# SOCCRATES

SOC & CSIRT Response to Attacks & Threats based on attack defence graphs Evaluation Systems

# D3.2

# Final ADG based Attack prototypes

| | |
|---|---|
| Deliverable type: | DEM |
| Contributing work packages: | WP3 ADG based Attack Analysis |
| Due date of deliverable: | 28/02/2022 |
| Submission date: | 28/02/2022 |
| Dissemination level: | PU |

| | |
|---|---|
| Responsible organisation: | F-Secure |
| Editors: | Marko Komssi & Frank Fransen |
| Revision: | 1.0 |

| | |
|---|---|
| Abstract | The overall objective of Work Package 3 is to develop the Infrastructure Modelling and Attack Defence Graph analyser & Course of Action Generator components of the SOCCRATES Platform. This deliverable describes the final versions of the Infrastructure Modelling Component, Attack Defence Graph analyser and Course of Action generator components. |
| Keywords: | Infrastructure modelling, Attack Defense Graphs, Attack Simulations, Course of Action |

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 833481
Call H2020-SU-ICT-2018 • Innovation Action • Start date: September 1st, 2019

**Classification level: Public**

# Management summary

The main objective of SOCCRATES is to develop and implement a security automation and decision support platform for SOCs and CSIRTs that will significantly improve an organisation's capability to quickly and effectively detect and respond to new cyber threats and ongoing attacks. The SOCCRATES platform consists of eleven components. This deliverable describes the final prototype implementations of three of those components – the Infrastructure Modelling Component (IMC), Attack Defence Graph (ADG) and Course of Action Generator (COA).

The reader is assumed to be familiar with the following SOCCRATES deliverables:

- D2.2, which contains a description of all SOCCRATES components and where IMC and ADG are introduced (in Sections 4 and 5 respectively).
- D3.1, which is the preliminary version of this deliverable. It describes the capabilities of IMC and ADG and documents the initial integration of prototypes of those two components.
- D3.3, which is the specification of the common ICT infrastructure reference meta model.

This deliverable presents the final Attack Defence Graph prototype components. Separate deliverables will detail future development and use of these components. D6.4 will present the components in a final version of the SOCCRATES platform. D7.2 will provide details of a pilot evaluation. Finally, D8.6 will introduce a commercial exploitation plan for the components.

| | |
|---|---|
| Contributor(s) | Marko Komssi (F-Secure)<br>Ville Alkkiomäki (F-Secure)<br>Seamus Moloney (F-Secure)<br>Erik Ringdahl (Foreseeti)<br>Per Eliasson (Foreseeti)<br>Martin Eian (Mnenomic)<br>Mathias Ekstedt (KTH)<br>Ashish Kumar Dwivedi (KTH)<br>Giuseppe Nebbione (KTH)<br>Frank Fransen (TNO)<br>Sebastiaan Tesink (TNO) |

| | |
|---|---|
| Reviewer(s) | Christophe Kiennert (Institut Mines-Télécom)<br>Andrew Patel (F-Secure) |

| **Security Assessment** | |
|---|---|
| Approval Date | 19/02/2022 |
| Remarks | none |

# TABLE OF CONTENTS


1 INTRODUCTION – RATIONALE OF THIS DOCUMENT ..... 6

1.1 The SOCCRATES project ..... 6
1.2 Final prototype versions of the three components ..... 7
1.3 Structure of this deliverable with the main changes to D3.1 ..... 7

2 INFRASTRUCTURE MODELLING COMPONENT ..... 9

2.1 Introduction ..... 9
2.2 System architecture ..... 9
2.3 Demonstration infrastructure ..... 12
*2.3.1 F-Secure Elements Security Center ..... 12*
*2.3.2 Elements Vulnerability Management scan node ..... 13*
*2.3.3 Elements Vulnerability Management - IMC Adapter ..... 13*
*2.3.4 IMC Database and REST APIs ..... 13*
2.4 Demonstration of IMC capabilities ..... 13
*2.4.1 Automated infrastructure scanning ..... 14*
*2.4.2 Continuous updates ..... 14*
*2.4.3 Time dimension ..... 15*
*2.4.4 Extendable model ..... 15*
*2.4.5 Multi-organization ..... 16*

3 ATTACK DEFENCE GRAPHS ..... 17

3.1 Introduction ..... 17
3.2 Current system architecture ..... 17
*3.2.1 Model generation ..... 17*
*3.2.2 Attack simulation ..... 18*
*3.2.3 Analysis ..... 18*
*3.2.4 Threat information platform (TIP) ..... 18*
*3.2.5 Course of action generator (CoA) ..... 18*
3.3 Domains and integration ..... 18
*3.3.1 Support for the Meta Attack Language (MAL) framework ..... 18*
*3.3.2 Support for on-premise environments ..... 19*
*3.3.3 Support for Amazon Web Services (AWS) ..... 19*
*3.3.4 Support for Microsoft Azure ..... 19*
*3.3.5 Integration ..... 19*
3.4 Deployment ..... 20
*3.4.1 Requirements and installation ..... 20*
*3.4.2 Target domain ..... 21*
3.5 Current capabilities ..... 21

4 COUSE OF ACTION (COA) GENERATOR ..... 23

4.1 Introduction ..... 23
4.2 System Architecture ..... 23
*4.2.1 Deployment of CoA Generator ..... 23*
*4.2.2 Requirements and installation ..... 24*
*4.2.3 Language Support ..... 24*

4.3 Model Structure ..... 24
*4.3.1 Attack Graphs ..... 24*
*4.3.2 Attack Simulation ..... 24*
4.4 Algorithm ..... 25
*4.4.1 Criticality score ..... 25*
*4.4.2 Cost and Budget for the Selected Countermeasures ..... 27*
4.5 Example ..... 27
*4.5.1 Application of CoA Generator for an IT infrastructure model ..... 27*
*4.5.2 Result of CoA Generator ..... 28*
5 FINAL PROTOTYPE VERSIONS AND TEST ENVIRONMENT ..... 33
6 ABBREVIATIONS ..... 34

# 1 Introduction – Rationale of this document

This section introduces the SOCCRATES project and defines the goals of this deliverable.

## 1.1 The SOCCRATES project

SOCCRATES (SOC & CSIRT Response to Attacks & Threats based on attack defence graphs Evaluation Systems) is an EU funded project under the Horizon 2020 programme that addresses the following challenge:

***How can SOC and CSIRT operations effectively improve their capability in detecting and managing response to complex cyber-attacks and emerging threats, in complex and continuously evolving ICT infrastructures while there is a shortage of qualified cybersecurity talent?***

The main objective of SOCCRATES is to develop and implement a security automation and decision support platform (‘**the SOCCRATES platform’)** that will significantly improve an organisation’s capability (usually implemented by a SOC and/or CSIRT) to quickly and effectively detect and respond to new cyber threats and ongoing attacks.

The SOCCRATES platform (see Figure 1) consists of an orchestrating function and a set of innovative components for automated infrastructure modelling, attack detection, cyber threat intelligence utilization, threat trend prediction, and automated analysis using attack defence graphs and business impact modelling to aid human analysis and decision making on response actions and enable the execution of defensive actions at machine-speed.

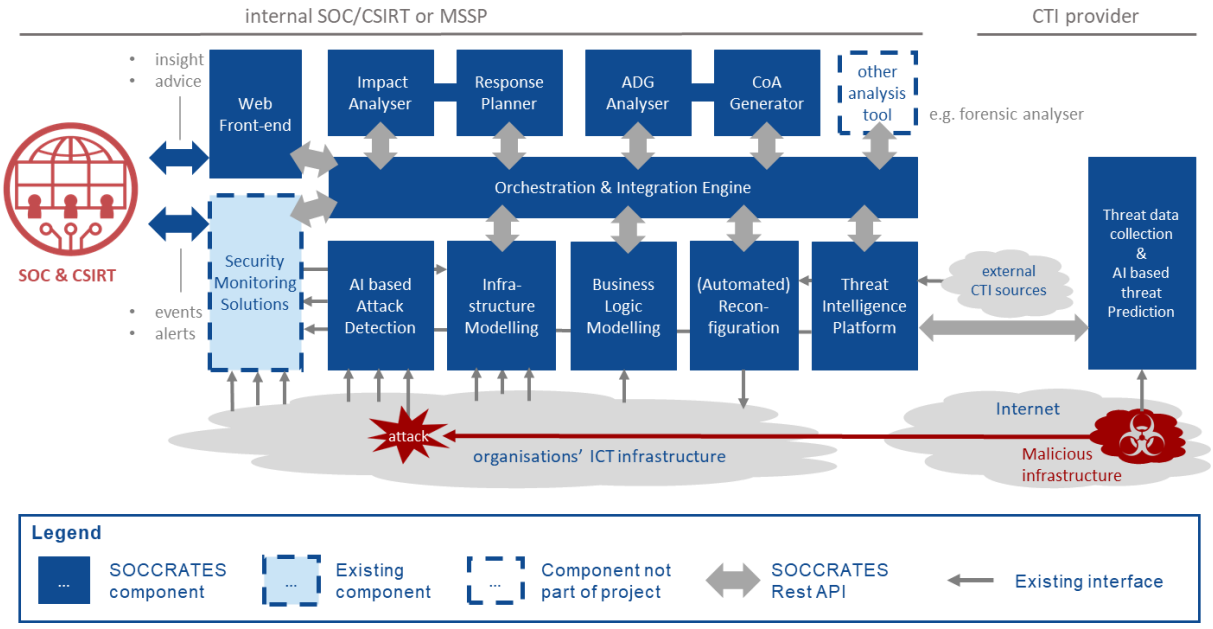


**Figure 1 The SOCCRATES platform**

SOCCRATES has the following concrete project objectives:

1. Deliver the SOCCRATES platform consisting of an orchestration function and a unique integration of innovative background solutions that seamlessly work together.
2. Demonstrate that the SOCCRATES platform can improve SOC operations by evaluating it in two diverse real-life pilot environments.
3. Examine and illustrate the benefits of automation for selected SOC activities to help manage the cyber security skills gap in organizations.
4. Prepare for successful exploitation by the SOCCRATES partners of the individual innovated components and the integrated SOCCRATES platform in commercial products that are offered to the market and are available for the European (business) community.

Please visit www.soccrates.eu for more information on the SOCCRATES project.

## 1.2 Final prototype versions of the three components

The SOCCRATES platform consists of eleven components that are introduced in D2.2. This deliverable documents the final prototype versions of Attack Defence Graph components: Infrastructure Modelling Component (IMC), Attack Defence Graph analyser (ADG) and Course of Action generator (COA).

IMC provides a model of an organisation's current ICT infrastructure in a common reference description language. The model allows SOC analysts to better understand system structures, configurations, and vulnerabilities and provides insight into how attacks propagate through their ICT infrastructure. An important goal of IMC is to provide an accurate and up-to-date model of the ICT infrastructure to other SOCCRATES components for further analysis. In WP3, however, it was observed that there is no commonly acknowledged ICT infrastructure modelling standard currently available that fulfils the needs of the SOCCRATES project. One such need is support for threat intelligence data. To address this problem, an appropriate reference (meta) model was developed based on the needs of the SOCCRATES project. The new reference model adopts relevant portions of existing initiatives and standards in the cyber security industry.

A role of ADG is to predict how attacks propagate over a given ICT infrastructure. ADG can be used to predict, for instance, the potential effect of a new threat, a new vulnerability, or a change in system configuration. ADG-based analysis can also be used to determine if and how an attacker may compromise other systems when an attack is detected. ADG utilizes data provided by the infrastructure modelling component (IMC) and performs an analysis based on probabilistic attack simulations that are further enriched with cyber threat information from a threat intelligence platform (TIP). As a result, ADG provides quantitative metrics on risk exposure and time to compromise (TTC), the most likely attack paths to high-value assets, and suggested mitigations.

The Course of Action (CoA) Generator operates on a model of an ICT infrastructure as described by the Attack Defense Graph (ADG) component and generates a set of defences that are currently inactive in the model and are suggested to be activated. The result from the CoA Generator can either serve as input to a SOC analyst or as a basis for automated response. The generated CoAs are sent to the Response Planner for ranking according to financial impact in the SOCCRATES platform.

## 1.3 Structure of this deliverable with the main changes to D3.1

Whilst D3.1 introduced preliminary prototype versions of two of the above components, deliverable D3.2 presents the final prototype versions of all three components. The next section briefly introduces the structure of this deliverable with the key changes between this deliverable and the previous.

Section 2 presents the final prototype of IMC, which includes some technical changes compared to the preliminary version (described in D3.1). The final prototype includes two new adapters – the IMC Active Directory adapter and the IMC Netflow adapter, a new IMC Notification rule engine and endpoint agent, and an updated version of the IMC graph database schema. The agent is named Elements VM endpoint agent and the naming of the background product, RADAR, was changed to F-Secure Elements VM.

Section 3 presents the final prototype of ADG, and updates to the integration between ADG and Threat information platform (TIP). The final version of ADG includes domain support and auto-modelling in Section 3.2 and Section 3.3. Section 3.5 also details four new ADG capabilities.

Section 4 presents the final prototype version of the COA Generator which was not introduced in D3.1. D3.1 described how IMC and ADG work jointly in three use case demos. Note that while the D3.1 demos are not covered in this deliverable, a full demonstration of all three components is presented in D7.2.

Section 5 introduces a new test environment.

# 2 Infrastructure modelling component

## 2.1 Introduction

The Infrastructure modelling component (IMC) is designed to scan infrastructure and generate a model of the pilot infrastructure (defined as KPI1 in deliverable D7.1). IMC also acts as a vulnerability scanning solution for SOCCRATES, providing information about vulnerabilities for other SOCCRATES components to use.

## 2.2 System architecture

Figure 2 illustrates the system architecture of IMC and its integration with the Orchestration and Integration Engine. IMC creates a model that describes the current ICT infrastructure using a common reference description language (IMC graph database schema). The model is accessed via a REST API. To build this model, the component collects data from a variety of available IT and security management tools. Moreover, IMC will trigger the Orchestration and Integration Engine when relevant changes in the infrastructure model are detected.

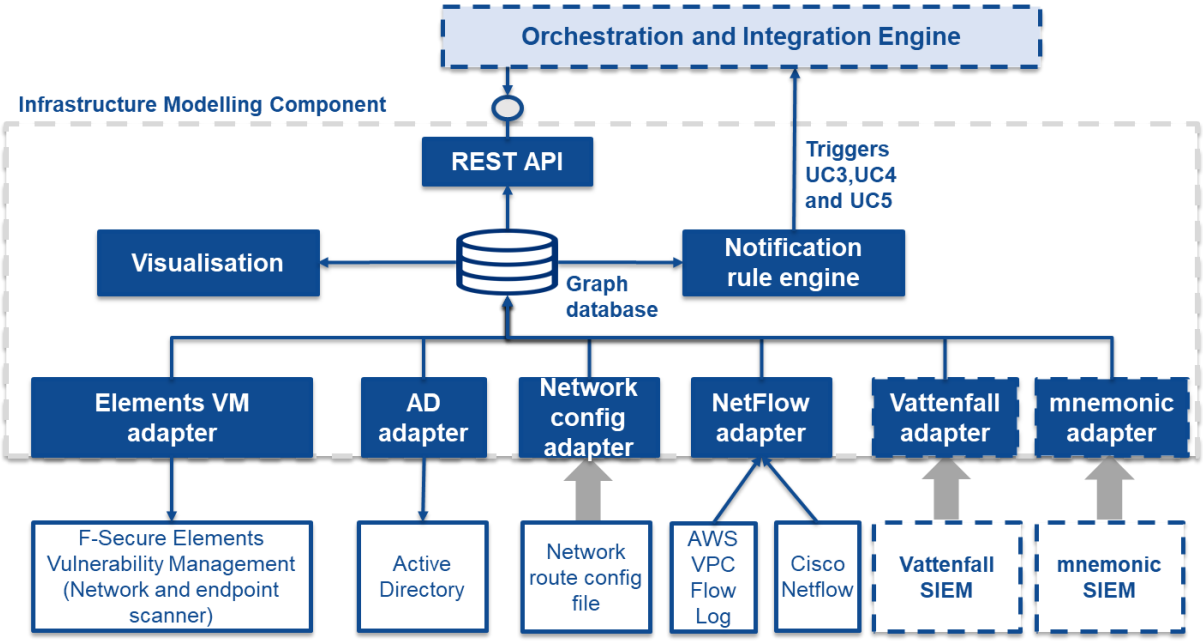


**Figure 2. The current system architecture of IMC**

Adapters and data sources used in the final prototype are:

- **F-Secure Elements VM (previously known as RADAR) network adapter**
  - Provides a comprehensive view to all hosts found from the configured target networks.
  - Collects software versions visible with the network scan together with their known vulnerabilities, i.e. only running software with visible open ports.
- **F-Secure Elements VM (previously known as RADAR) agent**
  - F-Secure Elements endpoint agent now has integrated vulnerability scanning functionality, which allows organizations to execute vulnerability scans on endpoints without needing to initiate SSH, WinRM or RPC connections to the scan node.
  - Agent-based approach makes it possible to scan devices outside company networks.

    - Agent-based scanning provides detailed host information, i.e. host type and hardware information, installed software and their known vulnerabilities.
    - Agent based scanning can identify software that does not have open ports.
- **Network configuration file importer**
    - Target network configuration file is imported with a separate adapter, providing network routes, firewall rulesets and rules.
- **NetFlow adapter**
    - Netflow adapter provides observed network connection information between hosts based on AWS VPC Flow logs or Cisco Netflow data sources.
    - This information complements the static network route and firewall rule information and enables Business Impact Analyzer to evaluate the importance of each host.
- **Active Directory adapter**
    - Active directory adapter fetches user and user access related information from the target organization's Microsoft Active Directory.
- **Custom Vattenfall adapter**
    - A custom adapter was developed for the Vattenfall pilot as the company policy did not allow separate vulnerability scanners to be installed on their computers. Vulnerability scan data was made available through their SIEM system.
- **Custom Mnemonic adapter**
    - A custom adapter was developed for the Mnemonic pilot as company policies did not allow separate vulnerability scanners to be installed on their network. Vulnerability scan data was made available through their SIEM system.

IMC uses a graph database to store collected data in a normalized data format. Figure 3 illustrates the objects used in the final prototype together with their relationships. All data is tagged with a unique organisation identifier to support multi-tenant deployment. The prototype uses data from a single-organization setup, i.e., all data is stored with the same organization identifier.

When a new fact is added to the IMC database it passes through an IMC Notification rule engine. This rule engine is designed to match specific fact types and/or values that indicate whether a use case should be triggered. When a match is found, the engine performs a request to the OIE REST API (Orchestration and Integration Engine), either directly (Use Case 3) or through a worker that first performs additional queries to the IMC to verify that the Use Case should be triggered (Use Cases 4 and 5). This mechanism is used to trigger Use Cases 3, 4 and 5 in the SOCCRATES platform.

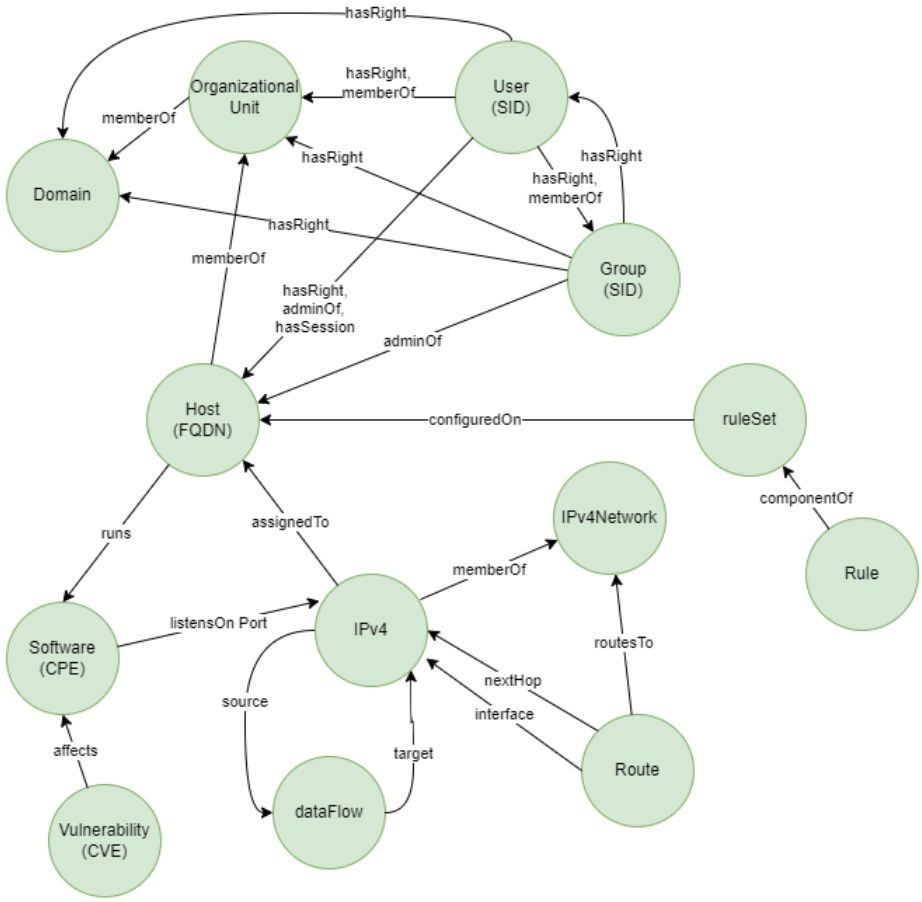


**Figure 3. IMC graph database schema**

Data objects include:

- **Host** - a device connected to the target organization's (IP) network, identified with FQDN.
- **Software** - a software product installed to a host, identified with CPE when available.
- **Vulnerability** - a known vulnerability, identified with a CVE ID.
- **IPv4 Network** - a network segment within the target organisation's ICT infrastructure that has one or more hosts connected to it.
- **Route** - a network route between IPv4 Networks.
- **Firewall ruleset** – group of rules that limit network traffic passed through a route.
- **Rule** - rule specifies the network traffic which is allowed or denied.
- **Data flow -** provides statistics about network traffic observed between source and target IPv4 addresses.
- **Domain -** Active Directory domain which uniquely identifies the Security Identifiers (SID) generated by the domain controllers in that network.
- **User** - a user account present in Active Directory.
- **Organization Unit** - a special object in Active Directory typically used to group users and computers together per site or function.
- **Group** - used to manage access control to computers, shared disks and business applications through Active Directory.

Detailed data model specification is available in a separate deliverable D3.3 Specification of the common ICT infrastructure reference meta model.

## 2.3 Demonstration infrastructure

Figure 4 is a depiction of networks and PCs monitored by IMC. Windows PCs are joined to Active Directory together with a set of test users, organization units, and user groups. IMC components and their configuration are described further in the chapters below.

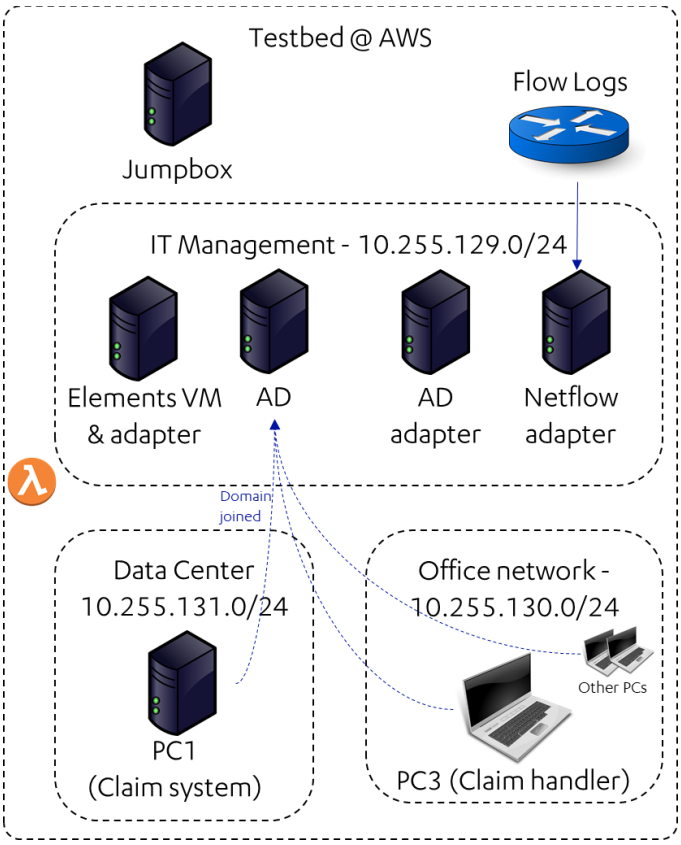


**Figure 4. Testbed environment**

The demonstration infrastructure described in this section represents the Archisurance company network. Archisurance is a fictitious company designed to illustrate the use of the ArchiMate® modelling language and is available through https://archimate-models.github.io/archisurance/.

The demonstration environment consists of three subnets and a separate jumpbox. An IT Management network is used to deploy servers and software used by IT and security operations, e.g. IMC adapters, Active Directory and Elements Vulnerability Management scan nodes. The office network contains virtual machines representing company office laptop. A Data Center network contains servers running business applications, e.g. claim system. An additional jumpbox is used to enable SSH and RDP access from whitelisted remote networks.

Furthermore, a custom AWS lambda function is used as a proxy between the Orchestration and Integration engines and this environment to demonstrate automated reconfiguration.

### 2.3.1 F-Secure Elements Security Center

RADAR Security Center was integrated into the F-Secure Elements Security Center and run in a multi-tenant F-Secure Cloud environment. The SOCCRATES demonstration environment is licensed with Elements VM and EDR licenses and is configured to be used as a data source both for network and endpoint scans.

Elements Vulnerability management configuration:

- Discovery scans are configured for all target networks
- Scan groups have been created for each subnet with the following options:
    - System scan enabled.
    - Default (TCP 0-65535, UDP 0-1024) template is used for network scan.
    - Asset update sources are set to use Discovery Scan as a source and triggering immediate scan for new hosts and new network services.
    - Scan schedule is set to Daily.
    - Web scans are disabled.
- Elements VM agent is installed on all Windows based PCs
    - Scheduled vulnerability scanning is enabled providing endpoint vulnerability scans.
- API Keys created for Elements Vulnerability Management-IMC Adapter

Elements EDR configuration is as follows:

- Elements EDR functionality is enabled on all Windows-based PCs.
- Elements Security Center is used to demonstrate F-Secure Elements response capabilities.

### 2.3.2 Elements Vulnerability Management scan node

Normal Windows Elements VM scan nodes were installed on dedicated servers together with Elements VM – IMC adapters. Scan nodes were configured to be administered through Elements Security Center following the installation guide available on F-Secure's web pages: https://help.f-secure.com/product.html#business/radar/3.0/en/concept_AFCE4EB5D8434AEA9D6C79496D3609D4-3.0-en

### 2.3.3 Elements Vulnerability Management - IMC Adapter

Elements VM-IMC adapter was installed on the same server as the scan node, together with a Python environment and required pyyaml & act-api libraries. The Elements VM-IMC adapter was configured to use the Elements Security Center and ACT cloud platform URLs with API credentials stored in a configuration file. The Elements VM-IMC adapter source code will be published as open source and will be made publicly available during the SOCCRATES project.

### 2.3.4 IMC Database and REST APIs

The IMC graph database and IMC Environment API were deployed to Mnemonic's multi-tenant cloud environment and API credentials were created for IMC adapters and ADG. IMC data model schema was used as a configuration for the database. The IMC graph database schema will be published as open source and will be made publicly available during the SOCCRATES project.

## 2.4 Demonstration of IMC capabilities

This section demonstrates the main capabilities of IMC, which include:

- Automated infrastructure scanning.
- Continuous updates.
- Time dimension.
- Extendable model.
- Multi-organization.

The following chapters describe how these capabilities can be leveraged.

### 2.4.1 Automated infrastructure scanning

IMC is designed to automatically scan a target infrastructure. This can be demonstrated by cleaning up the IMC database and then executing full scans manually. This illustrates how all assets are detected and imported into the graph database correctly. Figure 5 provides a screenshot from Elements Security Center listing network hosts found in the all subnets or the test infrastructure.

All virtual machines and network devices visible in the AWS environment were successfully discovered and imported into the graph database. Additional information, i.e. software with no open ports, was provided for Windows devices with Elements endpoint agent, enabling a more comprehensive attack path simulation. Elements VM was configured to scan on a daily basis, which may be too infrequent for real-world environments, but was more than enough for testing purposes.

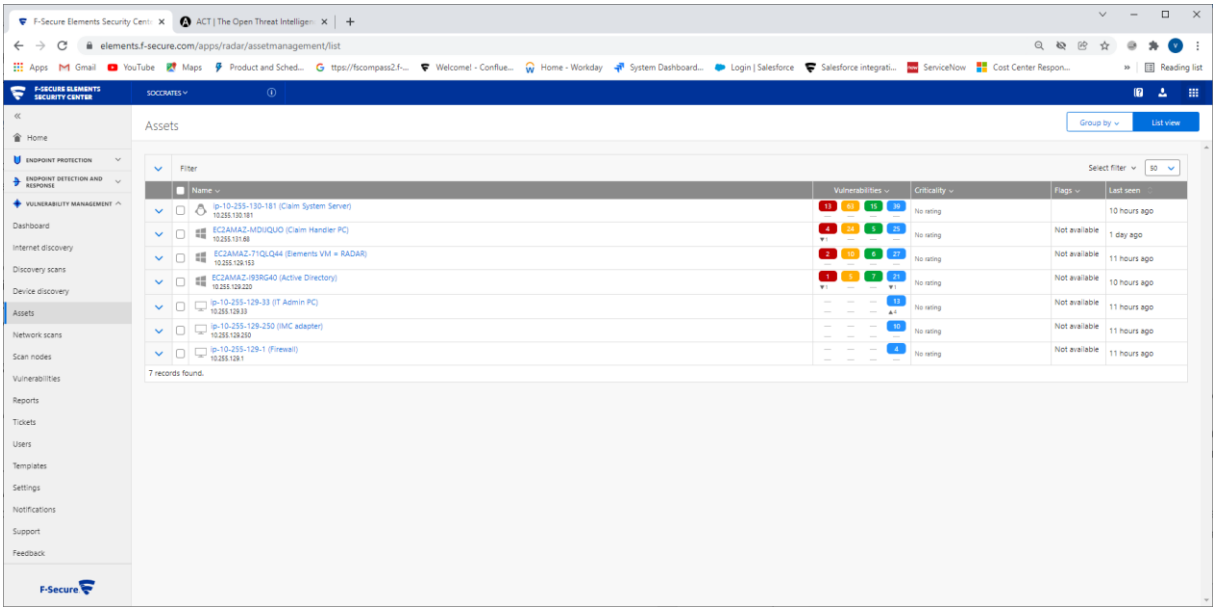


**Figure 5 : Elements Security Center and identified hosts**

### 2.4.2 Continuous updates

IMC can normalize and update its model simultaneously from multiple data sources. This is demonstrated by scanning some endpoints from the network and others from both the network and via endpoint scans. Furthermore, this host information was merged with the data sourced from Active Directory and Netflow adapters, resulting in a single customer infrastructure model in the graph database. Figure 6 shows how PC3 is linked in IMC to the “Example OU” organization unit in AD and to the IPv4 address 10.255.131.68.

Additionally, the network routes and firewall rules were imported from a configuration file enriching the ipv4network information gathered from an Element VM network scan and actual observed network traffic from Netflow adapter.

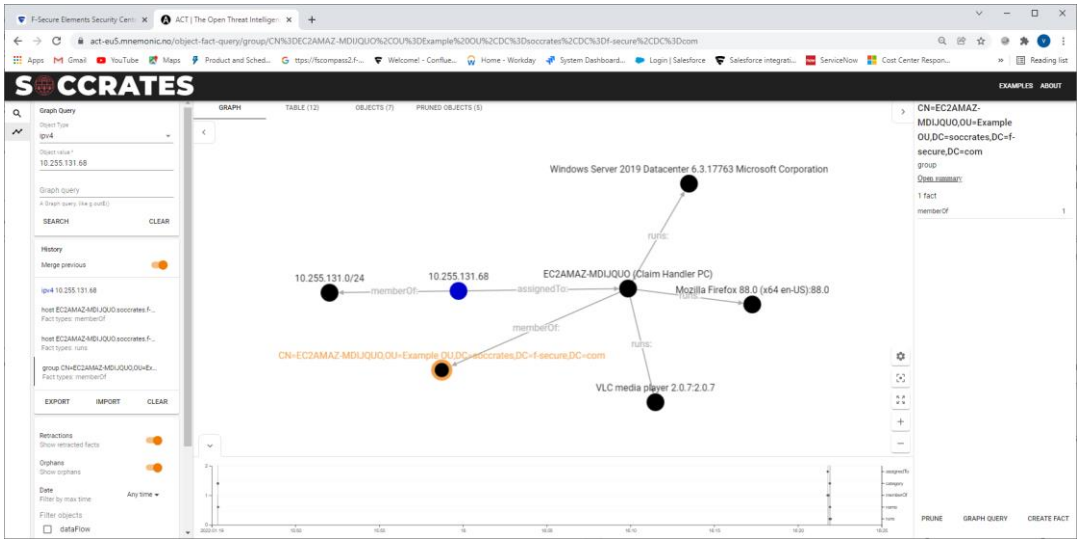


**Figure 6 : Vulnerability scanning results and Active Directory data linked in IMC graph database**

### 2.4.3 Time dimension

Time dimension refers to the ability to query the past IMC models via timestamps. This was tested by making changes to Active Directory Organization Units and rerunning the AD adapter to update the graph database with new timestamps. The time dimension is also nicely visible in the observed data-flow graph (Figure 7), illustrating when each of the facts were added to the graph.

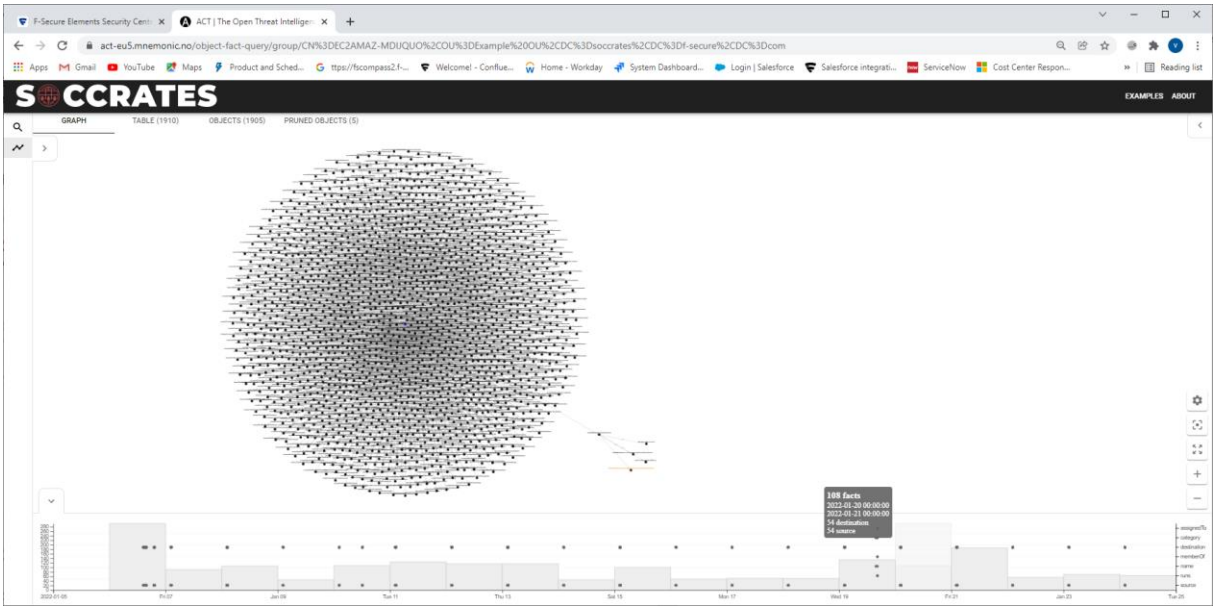

**Figure 7 : Observed traffic graph together with time dimension below it**

### 2.4.4 Extendable model

The IMC data model was extended and modified a few times during development to support changing of software cardinality rules and adding new data objects like users, groups, organization units and observed data flows. Changing existing objects in the IMC data model schema required coordinated schema changes to several components. Adding new data objects was easier as it was possible to gradually add support for new objects to each of the affected SOCCRATES components.

None of these schema changes required customer data to be deleted or migrated. This is important, since making schema changes in customer environments requires a change management process that prevents breaking customer facing services until all necessary components are updated.

### 2.4.5 Multi-organization

Although the final prototype was only tested in one organization, all IMC components were developed to support multi-tenancy. In tests, all data was created under one organization and all access rights were granted only for that single organization.

Fundamentally all current IMC components can be used in a multi-tenant setup and the fact that the same end customer data could be fetched from ADG, using the ACT credentials scoped to demo environment only, suggests that the basic support for multi-organization is already working.

# 3 Attack Defence Graphs

## 3.1 Introduction

The Attack Defence Graph analyser (ADG) component can be used to predict how attacks propagate over a given ICT infrastructure. In SOCCRATES this is used to predict the potential effect of a new threat, a new vulnerability, or a change in system configuration. In addition, when an attack is detected, ADG-based analysis can be used to determine if and how the attacker might compromise other systems. SOCCRATES will use this to provide the SOC / CSIRT analyst with detailed insight into what systems can be compromised, via which attack paths, and how likely they are. The ADG component is based on Foreseeti AB's securiCAD solution for automated threat modelling and predictive attack simulations. In the SOCCRATES platform ADG component role, securiCAD is also adapted for continuous operational monitoring.

## 3.2 Current system architecture

This section describes the high-level architecture of the current ADG component and its main components and functionality (see Figure 8). The ADG component is based on a service-oriented architecture where different services with specific responsibilities communicate using REST APIs or via a Message broker service. The internal functionality can be divided into three main processes: model generation, attack simulation and analysis.

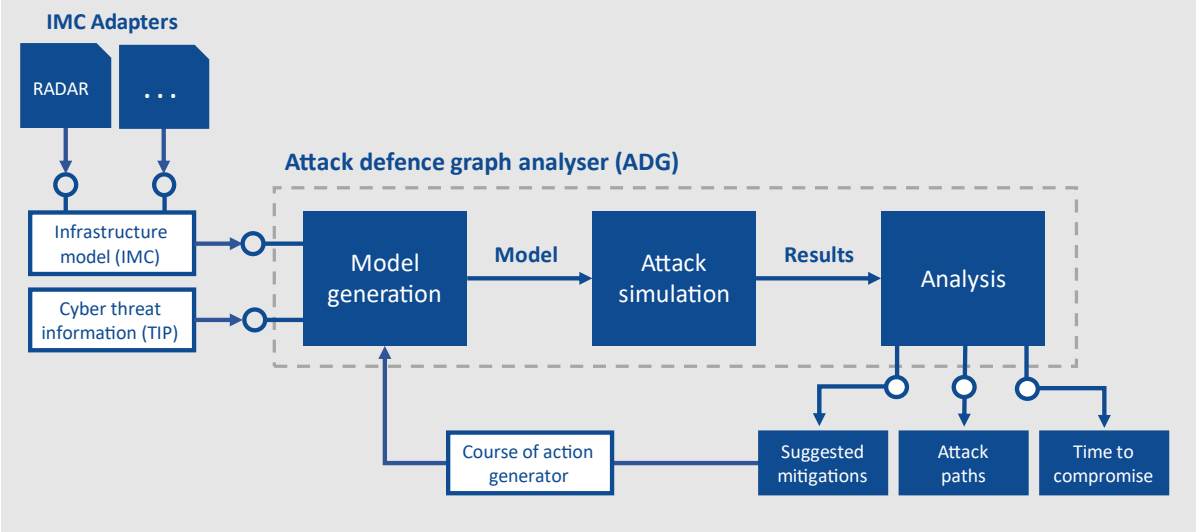


**Figure 8. High-level architecture of the current ADG component**

### 3.2.1 Model generation

The ADG component performs attack simulations on models of IT environments. These models can be generated automatically, created manually, or by a combination of both. In the context of the SOCCRATES project, models are generated using IMC data. Models are created by adding assets and connecting them to each other to represent machines, users, networks, cloud services, and any other type of asset that you typically find in your domain. The model definition, i.e., what asset types there are and how they relate to each other is defined in a domain specific language definition. This specification also contains attack logic, i.e., how an attacker may behave and move in the model by exploiting vulnerabilities, lateral movement through valid accounts, phishing of login credentials, and so on.

### 3.2.2 Attack simulation

The ADG component utilizes the securiCAD simulation engine to perform attack simulations on the generated model. Attack simulations generate attack graphs containing all possible steps and paths between an attacker's entry point and high value assets in the model. Monte Carlo simulations are used to sample the attack graph and its statistical data on success rates and expected effort required to conduct attack steps. Shortest path calculations are then used to find the most likely paths through the graph.

### 3.2.3 Analysis

Based on the results of the simulation, the securiCAD engine will return the most likely attack paths to high value assets, the probability of success, the Time-To-Compromise (TTC) and suggested mitigations in the form of a web-based report or as JSON data.

### 3.2.4 Threat information platform (TIP)

Cyber threat intelligence (CTI) from the Threat Information Platform (TIP) is used by the ADG, in certain use cases, to customize the attack simulation according to specific threat actors or attack capabilities and resources. The ADG Cortex Analyser fetches CTI from the TIP based on requests from the and applies it to the model generation or attack simulation stage, depending on the use case.

### 3.2.5 Course of action generator (CoA)

The Course of Action generator component uses the automation SDK to communicate with the ADG component and obtains simulation ids of ADG analyses from the orchestrator platform. The CoA runs multiple simulations against the ADG to arrive at a set of suggested mitigations which are returned to the OIE.

## 3.3 Domains and integration

To be able to model and simulate modern IT architectures, one must acknowledge the fact that most IT architectures are actually hybrid environments. Many organizations migrate their datacentres and workloads to the cloud, but even if they run their own on-premise solutions, many tools and services used are often hosted by other suppliers. The securiCAD simulation engine only had built-in support for general on-premise environments with little or no opportunity to extend or customize the language definition.

### 3.3.1 Support for the Meta Attack Language (MAL) framework

In order to support additional domains and modern hybrid environments, the underlying language definition must be dynamic and customizable. Meta Attack Language (MAL) is an open-source framework for specifying assets and associations of models as well as attack steps, defences, and probabilities in attack graphs. MAL specifications are compiled into a software package that can be used directly with a securiCAD simulation engine without any additional tweaks or modifications. By using and extending MAL, hybrid environments can be modelled and simulated in the ADG component.

New MAL specifications for specific domains such as cloud environments can be combined with specifications for "traditional" IT architectures to facilitate cross-domain analysis. The use of MAL has allowed the modelling and attack simulation logic of different domains to be decoupled from ADG, allowing a domain-specific language to be developed independently from, and dynamically loaded into ADG. In SOCCRATES, the scope of the ADG component ranges across both cloud environments and on-

premise infrastructures, but the focus is on on-premise environments as this is the target of both project pilots. The currently supported cloud service providers are Amazon Web Services (AWS) and Microsoft Azure.

### 3.3.2 Support for on-premise environments

On-premise ICT environments are the target domain of the SOCCRATES project, and this is in many ways a much more challenging target than a cloud environment, since there is no single consistent and homogenous data source from which such environments can be modelled. This is exactly what the Infrastructure Modelling Component (IMC) solves. ADG uses IMC as a single data source for continuously running the modelling – attack simulation – analysis loop to support SOCCRATES use cases. coreLang (https://github.com/mal-lang/coreLang) [2] is an open-source initiative to create a MAL-based threat modelling and attack simulation language based on few but powerful abstractions that allow flexible representation of many different environments. coreLang was chosen for the ADG component over the securiCAD-native securiLang due to its superior fit with the SOCCRATES cyber operational context in on-premise environments. In particular, coreLang has explicit support for software vulnerabilities, including automatic mapping from CVSS v2/v3 vectors to probability functions in the attack graphs and support for more direct and intuitive modelling of supply chain attacks.

### 3.3.3 Support for Amazon Web Services (AWS)

The ADG component is also capable of generating models and running attack simulations of AWS environments. This includes a MAL-based language specification that defines AWS attack logic, as well as a parser that can automatically build infrastructure models from standard AWS APIs. AWS is a comprehensive platform with a large set of functions and services and the AWS support package currently covers the most important set of core services: IAM, KMS, EC2, VPC, RDS, S3, DynamoDB, Lambda, API Gateway, ECS, ECR, Glue, DataPipeline, SageMaker, CloudFormation, CodeStar, and Amazon Inspector.

### 3.3.4 Support for Microsoft Azure

The ADG component supports Microsoft Azure cloud environments in the same manner as it supports AWS. Azure features currently supported by ADG are: RBAC (IAM - via Management Groups, Subscriptions and Resource Groups), Virtual Machines, Virtual Networks, Network Security Groups, Key Vault, Cosmos DB, App service Apps, Function Apps, and Service Bus.

### 3.3.5 Integration

In order to achieve continuous and near real-time modelling and analysis of environments where the SOCCRATES platform is deployed, the ADG component must also be able to automatically and efficiently generate models from IMC data or other data sources for any target domain. These integration points between environment data and the ADG component are called parsers. Parsers effectively transform data collected from the target environment into the correct language specification format.

In Figure 9, **Domain** denotes a particular type of environment which the ADG component is targeted to support. Selection of the environment is in theory arbitrary, but in practice chosen by aspects of cohesion as the domain is to be formally represented in a MAL-based language and models auto-generated through a parser. There are currently two types of parser – the **IMC Parser** which parses data from the **IMC** and **Cloud Parsers** which parse data directly from cloud service providers.

- The connection between **Domain** and **Domain Language** denotes the representation of a particular type of environment in a formal Domain language specified in the MAL framework.
- The connections between **Domain Language**, **MAL compiler** and **Domain language plug-in** illustrate that the domain language is compiled into a securiCAD "plug-in" i.e., the simulation and modelling logic which is used for the attack simulation.
- The **Domain Language plug-in** and **ADG platform** connections show that the plug-in is loaded by securiCAD to provide the simulation functionality for that domain.
- **Parsers** are linked to domain languages and are written to utilize data sources describing a real environment of a particular domain. The source data is transformed into a model that can be simulated by securiCAD in the ADG component. Parsers are written with a standard **Parser SDK**, allowing for the creation of tailored parsers for different domains.
- The **Automation SDK** component is used to automate both the modelling process as well as the simulation and analysis process of the ADG component. It is utilized to integrate the ADG component into the larger SOCCRATES system context. This is illustrated by connections to the Cortex analyser of the Orchestrator and the COA.

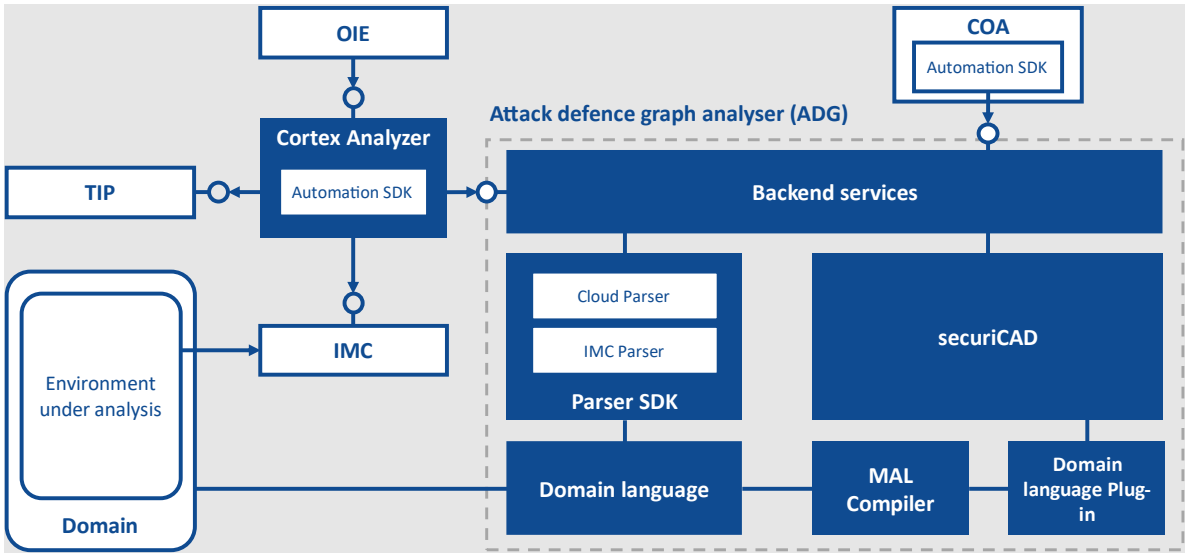


**Figure 9. Domain support and auto-modelling**

## 3.4 Deployment

The ADG component is a multi-user and multi-tenant web-based platform. An ADG deployment includes a centralized server instance holding the project and user databases. System services and other components are accessed via a browser, APIs, or the SDK. The ADG component can be installed directly on dedicated hardware or in a VM with a supported guest OS and can be deployed on most modern servers, workstations or laptops.

### 3.4.1 Requirements and installation

The ADG component requires a machine (dedicated hardware or virtual) with at least 8GB RAM and 30GB of storage, and currently supports the following operating systems: Ubuntu Server 16.04/18.04/20.04, RHEL 7 and CentOS 7. The installation requires a dedicated account with sudo privileges. By default, the installer connects to the Internet to download needed packages. If required, Foreseeti can provide installers with all packages included where Internet access is not available or allowed.

### 3.4.2 Target domain

The ADG component currently requires that the intended target domain is selected before deployment so the required language specification can be installed (based on whether it will be deployed in AWS, Azure or on-premise). Support for deployments for hybrid-environments will be added.

## 3.5 Current capabilities

In this section we introduce the current capabilities of the ADG component and our approach to verifying them. Verification was carried out in the SOCCRATES testbed. Cloud environments were used to verify AWS and Azure integration functionality. The functions provided by the current ADG component prototype are:

- Perform probabilistic attack simulation on a given infrastructure model provided by the IMC and supported cloud service providers, including:
  - attack simulations for a particular set of high-value assets and specific starting point and capabilities of the simulated attacker (Use Case 1).
  - assess the probability of a successful attack with regards to specific security measures and model configurations.
  - find the most probable attack paths to high-value assets in the model.
- Quantify and provide a metric based on the most probable attack path to each of the high-value assets, including:
  - Time to compromise (TTC) i.e., the time to a successful attack, will be provided for each high value asset
- Compare and quantify the difference between simulation results with regards to the aforementioned metric. Difference with previous analysis could be due to:
  - Newly discovered vulnerable assets from the IMC (Use Case 3).
  - Change in the infrastructure model from the IMC (Use Case 4 and 5).
- Compare and quantify the difference between simulation results with regards to:
  - New threat information from the TIP (Use Case 2).
- Update or add new attack steps to the attack simulations based on, for example, increased use of a particular attack technique (Use Case 2.1.1).
- Update the parameters of vulnerabilities in the model based on new information about the vulnerability. For example, the availability of public exploits or released patches (Use Case 2.1.2).
- Define a threat actor with specific set of preferred attack techniques or capabilities (based on Adversary Emulation Plan) (Use Case 2.1.3).

The above capabilities performed as expected and delivered the expected data to the SOCCRATES testbed environment as well as tested cloud environments. However, some expected challenges arose in more complex environments where not all data points were as neatly curated as in the testbed:

- The ADG component relies on a statistical simulation engine to calculate the most likely attack paths. These attack path likelihoods often depend on security controls or configuration points in the environment e.g., the existence of a vulnerability scanner or 2FA on an account. If these data points are not available in the IMC, the ADG component makes assumptions about the model. The ADG component must be able to communicate these assumptions together with results, and present the recipient with an opportunity to change them.

- Changes to the underlying simulation logic can greatly affect results and create large changes in the quantified results as they are compared over time. These changes must be communicated by the ADG component to the recipient to note that changes have been made to the simulation logic and that they can be the reason behind changes in the results.

# 4 Couse of Action (CoA) Generator

## 4.1 Introduction

The Course of Action (CoA) generator component operates on a model of an ICT infrastructure produced by the ADG component. The CoA generator component searches for "inactive defences" in the model and suggests to activate a set of them that have been identified as the most effective given a predefined budget. In SOCCRATES this is used to suggest defences to improve the security of ICT infrastructure to an SOC analyst. The CoA generator forms a feedback loop with the ADG analyser. To drive the optimization process, the generator computes a partial response to the threat based on the input attack path, runs the ADG analyser to obtain the expected behaviour of the attacker under this partial response, and then repeats the procedure on the newly obtained attack path. These iterations thus happen *inside* the CoA generator component without the involvement of the central orchestrator (OIE).

## 4.2 System Architecture

This section describes the high-level architecture of the current CoA generator component and its main components and functionality. The CoA generator component internally leverages the ADG component to obtain a simulation id of an ADG analysis from the orchestrator platform to ensure it is working with the same models (see Figure 10). Once this is achieved, the ADG component is contacted multiple times by internal bi-lateral communication. Once the CoA generator has arrived at a result with a set of suggested defences to implement in the model, a list of data is forwarded to the orchestrator platform for further analysis and basis for actions.

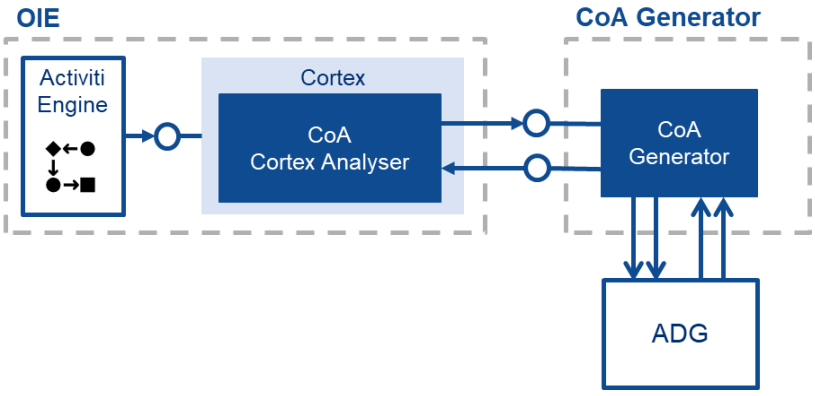


**Figure 10. Relationships of CoA Generator with other components**

### 4.2.1 Deployment of CoA Generator

The CoA generator component is a multi-user web-based platform. The CoA generator deployment includes a virtual machine instance containing the CoA generator component and user databases. Services of the CoA generator are accessed via APIs. The CoA generator component can be executed directly on dedicated hardware or in a VM with a supported guest OS and can be deployed on most modern servers, workstations or laptops.

### 4.2.2 Requirements and installation

The CoA generator component requires a machine (dedicated hardware or virtual) with at least 8GB RAM and 20GB of storage, and currently supports the following operating systems: Windows and Ubuntu Server 18.04/20.04. Execution of the CoA generator component requires user credentials.

### 4.2.3 Language Support

The CoA generator component is independent of the modelling language ADG is using, as long as it is defined in the Meta Attack Language. However, defence step implementation costs need to be defined separately per model (or language). In SOCCRATES we use the language named coreLang for which we have assigned some default costs for a subset of all defences in the language.

## 4.3 Model Structure

### 4.3.1 Attack Graphs

An attack graph, generated from an ADG model, is a directed acyclic graph. Its nodes correspond to attack and defence steps that can be performed in the modelled system. The edges model attack logic, as encoded by ADG's domain specific language. An edge from one attack step e.g., AS1 to another attack e.g., AS2 step means that execution of *AS1* is a prerequisite for execution of *AS2*. Furthermore, attack steps are typed: there are AND attack steps and OR attack steps. An AND attack step can be performed only if all of its prerequisites have been executed; an OR attack step can be executed as soon as one of its prerequisites has been performed. An edge from a defence step *e.g., DS* to an attack step *AS2* means that if the defence *DS* is successfully implemented, then *AS2* cannot be executed, even if its prerequisites have been executed (the attack steps are here thus following AND logic). Finally, each of the nodes in an attack graph is assigned a probability distribution of Time To Compromise (TTC), representing the expected difficulty for an attacker to succeed once it is attempted. With $F$ denoting the set of all probability distributions, attack graphs considered in the CoA generator are defined as follows.

**Definition 1.** An attack graph is a tuple $G = (V,E,\text{type},\text{ttc})$, where

- $(V,E)$ is a directed acyclic graph,
- type: $V \rightarrow \{\&, |, \#\}$ is a function assigning types to nodes in $V$, in such a way that none of the nodes of type # has a parent,
- ttc: $V \rightarrow F$ is a function assigning probability distributions to nodes in $V$, in such a way that each of the nodes of type # is assigned a Bernoulli distribution.

The AND attack steps, OR attack steps, and defence steps are assigned types &, | and #, respectively.

### 4.3.2 Attack Simulation

An attack graph describes possible behaviour of the attacker and defender in the modelled infrastructure. The attacks, that is, ways in which the attacker can reach their target steps from initial position while respecting the AND and OR requirements of particular steps, correspond to subgraphs of the graph. While these subgraphs are not necessarily paths in the standard meaning of the word, we will call them attack paths.

**Definition 2** (Attack path)**.** Let $G = (V,E,\text{type},\text{ttc})$ be an attack graph and let $X \subseteq A_G$. An attack graph $P = (V',E',\text{type},\text{ttc})$ is an *attack path in G starting in X* if

- $(V',E')$ is a subgraph of $(V,E)$ and $V' \subseteq A_G$,
- for every $v \in V'$, there is a node $w \in X$ for which there is a path in $G$ starting in $w$ and ending in $v$,
- if $v \in V' \setminus X$ and type($v$) = &, then for every parent of $v$ in $G$ the edge connecting $v$ with the parent belongs to $E'$,
- if $v \in V' \setminus X$ and type($v$) = |, then there is a parent of $v$ in $G$ such that the edge connecting $v$ with the parent belongs to $E'$.

Let $X_e$, $X_t \subseteq A_G$ be disjoint sets of attack steps, representing attack steps already executed by the attacker, and the attacker's target steps, respectively. Furthermore, let $Y \subseteq D_G$ be a set of defence steps implemented in the modelled infrastructure. An attack path $P$ in $G$ starting at $X_e$ compromises $X_t$ under $Y$ if each of the nodes in $X_t$ belongs to the path, and none of the nodes in the path has a parent in $Y$. Intuitively, such a path describes an attack that starts with the attacker executing the steps in $X_e$, and eventually, respecting the AND/OR dependencies between the steps, leads to compromising the target attack steps, in the presence of the set of countermeasures $Y$.

When selecting countermeasures for improving the security of a system, one generally prefers securing the system against the attacks that are most critical in some sense. One way of determining the most critical attacks would be to extract all attacks from the model, and then compare them using a metric of choice. The goal of attack simulations is to assess the security of valuable assets in the modelled system and to identify ways in which they could be compromised.

## 4.4 Algorithm

The CoA generator handles the problem at hand in the following way: given an attack graph, generated from a model of an ICT infrastructure including knowledge of possible defence steps that could be implemented, the CoA generator suggests a set of defence steps the implementation of which increases TTC of specified attack steps, while respecting some constraints. We consider three elementary constraints relevant to implementation of security measures:

- limitations on available resources (e.g., fixed budget for operating costs for securing the system),
- order dependencies, specifying that some measures can be implemented only in a specific order (e.g., to run an antivirus scan, one needs to install an antivirus component beforehand),
- mutual exclusivity between the measures (e.g., if we have two accounts for a service we might for business reasons not be allowed to disable both simultaneously).

In the SOCCRATES project only resource constraints in terms of an expected monetary cost for activating defence mechanisms have been used.

### 4.4.1 Criticality score

In order to select defence mechanisms to suggest for activation, the algorithm relies on the notion of a criticality measure of an attack step with respect to target steps in the attack graph. Formally, our criticality measure is a family of functions (including the non-deterministic ones) parameterized over attack graphs.

$$Crit_G : X \times 2^X \rightarrow R_{\geq 0}$$

Intuitively, if for a set $X^T$ of attack steps in an attack graph G the value of Crit(x, $X^T$) is high, then the attack step x plays an important role in the attacker's reaching of the target steps.

At each iteration of the method of CoA generation, an attempt will be made to add a set consisting of one or more defence steps and all of its prerequisites to the solution generated so far. At every iteration, attack simulations are performed under the assumption that the defence steps in the solution generated so far are implemented in the system. The simulations return attack paths, which describe expected reasonable behaviour of the attacker, and the corresponding global TTC values for the target attack steps. The attack paths are merged into a single graph, and a criticality measure is computed for every attack step in the graph that has a defence step as a parent. As long as there are attack steps with defence step parents in the attack paths, the most critical of these steps is picked, and the defences that counter it are examined. Once a defence step that can be added to the solution without violating any of the constraints is encountered, this defence step and all of its prerequisites are added to the solution, and the set of counterable steps is updated. Analysis of the counterable attack steps stops once all of them have been examined or when the number of defence steps added to the solution in the current iteration exceeds a prespecified bound of fidelity for the iterations. The fidelity parameter defines how many defences that are added to the CoA per iteration, thus enabling a trade-off between the quality of the solution produced and the number of simulations performed. To obtain a high quality solution, one could run a large number of iterations in the algorithm. Then, the next iteration starts. The main loop of the algorithm stops once a stop condition is met (e.g., a desired improvement in TTC values has been achieved) or an iteration passed without altering the solution.

We could imagine several different measures for deriving criticality scores of attack steps. In this work we are combining two metrics of criticality; Frequency and Fan-out of attack steps.

Let G=(X, E, type, ttc) be an attack graph, where X denotes a set of vertices, E denotes set of edges, type represents type of attack steps and ttc denotes Time-to-Compromise value. Let $P = \{P^{(x)} : x \in X^T\}$ be a set of lists of attack paths in G compromising target attack steps in $X^T \subseteq X$. Set Set $G' := \bigcup_{x \in X^T} \bigcup_{P \in P^{(x)}} P$. Therefore, $G' = (X', E', type, ttc)$.
For $x \in X^0$, let

$$freq(x) := \sum_{x \in X^0} |\{P \in P^{(x)} : x \in X(P)\}| \quad (4.1)$$

$$out_w(x) := \sum_{(x, x') \in E(G')} freq(x'), \quad (4.2)$$

The frequency function *freq,* defined by equation 4.1, returns the number of attack paths in which the attack step x appears. In other words, it counts how frequently the attacker used this step in the simulations. If this value is high, then the attack step is a likely contributor to the attack on the target assets, and so the step can be deemed critical.

The function defined by equation 4.2 is the weighted "out-degree" of x in G', meaning the number of children step to x. Intuitively, the more attack steps that are enabled by x, the more critical x is. However, x will be more critical if its children attack steps also have high frequencies. Thus, instead of simply counting the children, the sum of their frequencies is computed.

As a side comment we note that using the frequency and out-degree metrics in no way are the only potential criticality scores. Alternatively, perhaps most intuitively, a measure of quality of mitigations could be used, and the focus could be put on selecting a mitigation technique of the highest quality. In the context of our framework, however, it is important to remember that at each iteration of CoA generation only partial knowledge about the attacks against the modelled infrastructure are available. In particular, while it is known which of the attack steps lie on the currently optimal attack paths, the exact importance of the remaining attack steps is not known. In consequence, measuring quality of mitigations countering only the latter attack steps is not straightforward. Also, the overall assessment of implementing a defence mechanism is evaluated in later simulation steps. So, the criticality score can be considered a generator of hypotheses for defence mechanisms that should be applied.

### 4.4.2 Cost and Budget for the Selected Countermeasures

As mentioned previously, monetary costs for the implementation of defences are the constraining factor. For costs we will assume that activation of different types of defences are independent of the ordering of the activation, e.g. the total cost of sending staff to security awareness training and deploying an intrusion detection system is the same regardless of the order in which these to defences are implemented. The costs are simply added. However, we will take marginal cost decrease or increase into account for the implementation of the same type of defence, as specified by the underlying ADG model language. E.g. we can specify that implementing an intrusion detection system on one network has one cost and another, possibly lower, cost for implementing it on a second or third (and so on) network. The overall cost budget is simply defined by a single value.

## 4.5 Example

### 4.5.1 Application of CoA Generator for an IT infrastructure model

ADG uses a DSL (Domain Specific Language) which encodes security knowledge by assigning attack steps and defence steps and how they relate to ICT infrastructure asset types. In Figure 11, an example model devised using the DSL coreLang is illustrated (attack and defence steps are not visualized).

The model represents a threat scenario where an attacker is present on the internet and is also able to launch a phishing campaign towards a user in the system environment. The system environment is a small infrastructure consisting of two virtual machines connected to separate networks and a VPN service. Virtual Machine 2 is running a VPN client that is connected to a VPN server. The VPN server in turn has a vulnerability and credentials for logging onto it are stored on Virtual Machine 1. All these assets have a number of attack and defence steps associated with them that are not described in completeness for space reasons, but this includes for instance that vulnerabilities can be exploited, data and credentials can be read, and networks can be eavesdropped. In our scenario we have defined the VPN server as the target we want to protect.

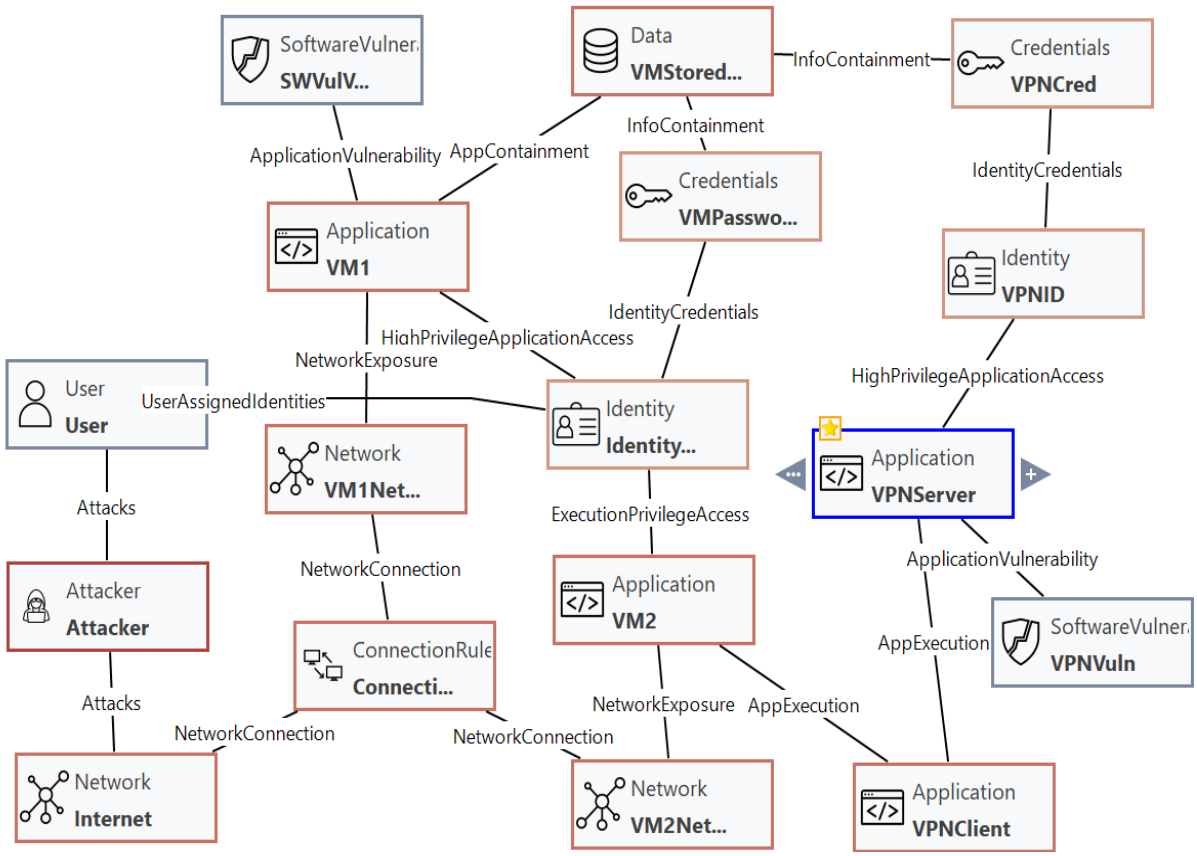


**Figure 11. An illustration of an example model devised using the DSL coreLang**

### 4.5.2 Result of CoA Generator

The model in Figure 11 is simulated using the ADG component that generates TTC values for all attack steps encoded on assets in the model. The ADG also generates a "critical path" for the attacker to reach our selected target, the VPNServer, as illustrated in Figure 11. This "critical path" is the aggregate of the most frequently traversed attack paths in the simulation over the different simulation samples. In the critical path we see that the attacker exploits a vulnerability in Virtual Machine 1, gains access to the credentials for Virtual Machine 2, and then legitimately connects to the VPN server via the VPN client. For all of the attack steps the ADG calculates a TTC distribution. For our target, the attack step '*FullAccess*' on the VPN server, the initial distribution is illustrated in Table 2. Along the critical path in Figure 12, we find defence steps that have not yet been implemented (green nodes). Along with many other defence steps, not illustrated in the critical path, these are the potential action space for the defender. In the SOCCRATES project a subset of these defence steps have been assigned with cost values based on approximations[1] made by project members that has been averaged. The cost values of the used defences are described in Table 1. The marginal cost for implementing the same type of defence is in Table 1 represented by a vector where the i:th item in the vector represents the cost of implementing the defence the i:th time. For instance, applying *'Disabled'* on Virtual Machine 2

[1] To find exact cost values is generally challenging and also largely system and context dependent. The values elicited for the SOCCRATES project are mainly to be considered exemplary values that serve to illustrate the GoA generator functionality but that hopefully also is in a reasonable order of magnitude.



**Classification level: Public**

costs 30 units (as it is the first time this type of defence is activated, in subsequent disabling of Applications this action will cost 20 units).

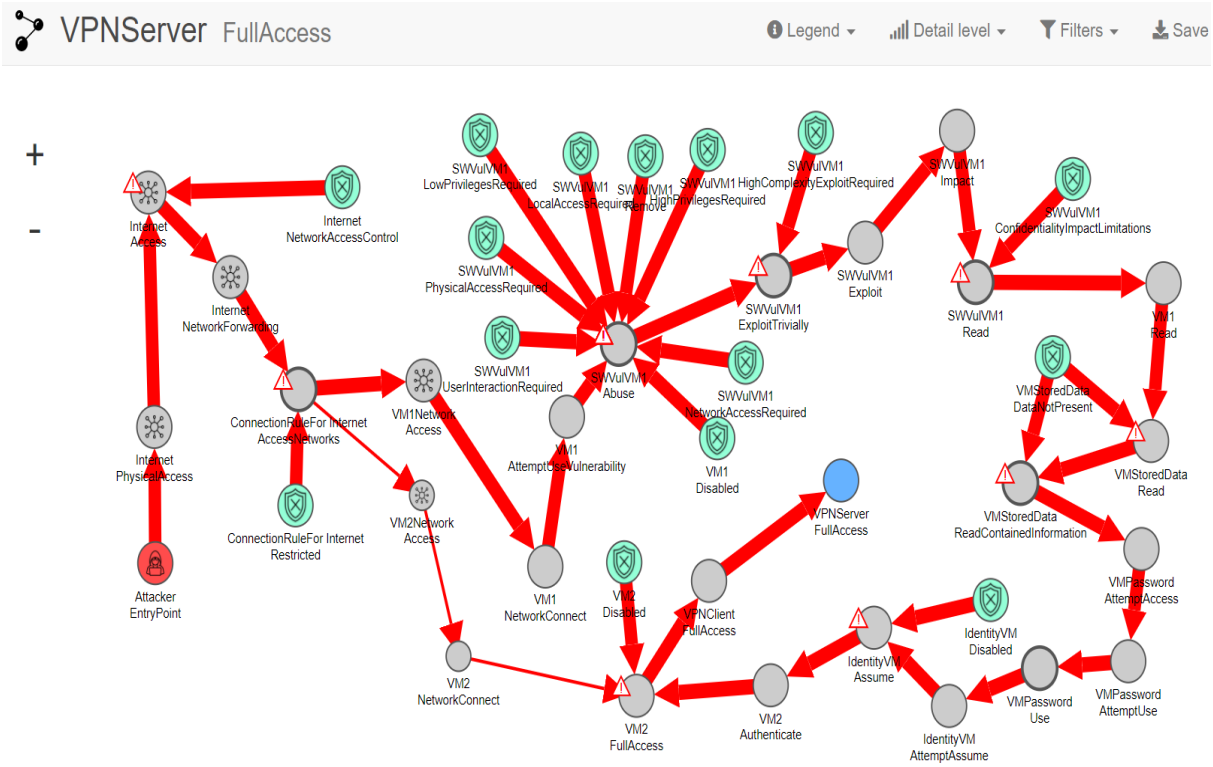


**Figure 12. Critical attack path of the example model.**

In Table 2 we summarize the results of the CoA generator algorithm iterations. The initial budget is 1000 and the initial cumulative distribution function for the TTC value is 19/26/50 for 5$^{th}$/50$^{th}$/95$^{th}$ percentile. In the first iteration of CoA generation, the defence step '*SWVulVM1.Remove*' is implemented (with a cost of 10 units), which increases the TTC values 30, 75 and 1.797e+308 (the value representing infinity) and the efficiency value of that iteration is 170. After implementing the defence step *'SWVulVM1.Remove',* a new critical path is generated by ADG as shown in Figure 13. The '*SWVulVM1.Remove'* defence step represents patching of the vulnerability in Virtual Machine 1. In iteration 2, the '*VM2.disabled*' defense step is implemented at a cost of 30 units, which significantly increases the target TTC. This defence represents shutting down Virtual Machine 2 (an action that might not be allowed in practice for various reasons). In the next iteration, the CoA generator does not suggest any further defence steps because the TTC value for the target attack step has become infinite. (Other reasons for the CoA generator to halt is if additional defences do not improve TTC more than a threshold value or that we run out of budget.)

**Table 1. List of Defence steps, object type and associated cost values for i:th use**

| Defense Name | Object Type | Costs | Description of defences |
|---|---|---|---|
| SecurityAware-ness | User | 1st: 64; 2nd: 65; ...; i:th: 65 | What is the cost associated to in-struct one employee on security best practices? |
| Remove | SoftwareVulnerability | 1st: 10; 2nd: 5; ...; i:th: 5 | What is the cost associated to patch an application? |
| Disabled | IAMObject | 1st: 64; 2nd: 76; ...; i:th: 76 | What is the cost associated to disa-bling an account? |
| HardwareModifi-cationsProtection | Hardware | 1st: 224; 2nd: 244; ...; i:th: 244 | What is the cost associated to add protection that prevent hardware tampering for a device? |
| Disabled | Application | 1st: 30; 2nd: 20; ...; i:th: 20 | What is the cost associated to shutting down an application? |
| Restricted | ConnectionRule | 1st: 226; 2nd: 247; ...; i:th: 247 | What is the cost associated to iso-late the communication between two applications? |
| NotDisclosed | Credentials | 1st: 242; 2nd: 122; ...; i:th: 122 | What is the cost associated to add-ing a monitor for all traffic be-tween two applications? |
| NotPhishable | Credentials | 1st: 242; 2nd: 122; ...; i:th: 122 | What is the cost associated to im-plement alternative ways of au-thentication (such as iris scans, key cards or fingerprints) for an applica-tion? |
| ManInTheMiddle-Defense | Network | 1st: 422; 2nd: 240; ...; i:th: 240 | What is the cost associated to monitoring a network and imple-ment robust encryption mecha-nism aimed at preventing Man-in-the-Middle attacks on a computer network? |

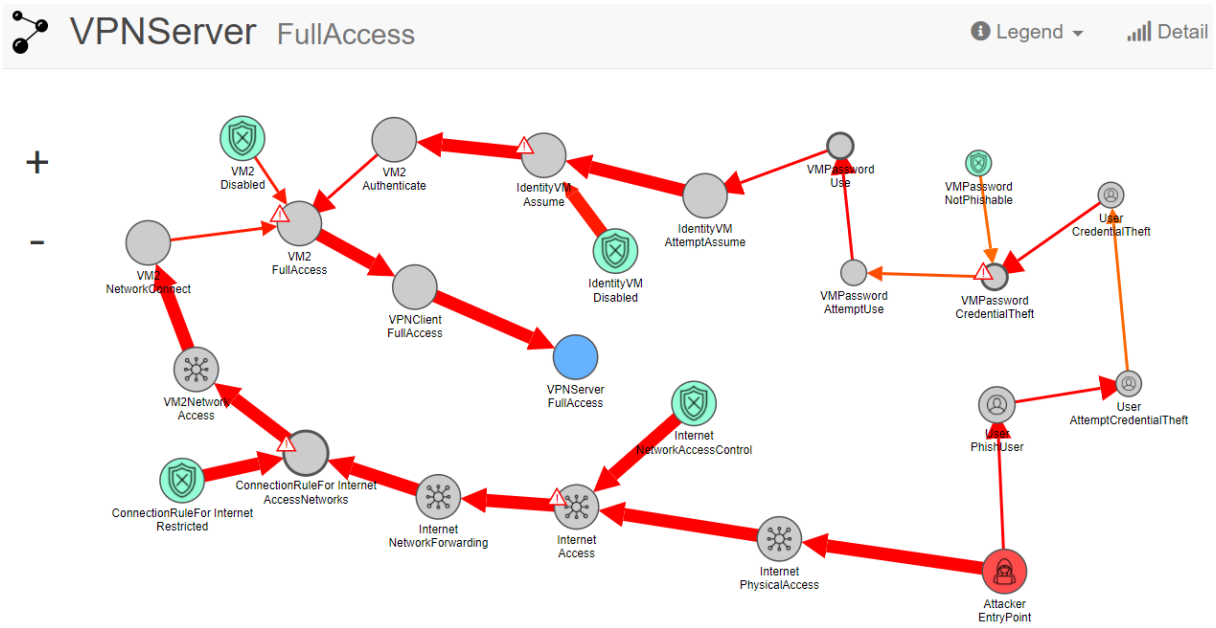


**Figure 13. Critical attack path of the example model after iteration 1.**

**Table 2. Result of CoA generation for different iteration.**

| | Base line | Iteration 1 | Iteration 2 |
|---|---|---|---|
| Defence steps added | N/A | SWVulVM1.Remove | VM2.Disabled |
| Efficiency score | N/A | 170 | 285 |
| Budget remaining | 1000 | 990 | 960 |
| Target TTC value {5%, 50%, 95 % probability} | *VPNServer.FullAccess* { 19, 26, 50 } | *VPNServer.FullAccess* { 30, 75, 1.797e+308 } | *VPNServer.FullAccess* {1.797e+308, 1.797e+308, 1.797e+308 } |

This example serves to illustrate how the CoA generator algorithm works. Results may seem trivial when using an overly simplified example (such as the one used here). However in large models perhaps containing millions of assets, the selection of defences will of course be less trivial, and quickly difficult for the human eye to determine. In more complex cases, the CoA generator will likely also run more iterations before halting. Thus it is also for large models that the end user value of the CoA generator is created.

To validate the CoA Generator, we have tested it on the model shown in Figure 11. The model consists of 17 objects. In the DSL used, there are on average 5.8 attack-steps related to an object, hence the attack graphs corresponding to the models are expected to have approximately 90 nodes. The number of simulations performed by securiCAD was set to 100. Variants of metric are used in the experiment such as criticality score for attack steps for frequency and weighted out-degree as shown in equations 4.1 and 4.2 respectively. A stopping condition is used to limit the number of executions of the algorithm. The execution of the algorithm is stopped if the fifth percentile of the TTC value for the target attack step has become infinite. The increased value of efficiency of CoA generator in iteration 2 represents the better performance of the algorithm as shown in Table 2.

The CoA generator is scalable in nature that constructs sets of countermeasures by repeated analysis of optimal attack paths in the modelled system. This can be seen as repeated identification of the most critical threats and implementation of the smallest possible change in the system improving the

security against these threats. However, while simulations are an effective way of identifying some of the optimal attack paths. After each iteration, the efficiency of the solution generated so far is computed that represent quality indicator of the solution.

# 5 Final prototype versions and test environment

The described test environment was built to demonstrate the capabilities of the final prototype versions and it is configured to protect the demo infrastructure. The demo infrastructure is a virtual environment representing a company network with a set of virtual LANs, routers, workstations and Active Directory.

Figure 14 illustrates the overall SOCCRATES platform deployment, which is split between TNO, F-Secure, mnemonic and ShadowServer. Deliverable D6.3 describes the approach and results of testing and verification of the SOCCRATES Platform. This document describes the final versions of the Infrastructure modelling, Attack Defence Graph and Course of Action generator components.

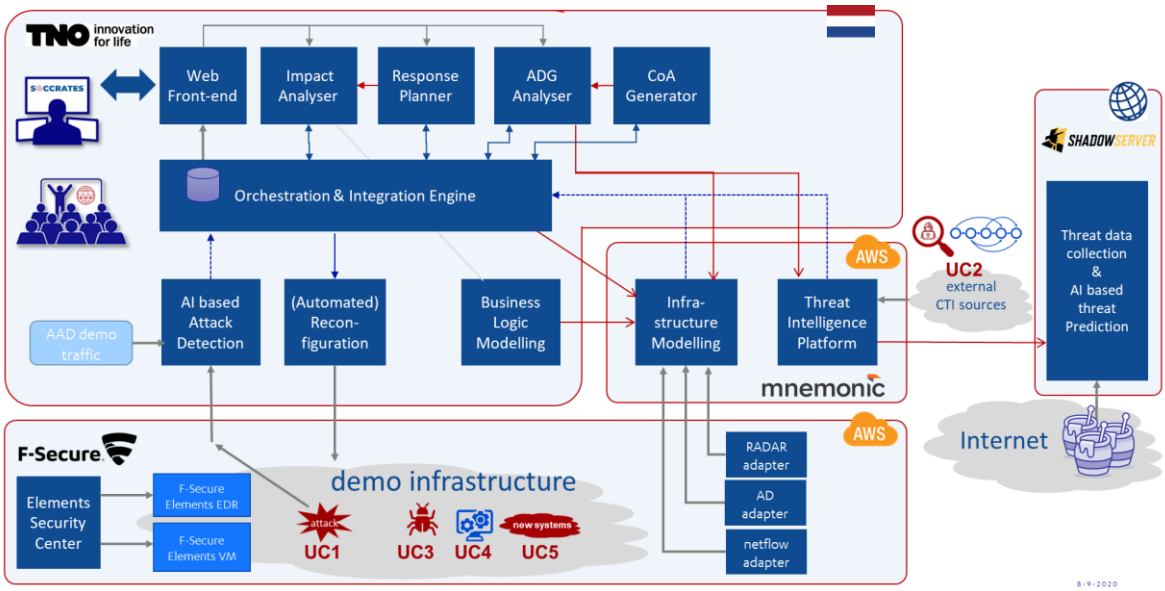


**Figure 14 : Final prototype deployment**

IMC is deployed in a hybrid mode, meaning that cloud deployed Elements Security Center is used to manage Elements scan nodes installed to the demo infrastructure and similarly cloud version of the ACT platform is used as a core database of IMC.

Other SOCCRATES components, including ADG and CoA Generator, are deployed to TNO test environment.

# 6 Abbreviations

This glossary serves as inventory of abbreviations used in the document.

| Acronym | Description |
|---|---|
| ACT | semi-Automated Cyber Threat intelligence |
| ADG | Attack Defence Graph |
| AEF | Argus Event Format |
| AI | Artificial Intelligence |
| AIT | AIT Austrian Institute of technology |
| API | Application Programming Interface |
| APT | Advance Persistent Threat |
| ATOS | ATOS Spain |
| AV | AntiVirus |
| AWS | Amazon Web Services |
| BPMN | Business Process Model and Notation |
| CC | Command and Control |
| CERT | Computer Emergency Response Team |
| CMDB | Configuration Management Database |
| CSIRT | Computer Security Incident Response Team |
| CoA | Course of Action |
| CTI | Cyber Threat Intelligence |
| CVE | Common Vulnerabilities and Exposures |
| DC | DataCentre |
| DGA | Domain Generated Algorithm |
| DNS | Domain Name System |
| EDR | Endpoint Detection and Response |
| ELK | Elasticsearch/Logstash/Kibana |
| FRS | Foreseeti |
| FSC | F-secure |
| ICT | Information and Communication Technology |
| IDS | Intrusion Detection System |
| IMC | Infrastructure Modelling Component |
| IMT | Institut Mines Télécom – Télécom SudParis |
| INTF | Interface |
| IoC | Indicators of Compromise |
| IP | Internet Protocol |
| IPS | Intrusion Prevention System |
| IRM | Incident Response and Management |
| ITIL | Information Technology Infrastructure Library |
| KTH | Kungliga Tekniska högskolan - Royal Institute of Technology |
| LAN | Local Area Network |
| LDAP | Lightweight Directory Access Protocol |
| MAL | Meta Attack Language |
| ML | Machine Learning |
| $M_n$ | Infrastructure Model (at time *n*) |
| MNM | mnemonic |

| | |
|---|---|
| MSSP | Managed Security Service Provider |
| MTTD | Mean Time To Detection |
| NOC | Network Operations Centre |
| OIE | Orchestration and Integration Engine |
| OT | Operational Technology |
| OS | Operating System |
| RORI | Return on Response Investment |
| SDN | Software Defined Network |
| SHS | Shadowserver |
| SIEM | Security information and event management |
| SOAR | Security Orchestration, Automation and Response |
| SOC | Security Operation Centre |
| SOCCRATES | SOC & CSIRT Response to Attacks & Threats based on attack defence graph Evaluation Systems |
| SSH | Secure Shell, a cryptographic network protocol |
| SSL | Secure Sockets Layer |
| TAP | Test Access Point |
| TIP | Threat Intelligence platform |
| TLS | Transport Layer Security |
| TNO | Nederlandse Organisatie voor toegepast natuurwetenschappelijk onderzoek |
| TTC | Time To Compromise |
| UC | Use Case |
| VLAN | Virtual LAN |
| VM | Virtual Machine |
| VM | Vulnerability Management |
| VTF | Vattenfall |