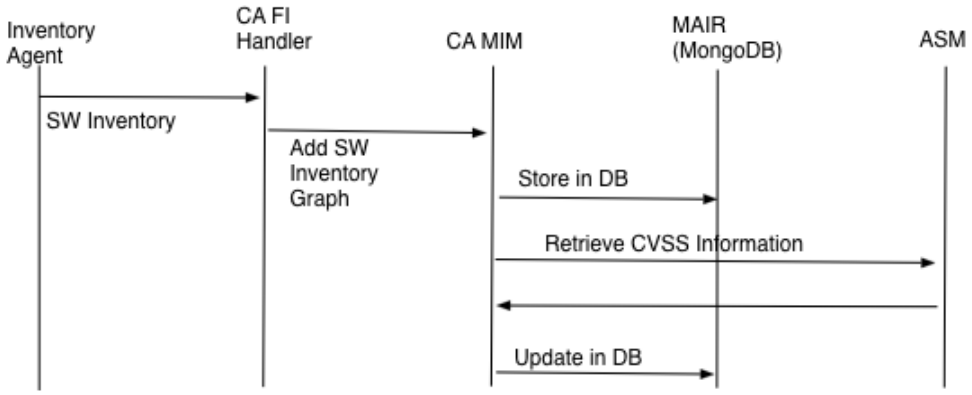

Figure 11: Inventory upload to CA

This interface contains primitives for creation and modification of CDG's. These operations are restricted.

3.5.2 CDG upload

The CDG is uploaded as file according to the format specified in Annex A. A graphical user interface is provided in Prot-Dash to enable a user to upload a CDG file. Note however that Inventory agent may already have uploaded part of the context graph for the Infrastructure layer. Dependency information may be added to existing objects by referencing them in the CDG upload file and adding dependency attributes such as weighting. CA-MIM will automatically update the object in the MAIR. If the object does not already exist in the DB then CA-MIM will create it and update its information.

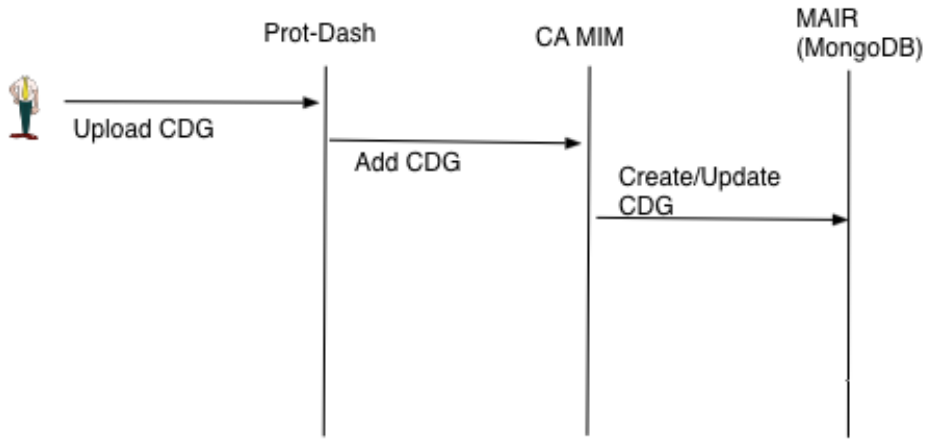

Figure 12: Upload a CDG

3.5.3 ASM upload

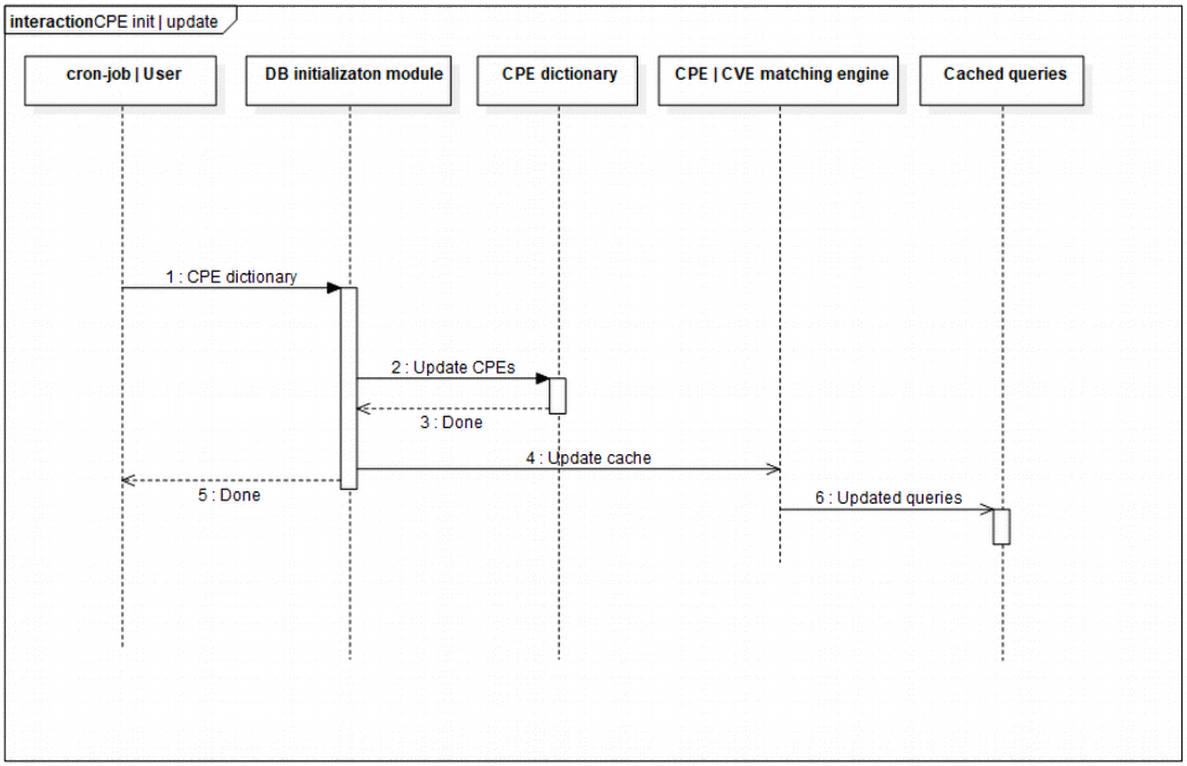

Figure 13: Upload / update CPE-dictionary / CVE data

Although Figure 13 presents sequence diagram for CPE upload / update, the sequence for CVE update and labelled data are similar. A (user-defined) cron-job is responsible for downloading new CPE-dictionary / NVD-feeds periodically (e.g. once a week) and executing an associated API call. For a description of the API, we refer the reader to Annex B.

CPE upload / update

The CPEs are uploaded as files according to the format specified in CPE-dictionary specification and are provided in a dictionary in XML format. The CPEs are then processed by the DB's initialization module. New CPEs are added and deprecated ones removed from the database. The DB initialization can handle both an install of a full CPE-dictionary, as well as an incremental upload of new / deprecated CPEs. If the CPE does not exist already in the DB then the AS will create it and update its information. If any of the cached queries in the DB are marked as not having any CPE matches, the AS will automatically and asynchronously attempt to update relevant cached results to see if the cached results match any of the new CPEs in the DB.

CVE upload / update

The process of updating CVE feeds is similar to the CPE upload. The main difference is in the changes made by the DB initialization service to the cached results. Cached queries are checked for CPEs that can be related to new CVE entries and the results of these CPEs are then updated.

Moreover CPE / CVE labelled data can be uploaded. In this case, the user can provide a pre-defined labelled dataset in which specific inventory elements have fixed CPE-binding and CPE-CVE bindings.

3.5.4 Query and Management of MIM

A variety of information can be queried from the MIM. This includes information about CDG themselves as well as their content. The latter queries are directed towards finding the most critical asset or the criticality of a particular asset identified by IP or DNS key. It also includes similar queries for the asset state information related to vulnerabilities. The information may be retrieved by either PROTECTIVE'S staff or by other PROTECTIVE applications such as the Correlation function.

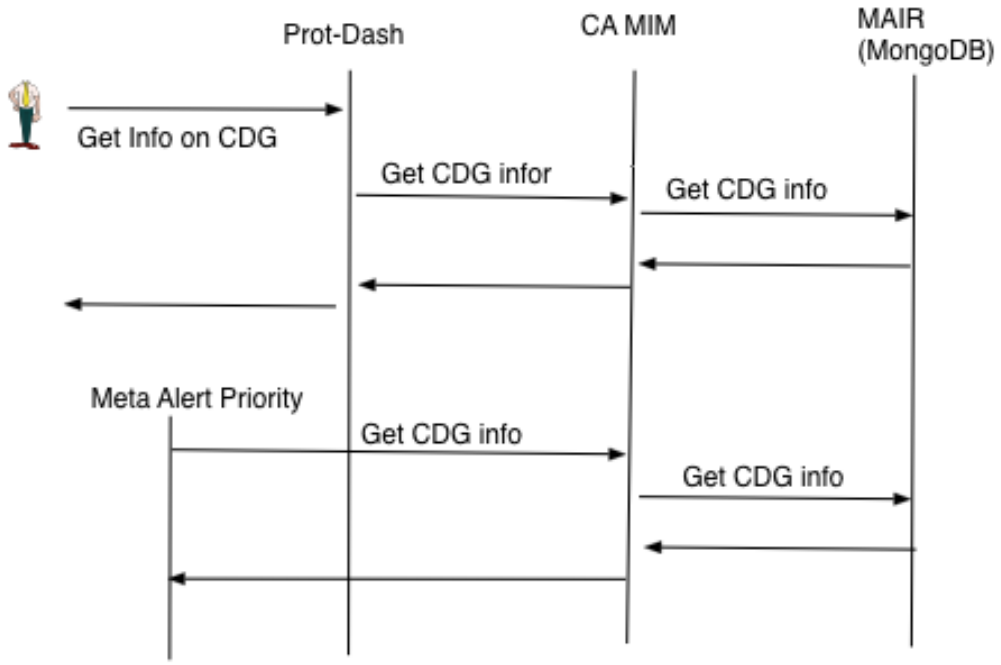

Figure 14: Query and Management of MIM

This use case also includes commands to create and/or delete CDG. Authority to perform these operations is restricted.

3.5.5 Rank Assets

Asset Ranking is currently implemented as a scheduled task. The way to schedule the ranking algorithm is to modify *application.properties* or a *profile properties* file as follows:

3,600,000 milliseconds = 1 hour

run.ranking.algorithm.on.a.schedule=true

delay.between.runs.of.ranking.algorithm=3600000

Because the properties files are configured to be embedded as resources in the application jar (default Spring Boot behaviour) using environment variables is preferred. The settings can be set or overridden by setting the following environment variables e.g. in a docker-compose file:

RUN_RANKING_ALGORITHM_ON_A_SCHEDULE=true

DELAY_BETWEEN_RUNS_OF_RANKING_ALGORITHM=3600000

4 Implementation

The software in this deliverable is focused on enabling the definition and manipulation of CDG's.

4.1 CA Software Architecture

The software architecture of the CA subsystem – defined in D4.2 – is shown in Figure 15.

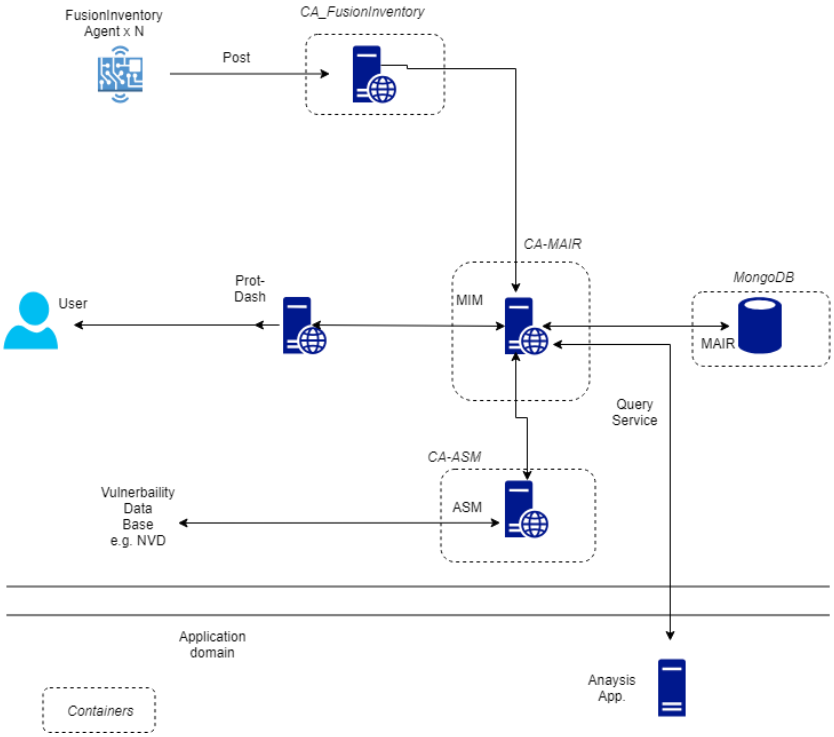

Figure 15: CA Software Architecture

Inventory data is uploaded via the CAFusionInventory (CAFI) container. The interface between FusioInventory agent and CAFI is FusionInventory specific. CAFI can be used as a base for other inventory agent designs. CAFI forwards the data to the CA-MAIR agent (encapsulates the MIM logical entity – the "MAIR" designation is historical). CA-MAIR provides a single REST interface that is used in all use-cases by CAFI, Prot-dash and the PROTECTIVE applications. CA-MAIR also carries out the assets ranking. It also acquires the CVSS scoring information from CA-ASM.

4.2 User Interface

The CA subsystem has a Prot-Dash UI to allow users to

- View CDG's. Users can view any layer as a separate graph or can view all layers as a single, composite, graph.
- Upload CDG files

Screenshots of the UI are shown below. Figure 16 shows the list of the CDG from which a user can select a CDG to view while Figure 17 shows an view of instance of the Application layer which the user has selected.

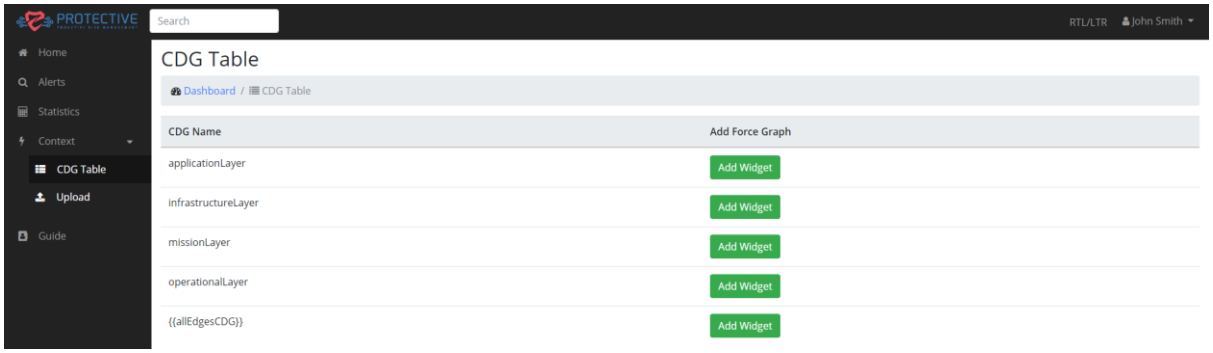

Figure 16: CDG Table

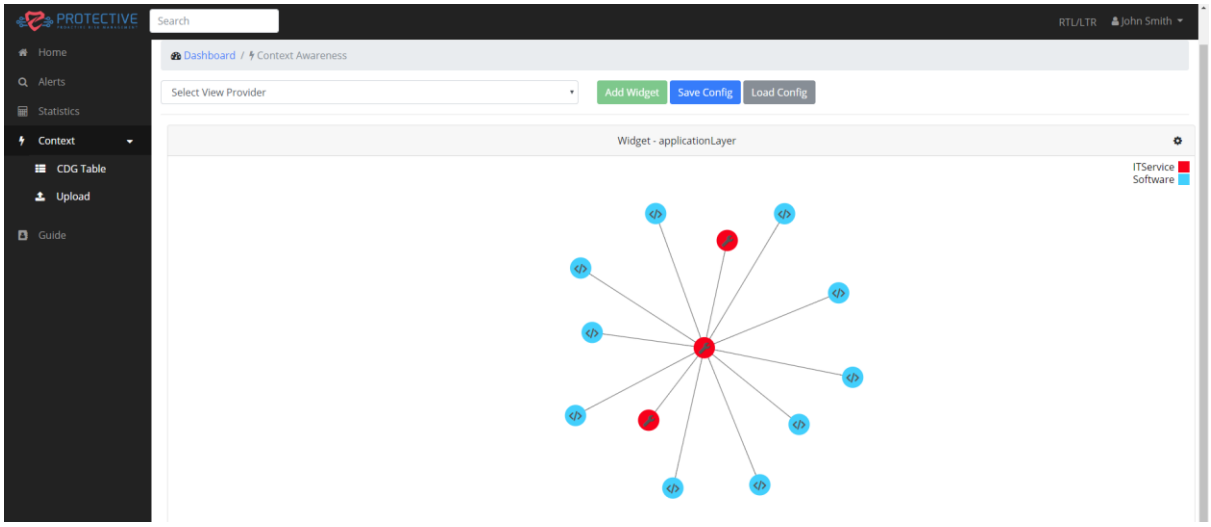

Figure 17: CDG Query - Application Layer

4.3 External Interfaces

The CA subsystem interacts with a number of different actors as described in the use-cases and shown on Figure 15. In some cases, CA offers API's (Application Programming Interfaces) to other services and in other case CA consumes services from other API's

CA-MAIR REST interface

All interactions provided by the MIM module are offered through a single interface to the variety of actors shown in Figure 15. The interface defines the following calls, which are largely self-explanatory:

Figure 18 CA-MAIR REST Interface

4.4 Prerequisites

The CA system is delivered with the following Docker containers:

- CA_MAIR,
- CA_FusionInventory
- MongoDB
- CA_ASM

The context awareness component software is available in GitLab[20] in https://gitlab.com/protective-h2020-eu/enrichment/context-awareness

4.5 Data Model

The CA meta-model has been introduced in Figure 5. Here we describe formats to

- capture dependencies for upload and
- the data base representation.

A CA CDG is implemented as one or more layers of the CA meta-model. The number of layers and type of layer depends on the particular use case being modelled. CDG's are defined according to the (ANTLR4) grammar shown below:

```
grammar mdep;
cdgdep: ('cdg' '=' ID dependency*)+
dependency: mission_dep | bp_dep | ia_dep | its_dep | sw_dep;
mission_dep:'<''m' '='ID',' (bdeps)?(iadeps)?
         (itsdeps)? agg_func?'>
bp_dep:'<' 'bp' '=' ID ','(bdeps)?(iadeps)? (itsdeps)? agg_func?'>';
```

[20] Please note that – as discussed in WP8 - the gitlab will be made fully available at the end of the project, so not all components may be fully available before,

```
ia_dep: '<' 'ia' '=' ID ',' (iadeps)? (itsdeps)? agg_func? '>';
its_dep:'<' 'its' '=' ID ',' (itsdeps)? (sw_deps)?
        agg_func? '>';
sw_dep: '<' 'sw' '=' ID ',' 'node' '=' ID ',' (sw_deps)? agg_func?
'>';
net_dep: '<' 'node' '=' ID ',' (netdeps)? (agg_func)? '>';

bdeps: 'bpdep' '=' dep_list;
iadeps: 'iadep' '=' dep_list;
itsdeps: 'itsdep' '=' dep_list;
sw_deps: 'swdep' '=' dep_list;
netdeps: 'netdep' '=' dep_list;

dep_list: '{' dep_nodes (dep_set)* '}';
dep_nodes: dep_node* ; //(dep_node)*;
dep_set: '[' dep_nodes agg_func ']';
dep_node: '('ID ',' MGRADE ')';
agg_func: 'agfunc' '=' ID;
```

Figure 19: CDG Data Capture Grammar

The full grammar is shown in Annex A.

CDG dependency files structured according to the above grammar are parsed and loaded into the database by a Spring Boot ANTLR4 based Java implementation, part of the CA_MIM module.

CDG's are stored in the Mongo database in graph format using MongoDB Collections data structure, as follows.

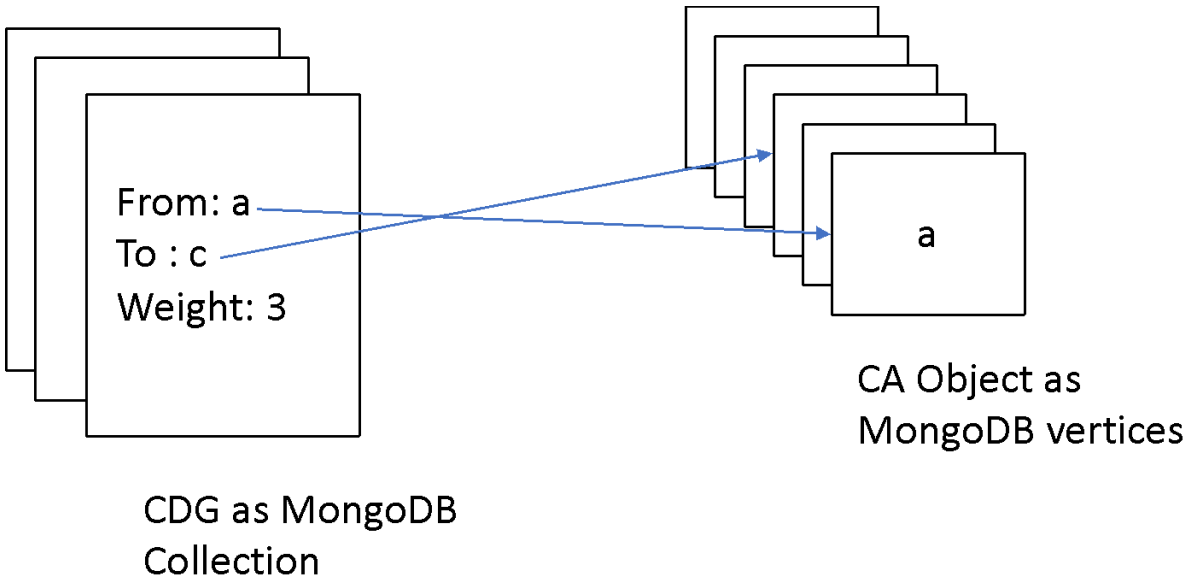

Figure 20: CDG representation in MongoDB

CA Objects (assets etc.) may belong to more than one collection. Furthermore CDG's are distinct. It is not possible to search across CDG's and CDG cannot be merged or joined. Each layer of the meta-model shown in Figure 5 is stored as a CDG and there is also an overall or global CDG to represent the composite model. These correspond to the options in the 'CDG Table' UI screenshot in Figure 16.

5 IPR Management

No background IPR has been used as input in this work. AS part of CA will be made available to PROTECTIVE NRENs with a free licence for use after the PROTECTIVE project and its details will be provided within WP8 deliverables (i.e. D8.3).

6 Ethical Impacts

There are no ethical impacts arising from this function as none of the data is shared with other partners.

7 References

[1] R. Sawilla, “Abstracting PageRank To Dynamic Asset Valuation,” Defence R&D Canada , Ottawa, 2007.

[2] G. Jakobsson, “Mission Cyber Security Situation Assessment Using Impact Dependency Graphs. Chicago: 14th International Conference on Information Fusion,” in *14th International Conference on Information Fusion*, Chicago, 2014.

[3] J. Holsopple and S. Yang, “Mission Impact Assessment,” in *IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA).*, San Diego, 2013.

[4] J. Jiangx, L. Dingx, E. Zhai and Y. Yu, “VRank: A Context-Aware Approach to Vulnerability Scoring and Ranking in SOA,” in *IEEE Sixth International Conference on Software Security and Reliability*, Gaitherburg, 2012.

[5] H. Zhuang and K. Aberer, “A Non-Intrusive and Context-Based Vulnerability Scoring Framework for Cloud Services,” in *Proceedings of the International Symposium on Physical Design*, Santa Rosa, 2016.

[6] J. Watters, S. Morrissey and S. Powers, “The Risk-to-Mission Assessment Process (RiskMAP): A Sensitivity Analysis and an Extension to Treat Confidentiality Issues,” Mitre Corporation, Dartmouth , 2009.

[7] A. Kim, M. Kang, J. Luo and A. Velazazquez, “A Framework for Event Prioritization in Cyber Network Defense,” Naval Research Laboratory, Washington, 2014.

[8] E. e. a. Zhai, “Heading Off Correlated Failures through Independence-as-a-Service,” in *USENIX Symposium on Operating System Design and Implementation*, Broomfield, CO, 2014.

[9] F. Innerhofer–Oberperfler and B. Breu, “Using an Enterprise Architecture for IT Risk Management,” in *ISSA from Insight to Foresight Conference*, Sandton, S.A., 2006.

[10] R. Ko, P. Jagadpramana and B. Sung Lee, “Flogger: A File-Centric Logger for Monitoring File Access and Transfers within Cloud Computing Environments,” in *10th International Conference on Trust, Security and Privacy in Computing and Communications*, Changsa, 2011.

[11] B. Suh and I. Jan, “The IS risk analysis based on a business model,” *Elsevier Information & Management ,* vol. 41, pp. 149-158, 2003.

[12] J. Dai, X. Sun, P. Liu and N. Giacobe, “Gaining Big Picture Awareness through an Interconnected Cross-layer Situation Knowledge Reference Model,” in *International Conference on Cyber Security*, 2012.

[13] J. Medved, “RFC 8345 A YANG Model for Network Topologies,” January 2018. [Online]. Available: https://tools.ietf.org/html/rfc8345. [Accessed May 2018].

[14] A. Zand and e. al., “Know Your Achilles’ Heel: Automatic Detection of Network Critical Services,” in *ACSAC ’15*, Los Angeles, 2015.

[15] S. Wang, R. State, M. Ourdane and Engel, “Mining netflow records for critical network activities,” in *6th International Wireless Communications & Mobile Computing Conference,*, 2010.

[16] S. Wang, R. State, M. Ourdane and Engel, “Mining NetFlow Records for Critical Network Activities,” in *4th International Conference on Autonomous Infrastructure, Management and Security (AIMS)*, Zurich, 2010.

[17] R. Caralli, J. Stevens, L. Young and W. Wilson, “Introducing OCTAVE Allegro : Improving the Information Security Risk Assessment Process,” Software Engineering Institute , Hanscom, 2007.

[18] Cheng, “Metrics of Security,” in *Cyber Defense and Situational Awareness*, A. Kott, C. Wang and R. Erbacher, Eds., Heidelberg, Springer International, 2014, pp. 263-297.

[19] Sanguino, Luis Alberto Benthin and Rafael Uetz. “Software Vulnerability Analysis Using CPE and CVE.” CoRR abs/1705.05347, 2017

[20] Matthias Niedermaier, Thomas Hanka, Sven Plaga, Alexander von Bodisco & Dominik Merli, “Efficient Passive ICS Device Discovery and Identification by MAC Address Correlation”, 5th International Symposium for ICS & SCADA Cyber Security Research 2018 (ICS-CSR 2018), University of Hamburg, Germany, 2018

[21] R. Sawilla and X. Ou, “Googling Attack Graphs,” Defence R&D Canada, Ottawa, 2007.

[22] Warden, "WARDEN," 2014. [Online]. Available: https://warden.cesnet.cz/en/index. [Accessed 25 04 2017].

[23] Mentat, "Mentat," 2017. [Online]. Available: https://mentat.cesnet.cz/en/index. [Accessed 25 04 2017].

[24] IDEA, "IDEA," 2017. [Online]. Available: https://idea.cesnet.cz/en/index. [Accessed 25 4 2017].

Annexes

Annex A: CA CDG Dependency File Grammar

```
grammar mdep;
cdgdep: ('cdg' '=' ID dependency*)+
dependency: mission_dep | bp_dep | ia_dep | its_dep | sw_dep;
mission_dep:'<''m' '='ID',' (bdeps)?(iadeps)?
         (itsdeps)? agg_func?'>
bp_dep:'<' 'bp' '=' ID ','(bdeps)?(iadeps)? (itsdeps)? agg_func?'>';
ia_dep: '<' 'ia' '=' ID ',' (iadeps)? (itsdeps)? agg_func? '>'; its_dep:'<' 'its'
'=' ID ',' (itsdeps)? (sw_deps)?
        agg_func? '>';
sw_dep: '<' 'sw' '=' ID ',' 'node' '=' ID ',' (sw_deps)? agg_func? '>';
net_dep: '<' 'node' '=' ID ',' (netdeps)? (agg_func)? '>';

bdeps: 'bpdep' '=' dep_list;
iadeps: 'iadep' '=' dep_list;
itsdeps: 'itsdep' '=' dep_list;
sw_deps: 'swdep' '=' dep_list;
netdeps: 'netdep' '=' dep_list;

dep_list: '{' dep_nodes (dep_set)* '}';
dep_nodes: dep_node* ; //(dep_node)*;
dep_set: '[' dep_nodes agg_func ']';
dep_node: '('ID ',' MGRADE ')';
agg_func: 'agfunc' '=' ID;


/////////Lexer ////////////////////

MGRADE: 'none' | 'low' | 'medium' | 'high' | 'failure';
RELOP: '==' | '<=' | '>=' | '!=' | '<' | '>' ;
EQ: '=';
OPSELECT:'->' ;

LITERAL: (STRINGLITERAL) | (INTLITERAL) | (REALITERAL) ;

STRINGLITERAL: '"'ID'"';
INTLITERAL: DIGIT+;
REALITERAL: DIGIT+'.'DIGIT+;

ID: IDENT | COMPOSED_ID ;
IDENT: (LETTER) (LETTER|DIGIT|'_')* ;
fragment COMPOSED_ID: IDENT ('.' IDENT)* ;
STRINGS: (LETTER|DIGIT|OTHERCHAR)+ ;
OTHERCHAR: '%' |'*'|'$'|'&'|'@'|'?'|'"'|'.' ;
//ID : [a-zA-Z] [a-zA-Z0-9]* ;

LETTER:[a-zA-Z];
DIGIT:[0-9];

FCL: (LETTER | DIGIT | ';' | ':=' | ':' |','|'('|')'| '//');

WS:[ \t\r\n]+ -> skip; // skip spaces, tabs, newlines


SL_COMMENT: '//' .*? '\n' -> skip ;
```

Annex B: Asset State REST API

POST /api/get-software-max-cvss

Find the maximum value of CVSS by searching all vulnerabilities related to the software given as an input.

Authorization

Resource is public

Path parameters

No parameters

Query parameters

No parameters

Request fields

Path	Type	Optional	Description
key	String	true	An identifier of software instance
product	String	false	Name of the product
vendor	String	false	Name of the publisher
version	String	false	Version identifier

Response fields

Type	Description
String	Maximum value of CVSS

Example request

curl -i -H "Content-Type: application/json" -X POST -d "

{

"""key""":"""Internet Explorer""",

"""product""": """Internet Explorer""",

"""vendor""":"""Microsoft Corporation""",

"""version""":"""11.0.9600.18792"""

}" http://127.0.0.1:5000/api/get-software-max-cvss

Example response

HTTP/1.0 201 CREATED

Content-Type: text/html; charset=utf-8

Content-Length: 2

Server: Werkzeug/0.10.4 Python/3.5.2

Date: Wed, 17 Oct 2018 11:49:05 GMT

POST /api/get-asset-max-cvss

Find the maximum value of CVSS by searching all vulnerabilities related to the software list given as an input.

Authorization

Resource is public

Path parameters

No parameters

Query parameters

No parameters

Request fields

Path	Type	Optional	Description
	Array	false	List of software
[].key	String	true	An identifier of software instance
[].product	String	false	Name of the product
[].vendor	String	false	Name of the publisher
[].version	String	false	Version identifier

Response fields

Type	Description
String	Maximum value of CVSS

Example request

```
curl -i -H "Content-Type: application/json" -X POST -d "
[
{"""key""":"""Microsoft .NET Framework 4.5 Multi-Targeting Pack""",
"""product""": """Microsoft .NET Framework 4.5 Multi-Targeting Pack""",
"""vendor""":"""Microsoft Corporation""",
"""version""":"""4.5.50710"""
},
{"""key""":"""Internet Explorer""",
"""product""": """Internet Explorer""",
"""vendor""":"""Microsoft Corporation""",
"""version""":"""11.0.9600.18792"""
},
{
"""key""":"""VLC""",
"""product""": """VLC media player""",
 """vendor""":"""VideoLAN""",
"""version""":"""2.2.4"""
},
{
"""key""":"""Adobe AIR""",
"""product""": """Adobe AIR""",
"""vendor""":"""Adobe Systems Incorporated""",
"""version""":"""20.0.0.260"""
},
{
"""key""":"""Mozilla Thunderbird""",
"""product""": """Mozilla Thunderbird 38.6.0 (x86 pl)""",
"""vendor""":"""Mozilla""",
"""version""":"""38.6.0"""
}
]" http://127.0.0.1:5000/api/get-asset-max-cvss
```

Example response

HTTP/1.0 201 CREATED

Content-Type: text/html; charset=utf-8

Content-Length: 2

Server: Werkzeug/0.10.4 Python/3.5.2

Date: Wed, 17 Oct 2018 12:00:14 GMT

POST /api/load-cpe file=@<file>

Upload CPE dictionary.

Authorization

Resource is public

Path parameters

No parameters

Query parameters

Parameter	Type	Optional	Description
file	String	false	The name of CPE dictionary file

Request fields

No request body

Response fields

No response body

Example request

curl -i -X POST -F file=@official-cpe-dictionary_v2.3.xml "http://127.0.0.1:5000/api/load-cpe

Example response

HTTP/1.1 100 Continue

HTTP/1.0 200 OK

Content-Type: text/html; charset=utf-8

Content-Length: 25

Server: Werkzeug/0.10.4 Python/3.5.2

Date: Fri, 19 Oct 2018 09:35:13 GMT

File uploaded successfully

POST /api/load-cve file=@<file>

Upload CVE file.

Authorization

Resource is public

Path parameters

No parameters

Query parameters

Parameter	Type	Optional	Description
file	String	false	The name of the CVE file

Request fields

No request body

Response fields

No response body

Example request

curl -i -X POST -F file=@nvdcve-2.0-2017.xml "http://127.0.0.1:5000/api/load-cve

Example response

HTTP/1.1 100 Continue

HTTP/1.0 200 OK

Content-Type: text/html; charset=utf-8

Content-Length: 25

Server: Werkzeug/0.10.4 Python/3.5.2

Date: Fri, 19 Oct 2018 09:35:13 GMT

File uploaded successfully

GET /api/init-conf-db?option=<option>

Initialize CPE or CVE database using NVD data, clear CVE database or cached result.

Authorization

Resource is public

Path parameters

No parameters

Query parameters

Parameter	Type	Optional	Description
option	String	false	Following values can be entered: “cve” - this runs CVE database initialization process “cpe” - this runs CPE database initialization process “emptycve” - this removes all data from CVE database and creates an empty table. “emptycache” - this removes all data from cache database and creates an empty table.

Request fields

No request body

Response fields

No response body

Example request

curl 'http://127.0.0.1:5000/api/init-conf-db?option=cve

Example response

HTTP/1.0 201 CREATED

Content-Type: text/html; charset=utf-8

Content-Length: 2

Server: Werkzeug/0.10.4 Python/3.5.2

Date: Wed, 17 Oct 2018 12:00:14 GMT

Successfully initialized CVE database

GET /api/data-incr-cve?file=<file>

Add data from given CVE file into the CVE database

Authorization

Resource is public

Path parameters

No parameters

Query parameters

Parameter	Type	Optional	Description
file	String	false	Name of the file from which the data should be inserted into CVE database

Request fields

No request body

Response fields

No response body

Example request

curl 'http://127.0.0.1:5000/api/data-incr-cve?file=nvdcve-2.0-2017.xml

Example response

HTTP/1.0 201 CREATED

Content-Type: text/html; charset=utf-8

Content-Length: 2

Server: Werkzeug/0.10.4 Python/3.5.2

Date: Wed, 17 Oct 2018 12:00:14 GMT

Successfully inserted data

POST /api/add-cpe-definition

Upload labelled data / pair given CPE identifier with given name of product, vendor and version of software.

Authorization

Resource is public

Path parameters

No parameters

Query parameters

No parameters

Request fields

Path	Type	Optional	Description
product	String	false	Name of the product
vendor	String	false	Name of the publisher
version	String	false	Version identifier
cpe	String	false	CPE identifier
	Array	True	NVD bindings and labels
[].cve	String	True	CVE identifier
[].label	String	True	Additional label

Response fields

No response body

Example request

```
curl -i -H "Content-Type: application/json" -X POST -d "
{
```

"""product""": """Internet Explorer""",

"""vendor""":"""Microsoft Corporation""",

"""version""":"""11.0.9600.18792"""

"""cpe""":"""cpe:/a:microsoft:internet_explorer:11:-"""

}" http://127.0.0.1:5000/api/add-cpe-definition

Example response

HTTP/1.0 201 CREATED

Content-Type: text/html; charset=utf-8

Content-Length: 2

Server: Werkzeug/0.10.4 Python/3.5.2

Date: Wed, 17 Oct 2018 11:49:05 GMT

Definition has been added.

Annex C: Asset State – system configuration used for testing WFN weights

Table: List of computers used in test

ID	Operating system
1	Windows 7
2	Windows 10
3	Windows 8.1 (ks48)
4	Windows 8.1 (ks54)

Table: Number of software items including / excluding Windows Updates

Computer ID	Number of software without removing Windows updates	Number of software after removing Windows updates
1	390	136
2	73	70
3	406	215
4	312	86

Table: List of vulnerable software tested, information available in Inventory System

	product	vendor	version
1	Internet Explorer	Microsoft Corporation	11.0.9600.18792
2	Microsoft .NET Framework 4.5 Multi-Targeting Pack	Microsoft Corporation	4.5.50710
3	Microsoft Office 2013 dla Użytkowników Domowych i Małych Firm - pl-pl	Microsoft Corporation	15.0.4953.1001

4	7-Zip 9.20 (x64 edition)	Igor Pavlov	9.20.00.0
5	Mozilla Thunderbird 38.6.0 (x86 pl)	Mozilla	38.6.0
6	Epson Print CD	SEIKO EPSON CORPORATION	2.00.00
7	VLC media player	VideoLAN	2.2.4
8	Camtasia Studio 7	TechSmith Corporation	7.1.1
9	Python 3.4.3	Python Software Foundation	3.4.16490
10	Adobe AIR	Adobe Systems Incorporated	20.0.0.260