SOCCRATES

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

# D3.3

# Specification of the common ICT infrastructure reference meta model

| | |
|---|---|
| Deliverable type: | Report |
| Contributing work packages: | WP3 ADG based Attack Analysis |
| Due date of deliverable: | 31/08/2021 |
| Submission date: | 30/08/2021 |
| Dissemination level: | PU |

| | |
|---|---|
| Responsible organisation: | F-Secure |
| Editors: | Seamus Moloney |
| 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 concepts and structures of the infrastructure reference meta model which underpins these components. |
| Keywords: | Infrastructure modelling, standards, attack defence graph, Meta Attack Language |

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

# 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. This deliverable describes how data is stored in the SOCCRATES infrastructure model and how the data model is used in the Attack Defence Graph (ADG) component.

The reader is assumed to be familiar with the other SOCCRATES deliverables. The goal of this report is to specify the common reference meta model for describing ICT infrastructure, including the concepts and structures of the model.

| | |
|---|---|
| Contributor(s) | Seamus Moloney (F-Secure)<br>Ville Alkkiomäki (F-Secure)<br>Andrew Patel (F-Secure)<br>Marko Komssi (F-Secure)<br>Preetham Mukherjee (KTH)<br>Ashish Kumar Dwivedi (KTH)<br>Mathias Ekstedt (KTH)<br>Martin Eian (Mnemonic) |

| | |
|---|---|
| Reviewer(s) | Frank Fransen (TNO) |

| **Security Assessment** | **See deliverables table for security assessment requirements** |
|---|---|
| Approval Date | 25/08/2021 |
| Remarks | None. |

# TABLE OF CONTENTS

1 INTRODUCTION – RATIONALE OF THIS DOCUMENT ..... 6
1.1 The SOCCRATES project ..... 6
1.2 This deliverable ..... 7
1.3 Structure of this deliverable ..... 7
2 BACKGROUND TO INFRASTRUCTURE MODEL ..... 8
2.1 Introduction ..... 8
2.2 ACT Platform ..... 8
3 DETAILED DESCRIPTION OF THE IMC MODEL ..... 10
3.1 IMC objects and facts ..... 10
3.2 Model challenges ..... 15
3.3 Interoperability through standards ..... 16
4 ATTACK DEFENCE GRAPH ..... 19
4.1 MAL (Meta-Attack Language) ..... 19
4.2 coreLang ..... 21
5 REFERENCE ..... 29
6 ABBREVIATIONS ..... 30
7 APPENDIX ..... 32
7.1 fact-types-imc.json ..... 32
7.2 meta-fact-types-imc.json ..... 40
7.3 object-types-imc.json ..... 40

# 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 Horizon2020 programme that has the following main 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.

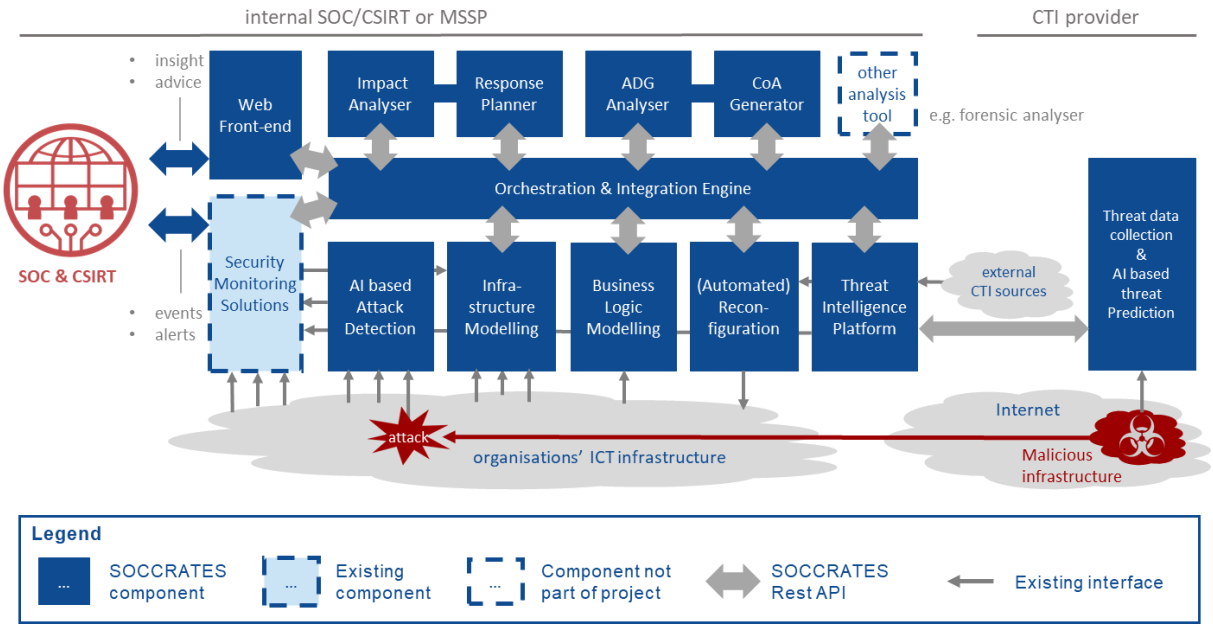


**Figure 1 The SOCCRATES platform**

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.

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. Show that the SOCCRATES platform can improve SOC operations by evaluating the SOCCRATES platform 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 This deliverable

The purpose of the Infrastructure Modelling Component (IMC) is to provide an accurate and up-to-date model of the ICT infrastructure to the other SOCCRATES components to function. This deliverable [D3.3] describes the IMC data model in detail and is meant to be used as a specification of IMC as seen from other SOCCRATES component point of view.

The initial specification of the IMC data model has been described in deliverable D2.2 and deliverable D3.1 described the revised data model after the IMC prototype had been implemented. This deliverable contains the final specification of the IMC data model.

In the ADG component of the SOCCRATES project, attack simulations on system architecture models are performed to support the identification of relevant security properties of a system, and the collection of this security related information [1], [2]. These approaches facilitate a model of the organization and simulate cyber-attacks in order to identify the greatest weaknesses. This can be imagined as the execution of a large number of parallel virtual penetration tests. Such an attack simulation tool enables the security assessor to focus on the collection of the information about the system itself required for the simulations. The attack simulations are based on Meta-Attack Language (MAL) which is a framework for creating Domain Specific Languages (DSLs). DSLs are used to perform attack simulations within that specific domain (e.g. enterprise IT network, industrial control systems, or in-car systems). In this deliverable the Meta-Attack Language (MAL) is described, and coreLang, a common set of Attack Language concepts that can be used as basis to define a DSL, is introduced.

## 1.3 Structure of this deliverable

This deliverable is structured as follows. Section 2 introduces the ACT platform which is used as a basis for IMC, section 3 provides a specification of the IMC data model and section 4 describes data models and concepts used in Attack Defence Graph (ADG) component merging organization specific IMC data with generic attack and defence information in order to make attack path simulations.

Technical schema files used to create the IMC database are provided as a part of this deliverable to allow further development of IMC outside the SOCCRATES project.

# 2 Background to Infrastructure Model

## 2.1 Introduction

The infrastructure modelling component (IMC) is a basis for producing a current and accurate model of the ICT infrastructure to enable automated security analysis tools to reason about the security state and for visualizations that accommodate human comprehension of the IT infrastructure. The figure below illustrates the IMC sub-components and their integrations with other components.

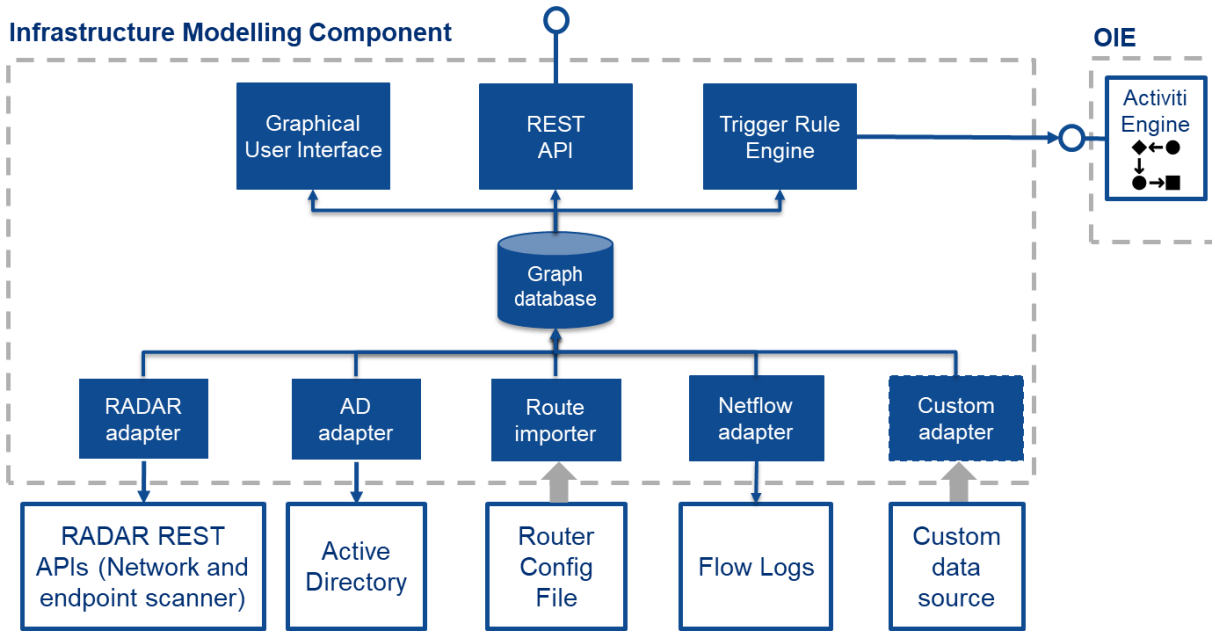


**Figure 2 IMC integration architecture**

The graph database is at the heart of the Infrastructure Modelling Component and stores the data fetched from the data source adapters. The graph database stores data with timestamp, allowing the data to be queried for a given period of time. The Graph database is based on the open source ACT platform from mnemonic.

## 2.2 ACT Platform

The ACT platform provides a set of versioned REST APIs in order to interact with the system. A Python library[1] to interact with the API is available, as well as a graphical user interface[2] (GUI). The API endpoints are documented with Swagger, which contains a full description of each endpoint, including requests and responses. The Swagger documentation is available at https://act-eu1.mnemonic.no/swagger/.

The Object-Fact-Model of the IMC consists of a set of objects, such as infrastructure components, access control rules, and configuration settings, on which the system can attribute facts collected from

[1] https://github.com/mnemonic-no/act-api-python
[2] https://github.com/mnemonic-no/act-frontend

IMC adapters. Each object represents a real world eternal entity, which exists (theoretically at least) independently of any other information, such as an IP address, a host, a routing table entry, etc.

Facts represent object properties and relations between objects, and may be added from different sources. Such facts may be uncertain, may be sensitive, and even contradicting or overlapping, and need to be traceable and backed by evidence and assessments.

The Object-Fact-Model therefore supports:

- Any number of unique objects which are globally defined.
- Any number of facts added to each object. A fact may be connected to a single object, or may be connected to two objects, creating a relation between them. A fact can also directly reference another fact to add more information to that fact (called a meta fact).
- Each fact is a simple statement of a property, is clearly owned by an organization and traceable to a source and point in time.
- Multiple facts may overlap and/or contradict, it is up to the analyst to determine which information to trust.
- Facts are immutable, that is, once a fact is added to the system, it cannot be altered or removed. New information is added as additional facts. To contest a false fact, a user may issue a new fact of type "retraction", asserting that the referenced fact is invalid/retracted.
- The sharing level is set on each fact, to allow granular access control to information. Access control is both role-based and explicit.
- Each fact can be backed by a set of evidence and comments.
- Since every piece of data is timestamped and immutable, this allows the system to "rewind time" or "replay events", and create exact timelines.
- Even changing the access mode of a fact is not allowed. To make a fact more public, the fact needs to be retracted and republished with a less restrictive access mode. Adding a fact with a more restrictive access mode does not make sense, since any user who had access to the original fact will still do so.

The IMC is configured with a schema that defines:

- Object types and fact types.
- Validators (regular expressions) for the value of each object type and fact type.
- Allowed source and destination objects for the different fact types.

# 3 Detailed description of the IMC Model

## 3.1 IMC objects and facts

This section describes each type of object and fact in the current infrastructure model and explains the reasoning for these as well as providing some example values. Facts can represent either properties of an object or the relationships between objects.

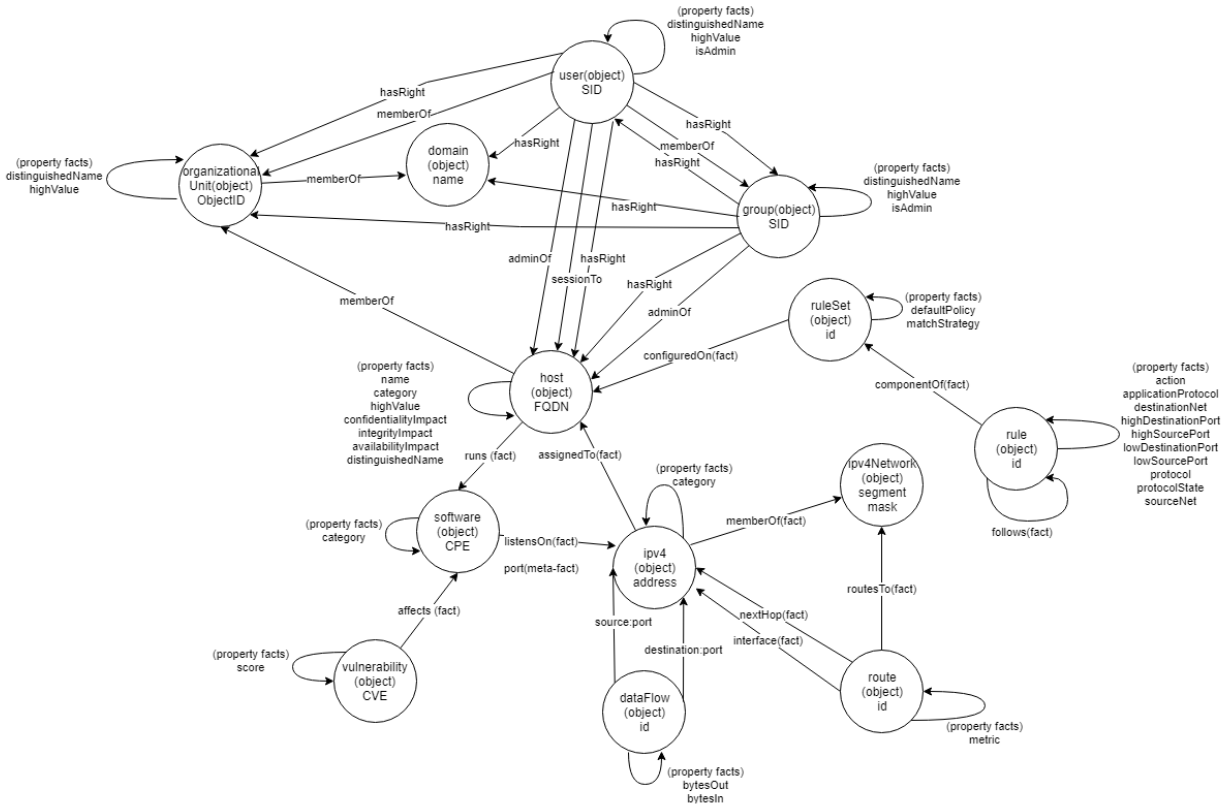


**Figure 3 Infrastructure Modelling Component Objects and their relationships**

Note that in addition to the properties below on objects and facts, there is a meta-fact stored about each object and fact in the data model called observationTime.

**dataflow –** The dataflow object is a calculated aggregate representing traffic which happened between two IP addresses. The Flow log adapter collects network information between hosts based on Cisco Netflow and AWS VPC Flow log formats. This will complement the static network route and firewall rule information defining what connections are possible in general with automatically gathered information of the actual network connections between the ipv4 addresses.
The dataflow objects contain two property facts:

- bytesIn / bytesOut: provide information about the volume of traffic between the two ipv4 objects.

The observationTime meta-fact can in this case be used to see when the particular aggregate dataflow object was created and by combining the dataflow objects, the volume of traffic between the two ipv4 objects over time can be plotted.

The dataflow object also has 2 facts which relate it to the ipv4 address objects:

- source / destination: These facts link the dataflow object to the source and destination ipv4 addresses which the aggregate is measuring the traffic between. This fact also contains a value for the port number which was used for the data transfer.

**domain –** The domain object is provided by the Active Directory adapter. It contains the following properties. It is uniquely identified by a Security Identifier (SID) which is generated by the domain controller in that network.

- distinguishedName: This describes the hierarchical paths in Active Directory to the domain and can be used to uniquely reference this object. For example, "DC=soccrates,DC=f-secure,DC=com".

The domain can have the following facts relating them to other objects:

- memberOf: The Active Directory adapter provides information about which organizationalUnits are members of each domain.
- hasRight: Both user and group objects can have rights to perform actions on the domain objects. The details about what actions are allowed are stored in the value of the hasRight fact.
- trusts: The model supports trust between domain objects. Active directory supports trusts between different domains, although this has not been tested yet in pilots.

**group –** The group object is used to model Active Directory groups. In Active Directory, groups are used to collect user accounts, computer accounts, and other groups into manageable units. Working with groups instead of with individual users helps simplify network maintenance and administration. As for domain objects, the groups are uniquely defined by their SID values which are collected with the Active Directory adapter.

Groups have the following property facts in the IMC data model:

- distinguishedName: This describes the hierarchical paths in Active Directory to the group and can be used to uniquely reference this object. " CN=IIS_IUSRS,CN=Builtin,DC=soccrates,DC=f-secure,DC=com" is an example for this field.
- highValue: This is a Boolean field which indicates if the Active Directory adaptor has flagged that the group is of high value to an attacker. A high value target is by default any group that belongs to the domain admins, domain controllers, and several other high privilege Active Directory groups.
- isAdmin: This is set AD adapter to indicate if the group should be considered as an admin level group.

Active directory groups can have the following facts relating them to other objects in the model:

- adminOf: This details which hosts the group has been declared as a local administrator for. In practice the users who are in the group would be able to log into those hosts as local administrators.
- memberOf: The Active Directory adapter provides information about which users or groups are the members of this group.
- hasRight: Both user and group objects can have rights to perform actions on the group objects. The details about what actions are allowed are stored in the value of the hasRight fact. Groups can have both incoming and outgoing hasRight facts and the direction of the link is important in that the source indicates the right holder.

**host -** This object type represents a computer connected to the target organizations network. Hosts are uniquely identified by their fully qualified domain names (FQDN). An example of an identifier would be EC2AMAZ-71QLQ44.soccrates.f-secure.com.
Most of the information in the model about hosts comes from the Radar adapter, but also the Active Directory adapter collects information about the hosts seen and the groups these hosts belong to as well as details about the local administrators of the hosts and those who have rights to make changes on the hosts.

Hosts are allowed to have the following properties stored as facts:

- availabilityImpact: This is a number between 0 and 5 (high) representing the impact to the company of this host becoming unavailable. The data is sourced from Radar where administrators manually can select the value. The field is not present for all hosts as it is a manual configuration in Radar. See the CIA triad model for more details.
- category: This field can indicate if the host is a client or a server machine. Sourced from Radar
- confidentialityImpact: Similarly, the impact to the organization of data on the host being leaked may be stored as number between 0 and 5 (high) and is sourced from Radar if available.
- distinguishedName: This describes the hierarchical paths in Active Directory to the host and can be used to uniquely reference this object. It contains details about the domain and organizational unit the host is a member of. This is sourced from the Active Directory adaptor.
- highValue: This is a Boolean field which indicates if the Active Directory adaptor has flagged that the host is of high value to an attacker. A high value target is by default any computer that belongs to the domain admins, domain controllers, and several other high privilege Active Directory groups.
- integrityImpact: This property on host objects details the impact that a successful attack on the host which would result in being able to manipulate data and settings on the host would have. Again, it is a number between 0 and 5 (high) which is sourced from Radar if available.
- name: This is a friendly name for the host, sourced from the Radar data. For instance, this might be something like “PC2 (Web Server)”. It is mostly intended to help with using the ACT UI and navigating the data.

Hosts can have the following facts relating them to other types of objects in the data model:

- adminOf: This relationship identifies which users or groups are the named local administrators of the host. The data is provided by the Active Directory adapter.
- assignedTo: IP addresses (ipv4 objects) are assigned to host objects and these assignments change over time. This data is provided by the Radar adapter.
- configuredOn: The firewall logs adaptor provides facts about the firewall rulesets which are configured on hosts, and these are stored to this fact.
- hasRight: The active directory adapter provides a lot of details about the rights which users and groups have regarding the hosts and these rights are stored as facts with values corresponding to the name of the right which the user / group has regarding the host.
- memberOf: host objects can be members of organization units in active directory.
- runs: host objects are linked to software objects through this relationship.
- sessionTo: The active directory adapter provides information about the sessions which are active between users and hosts.

**Ipv4 -** The ipv4 object is a simple representation of a network address. For example, 10.255.157.245. ipv4 objects can have a sole property fact allowed called “category” but support the several following facts relating them to the other objects:

- assignedTo: The host object which the IP address is assigned to at that point in time.
- destination (and source): dataflow objects can be connected to ipv4 objects via these facts
- interface: This links the ipv4 and route objects.
- memberOf: The ipv4Network object which the ipv4 object belongs to. This basically means that the netmask.
- nextHop: route objects have a nextHop relation to the ipv4 objects. This data is provided by the Flow log adapter.

**ipv4Network -** The ipv4Network object represents a network segment is identified by a netmask in CIDR notation. An example value for the object would be 10.255.130.0/24. These objects are provided by the Flow log and Radar adapters. There are not yet any property facts stored for ipv4Network objects, but they are linked to several other objects via relationship facts:

- memberOf: this records the ipv4 objects which have been identified in the data from the various adapters about which addresses belong to which sub-networks.
- routesTo: facts link the ipv4Network and the routes objects.

**organizationalUnit -** Organization units are an Active Directory concept used to organize the users, hosts and groups in the same Active Directory domain. They are typically used to mirror an organization's functional or business structure. Each domain can implement its own organizational unit hierarchy. The data is collected by the AD adapter and each object is identified by its SID and contains the following property facts:

- distinguishedName: This describes the hierarchical paths in Active Directory to the organizational unitand can be used to uniquely reference this object. "OU=Example OU,DC=soccrates,DC=f-secure,DC=com" is an example for this field.
- highValue : This is a Boolean field which indicates if the Active Directory adaptor has flagged that the group is of high value to an attacker. A high value target is by default any group that belongs to the domain admins, domain controllers, and several other high privilege Active Directory groups.

These objects can have the following types of relationship facts:

- memberOf : users, groups, hosts and even other organizationalUnits can be members of a particular organizationalUnit. Also, the organizationalUnit can itself be a member of both domain objects and other organizationalUnit objects.
- hasRight : The active directory adapter provides a lot of details about the rights which users and groups have with regard to organizationalUnits and these rights are stored as facts with values corresponding to the name of the right which the user / group has with regard to the organizationalUnit.

**route -** This object is used to model the network routes between ipv4network objects. The data is collected using the Route importer and route objects have the following property facts:

- metric: the priority of the route.

Routes are linked to other objects in the data model using the following facts:

- nextHop: this links the route to an ipv4 object.

- routesTo: this fact is used to identify which network segment (identified by ipv4network object) the route is directing traffic to.

**rule -** This object is used to model the firewall rules in use in the corporate network and the data is collected by the Route importer and identified using the rule name. The rule specifies the type of traffic and the **ports** that this type of traffic uses.
These objects have the following property facts:

- action: The action to be performed on the traffic, e.g., ALLOW / DENY.
- applicationProtocol: The name of the application protocol, such as HTTP.
- destinationNet: The destination network for the traffic to be filtered. (Can be Any)
- highDestinationPort: The highest destination port the rule applies to.
- highSourcePort: The highest source port the rule applies to.
- lowDestinationPort: The lowest destination port number the rule applies to. for the destination.
- lowSourcePort: The lowest source port number the rule applies to.
- protocol: The name of the network protocol, e.g., TCP
- protocolState: The state of the protocol as described in RFC793, e.g. SYN-RECEIVED
- sourceNet: The target network for the traffic. (Can be Any)

Rules can have the following fact relationships to other objects:

- componentOf: Which ruleset a particular rule belongs to.
- follows: The next rule which will be applied after this rule.

**Ruleset -** This object is used to model collections of rules and limits the network traffic which is passed through a set of routes. The data is collected by the Route importer and is identified using a unique ruleset name. These objects have the following property facts:

- defaultPolicy: This specifies what the firewall should do if the message does not fit to any route.
- matchStrategy: This field specifies the strategy employed for the ruleset to find the matching rules for the input data. Example values can be “firstmatch”, “bestmatch” and “lastmatch”.

Rulesets have the following fact relationships to other objects:

- componentOf: Which ruleset a particular rule belongs to.
- configuredOn: Which host object a particular ruleset is configured to run on.

**software -** This object represents software applications, packages and libraries which are identified by the adaptors as running on hosts in the network. The current approach used is to use the software package name and version number as the unique identifier for these objects. As explained later, ideally, we should be using the Common Platform Enumeration (CPE) for such identifiers, but Radar does not supply these currently.
Software objects have the following property facts:

- category: A simple string to provide some context to the software description, e.g., “os”.

Software objects have the following fact relationships to other objects:

- affects: Software objects can be affected by named vulnerability objects. This data is sourced by the Radar Adaptor.

- listensOn: This relationship fact is used to model that a software application is running on a certain ipv4 object.
- runs: This indicates the host object which has been identified by the adaptors as running the software in question.

**vulnerability -** The Radar adapter provides data to the IMC about vulnerabilities seen on hosts during scans of the networks. The vulnerabilities are identified uniquely by their CVE IDs. There are many types of vulnerabilities and currently INFO level vulnerabilities are not processed into the IMC model. Vulnerability objects have the following property facts:

- score: This represents the CVSS score (Radar provides these scores according to the CVSS v2 scoring scheme). An example score would be 7.5 (0-10 scale, 10 meaning critical).

Vulnerability objects have the following fact relationships to other objects:

- affects: This indicates which software objects the named vulnerability impacts.

**user -** This object type is used to represent Active Directory user accounts which have been seen in the network. The data is gathered by the Active Directory adaptor. These objects are identified by their Security identifier (SID) and can have the following property facts:

- distinguishedName: This describes the hierarchical paths in Active Directory to the user account and can be used to uniquely reference this object. "CN=IIS_IUSRS,CN=Builtin,DC=soccrates,DC=f-secure,DC=com" is an example for this field.
- highValue : : This is a Boolean field which indicates if the Active Directory adaptor has flagged that the user account is of high value to an attacker. A high value target is by default any group that belongs to the domain admins, domain controllers, and several other high privilege Active Directory groups.
- isAdmin: This is set AD adapter to indicate if the user should be considered as an admin level user.

User objects also have the following fact relationships to other objects:

- adminOf: This fact provides details of which hosts the user has been declared as a local administrator.
- hasRight : The active directory adapter provides a lot of details about the rights which users and groups have regarding organizationalUnits and these rights are stored as facts with values corresponding to the name of the right which the user / group has with regard to the organizationalUnit.
- memberOf : users can be members of Active Directory organizationalUnits and groups.
- sessionTo: The active directory adapter provides information about the sessions which are active between users and hosts.

## 3.2 Model challenges

While developing the IMC model several issues became apparent, most notably the challenges in identifying vulnerabilities and software packages and how to integrate with various network devices with various proprietary configuration and logs format.

It is important in SOCCRATES project for the ADG and other components being developed to refer to both software packages and vulnerabilities using globally unique identifiers. While the ambition was to use existing standards (i.e., the CVE & CPE values) for this purpose, this turned out to be more difficult than predicted. For instance, not all vulnerabilities identified by Radar correspond to a known CVE and a vulnerability such as a remote web server configured to still allow the deprecated TLS 1.0 protocol is not covered by an existing CVE value. These are generally less serious vulnerabilities, but we continue to look for other approaches to identifying such vulnerabilities so other components can understand them. One approach here could be to use a URL which would link to the vendors' own knowledge base article about the vulnerability.

Similar issues were faced with software identification. CPE lists are published by NVD but have been found in various research (e.g., [5]] to be far from complete, so for many assets there is simply no existing CPE value. Radar itself can only produce detailed library version and publisher information when the authenticated scan feature is turned on, which is not always feasible in large networks. During the project, a new capability called RADAR Endpoint Agent was developed to make the authenticated scan easier to use and extend it to also cover Windows devices outside company network.

Another type of challenge was faced with the networking infrastructure as there are many, often vendor specific, types of firewall configuration and log formats. Checkpoint, Cisco, Amazon and a range of others all have their own formats and while the original idea was to focus solely on Cisco Netflow, the adapter built during the project supports already three different formats, e.g., nfdump, AWS VPC Flow Logs and custom json format used with Vattenfall.

## 3.3 Interoperability through standards

Selected information security standards have been applied in IMC to enable external data sources to be used, but the IMC data model, i.e., graph schema itself, is not directly derived from any existing standard.

IMC uses CVSS[3], CVE[4] and CPE[5] to enable integration with external vulnerability databases. The use of these de-facto standards allows IMC adapters to be built against most vulnerability scanning tools, as most of them further integrate with the same vulnerability databases.

- CVSS, the Common Vulnerability Scoring System, provides a standard way to describe characteristics of a vulnerability and a numerical score for its severity. IMC uses CVSS scores to rate the vulnerabilities.
- CVE, Common Vulnerabilities and Exposures, is a de-facto identifier for vulnerabilities and IMC uses CVE Entry ID as unique identifier for vulnerabilities.
- CPE, Common Platform Enumeration, provides de-facto naming scheme for software and packages and is used in IMC as unique identifier for software objects.

[3] https://www.first.org/cvss/
[4] https://cve.mitre.org/cve/
[5] https://nvd.nist.gov/products/cpe

Multiple standards with full data models were investigated during the project to provide a basis for IMC data model, but none of them were designed to be used with graph databases, making them unsuitable technically to build IMC. However, conceptual thinking of these standards can be applied to graph databases as well and while IMC is not directly built on top of any of the standards, adapters can be built to integrate with tools supporting them.

Conceptually IMC data model is quite close to draft SACM Information Model's endpoint-software instance-software component relationships and it should be quite easy to build SACM adapter to integrate with tools supporting it. Recently an update of the proposed draft Internet standard has been published [6]. Since none of the tools considered as a data source for IMC supported it, no such adapter was built.

OpenDXL Ontology[7] is a rather new project focused on the development of an open and interoperable cybersecurity messaging format to be used with the OpenDXL messaging bus. OpenDXL defines simple message formats for actions and notifications but lack definitions of the ICT infrastructure assets themselves and is not suitable in its current form to be used as a data source format.

DMTF Common Information Model (CIM [8]) provides extensive information model, defining CIM schemas like Application, Device, Network and User, using its own CIM Metamodel. CIM metamodel is used in CIM to define the CIM schemas in similar manner as ACT Object-Fact model is used to define IMC data model. CIM schema using CIM metamodel cannot be easily converted into Object-Fact model, making it unpractical to use CIM as a basis for IMC. Instead, separate IMC adapters towards tools using CIM could be built if suitable data sources would be available.

Security Content Automation Protocol (SCAP [9]) is a suite of specifications for exchanging security automation, including Standard for software identification (SWID), ISO/IEC 19770-2:2015, which is supposed to automate software inventory management and complement or replace CPEs in the vulnerability databases, but currently SWID tags are not widely supported yet neither by vulnerability databases nor software vendors.

STIX™ Standard [10] is intended for malware characterization, intrusion detection, incident response & management and digital forensics, but not to describe the ICT infrastructure itself. STIX™ Version 2.0. Part 4: Cyber Observable Objects define objects like software and user, but it does not define relationships between these objects and thus not suitable as a basis for IMC data model. However, STIX™ is used in SOCCRATES platform to exchange threat intel, but that data is not stored in IMC.

In the U.S., the National Telecommunications and Information Administration has recently published guidance on how to declare and share software inventories as a Software Bill of Materials (SBOM) [11]. The goal is to increase the transparency of software components and so identify vulnerable software

[6] https://datatracker.ietf.org/doc/draft-ietf-sacm-information-model/
[7] https://github.com/opencybersecurityalliance/opendxl-ontology
[8] https://www.dmtf.org/sites/default/files/cim/cim_schema_v2540/cim_schema_2.54.0-PDFs.zip
[9] https://nvd.nist.gov/products/swid
[10] https://oasis-open.github.io/cti-documentation/stix/intro & http://docs.oasis-open.org/cti/stix/v2.0/stix-v2.0-part4-cyber-observable-objects.pdf
[11] https://en.wikipedia.org/wiki/Software_bill_of_materials & https://www.ntia.gov/sbom

more effectively. This is clearly a promising development, but as it is still in development, we did not include it in our IMC data model yet.

In general, there is no existing suitable single standard which would be widely supported and would contain all necessary data objects to serve as a IMC data model. Furthermore, different data sources provide a different viewpoint to the infrastructure meaning also different set of data and if the data model would be too static, then the model would not work with all data sources. For example, if CPE would be mandatory field for software object, then vulnerabilities and identified software packages, which cannot be mapped to any CPE, could not be stored to the database. Graph database was found to be flexible enough to add new objects and relationships (facts) as needed when new data sources were added and while the resulting model may not be conceptually perfect, it allows the available data to be stored.

Another key aspect of the data model is the abstraction level. In context of IMC data objects, the Active Directory data was most problematic, e.g., whether to model user groups, organization units and domains simply as generic groups, or model them as separate objects. These different types of groups have different conceptual meaning in Active Directory together with different attributes and relationships, so in case these groups would have been generalized as generic group, then IMC would have needed to have group type specific validation and processing logic for the facts. In practice this would have moved the data model details from the data model itself to the IMC program code without other benefits than more simplified data model. The data model is not too complex yet and thus the decision was to model these groups separately.

As a lesson learned from the project, the benefits of standards in this context are in interoperability between systems, not in the internal database design of a system.

# 4 Attack Defence Graph

## 4.1 MAL (Meta-Attack Language)

In the ADG module of the SOCCRATES project we use the Meta-Attack Language (MAL) based attack simulations on system architecture models to support the identification of relevant security properties of a system, and the collection of this security related information [1], [2]. These approaches facilitate a model of the organization and simulate cyber-attacks in order to identify the greatest weaknesses. This can be imagined as the execution of a large number of parallel virtual penetration tests. Such an attack simulation tool enables the security assessor to focus on the collection of the information about the system itself required for the simulations.

As the previous approaches rely on a static implementation, we developed MAL [3]. This is a framework for Domain Specific Languages (DSLs) that defines which information about a system is required and specifies the generic attack logic. Since MAL is a meta language (i.e. the set of rules that should be used to create a new DSL), no particular domain of interest is represented, but it provides a set of techniques to create such a language. Before we explain such concrete languages, we present the basic building blocks of MAL.

Similar to UML [4], MAL describes languages in terms of classes and associations defining how object models are instantiated. In MAL, classes are called assets. Assets have associations between each other which are used for the creation of the model. As opposed to UML, MAL does not allow assets (classes) to contain any type of property instead it encodes attack graph logic. Assets contain attack steps, which represent the actual attacks/threats that can happen on them. Inheritance between assets is also possible and each child asset inherits all the attack steps of the parent asset. An attack step can be connected with one or more following attack steps to create an attack path. Those are used to create attack graphs which are facilitated when running simulations. Finally, assets can be organized into categories and probability distributions can be assigned to attack steps in order to represent the effort needed to complete the related attack step.

An example DSL (exampleLang) written using MAL syntax is given below. (Source: https://github.com/mal-lang/exampleLang).

In the exampleLang System is depicted as the collection of four types of assets viz. Network, Host, User, and Password. Different potential attack steps on those assets are also specified. This attack steps can be of two types OR (|) attack steps and AND (&) attack steps. An OR attack step can be reached if any of the attack steps which refers to A is reached. Whereas, an AND attack step can be reached only if all of the attack steps which refer to A are reached. Attack steps may have probability distributions set to express their uncertainties. Leads to (->) symbol indicates the successful compromise of one attack step allows the attacker to consequently compromise further attack steps.

After the assets and attack steps, exampleLang describes the associations in it. As said before, association is analogous to UML class diagram associations. It denotes a potential bidirectional structural relationship (physical or logical) between two assets. The syntax is given below,

Asset1 [role1] <multiplicity1> <-- AssociationName --> <multiplicity2> [role2] Asset2

The multiplicity of a composite association can be [1], [1..*] or [*] on the composite end. Role names determine the directional references to other assets used in code. For example, Asset1 references Asset2 as *role2.A*, where A is an attack step. Conversely, Asset2 references Asset1 as *role1.A*.

**exampleLang**

```
category System {
        asset Network {
                | access
                 -> hosts.connect
        }
        asset Host {
                | connect
                 -> access
                | authenticate
                 -> access
                | guessPassword
                 -> guessedPassword
                | guessedPassword [Exponential(0.02)]
                 -> authenticate
                & access
        }
        asset User {
                | attemptPhishing
                 -> phish
                | phish [Exponential(0.1)]
                 -> passwords.obtain
        }
        asset Password {
                | obtain
                 -> host.authenticate
        }
}

associations {
        Network [networks] * <-- NetworkAccess --> * [hosts] Host
        Host [host] 1 <-- Credentials --> * [passwords] Password
        User [user] 1 <-- Credentials --> * [passwords] Password
}
```

Defence is another important feature of MAL, which is missing in the exampleLang described above. As opposed to attack steps defences are boolean. A defence step represents the countermeasure of an attack step (e.g., passwordBruteForce attack step can be defended by twoFactorAuthentications defence step). While declaring an asset, it is possible to either enable or disable a defence step by setting the defence step to either TRUE or FALSE. It is also possible to assign a Bernoulli distribution to govern the activation. Example of defence step given below describes If data is encrypted, then it cannot be read.

```
# encrypted
        -> read
```

There are many other more advanced features of MAL which are available at https://github.com/mal-lang/mal-documentation/wiki/MAL-Syntax

**MITRE ATT&CK Mapping**

The MAL specifications are not bound to any specific framework or threat model by design as they are typically not well adjusted to the attack defence graph logic of MAL. However, taxonomies of adversary actions like MITRE ATT&CK are quickly becoming the de facto standard when describing and categorizing hacker attacks and attacker actions. To be able to provide a mapping to the MITRE ATT&CK taxonomy as well as other frameworks and threats models MAL implements a tagging system where attack steps and defences can be mapped to known taxonomies without compromising the strict syntax of MAL specifications. These mapping are then presented to analysts as an alternative description to the attack steps defined in MAL. A single attack step can be mapped to multiple taxonomies and frameworks at the same time.

**Example:**

```
| getRoleCredentials

    user info: "The attacker can get access to the permissions of the attached instance profile."

    mitigation info: "Follow the standard security advice of granting least privilege.”

    stride info: "Information Disclosure"

    mitre info: "T1552.005 Unsecured Credentials: Cloud Instance Metadata API"

        -> instanceProfile.assumeRoles
```

## 4.2 coreLang

As illustrated before, MAL provides the basics to create a threat modeling language from scratch. However, many languages created with MAL, share a common set of concepts. To reduce unnecessary redundant work, we developed coreLang that is comprised of assets such as software, data, and networks, that we envision will be present in (almost) all domains. As the project pilot features a standard enterprise environment, corelang will be used as is, but it can be noted that it can also serve as a starting point to model more domain specific languages.

Figure 4 given below, depicts an example coreLang model that showcases a simple attack involving social engineering. The securiCAD tool was used to create the visualization. The attacker targets Mary, a regular user that has access to her workstation (*Corp. WS*) that is a part of the corporate network (*Corp. NW*), with a social engineering operation in order to get access to the sensitive data (*Sens. Data*) stored on the corporate server (*Corp. Serv.*) that Mary also has access to. The attack defence graph generated by simulating the attacker's reachability for the *CredentialTheft* step on the Credentials asset representing Mary's username and password("UN&PWD") is given in Figure 5. How an attacker (in

the red circle) can reach an attack step (in blue, *CredentialTheft* in this example) with the help of intermediate attack steps (in grey, e. g. *PhishUser*) is shown. Defence steps (in green, e. g. *SecurityAwareness*), which can be used to counter the attack steps, are also given in the attack graph. Red arrows depict attack steps with a high degree of certainty, while yellow arrows represent attack steps that have a smaller likelihood of succeeding. These attack defence graphs can become very complex for large models (in Figure 6 a slightly complex attack defence graph presenting attacker's reachability to *read* attack step on the "sensitive data" is shown). We will return to the example in Figure 4 in more detail after we have given a description of the various components that comprise coreLang, for now it should just serve as an illustration of a typical scenario.

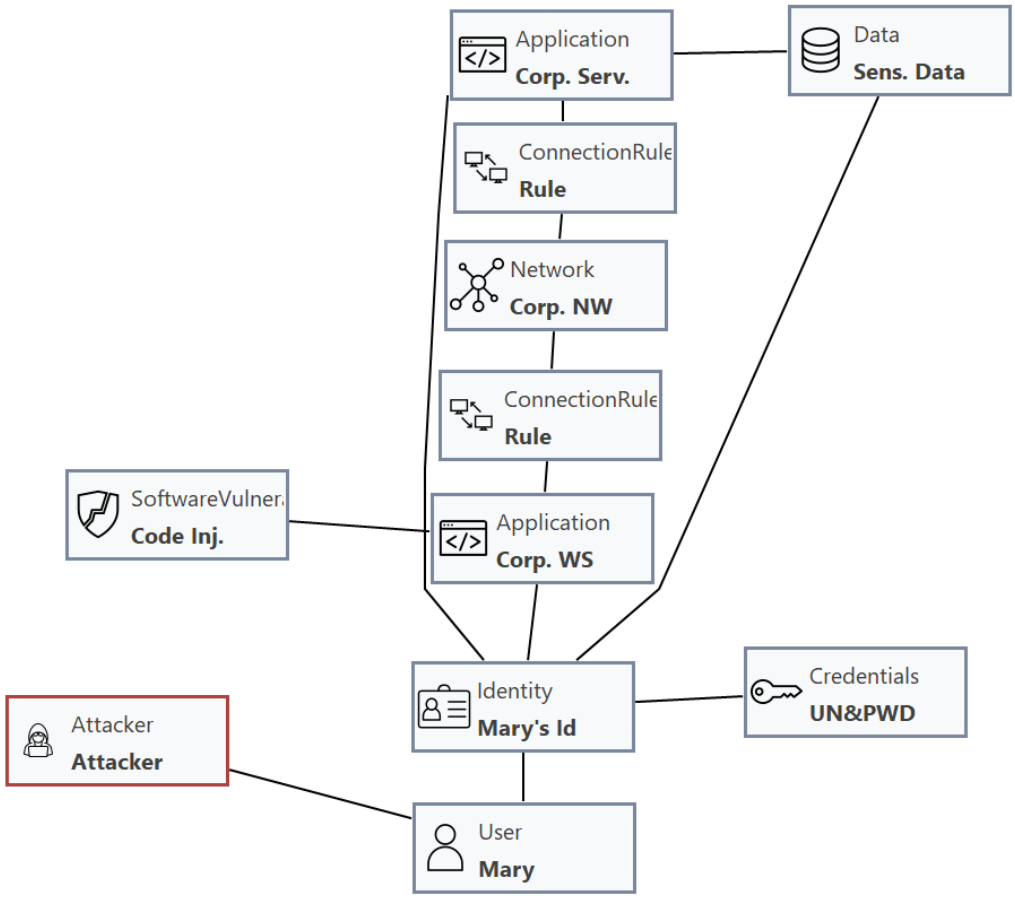


**Figure 4 Example coreLang model**

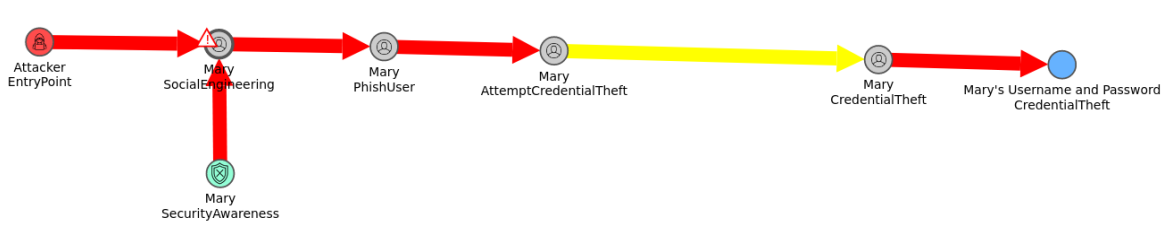


**Figure 5 Attack Defence Graph**

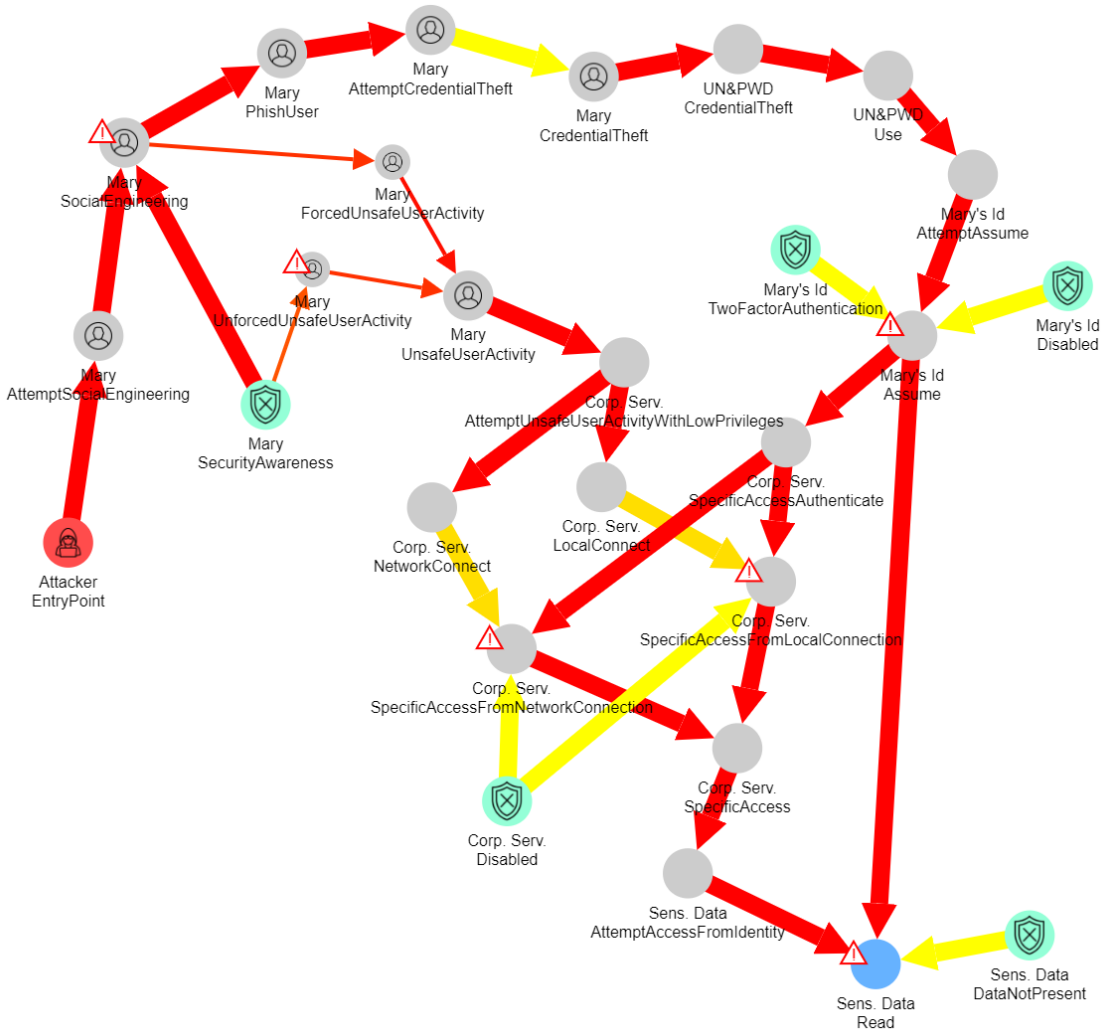


**Figure 6 Complex attack defence graph**

Figure 7 presents an overview of the assets that currently exist in coreLang and the relationships between them. The six different main categories of coreLang, Compute resources, Data resources, IAM (Identity and Access Management), User, Networking and Vulnerability, will be described in the following paragraphs. We will explain the reasoning behind the most commonly used assets and their salient attack steps and defences. Where appropriate we will relate them to the example mentioned above.

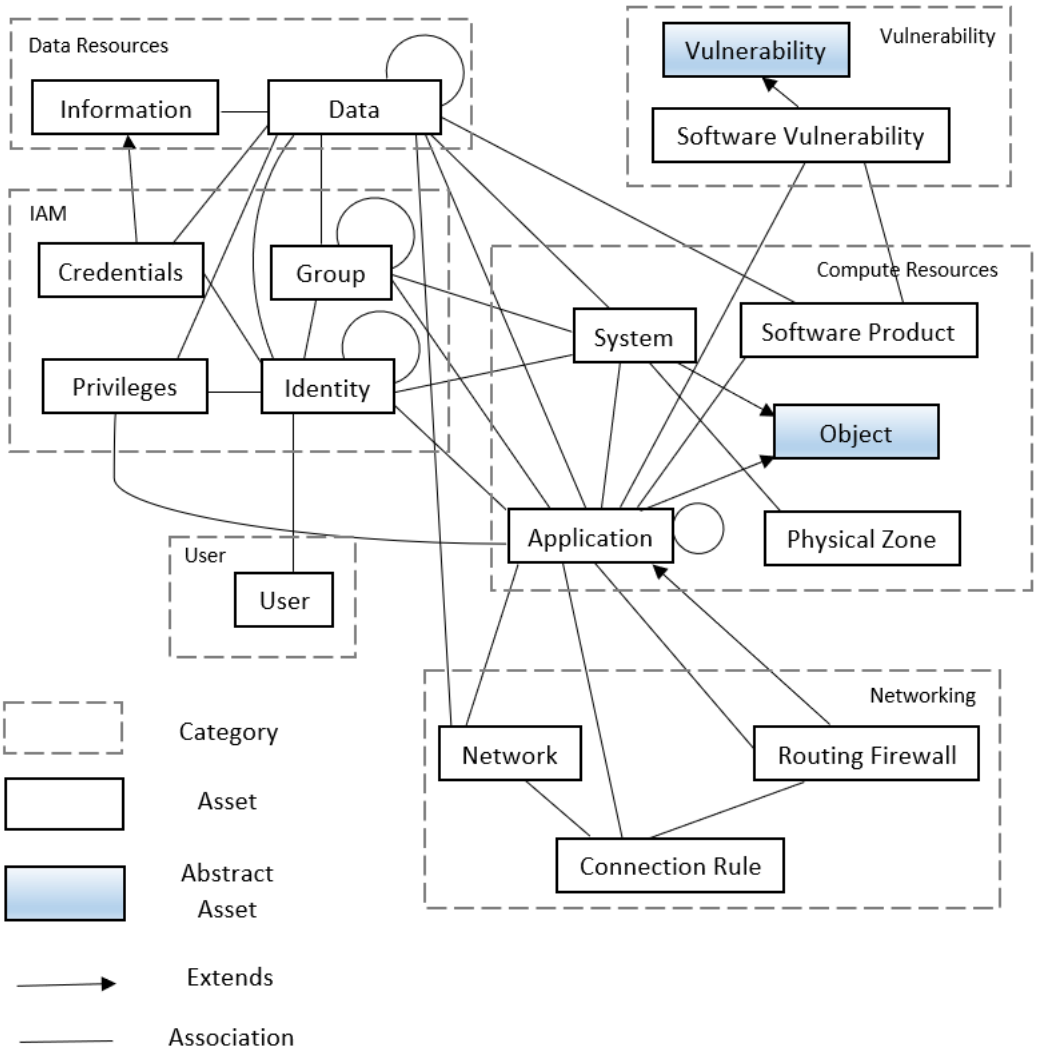


**Figure 7 coreLang assets overview**

**ComputeResources**

The Application asset represents any code being executed, this is a very broad range that includes everything from an operating system, a subcomponent of a particular piece of software, or even a group of programs that together serve a specific purpose. As such, it is one of the most commonly used assets in coreLang and, as seen in Figure 7, interacts with most of the other assets in the language. Two types of attack steps on the Application asset are especially important to the design of coreLang. First, we have attack steps that describe how an attacker can reach a particular Application, either *networkConnect* if reached via network components or *localConnect* if reached physically or through the malicious actions of someone with access to the Application.

Second, the *authenticate* and *specificAccessAuthenticate* attack steps represent the attacker having attained high or low privileges on the Application, respectively. The reason why these two categories are important is because in order to successfully access an Application an attacker must both be able to reach it and authenticate on it. The level of privileges used for authenticating will determine the

impact of the intrusion, either *specificAccess* or *fullAccess*. The specific design of privileges will be discussed in more detail in the IAM category later on.

In our previous example the attacker is able to trick the User into performing unsafe actions that it then exploits for its operations. As such, the attacker is able to reach the "Corporate Workstation" (*Corp. WS*) Application via both *NetworkConnect* and *LocalConnect* since the unsafe actions taken by the user could be of either type. In addition to the reachability the attacker is also able to make use of user Mary's privileges which are low on the "Corporate Workstation". This means that they have achieved *specificAccess*, they are able to perform some deleterious actions, but are not in full control of the Application.

The other frequently used asset in this category is the System asset. It is used to represent the hardware that applications are run on. For scenarios where the hardware components of an environment are relevant, such as the attacker getting direct or indirect access to the physical environment or supply chain attacks, the System asset should be employed. For circumstances that do not hinge on hardware aspects they can be omitted to maintain simplicity.

**DataResources**

As the name would suggest, the Data asset is the relevant asset in the Data Resources category. The Data represents data that are either in transit over network components or stored in particular applications. Four important attack steps on the Data asset are *read*, *write*, *deny*, and *delete*. The concept behind each of these should be rather self-explanatory. The *deny* and *delete* attack steps are seen as generally equivalent, the *deny* step is just used to explicitly depict a denial of service. The *authenticated* defence protects the data from being overwritten as the authentication would signal that the contents had been tampered with. As we will see in the IAM category, encryption is also provided to Data via the Credentials asset. In our example, the attacker sought out to access the "Sensitive Data" (*Sens. Data*) stored on the "Corporate Server" (*Corp. Serv.*) application. Because the attacker is able to access the "Corporate Server" it can try to *read*, *write*, and *delete* the "Sensitive Data".

IAM

The Identity asset serves as a collection of access roles available to a particular user. The roles are depicted via associations in coreLang, the code snippet below outline some of the most typically used sets of associations, with Application and Data assets.

```
Identity [highPrivAppIds] * <-- HighPrivilegeApplicationAccess --> * [highPrivApps] Application

Identity [lowPrivAppIds] * <-- LowPrivilegeApplicationAccess --> * [lowPrivApps] Application

Identity [readingIds] * <-- ReadPrivileges --> * [readPrivData] Data

Identity [writingIds] * <-- WritePrivileges --> * [writePrivData] Data

Identity [deletingIds] * <-- DeletePrivileges --> * [deletePrivData] Data
```

As can be seen in the snippet, the privileges for applications can be either *high* or *low* and their impact was briefly discussed when addressing the Application asset. For Data the privileges match three of the operations mentioned, *read*, *write*, and *delete*. Our previous example includes both types of associations. The "Mary's Id" Identity asset has low privileges on both the "Corporate Workstation" and the "Corporate Server" applications which allows the attacker to attain the *specificAccess* normally associated with low privileges once it is able to reach those applications. Additionally, the "Mary's Id"

Identity asset also has *read* privileges on the "Sensitive Data" stored on the "Corporate Server" which enables the attacker to read those data once reached.

Credentials serve a dual purpose in coreLang. First, they can be used to depict the authentication credentials associated with a particular Identity. Conceptually this could either be a username and password pair, a private key, or any other means of authentication. In our example the "Mary's Id" asset has a Credentials asset called "Mary's username and password" (*UN&PWD*) that represents user Mary's username and password in the corporate authentication services. The attacker may be able to get access to these credentials via the social engineering operations it is performing that may trick Mary into disclosing her username and password. Second, Credentials are used for encryption of data. A Credentials asset associated with a Data asset in an *EncryptionCredentials* relationship would protect the data from any of undesired access. If the attacker is able to somehow access those credentials then they are able to decrypt the data access it normally.

Group assets are typically used to associate identities together and structure hierarchies in access management. Privileges assets refer to specific subsets of access roles available to an Identity and are primarily used in speculative models to reason about the likelihood of superfluous or misconfigured permissions given to particular identities.

**User**

The eponymous asset User is the only one in this category. It primarily serves as an attack surface for social engineering attacks. An attacker triggering the *socialEngineering* attack step leads to phishing for credentials associated with identities belonging to the user and inducing the user to perform unsafe actions, for example. These social engineering attacks propagate via the User - Identity associations and the associations between the Identity asset and other assets that it has privileges for, some of which, Application and Data, have been described in the IAM discussions.

The *securityAwareness* defence on the User asset reduces the likelihood that the attacker can successfully deceive the user into performing the unsafe activities entailed in the *socialEngineering* attack. In our example the User "Mary" is assumed to have some basic cyber security training and therefore has a *securityAwareness* defence value of 0.4 which means that the social engineering activities of the attacker are somewhat hindered, but they can still succeed over longer periods of time with some effort4. From there the attacker can steal the Credentials "Mary's Username and Password" and cause unsafe activity through the Identity "Mary's Id" on the "Corporate Workstation" application from where the attack can expand to the "Corporate Server".

**Networking**

The Networking assets represent the hardware and software logic used to connect applications to one another. First, the Network asset represents all of the layers of the OSI stack involved in transmitting information between applications. Two attack steps exemplify the conceptual design behind the Network asset, *eavesDrop* and *manInTheMiddle.* The *eavesdrop* attack allows an intruder to read the information available over the network, while *manInTheMiddle* enables both reading and writing of the information. Defences exist for both of these attacks that limit their impacts, but the design assumes that the defences are bypassable, albeit with considerable effort.

Second, the ConnectionRule asset is used to depict individual or collections of firewall rules. These can be deployed between applications, between networks, or, most commonly, between a network and an application. The existence of a ConnectionRule between two entities suggests that there are firewall rules that allow that particular traffic to transit.

In addition to the specific attack steps mentioned above, Network and Connection Rule assets are used to simulate an attacker being able to reach Applications, Systems, and Data, over the network. Both the Network - ConnectionRule and ConnectionRule - Application associations are configurable as inbound, outbound, or bidirectional. This influences the extent of the attacker's reach over networking assets.

In our example model, the attacker is able to reach the "Corporate Server" from the "Corporate Workstation" by going through the "Corporate Network" (*Corp. NW*) and its two associated ConnectionRules. This provides the attacker with a *networkConnect* on the "Corporate Server", which together with the credentials phished from the user, "Mary's Username and Password", allows them to attain *specificAccess* and try to read the "Sensitive Data".

**Vulnerability**

The basic idea of creating a MAL related language is to provide a set of already defined attack steps to the modeller. Currently only software vulnerabilities are implemented in coreLang. A SoftwareVulnerability asset can be associated with either an Application or a SoftwareProduct. The SoftwareVulnerability can be configured using the defences on it in order to adequately represent the charactersitics of the vulnerability to be depicted. The Common Vulnerability Scoring System (CVSS)[12] served as the inspiration for the choice of attributes used to describe vulnerabilities. These properties allow the modeller to both restrict the circumstances under which the vulnerability can be exploited and define the impact of the vulnerability on the associated asset. For example, a vulnerability can require physical, local, or network access. Privileges can also be set as requirements for successful exploitation of the vulnerability. The impact of the exploit is configured based on confidentiality, integrity, and availability scores, just as in the CVSS documentation. If a SoftwareVulnerability is associated with a SoftwareProduct that suggests that all of the Applications connected to the SoftwareProduct asset are susceptible to the vulnerability.

A SoftwareVulnerability called "Browser Code Injection" (*Code Inj.*) is present in our example model which represents a vulnerability in the browser that allows for malicious code to be executed. Figure 8 shows the CVSS 3.1 score of the vulnerability in the CVSS calculcator[13]. This score is translated into the following three defences being set to 1.0 (fully enabled) on the "Browser Code Injection" SoftwareVulnerability asset: NetworkAccessRequired, LowPrivilegesRequired, and UserInteractionRequired. The attacker is able to fulfill all three of these requirements through their social engineering operations which is why they are able to escalate to *fullAccess* on the "Corporate Workstation".

[12] https://www.first.org/cvss/

[13] https://www.first.org/cvss/calculator/3.1

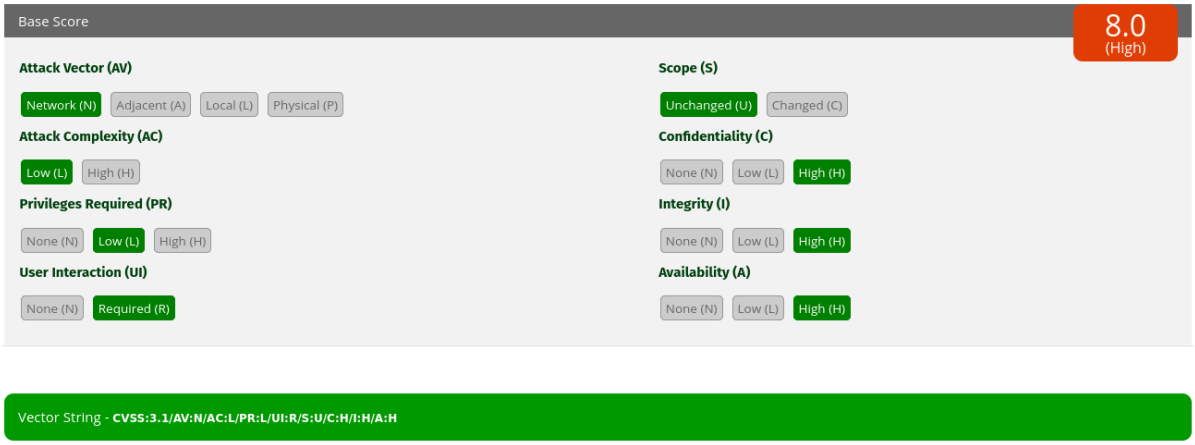


**Figure 8 "Browser Code Injection" SoftwareVulnerability CVSS 3.1 Score**

The coreLang repository is publicly available via the following link:
https://github.com/mal-lang/coreLang.

# 5 Reference


[1] Ekstedt, Mathias, Pontus Johnson, Robert Lagerström, Dan Gorton, Joakim Nydren, & Khurram Shahzad 2015. securiCAD by foreseeti: A CAD tool for enterprise cyber security management. In Enterprise Distributed Object Computing Workshop (EDOCW), 2015 IEEE 19th International, pages 152–155. IEEE.

[2] Holm, H., K. Shahzad, M. Buschle, & M. Ekstedt 2015. $P^2$CySeMoL: Predictive, Probabilistic Cyber Security Modeling Language. IEEE Transactions on Dependable and Secure Computing, 12(6):626–639.

[3] Johnson, Pontus, Robert Lagerström, & Mathias Ekstedt 2018. A Meta Language for Threat Modeling and Attack Simulations. In Proceedings of the 13th International Conference on Availability, Reliability and Security, page 38:1-10. ACM.

[4] Unified Modeling Language (UML) [online] Available: https://www.omg.org/spec/UML/2.5.https://www.omg.org/spec/UML

[5] Takahashi, Takeshi, and Daisuke Inoue. "Generating software identifier dictionaries from Vulnerability Database." 2016 14th Annual Conference on Privacy, Security and Trust (PST). IEEE, 2016.

# 6 Abbreviations

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

| Acronym | Description |
|---|---|
| ACT | semi-Automated Cyber Threat intelligence |
| AD | Active Directory |
| 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 |
| CIA | Confidentiality, Integrity, Availability |
| CIM | Common Information Model |
| CMDB | Configuration Management Database |
| CSIRT | Computer Security Incident Response Team |
| CoA | Course of Action |
| COA | Course of Action generator |
| CPE | Common Platform Enumeration |
| CTI | Cyber Threat Intelligence |
| CVE | Common Vulnerabilities and Exposures |
| CVSS | Common Vulnerability Scoring System |
| DC | DataCentre |
| DGA | Domain Generated Algorithm |
| DMTF | Distributed Management Task Force |
| DNS | Domain Name System |
| DSL | Domain Specific Language |
| EDR | Endpoint Detection and Response |
| ELK | Elasticsearch/Logstash/Kibana |
| FQDN | Fully Qualified Domain Name |
| FRS | Foreseeti |
| FSC | F-secure |
| IAM | Identity and Access Management |
| 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 |
| OT | Operational Technology |
| OS | Operating System |
| RORI | Return on Response Investment |
| SACM | Security Automation and Continuous Monitoring |
| SBOM | Software bill of materials |
| SCAP | Security Content Automation Protocol |
| SDN | Software Defined Network |
| SHS | Shadowserver |
| SID | Security Identifier |
| 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 |
| STIX™ | Structured Threat Information eXpression |
| SWID | Software identification |
| 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 |
| UML | Unified Modeling Language |
| VLAN | Virtual LAN |
| VM | Virtual Machine |
| VPC | Virtual Private Cloud |
| VTF | Vattenfall |

# 7 Appendix

Three schema files are needed to create IMC database using the ACT platform.

- Fact types are defined in fact-types-imc.json
- Meta facts are defied in meta-fact-types-imc.json
- Objects are defined in object-types-imc.json

## 7.1 fact-types-imc.json

```
[
 {
  "name": "action",
  "objectBindings": [
   {
    "sourceObjectType": "rule"
   }
  ]
 },
 {
  "name": "adminOf",
  "objectBindings": [
   {
    "destinationObjectType": "host",
    "sourceObjectType": ["group", "user"]
   }
  ]
 },
 {
  "name": "affects",
  "objectBindings": [
   {
    "destinationObjectType": "software",
    "sourceObjectType": "vulnerability"
   }
  ]
 },
 {
  "name": "applicationProtocol",
  "objectBindings": [
   {
    "sourceObjectType": "rule"
   }
  ]
 },
 {
```

```
  "name": "assignedTo",
  "objectBindings": [
   {
    "destinationObjectType": "host",
    "sourceObjectType": "ipv4"
   }
  ]
 },
 {
  "name": "availabilityImpact",
  "objectBindings": [
   {
    "sourceObjectType": [
     "host"
    ]
   }
  ],
  "validator": "^[1-5]$"
 },
 {
  "name": "bytesIn",
  "objectBindings": [
   {
    "sourceObjectType": "dataFlow"
   }
  ]
 },
 {
  "name": "bytesOut",
  "objectBindings": [
   {
    "sourceObjectType": "dataFlow"
   }
  ]
 },
 {
  "name": "category",
  "objectBindings": [
   {
    "sourceObjectType": [
     "host",
     "software",
     "ipv4"
    ]
   }
  ]
 },
```

```
{
 "name": "componentOf",
 "objectBindings": [
  {
   "destinationObjectType": "ruleSet",
   "sourceObjectType": "rule"
  }
 ]
},
{
 "name": "confidentialityImpact",
 "objectBindings": [
  {
   "sourceObjectType": [
    "host"
   ]
  }
 ],
 "validator": "^[1-5]$"
},
{
 "name": "configuredOn",
 "objectBindings": [
  {
   "destinationObjectType": "host",
   "sourceObjectType": "ruleSet"
  }
 ]
},
{
 "name": "defaultPolicy",
 "objectBindings": [
  {
   "sourceObjectType": "ruleSet"
  }
 ]
},
{
 "name": "destination",
 "objectBindings": [
  {
   "destinationObjectType": "ipv4",
   "sourceObjectType": "dataFlow"
  }
 ]
},
{
```

```
  "name": "destinationNet",
  "objectBindings": [
   {
    "sourceObjectType": "rule"
   }
  ]
 },
 {
  "name": "distinguishedName",
  "objectBindings": [
   {
    "sourceObjectType": [
     "domain",
     "group",
     "host",
     "organizationalUnit",
     "user"
    ]
   }
  ]
 },
 {
  "name": "follows",
  "objectBindings": [
   {
    "destinationObjectType": "rule",
    "sourceObjectType": "rule"
   }
  ]
 },
 {
  "name": "hasRight",
  "objectBindings": [
   {
    "destinationObjectType": [
     "domain",
     "group",
     "host",
     "organizationalUnit",
     "user"
    ],
    "sourceObjectType": ["host", "group", "user"]
   }
  ]
 },
 {
  "name": "highDestinationPort",
```

```
  "objectBindings": [
   {
    "sourceObjectType": "rule"
   }
  ]
 },
 {
  "name": "highSourcePort",
  "objectBindings": [
   {
    "sourceObjectType": "rule"
   }
  ]
 },
 {
  "name": "highValue",
  "objectBindings": [
   {
    "sourceObjectType": [
     "domain",
     "group",
     "host",
     "organizationalUnit",
     "user"
    ]
   }
  ]
 },
 {
  "name": "integrityImpact",
  "objectBindings": [
   {
     "sourceObjectType": [
          "host"
        ]
   }
  ],
  "validator": "^[1-5]$"
 },
 {
  "name": "interface",
  "objectBindings": [
   {
    "destinationObjectType": "ipv4",
    "sourceObjectType": "route"
   }
  ]
```

```
},
{
 "name": "isAdmin",
 "objectBindings": [
  {
     "sourceObjectType": [
           "group",
           "user"
        ]
  }
 ]
},
{
 "name": "listensOn",
 "objectBindings": [
  {
   "destinationObjectType": "ipv4",
   "sourceObjectType": "software"
  }
 ]
},
{
 "name": "lowDestinationPort",
 "objectBindings": [
  {
   "sourceObjectType": "rule"
  }
 ]
},
{
 "name": "lowSourcePort",
 "objectBindings": [
  {
   "sourceObjectType": "rule"
  }
 ]
},
{
 "name": "matchStrategy",
 "objectBindings": [
  {
   "sourceObjectType": "ruleSet"
  }
 ]
},
{
 "name": "memberOf",
```

```
  "objectBindings": [
   {
    "destinationObjectType": [
     "ipv4Network",
     "group",
     "organizationalUnit",
     "domain"
    ],
    "sourceObjectType": [
     "host",
     "ipv4",
     "user",
     "group",
     "organizationalUnit"
    ]
   }
  ]
 },
 {
  "name": "metric",
  "objectBindings": [
   {
    "sourceObjectType": "route"
   }
  ]
 },
 {
  "name": "name",
  "objectBindings": [
   {
    "sourceObjectType": "host"
   }
  ]
 },
 {
  "name": "nextHop",
  "objectBindings": [
   {
    "destinationObjectType": "ipv4",
    "sourceObjectType": "route"
   }
  ]
 },
 {
  "name": "protocol",
  "objectBindings": [
   {
```

```
      "sourceObjectType": "rule"
    }
   ]
  },
  {
   "name": "protocolState",
   "objectBindings": [
    {
     "sourceObjectType": "rule"
    }
   ]
  },
  {
   "name": "routesTo",
   "objectBindings": [
    {
     "destinationObjectType": "ipv4Network",
     "sourceObjectType": "route"
    }
   ]
  },
  {
   "name": "runs",
   "objectBindings": [
    {
     "destinationObjectType": "software",
     "sourceObjectType": "host"
    }
   ]
  },
  {
   "name": "score",
   "objectBindings": [
    {
     "sourceObjectType": "vulnerability"
    }
   ]
  },
  {
   "name": "sessionTo",
   "objectBindings": [
    {
     "destinationObjectType": "host",
     "sourceObjectType": "user"
    }
   ]
  },
```

```
 {
  "name": "source",
  "objectBindings": [
   {
    "destinationObjectType": "ipv4",
    "sourceObjectType": "dataFlow"
   }
  ]
 },
 {
  "name": "sourceNet",
  "objectBindings": [
   {
    "sourceObjectType": "rule"
   }
  ]
 },
 {
  "name": "trusts",
  "objectBindings": [
   {
    "destinationObjectType": "domain",
    "sourceObjectType": "domain"
   }
  ]
 }
]
```


## 7.2 meta-fact-types-imc.json


```
[
 {
  "name": "observationTime"
 },
 {
  "name": "isInherited"
 }
]
```


## 7.3 object-types-imc.json


```
[
 {
  "name": "dataFlow"
 },
 {
```

```
  "name": "domain"
 },
 {
  "name": "group"
 },
 {
  "name": "host"
 },
 {
  "name": "ipv4",
  "validator": "^((?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?))|(\\[placeholder\\[[a-z0-9]{64}\\]\\])$"
 },
 {
  "name": "ipv4Network",
  "validator": "^((?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)/([0-9]|[1-2][0-9]|3[0-2]))|(\\[placeholder\\[[a-z0-9]{64}\\]\\])$"
 },
 {
  "name": "organizationalUnit"
 },
 {
  "name": "route"
 },
 {
  "name": "rule"
 },
 {
  "name": "ruleSet"
 },
 {
  "name": "software"
 },
 {
  "name": "user"
 },
 {
  "name": "vulnerability"
 }
]
```