Rest Assured

SECURE DATA PROCESSING IN THE CLOUD

Deliverable D7.1
RestAssured Security and Privacy Engineering Methodology
Release 1.0

The research leading to these results has received funding from the European Community’s H2020 research and innovation programme under grant agreement n° 731678.
<table>
<thead>
<tr>
<th>Project Number</th>
<th>: 731678</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Title</td>
<td>: RestAssured - Secure Data Processing in the Cloud</td>
</tr>
<tr>
<td>Deliverable Type</td>
<td>: External Report</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Title of Deliverable</th>
<th>: RestAssured Security and Privacy Engineering Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nature of Deliverable</td>
<td>: External Report</td>
</tr>
<tr>
<td>Dissemination Level</td>
<td>: Public</td>
</tr>
<tr>
<td>Contractual Delivery Date</td>
<td>: 30 April 2018</td>
</tr>
<tr>
<td>Actual Delivery Date</td>
<td>: 10 May 2018</td>
</tr>
<tr>
<td>Contributing WPs</td>
<td>: WP7</td>
</tr>
<tr>
<td>Editor(s)</td>
<td>: Mike Surridge, Toby Wilkinson</td>
</tr>
</tbody>
</table>
## Contributors

<table>
<thead>
<tr>
<th>Name</th>
<th>Organization</th>
<th>Sections</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike Surridge</td>
<td>IT Innovation</td>
<td>2</td>
</tr>
<tr>
<td>Toby Wilkinson</td>
<td>IT Innovation</td>
<td>1, 3.2, 3.3, 4.2.4, 5</td>
</tr>
<tr>
<td>Stefanie Wiegand</td>
<td>IT Innovation</td>
<td>3.2, 3.4</td>
</tr>
<tr>
<td>Ludger Goekte</td>
<td>UDE</td>
<td>3.1, 3.3, 4.1</td>
</tr>
<tr>
<td>Nazila Gol Mohammadi</td>
<td>UDE</td>
<td>3.1, 3.3</td>
</tr>
</tbody>
</table>
## Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1</td>
<td>05.03.2018</td>
<td>Initial version</td>
</tr>
<tr>
<td>V0.2</td>
<td>13.04.2018</td>
<td>Draft for internal review</td>
</tr>
<tr>
<td>V0.3</td>
<td>23.04.2018</td>
<td>Draft for cross-WP review</td>
</tr>
<tr>
<td>V1.0</td>
<td>09.05.2018</td>
<td>Final version</td>
</tr>
</tbody>
</table>
Contents

List of Figures ........................................................................................................................................... 6
List of Tables .................................................................................................................................................. 7
1 Introduction ............................................................................................................................................. 8
2 Risk Assessment Methodology ............................................................................................................... 10
  2.1 A Standards Based Approach ........................................................................................................... 10
  2.2 System Assets and Structure Modelling ............................................................................................ 11
  2.3 Risk Levels and Adaptation ............................................................................................................... 12
  2.4 Summary ........................................................................................................................................... 14
3 Threat Models .......................................................................................................................................... 15
  3.1 Modelling the System Context .......................................................................................................... 15
    3.1.1 General context definition ........................................................................................................... 15
    3.1.2 Context definition by using the RestAssured Cloud System Analysis Pattern ......................... 15
  3.2 Detailed System Modelling ................................................................................................................ 25
    3.2.1 Graph Based System Modelling ................................................................................................. 25
    3.2.2 Threat Patterns ........................................................................................................................... 29
    3.2.3 Root Causes and Secondary Effects .......................................................................................... 32
    3.2.4 Threat Pattern Coverage ........................................................................................................... 34
    3.2.5 Risk Calculation .......................................................................................................................... 35
  3.3 Mapping from ReAs-CSAP to Secure System Modeller .................................................................. 39
4 Modelling Tools ....................................................................................................................................... 41
  4.1 CSAP-tool ............................................................................................................................................ 41
    4.1.1 The Designer-editor .................................................................................................................... 41
    4.1.2 The User-editor ............................................................................................................................ 44
  4.2 System Security Modeller .................................................................................................................. 47
    4.2.1 Architecture .................................................................................................................................. 47
    4.2.2 User Interface ............................................................................................................................... 47
    4.2.3 RestAssured Extensions .............................................................................................................. 50
    4.2.4 Workflow ..................................................................................................................................... 51
5 Conclusion .............................................................................................................................................. 53
List of Figures

Figure 2.1: Information risk management system - a standardised approach ........................................... 11
Figure 2.2: Run-time risk assessment in RestAssured ................................................................................. 13
Figure 3.1: Modified RestAssured-Cloud System Analysis Pattern (ReAs-CSAP) ........................................ 16
Figure 3.2: Instantiation of the ReAs-CSAP for the Ami volunteer service use case ................................. 23
Figure 3.3: Instantiation of the ReAs-CSAP for the use case Ami volunteer service with the additional component SCANT .......................................................... 24
Figure 3.4: Instantiation of the ReAs-CSAP for the use case Pay-as-you-Drive Insurance ....................... 25
Figure 3.5: The core model (simplified version) ............................................................................................. 26
Figure 3.6: The Asset class hierarchy .......................................................................................................... 28
Figure 3.7: Threat pattern for a remote attack on a host ............................................................................. 29
Figure 3.8: Ensure all security patches are applied to the host ................................................................. 31
Figure 3.9: Perform rigorous testing of the software on the host ............................................................... 31
Figure 3.10: Employ formal verification techniques on the host software ............................................... 31
Figure 3.11: Block all access to the host from Subnet .................................................................................. 32
Figure 3.12: Restrict all connections from Subnet to those initiated by the host ....................................... 32
Figure 4.1: Editor modes of the CSAP-tool ................................................................................................. 41
Figure 4.2: Creation of a user-specific CSAP on basis of an empty CSAP .................................................. 42
Figure 4.3: Addition of a Cloud Element to a user-specific CSAP ............................................................ 42
Figure 4.4: Defining the name of an added Cloud Element ........................................................................ 43
Figure 4.5: Equality regarding the name and instance type of an added Cloud Element ...................... 43
Figure 4.6: Instantiation of a Cloud Element of the instance type Data in a ReAs-CSAP ....................... 44
Figure 4.7: Representation of the instantiated Cloud Element of the instance type Data in a ReAs-CSAP .................................................................................................................... 45
Figure 4.8: Definition of Properties of the instantiated Cloud Element of the instance type Data ........... 45
Figure 4.9: Representation of the association between the instantiated Cloud Elements of the instance types Data and Sticky Policy

Figure 4.10: Specification of the type of the Cloud

Figure 4.11: The System Modeller high-level architecture

Figure 4.12: The System Modeller canvas

Figure 4.13: The Threat Editor window

Figure 4.14: The Misbehaviour Explorer

Figure 4.15: Re-calculating the risks in a system model

Figure 4.16: Updating the Impact of a Misbehaviour

Figure 4.17: Updating the level of a Trustworthiness Attribute of an Asset
List of Tables

Table 2.1: Possible responses to risk ................................................................. 10
Table 2.2: Risk levels adopted by RestAssured .................................................. 13
Table 3.1: Likelihood and trustworthiness levels .............................................. 30
Table 3.2: Trustworthiness attributes ................................................................. 30
Table 3.3: Misbehaviour impact levels ............................................................... 36
Table 3.4: Mapping misbehaviour likelihood and impact to risk level ............... 36
Table 3.5: Risk calculation symbols and definitions ......................................... 37
Table 3.6: Mapping from ReAs-CSAP-elements to System Security Modeller-elements .... 40
1 Introduction

This deliverable, D7.1, describes the outcomes of Work Package 7 in the first reporting period. WP7 comprises three tasks:

- Task 7.1: Security and privacy by design methodology,
- Task 7.2: Security and privacy threat identification tools,
- Task 7.3: Security and privacy threat mitigation tools.

Task 7.1 covers the overall RestAssured methodology and process, Task 7.2 the development of tools for design-time risk analysis of Cloud-based systems, and Task 7.3 tools for the run-time monitoring of risk levels and mitigation of threats. The run-time tools of Task 7.3 consume the design-time system models produced by the tools of Task 7.2, and work in conjunction with the Adaptation components of RestAssured to address threats to security and privacy in live systems.

In RestAssured we take the view that security and privacy must be designed into systems, but that this on its own is insufficient, and the security of systems once in production must also be continuously monitored and assessed. The security landscape constantly evolves, and new attack techniques are discovered (e.g. the recently discovered Meltdown and Spectre vulnerabilities), and this necessitates continuous assessment of a system’s security. In addition the systems themselves evolve, often in response to changes in the infrastructure the system is deployed upon. This is especially the true in Cloud-based systems where the infrastructure is frequently not owned by the system operator.

Security and privacy issues must therefore be addressed both at design-time, when a system is designed and specified, and at run-time when the system is deployed. In RestAssured we are developing tools for performing design-time risk analysis of systems, and a run-time component that operates in conjunction with the RestAssured Adaptation components to monitor the risk level of a deployed system, and assess the risk level of any change to the system proposed by Adaptation.

The major achievements in the first reporting period are:

- Development of the theoretical underpinnings for two design-time risk assessment tools.
- Development of prototypes of those two tools: CSAP and System Security Modeller.

The two design-time tools fulfill different roles within the process of system design. CSAP takes a high-level approach and models a system within its context. It employs patterns to identify the relationships between the system and the different stakeholders (Section 3.1). Here stakeholders include Data Subjects, Data Controllers, and Cloud Providers, and the emphasis is on ensuring coverage of all possible stakeholders, and their relationships to the system.

System Security Modeller takes a more detailed view of a system. Graph-based models capture the assets of a system and their relationships, and employs a catalogue of threat patterns to identify threats to a system. The risk level can be computed for the primary assets of the system.

This deliverable is organized as follows:

Chapter 2 describes the overall risk based security and privacy by design methodology.

Chapter 3 describes in detail the formal modelling of system threats to security and privacy. We employ two approaches. The first is a high-level approach that models the system within its context. This is a pattern based approach that identifies all the stakeholders that relate to the system (Data Subjects, Data Controllers, Cloud Providers etc.). The second is a graph-based approach that models the assets of a system and their relationships, and employs a catalogue of threat patterns to identify threats to
the system. This approach allows for the calculation of threat likelihoods and the corresponding risk levels.

Chapter 4 describes the CSAP and System Security Modeller tools for design-time risk assessment. The CSAP tool models the system context, and System Security Modeller provides for more detailed analysis of system risks.

Chapter 5 sums up the main points of the document and discusses conclusions as well as directions for future work.
2 Risk Assessment Methodology

2.1 A Standards Based Approach

The gold standard for managing risks in information systems is ISO/IEC 27001 [?]. This international standard is also a European Normative (EN) standard, having been endorsed by European standards organisations including CEN and CENELEC.

ISO 27001 applies at the level of an organisation, so it is not a standard whose implementation is the focus for RestAssured. However, the RestAssured risk assessment methodology should be usable by organisations that comply (or wish to comply) with ISO 27001.

ISO 27001 specifies the requirements for establishing, implementing and continually improving an information security risk management system within an organisation. It specifies 4 processes to be carried out by a compliant organisation with respect to an information system (see Figure 2.1):

1. Plan and improve the overall information risk management system;
2. Risk identification, assessment and prioritisation;
3. Implementation of risk reduction measures; and
4. Monitor incidents and evaluate risk management measures.

These four processes are meant to operate as a cycle, so when an incident occurs it is evaluated and lessons learned, as a result of which the overall risk management system can be improved. Steps (1) and (4) are concerned with organisational level aspects of the overall risk management framework. This includes the scope (i.e. which system or collection of systems being addressed), the policies for assessing risks (w.r.t. the purpose of the system), and executive and operational responsibilities.

The steps that are of most interest to RestAssured are steps (2) and (3), each of which has its own associated standard. Step (2) is to identify threats (i.e. sources of risk), and determine what security measures are needed to address them. This is covered by ISO/IEC 27005 [?], which mandates an asset based threat identification procedure, and derivation of risk levels per threat based on their impact and likelihood. Threats can then be prioritised for treatment based on the risk level, using one of the responses in Table 2.1.

<table>
<thead>
<tr>
<th>Risk response</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Risk acceptance</td>
<td>Go ahead with implementation and use of the system despite the associated threat, which is appropriate if the risk level is sufficiently low.</td>
</tr>
<tr>
<td>Risk reduction</td>
<td>Introduce security measures to reduce the risk level from the associated threat by making the threat less likely or by mitigating its impact.</td>
</tr>
<tr>
<td>Risk transfer</td>
<td>Rely on some other stakeholder to mitigate the risk - this includes relying on them to introduce security measures, or transferring liability by insuring against the risk.</td>
</tr>
<tr>
<td>Risk avoidance</td>
<td>Don’t use the system features that embody the associated threat, e.g. by replacing or disabling those features, or by just not using the system.</td>
</tr>
</tbody>
</table>

Table 2.1: Possible responses to risk

The level of risk after the introduction of risk reduction or transfer measures is sometimes called the residual risk level. The process we need is one in which risk levels are reduced via these measures to counteract threats, until the residual risk level becomes acceptable. The security requirements for the system can then be taken from the measures needed to reach that point. These measures can then be implemented in the system - step (3), and if necessary verified using methods specified by ISO/IEC 15408 [?].
At this point it is worth mentioning the relationship of RestAssured to several other standards. The 2011 revision of ISO 27005, and the 2013 revision of ISO 27001, align the overall risk management process with that of ISO 31000 [1]. ISO 31000 is broader in scope, in that it is not restricted to information security risks, but instead covers risk and opportunity management more generally. This has an obvious relationship to run-time adaptation in RestAssured, where system configuration may evolve for reasons of performance or cost optimisation, and not simply in response to information security risks.

Another relevant standard is ISO/IEC 27018 [2]. ISO 27018 augments ISO/IEC 27002 [3] with additional security controls that are applicable when processing personally identifiable information in public clouds. This is obviously relevant to RestAssured, as the RestAssured risk assessment methodology includes selection of controls to treat identified risks, and in particular, the risks addressed include threats to personally identifiable information.

2.2 System Assets and Structure Modelling

RestAssured provides tools for defining the composition of a system at design time (whether the design of the system, or design of security measures for an existing system). Our methodology uses these tools in the following way.

The first step is to use the Cloud System Analysis Pattern (CSAP) tool from UDE to capture the overall context for the system. This tool is designed to encourage users to think about assets they might at first overlook, including assets that are not directly participating in the system at run-time. The CSAP tool provides a pre-defined classification of system components, including data, cloud resources, and software components. More importantly, it also includes placeholders for different types of users such as data subjects (or owners of non-personal data), data controllers and processors, and data providers and consumers. Finally, it incorporates third parties, legislators and regulatory assessors, contracts, and other actors and artefacts that play an important role with respect to the system, even though they may not be participants in the system when it is operating. This tool is described in more detail in Section 4.1.
The second step is to use the System Security Modeller from IT Innovation to introduce a more detailed picture of the relationships between the assets, which can be automatically analysed to determine which threats could arise, and what measures could be used to treat the risks. The System Security Modeller tool allows users to specify the trustworthiness of system components, and the impact of potential threat-induced adverse behaviours in system assets. From these inputs one can calculate the likelihood of each threat and adverse behaviour, and combine this with the specified impact level to evaluate risks. The System Security Modeller thereby supports an iterative process for determining security requirements in the system at design time. For more details, see Section 4.2.

At this stage, the interaction between these two levels of analysis has been left open. We envisage the high-level structure would be captured first using CSAP, and the resulting model imported into System Security Modeller and refined there. However, it is also possible that SSM could be used to create pre-defined workflows (i.e. capturing the system structure), and CSAP used to add context in a specific, planned deployment. This second option may make sense if the structure of the system is pre-defined by the suppliers of software components, so the only decisions left for users is how and where the software will be deployed. We plan to reanalyse these options based on the feedback we obtain from the initial validation studies in WP8.

2.3 Risk Levels and Adaptation

In a modern, cloud based system composed of micro-services, the configuration of the system is not static. The above tools can capture the intended structure of the system (which types of services are connected together in which ways), but not the configuration (which services, deployed where, with what security measures, and what dataflows). This can only be determined when the system is deployed, creating specific service networks hosted at specific locations under known jurisdictions. The configuration may then change due to performance and cost optimisation adaptations, which will most likely be triggered by an autonomic cloud infrastructure management system. Changes may also become necessary due to failures in certain system components, forcing the system to adapt to use alternatives.

We envisage that the machine understandable models produced at design time will also be used at run-time to evaluate risks during the operation of the system. The idea is that the design time model specifies the intended structure of the system as defined by its designer, including security measures that could be used to protect system assets. For each type of threat identified at design time, we can extract the assets involved, and find the control strategies (combinations of security measures) that are considered adequate to manage the risk.

Figure 2.2 provides a conceptual view of the logical components and interfaces required for risk assessment and adaptation in RestAssured. The actual detailed software architecture may differ in the final implementation.

The risk monitor in Figure 2.2 maintains risk assessment’s view of the run-time system configuration. This is updated by the RestAssured adaptation components whenever the run-time system model changes, as shown in Figure 2.2, and is a fusion of the actual system state and the design-time risk model.

The risk evaluator in Figure 2.2 performs the run-time risk assessment. If the risk level rises above a certain threshold, action would need to be taken. To do this, the run-time risk model would signal the RestAssured adaptation component, using channel ⊗ in Figure 2.2.

At present we envisage using a 5-point Likert scale to represent risk levels (Table 2.2). With this scale, it would be acceptable for the risk level to be as high as ‘low’, but if it goes above this, the adaptation system should be told. The information supplied should include the assets involved in the associated threat, and which of those assets are non-compliant with respect to the design model. If the risk level is ‘medium’ then the adaptation component should look for a system change to reduce the risk level, e.g. swapping out non-compliant asset(s). If the risk level is ‘high’ then emergency action would need to be taken, e.g. shutting down services to avoid the risk if necessary.
RestAssured Consortium

Figure 2.2: Run-time risk assessment in RestAssured

<table>
<thead>
<tr>
<th>Risk level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very High</td>
<td>Cannot be allowed to arise</td>
</tr>
<tr>
<td>High</td>
<td>Control measures should be introduced immediately, by shutting down parts of the system if necessary</td>
</tr>
<tr>
<td>Medium</td>
<td>Can be tolerated for a short time while measures are introduced to address it</td>
</tr>
<tr>
<td>Low</td>
<td>Can be tolerated for a longer period if necessary</td>
</tr>
<tr>
<td>Very Low</td>
<td>Risk can be accepted</td>
</tr>
</tbody>
</table>

Table 2.2: Risk levels adopted by RestAssured

It is also possible that the adaptation system may wish to trigger a change for other reasons, e.g. to optimise the use of cloud resources, or to respond to a software crash or hardware failure.

Before it makes any changes, it should signal the risk evaluator using channel $\mathcal{y}$ in Figure 2.2. The information supplied to the risk evaluator should include a list of assets that would be added or removed, plus the relationships of added assets to those already in the system, and the security measures available at the added assets. Migration of a software component from one hardware node to another could therefore be represented by removing one asset and adding another.

When the risk evaluator receives such a change proposal, it will evaluate and report the risk level to the adaptation component, using channel $\mathcal{z}$ in Figure 2.2. There are two options for the information that can be passed. The simplest is to return yes or no depending upon whether the evaluated risk level is below a given threshold, in this case that the risk level is below medium. A better solution is to pass the same information as in channel $\mathcal{x}$. The adaptation component will then have to decide whether to go ahead with the adaptation, by weighing up the risk level with other criteria. For example, it may be acceptable to make...
a change to a medium risk configuration while recovering from some otherwise fatal system crash, provided it is possible to reconfigure again soon afterwards to reduce the risk again.

The choice of which approach to take will depend upon how the adaptation component selects candidate adaptations. This is something that may evolve during the project. We also envisage that some optimisation of the round trip (risk evaluation-adaptation-risk evaluation-adaptation) may be necessary in the second half of the project.

It is also unclear how human operators might play a role in this round trip. Figure 2.2 includes a ‘risk dashboard’ component which was intended to allow operators to monitor risk levels and (where appropriate) initiate changes on their own initiative. At this stage the risk dashboard will be a very simple interface that reports whether assets comply with (i.e. have the security controls specified in) the design time model. We will explore whether more or less human involvement is appropriate, based on results from the initial validation case studies in WP8.

2.4 Summary

The overall security and privacy engineering methodology is based on the analysis and treatment of risks, both at design time and during run-time. The benefits of this methodology are:

- it is well aligned to existing information security standards including ISO 27005 and ISO 27001, avoiding possible problems with adoption by standards compliant organisations;
- it allows users to supply all the key pieces of input, including trustworthiness and impact assumptions, while automating complex calculations of risk;
- it supports run-time as well as design-time risk assessment and management, making it possible to fulfil the General Data Protection Regulation (GDPR) [?] requirement for continuous risk assessment for any changes.

In the second half of the project, some refinements are anticipated. These will focus on how the design-time methodology can best support commercial software and service providers, e.g. using SSM and CSAP in a different way to more clearly delineate the roles of suppliers and end users, the involvement of human operators in run-time risk management, and optimisation of the interfaces between risk evaluation and the adaptation components in RestAssured.
3 Threat Models

As already described in Section 2.2 in RestAssured we have two risk analysis tools, CSAP and System Security Modeller, that model different aspects of a system. CSAP models the overall context for the system, while SSM models the detailed relationships between the assets of the system. In this chapter we describe the underpinning threat models of these two tools.

3.1 Modelling the System Context

3.1.1 General context definition

The modelling of the system context represents the basis for the risk assessment. The context defines the scope and boundaries that are relevant for performing the risk assessment. Furthermore, the context provides necessary information that has to be considered during the risk assessment. By means of the context information the assets for the risk assessment are derived. Context information is also considered regarding the determination of the impacts of security incidents. The definition of a context is divided into an external context and an internal context. The internal context refers to the considered system itself. It provides the following information:

- Relevant business service(s)
- Relevant internal stakeholders that are interacting with the considered service(s)
- Relevant locations
- High level information assets
- High level supporting assets (e.g. hardware, software)
- Interfaces

The external context specifies requirements that are indirectly relevant for the considered services. It provides information regarding legal, regulatory and contractual requirements to the provided services.

There exist logical connections between certain information from the external context and according information from the internal context. For example, the location of a service can imply particular legal requirements. Vice versa, regulatory requirements can influence the designated security level for information assets.

3.1.2 Context definition by using the RestAssured Cloud System Analysis Pattern

The Cloud System Analysis Pattern (CSAP) (see [?][]) represents a graphical pattern that supports the definition of a context regarding a cloud computing scenario. The pattern approach has the advantage that no necessary information is overlooked during the definition of the context.

Within the RestAssured project a specific CSAP named ReAs-CSAP has been created. This ReAs-CSAP allows the specification of contexts of cloud computing services that are provided by realizing concepts and technologies of RestAssured.

3.1.2.1 Description of the ReAs-CSAP

During the progress in the RestAssured project the ReAs-CSAP has been developed further in comparison to the ReAs-CSAP in the Deliverable 3.1 [?]. The modified ReAs-CSAP is shown in Figure 3.1.
Figure 3.1: Modified RestAssured-Cloud System Analysis Pattern (ReAs-CSAP)

In the following, the different elements of the ReAs-CSAP are discussed. Here, the ReAs-CSAP elements are associated to the general context definition from Section 3.1.1. The changes of the ReAs-CSAP regarding to [?] are described explicitly.

Within the ReAs-CSAP the information of the internal context (see Section 3.1.1) is represented by the Direct Environment. The Direct Environment contains the Direct Stakeholders that are relevant for the considered cloud computing services and the representation of the Cloud itself. In the ReSA-CSAP a Cloud is represented graphically by a grey box (see Figure 3.1). The Cloud represents its different parts in form of Cloud Elements. Here, Cloud Elements specify the relevant cloud computing services and the physical resources of the Cloud. Graphically the Cloud Elements are represented by white boxes inside the Cloud (see Figure 3.1). The associations between the different Cloud Elements are also represented. Direct Stakeholders are interacting actually with according Cloud Elements. The logical relationships between Direct Stakeholders are also considered within the ReAs-CSAP.

The different types of Direct Stakeholders are explained in the following:

- **Data Subject**: A Data Subject is an identified or identifiable natural person who makes use of the considered cloud computing service. In this context personally identifiable information (PII) and/or sensitive personal information (SPI) of the Data Subject is processed and/or stored by using the cloud computing service. Data Subjects represent a crucial role because assuring the privacy and security of their data is the main goal of RestAssured.

  The definition for Data Subjects in the context of RestAssured is compatible to the General Data Protection Regulation (GDPR) [?]. The GDPR is the successor of the DIRECTIVE 95/46/EC [?].

- **Data Controller**: A Data Controller is the provider of the considered cloud computing service in form of IaaS, PaaS or SaaS. Data controllers have contracts with the Data Subjects who are using the provided cloud computing services. A Data Controller is legally responsible for the compliance of the specified privacy and security requirements regarding the data of the Data Subjects. For insuring this compliance, Data Controller make use of the RestAssured concepts and technologies. To this, they have to register to the RestAssured Platform.
The definition for Data Controllers in the context of RestAssured is compatible to the General Data Protection Regulation (GDPR) [?]. The GDPR is the successor of the DIRECTIVE 95/46/EC [?].

For providing their SaaS Data Controllers can use

- their own Infrastructure as a Service (IaaS) and/or Platform as a Service (PaaS) infrastructure or
- they can use an external IaaS and/or PaaS infrastructure provided by particular Cloud Providers.

Within the ReAs-CSAP in [?] the role of the Data Controller was represented by an Indirect Stakeholder of the type Online Service Provider. By renaming the type of this Indirect Stakeholder, its logical role is described more precisely and conforms to the general definition within RestAssured. The associations HasContractWith and RegisterTo have also been added as compared to the ReAs-CSAP from [?].

- **Data Consumer:** Data Consumers are natural or legal persons, public authorities, agencies or any other bodies. Besides the Data Subjects, they are also users of the considered cloud computing service. Here, their use case can be different to the use case of the Data Subjects. Within this use case the accessible data of Data Subjects is disclosed to Data Consumers. The rights for Data Consumers regarding the access to and usage of particular data is defined in the corresponding Sticky Policies.

The definition for Data Consumers in the context of RestAssured is compatible to the General Data Protection Regulation (GDPR) [?]. The GDPR is the successor of the DIRECTIVE 95/46/EC [?].

In [?] Direct Stakeholders of different types like Online Service Client, Cloud Provider, Cloud Administrator, SaaS Operator, etc. have been considered as Data Consumers. Regarding the present definition for a Data Consumer, only the former Online Service Client represents the role of a Data Consumer. Accordingly, the type of the Online Service Client has been changed to Data Consumer. The IsA-association from Data Consumer to the particular Direct Stakeholder has been deleted. A Cloud Provider is now considered as a Data Processor that processes the data of a Data Subject in the provided cloud computing service. This role as a Data Processor is represented in the name of the instantiated Cloud Provider.

- **Cloud Provider:** Cloud Providers are legal entities that provide cloud computing services in the form of IaaS and/or PaaS that are used for providing the considered SaaS. In the case of IaaS-Providers, they also own the resources for providing this type of cloud computing service. Cloud Providers can have associations to the following other types of Direct Stakeholders that are working for Cloud Providers:

  - **Cloud Support:** The optional Cloud Support works for the Cloud Provider. It represents the point of contact for Cloud Customers if they have questions or problems regarding the relevant IaaS- and PaaS-Cloud computing service. Possible problems are delegated to the Cloud Administrators.
  - **Cloud Administrators:** Cloud Administrators work for Cloud Providers. They administrate the Resources of the Cloud as well as the Cloud Software Stack and handle problems that have been reported by customers.
  - **External Parties:** External Parties specify service providers that work for the Cloud Provider. Here, an External Party delivers services that are relevant for the cloud or affect the operation of the considered cloud computing service. For example, external parties could be represented by companies for the maintenance of IT-resources and air-condition or cleaning services.
  - **IaaS Operator:** IaaS Operators perform tasks regarding the operation of an IaaS service.

The types of Direct Stakeholders that are mentioned in the list above are not relevant at the current state of the RestAssured project.
• **Online Service Developer**: The *Online Service Developer* specifies a legal person or organisation that has developed the Software Product that is provided by the Data Controller by the corresponding SaaS. Online Service Developers work for the Data Controller. They need profound skills regarding the technologies (e.g. Intel SGX) that are used by the RestAssured platform.

• **External RestAssured Service Provider**: *External RestAssured Service Providers* host the RestAssured Data Protection Components in an according infrastructure and provide the functionality of these components as a service. This service can be used by Data Controllers that do not want to be in charge for hosting the the RestAssured Data Protection Components.

In comparison to the ReAs-CSAP in [?][1] the type of this Direct Stakeholder was renamed from *External Service Provider* to *External RestAssured Service Provider* to emphasise its logical role.

• **SaaS Operator**: *SaaS Operators* perform tasks regarding the operation of an SaaS. This type of Direct Stakeholder is not relevant in the current stage of the project.

Beside the Direct Stakeholders, the Direct Environment also contains the **Cloud** that in turn contains the different **Cloud Elements**. The Cloud can represent a public, private hybrid or community cloud. In the graphical representation of the ReAs-CSAP the type of the Cloud is displayed in the Cloud beside the key word ”Cloud” (see Figure [3.1]). The instantiated Cloud Elements represent

- Relevant business services specified by (cloud computing) **Services** (IaaS, PaaS, SaaS)
- High level information assets specified by **Data**
- High level supporting assets specified by **Resources** (Hardware, Software, Location)
- RestAssured concepts and technologies (Sticky Policies, RestAssured Platform)

The different types of Cloud Elements are described in the following:

- **Service** defines a central point for referencing all provided cloud computing services. Data has an association to Service, because the relevant Data is processed by all provided cloud computing services. The RestAssured Platform influences the processing of the Data on the PaaS-levels.

- **SaaS** represents a cloud computing service on the SaaS-level. In the current state of the RestAssured project only SaaS-cloud computing services are provided by the Data Controllers. Accordingly, SaaS-services are currently the main subject for the risk analysis. SaaS is complemented by the Software Product(s) that functionality is provided via the SaaS. SaaS uses a PaaS for the provision of its service.

- **PaaS** specifies a cloud computing service on the PaaS-level that is used by the SaaS. PaaS is complemented by the Development Environment and API that is provided via the PaaS. PaaS uses the resources that are provided by the IaaS. PaaS is provided by a Cloud Provider.

- **IaaS** defines a cloud computing service on the IaaS-level that provides the necessary infrastructure in form of different types of resources. An example for Resources is hardware for storage and processing power. Resources are used directly by the PaaS and indirectly by the SaaS. IaaS in complemented by the Cloud Software Stack and the infrastructure resources like software and hardware. The IaaS and the Cloud Software Stack are administrated by the Cloud Administrator.

- **Cloud Software Stack** represents the cloud software stack that is necessary for providing the corresponding IaaS.
• Development Environment and API specifies the API and Development Environment that is provided by the according PaaS. The provided API and Development Environment are used for developing the Software Product that is provided by the considered SaaS. The API provides functionalities that enable the use of the relevant PaaS- and IaaS-resources by the Software Product without the need of knowing any specific technical details regarding these resources.

• Software Product represents the software that is provided by the Data Controller via the corresponding SaaS. The Software Product has been developed by the Online Service Developer. For the development the API and Development Environment of the PaaS have been used.

• Pool is the central point for referencing all relevant physical resources of the Cloud that are necessary for providing the appropriate cloud services.

• Resource is the central point for referencing all resources in form of Locations, Software and Hardware.

• Location represents all locations that contain cloud resources (e.g. computing center) or are relevant for the provided cloud computing service in another way (e.g. development site).

• Hardware represents cloud resources in form of necessary hardware (e.g. server racks or network components).

• Software represents cloud resources in form of necessary software (e.g. software for managing the cloud or virtualization).

• Data specifies the personally identifiable information (PII) and/or sensitive personal information (SPI) of the Data Subject that is processed and or stored by the according SaaS. The rights regarding the access to and usage of this data by Data Consumers are specified as follows:

  – Legal specification by the contract between the Data Subject and the Data Controller
  – Formal specification in the associated Sticky Policies

• Sticky Policy defines requirements regarding the access and usage of the Data of a Data Subject. They are derived from the Contract between the Data Subject and Data Controller.

• RestAssured Platform provides security and privacy mechanisms that influence the PaaS-level by enforcing the relevant Sticky Policies. The RestAssured Platform provides different Data Protection Components that implement the appropriate security and privacy mechanisms. The different components can be hosted by

  – Data Controllers themselves in an according infrastructure or
  – provided by External RestAssured Providers that provide the functionality of the components as a service.

• Data Protection Component defines the RestAssured components that implement security and privacy mechanisms. They are used in the context of providing the considered cloud computing service.

The Indirect Environment represents the external context of a cloud computing service. The relevant information about the external context is represented by different types of Indirect Stakeholders that are contained in the Indirect Environment. Here, the different types of Indirect Stakeholders represent the follow information:
Legislator: Representation of laws and provisions of legislators (e.g. Germany or the European Union) that are relevant for the cloud computing service.

This type of Indirect Stakeholder is especially important within RestAssured, because it enables the representation of laws regarding data privacy. Here, the General Data Protection Regulation (GDPR) is especially relevant. Relevant Legislators can be derived from the Location of the considered cloud computing service.

Domain: Specification of domain-specific provisions and guidelines the cloud computing service has to comply with.

Contract: Representation of contractual provisions (e.g. Service Level Agreements with customers) that have to be fulfilled by the cloud computing service.

In the context of RestAssured the representation of the Contracts between Data Subjects and Data Controllers are particular important because they specify the security and privacy requirements for the particular data of the Data Subject. The sticky policies that specify these privacy requirements in a formal and machine-readable way are derived from such Contracts. It should be ensured that the content of a Contract conforms to the relevant Legislators and Domains.

Assessor: Evaluating the level of security and/or privacy that is provided by the considered cloud computing service. The evaluation can be performed regarding an existing standard like for example ISO 27001.

Within RestAssured, Assessors could analyse the results of the risk assessment approach at design time that is described in this document. Here, they could map information of the risk assessment to the real system. For example, they could check the implementation of countermeasures that shall reduce the levels of corresponding risks.

### 3.1.2.2 Instantiation of the ReAs-CSAP

The ReAs-CSAP specifies the context of a particular cloud computing service that is provided by a Data Controller. This cloud computing service represents the service (see Section 3.1.1) that is the target of the risk analysis. It can be an IaaS, a PaaS or a SaaS. The ReAs-CSAP is instantiated regarding to the cloud computing service that is provided by the Data Controller. Additionally, all cloud computing services that are used by the cloud computing service of the Data Controller have to be considered. For example, if the Data Controller provides a SaaS the directly used PaaS and the indirectly used IaaS have also to be instantiated.

The instantiation of the ReAs-CSAP has to be performed for every type of Data Subject that uses the cloud computing service that is provided by the Data Controller.

Only the instantiated ReAs-CSAP-elements are relevant for the context specification. An instantiated ReAS-CSAP-element can be identified by its name that is followed by its instance type. The instance type is enclosed in angle brackets. If the name of an instantiated ReAs-CSAP-element is equal to its instance type, the element is relevant for the context in general. Such an element is refined to concrete elements later in the risk analysis phase.

During the instantiation of the ReAs-CSAP the following relationships should be considered:

- Instances of Legislators can be derived from the Location(s) of the cloud system.
- The instances of Contracts have to be conform to the instances of Legislator that represent particular laws, regulations etc.
• Instances of Sticky Policies for the data of the Data Subject are derived from the corresponding instances of Contract. According to this, all Contract-instances are relevant that represent a legal contract between the Data Subject and Data Controller regarding privacy and security requirements that reference the relevant data of the Data Subject.

• All cloud computing services that are used directly and/or indirectly by the considered cloud computing service have to be instantiated.

• If the Data Controller is not in charge for providing the particular components of the RestAssured platform, the assigned External RestAssured Provider should be instantiated.

3.1.2.3 Usage of the ReAs-CSAP information in the Risk Assessment

This section describes how the information that is represented in an instantiated ReAs-CSAP is used for the risk analysis.

Primary assets are represented in form of

• Business processes by instances of the cloud elements of the instance types Service, IaaS, PaaS, SaaS
• Informational assets by instances of the cloud elements of the instance type Data
• Direct Stakeholders of the instance types Data Subject and Data Controllers

These primary assets are represented on a relatively high abstraction level. If necessary, these assets should be refined during the risk analysis. In this context, a cloud computing service (e.g. SaaS) can be refined into processes and sub-processes that are necessary for providing the considered service. By means of a cloud computing service and its processes, the identification of further informational assets in form of data can be performed. The data specified in an instance of the CSAP can be refined into a set of more specific data during the risk analysis.

Supporting assets are represented by cloud elements of the instance types

• Pool
• Resources
• Hardware
• Software
• Location
• Cloud Software Stack
• Development Environment / API
• Software Product
• Sticky Policy

The supporting assets that are represented in the ReAs-CSAP should also be refined during the risk analysis. The cloud elements of the instance types Hardware and Software are usually instantiated in each case only once whereby the name of an instance corresponds to its instance type (cf. Sec. 3.1.2.2). In this context, the respective Hardware- and Software-instance represent a category for supporting assets. The Hardware-instance for example could be refined into Hardware-instances that represent servers, network
components or personal computers. During the refinement, new Hardware- and Software-instances can be identified, by considering the hardware and software that supports the provision of the relevant cloud computing services in the ReAs-CSAP.

The instances of Indirect Stakeholders of the types Legislator, Domain and Contract provide information for the assessment of the values of the assets. This assessment determines the impact for the case that a privacy or security property of an asset is compromised. For example, the compromising of a particular privacy property of data or a data subject can lead to legal penalties regarding to the General Data Protection Regulation (GDPR).

3.1.2.4  Context definition for Ami volunteer service use case

The use case of this section describes the Ami volunteer service that is provided in form of a SaaS. This SaaS enables the use of the web-based application Ami that matches service providing volunteers (Ami volunteers) with people requiring help (Ami clients) (cf. [?], Sec. 3.3.1, p. 20).

Figure 3.2 shows the instantiation of the ReAs-CSAP for the Ami volunteer service that is the target of the risk analysis. Here, the volunteer service is represented by the instance of the cloud element Volunteer Service of the type SaaS. This service provides the functionality of the Ami-web-application that is represented by the instantiated cloud element Ami Web Application of the type Software Product. The data controller OCC who provides the volunteer service is specified by the particular instance of the direct stakeholder with the type Data Controller. The application is developed by the OCC Developers by using the Azure Development Environment/API that is provided over the PaaS-cloud element Azure PaaS. The provider of this PaaS is Microsoft. This Cloud Provider provides also the Pool of Resources for running the PaaS via the IaaS Azure IaaS that is defined by the instantiated IaaS-cloud element. The roles of Microsoft as IaaS- and PaaS-Provider are represented by the same direct stakeholder-instance of the type Cloud Provider. The location of the cloud infrastructure for the IaaS is represented by the particular instantiated Location-cloud element that has the name UK. The cloud elements of the type Pool, Resources, Hardware and Software have been instantiated because they are relevant for the context of the Ami Volunteer Service. A refinement of this cloud element is performed during the risk analysis.

The Ami Volunteers are the Data Subjects. They are defined by an instance of the direct stakeholder of the corresponding type. The Data Subjects provide their personal data over the Ami Web Application to the Ami Volunteer Service. This data is represented by the cloud element-instance Volunteer Data of the type Data. The instantiated Sticky Policy-cloud element Volunteer Data Policy restricts the access and usage of the Volunteer Data. The Volunteer Data is consumed by the Data Consumers in form of the indirect stakeholders Ami Clients and the Volunteer Organisation that also represent users of the Ami Volunteer Service. In this context, the access to and usage of the Volunteer Data for these Data Consumers is restricted by the Volunteer Data Policy. The compliance of the Volunteer Data Policy is realized by the cloud element-instance RestAssured Platform of the type RestAssured Platform. Thereby, the compliance of the data security and data privacy requirements, defined by the Volunteers, shall be ensured.

The relevant cloud elements for representing the RestAssured Platform and its contained Data Protection Components are instantiated in each case with a name equal to the according instance type. These instantiations specify that the RestAssured Platform and its Data Protection Components are used in the context of the Ami Volunteer Service, but no further specifications regarding the implementation of the RestAssured Platform and its actual Data Protection Components are made. This actual implementation of the RestAssured Platform is specified during the risk analysis by performing a refinement. The functionality of the RestAssured Platform shall be provided as a service by an External RestAssured Provider. The Data Controller OCC and the Data Subject in person of the Ami Clients have to register to the RestAssured Platform.

Because the Ami Volunteer Service is provided and the cloud infrastructure is located in the United Kingdom, the legislations of the United Kingdom and the European Union are relevant. These legisla-
tions are represented by the corresponding indirect stakeholder-instances of the type *Legislator*. Because the European Union is relevant as a Legislator, the GDPR has also to be represented by a particular indirect stakeholder-instance. The instance of the *Domain-*indirect stakeholder *Social Service* specifies requirements for social services. The instantiated *Contract-*indirect stakeholder *Volunteer Contract* represents the contract between the Data Subject and the Data Controller. This contract specifies restrictions regarding the access to and the usage of the Volunteer Data for the Data Consumers in form of the Ami Clients and the Volunteer Organisation. The Sticky Policies are derived from this contract.

### 3.1.2.5 Context definition for the SCANT Ami volunteer service use case

This section describes the *SCANT* Ami volunteer service that represents an extension to the Ami volunteer service described in Section 3.1.2.4. SCANT (*Social Care Analysis of Needs Tool*) is a component that complements the Ami volunteer service by enabling local authorities to perform queries regarding the personal data of the Ami Clients for the purpose of data analysis. The goal of this data analysis is to gain information regarding social issues. The Ami Clients can restrict or refuse these queries regarding their personal data by an according sticky policy.

Figure 3.3 presents the instantiation for the SCANT Ami volunteer service use case. Because this use case represents an extension of the Ami volunteer service (see Sec. 3.1.2.4), only the differences compared to the Ami volunteer service use case are described.

The *SaaS-*cloud element *Volunteer Service* represents still the target for the risk analysis. Compared to the instantiation of the ReAs-CSAP from Figure 3.2, the Volunteer Service is complemented additionally by the *Software Product SCANT*. The Data Subjects are represented by the *Ami Clients*, because their *Ami Client Data* is generally designated to be accessed by SCANT. Additional to the *Volunteer Organisation*, *Local Authorities* represent another Data Consumer that wants to get access to the Ami Client Data via SCANT. The access to the Ami Client Data is restricted by the *Ami Client Data Policy* that is derived from the *Ami Client Contract* between the Ami Clients and OCC in the role of the Data Controller.
The SCANT-software has been developed by the OCC Developers.

3.1.2.6 Context definition for the Pay-as-you-Drive Insurance use case

The Pay-as-you-Drive Insurance (PAYD) use case represents a cloud computing service for automotive insurances. PAYD enables insurers to offer innovative, cost effective and usage based automotive insurance products. This is achieved by recording and analysing the driving data of the insurance customers. Because this driving data represents personal data of the insurance customers, the requirements regarding data privacy and data security are very high.

The instantiation of the PAYD use case is represented in Figure 3.4. In the following this instantiation is explained. Because the corresponding cloud computing service for PAYD represents a SaaS, it is defined by an cloud element-instance of the type SaaS.

The PAYD-service is provided by the respective insurers. Accordingly, these insurers have the role of a DataController. The PAYD-service is complemented by two Software Products.

The first Software Product is the PAYD Web Application that has been developed by Adaptant. This web application is used by the Data Subjects in person of the Insurance Customers. By using the PAYD Web Application, the insurance customers are able to register to an insurance product and obtain information regarding their current insurance conditions. The Insurance Customers provide their personal data in form of Insurance Customer Data to the PAYD Web Application. The access to as well as the processing and storage of the Insurance Customer Data is restricted by an according Sticky Policy named Customer Data Policy.

The second Software Product specifies the Telematic Component. This component is responsible for receiving the Driving Data of the insurance customers in form of telematic data. The received Driving Data is delivered to the cloud infrastructure of the relevant insurer. If insurance customers consent to the secondary use of their driving data by the telematics provider for the purpose of statistical analysis and performance
Figure 3.4: Instantiation of the ReAs-CSAP for the use case Pay-as-you-Drive Insurance

diagnostics of the edge / mobile network infrastructure, the driving data will also be made accessible to
the telematics analysts. Accordingly, Telematics Analysts represent one Data Consumer. The second Data
Consumer is specified by the Insurance Analysts. They analyse the Driving Data to get information that
is relevant in the context of the respective, provided automotive insurance offers. The Driving Data Policy
corresponds to the Driving Data. This Sticky Policy restricts the amount of telematic data that is comprised
by the Driving Data. The sticky policy also restricts the access to as well as the processing and analysing of
the Driving Data. The Customer Data Policy as well as the Driving Data Policy are implied by the Insurance
Contract between the Insurer and the Insurance Customer. The Insurance Contract has to be conform to the
legislation in form of the GDPR (General Data Protection Regulation).

The PaaS and IaaS that are used in the context for providing the PAYD-service are also provided by the
respective insurers. Accordingly, insurers are the Cloud Providers for PaaS and IaaS and own the Resources
of the Pool that represent the cloud infrastructure. The Cloud Provider, IaaS, PaaS and its complementing
components are instantiated generally for insurers.

The RestAssured Platform and its Data Protection Components are provided by an External RestAssured
Provider.

3.2 Detailed System Modelling

3.2.1 Graph Based System Modelling

To perform a more detailed and complete risk assememnt the individual assets of the system must be iden-
tified and their relationships determined. Graphs provide a natural way of representing such relationships.
The assets in a system represent the nodes in a graph, and the relationships between them are the edges.

At the lowest level conceptually are directed graphs. They are commonly serialised into triples, where a
triple represents one edge in a graph and comprises of a subject (the source of the relationship), a property
(the label of the edge or relationship type) and an object (the target of the relationship). A commonly used model to manage graphs is RDF (https://www.w3.org/RDF/), which introduces the notion of different node types and properties. An RDF dataset typically consists of an ontology (the schema) and the data, which refers to concepts defined by the ontology and can be serialised in different ways, e.g. RDF/XML or Turtle. To efficiently manage such datasets, the concept of segregated graphs has been introduced. The information is recorded by adding an extra field to each triple, effectively turning it into a quad where the last field contains the URI of the graph to which this triple belongs. The resulting collection of datasets can be treated like different tables in a relational database: they are stored separately but can be combined for queries.

Building on these technologies, our approach utilises three distinct layers that build on each other:

A **core model**: the schema that captures the fundamental concepts, e.g. assets, roles, threats, and their relationships.

**Domain models**: datasets that encode domain- specific knowledge, e.g. detailed threats and their possible control strategies.

**System models**: that represent an actual system upon which the risk assessment is being performed.

In the following sections we describe each of these in more detail.

### 3.2.1.1 Core Model

The core model captures the fundamental concepts, and the relationships between them, that the domain models and system models build upon. A simplified version is shown in figure 3.5.
As the underlying schema for both the domain models and the system models, the core model defines a collection of concepts and relationships between them. Some of these concepts are used throughout the model stack, others are helper constructs, supporting inference of more obscure connections between these concepts. For this reason, some of the concepts have been left out of the core model overview as they appear only in the domain model but not in the validated system model.

The RestAssured core model is an enhancement of the core model from previous projects (SERSCIS, OPTET, ASSURED, 5G-ENSURE). The concepts of Asset, Role, Misbehaviour, Threat, Control etc. are carried over from this earlier work, along with relationships like:

- “Asset has Role”
- “Threat threatens Asset”
- “Threat causes Misbehaviour”
- “Control blocks Threat”

The major changes for RestAssured are the addition of impact levels associated with the misbehaviour of an asset, likelihood and risk levels for threats, and trustworthiness levels associated with assets.

Each asset can have multiple trustworthiness levels, each associated with a different attribute of the asset. In a system model the system designer or risk analyst would specify the trustworthiness levels based upon their knowledge of the system itself, what they know about the provenance of the asset, and general knowledge of the threat environment.

The trustworthiness levels are then used to generate the likelihood levels of the threats, and from that the likelihood levels for the possible asset misbehaviours, for example Loss of Availability, or Loss of Control.

Finally, the system designer specifies the impact level of these asset misbehaviours based upon how important each asset is to the business or any regulatory requirements, and from this, and the likelihood level of the misbehaviours, the risk level of each asset misbehaviour is computed.

### 3.2.1.2 Domain Models

The domain model encodes domain specific knowledge. The starting point is a definition of the different possible asset classes that might exist in a given domain. These will typically always include things like hosts, processes, data, stakeholders, etc., but may also include much more specialised asset classes that are specific to a given domain.

The asset classes in a domain model define a tree, so for example we can have an asset class for Human that is a subclass of the Stakeholder asset class. Figure 3.6 shows an excerpt from the RestAssured asset tree, with assertable assets highlighted in yellow. An assertable asset class is one which can be used to construct a system model. In the System Modeller tool (Section 4.2) this is done by dragging assertable assets onto the canvas. Rules, and in particular threat patterns (Section 3.2.2), that apply to parent asset classes will also apply to subclasses.

Other concepts from the core model like Roles, Controls or Misbehaviours need to be specified for each domain model. For example, for each asset class the domain model also needs to define the different roles it can perform in a system model (e.g. client or service, master or slave), its possible misbehaviours (e.g. loss of availability), and the security controls that can be applied to it. A domain model also contains default values for trustworthiness and impact levels (Section 3.2.2.2), though these can later be overridden by the system designer in a given system model.

The bulk of a domain model however consists of a catalogue of threat patterns (see Section 3.2.2). These represent the different possible ways that assets of the classes defined in the domain model can be threatened,
Figure 3.6: The Asset class hierarchy

and the possible control strategies that can be applied to address those threats. The threat pattern catalogue therefore encodes knowledge about the threats that exist within the domain represented by the domain model. System Modeller uses this knowledge base to generate a comprehensive catalogue of threats for each system model.

3.2.1.3 System Models

System models represent actual systems. A system model is constructed from the asset classes in a domain model. Each system model can only use one domain model, but each domain model can be used by any number of system models. The relationships between the assets in a system model are expressed using properties defined in the domain model. For example, that a web service uses a database.

Each system model must be validated. Validation of the system model checks whether the model is a legitimate model, and augments the model by adding inferred assets and relationships, before applying the threat patterns (Section 3.2.2) to identify instances of each threat within the system model.

Threats cause misbehaviours which have an impact level, and have a likelihood that derives from the trustworthiness levels of the assets involved in the threat pattern (Section 3.2.2.2). The system designer or risk analyst uses their knowledge of the system and its context to specify the impact levels for the primary assets of the system (the supporting assets are automatically given appropriate default impact levels by System Modeller). The system designer also specifies appropriate default trustworthiness levels. Once again System Modeller provides initial default values that are derived from the domain model. System Modeller then computes the threat likelihood levels, and from these and the impact levels, risk levels are derived.

The system designer is then able to use the computed risk levels to identify the most serious threats to the system, and then from the control strategies encoded in the domain model alongside the threat patterns (Section 3.2.2.3), select the appropriate controls to address the identified threats. System Modeller then recomputes the threat likelihoods and risk levels. This process is iterated until the system designer is happy that the residual risk has been reduced to an acceptable level.
3.2.2 Threat Patterns

Each domain model contains a catalogue of threat patterns that apply to that domain. These threat patterns capture knowledge of known attack techniques and weaknesses that exist in the given domain. A threat pattern is not a vulnerability though, in the sense of CVE\(^1\) but rather abstracts away specific details to leave just the general shape of the attack.

### 3.2.2.1 Example

![Diagram of a threat pattern for a remote attack on a host](image)

Figure 3.7: Threat pattern for a remote attack on a host

Figure 3.7 represents a remote attack on a host that exploits a software vulnerability on the host. This one pattern represents all possible such attacks. We do not have a separate pattern for each possible software vulnerability on the host. Instead the pattern abstracts away the specific details of the vulnerability, and focuses on the interaction between a remote attacker and a host with vulnerable software.

The pattern should be read as follows:

- The black boxes represent the system assets involved in the pattern. In Figure 3.7 the attacker is on the network “Subnet” and this is linked to the host via a network path.

- The red box indicates which asset is threatened. In Figure 3.7 this is the host.

- The red ellipse indicates the effect of the threat i.e. the misbehaviour it causes on the threatened asset. In Figure 3.7 this is loss of trustworthiness of the users on the host. This threat therefore represents an attack where the attacker acquires a shell on the host. It need not be a root shell though, as the misbehaviour in this particular threat pattern is not loss of control of the host, however this threat could be combined with a privilege escalation attack to take control of the host.

- The blue boxes represent the entry points through which the attack occurs. Entry points are discussed in more detail in Section 3.2.2.2. In Figure 3.7 these are the subset where the attacker is located, and the host that is threatened.

3.2.2.2 Entry Points and Trustworthiness

To attack a system an attacker needs some way in, these are the entry points. For example, in Figure 3.7 the threat is that an untrustworthy user on “Subnet” exploits a vulnerability in the software on the host. This threat exists if not all users of “Subnet” are trustworthy, and the software on the host is not trustworthy. If
the software is defect free, then there is nothing to exploit, and so the threat does not exist. Similarly, if “Subnet” is a secure network, and all the users are trustworthy, then again the threat does not exist. The likelihood of the threat therefore depends upon the actual trustworthiness levels of the users on “Subnet“ and the software on the host.

Trustworthiness and threat likelihood are therefore intimately related, in fact they can be thought of as the duals or inverses of each other. Table 3.1 shows the meaning and relationship between the trustworthiness and likelihood levels we have adopted in RestAssured. As can be seen, if one is high, the other is low, and vice versa.

<table>
<thead>
<tr>
<th>Likelihood</th>
<th>Description</th>
<th>Trustworthiness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very High</td>
<td>Something will definitely go wrong if the possibility exists even only for a short time.</td>
<td>Very Low</td>
</tr>
<tr>
<td>High</td>
<td>Something is likely to go wrong if the possibility exists even only for a short time. Something will definitely go wrong if the possibility exists for very long.</td>
<td>Low</td>
</tr>
<tr>
<td>Medium</td>
<td>Something is likely to go wrong if the possibility exists for very long, but not otherwise.</td>
<td>Medium</td>
</tr>
<tr>
<td>Low</td>
<td>It is unlikely that anything will go wrong without very prolonged exposure to the possibility.</td>
<td>High</td>
</tr>
<tr>
<td>Very Low</td>
<td>It is unlikely that anything will ever go wrong.</td>
<td>Very High</td>
</tr>
</tbody>
</table>

Table 3.1: Likelihood and trustworthiness levels

From the example in Figure 3.7, we can see that the existence of a threat depends upon the trustworthiness of certain asset attributes, rather some absolute notion of asset trustworthiness. Table 3.2 lists the trustworthiness attributes that we see in threat patterns.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intrinsic TW</td>
<td>The extent to which the asset is free of internal motives or flaws that cause it to misbehave.</td>
</tr>
<tr>
<td>Extrinsic TW</td>
<td>The extent to which the asset can resist external influences that cause it to misbehave.</td>
</tr>
<tr>
<td>Management TW</td>
<td>The extent to which the asset is under the control of those with that responsibility. Destroyed by attacks that gain admin rights over the asset.</td>
</tr>
<tr>
<td>User TW</td>
<td>The extent to which users of the asset are free of any tendency to misuse the assets capabilities, whether for malicious or accidental reasons.</td>
</tr>
<tr>
<td>Authenticity</td>
<td>The extent to which operators of the asset are who they claim to be.</td>
</tr>
</tbody>
</table>

Table 3.2: Trustworthiness attributes

The likelihood of a threat is a function of the trustworthiness levels of the attributes that form the entry points for that threat. If one or more of the entry points are highly trustworthy the threat is unlikely, but if all are untrustworthy then the threat likelihood is high. To be more precise, the likelihood of a threat is the inverse (as defined in Table 3.1) of the maximum trustworthiness level of all the attributes that form the entry points to the threat:

\[
\text{likelihood} = \text{inverse}(\max(\text{trustworthiness}))
\]
3.2.2.3 Control Strategies

So far we have not discussed how threats can be blocked or mitigated. To do this we employ security controls. A threat pattern includes possible control strategies that could be employed to address the threat. For the threat in Figure 3.7 several control strategies are available. These are shown in Figures 3.8 through 3.12.

Figure 3.8: Ensure all security patches are applied to the host

Figure 3.9: Perform rigorous testing of the software on the host

Figure 3.10: Employ formal verification techniques on the host software

Figures 3.8 through 3.10 show different strategies for hardening the host. Each of these will have a different effect on the likelihood of the threat. Formal verification of the software (Figure 3.10) is obviously a much stronger technique than just applying the latest security patches (Figure 3.8) as it (hopefully) guards against all possible zero-day attacks. Rigorous software testing (Figure 3.9) probably sits somewhere between the two. Our threat patterns incorporate this information, and System Modeller employs it in its reasoning (Section 3.2.3.3).

The other way to address the threat is to reduce the exposure to malicious users on “Subnet”. One way to do this is to block all access to the host from “Subnet” (Figure 3.11), this is a very effective technique, but is not possible if there are legitimate users of the host on “Subnet”. Alternatively we could restrict communication between “Subnet” and the host to only sessions initiated by the host (Figure 3.12).
3.2.3 Root Causes and Secondary Effects

Threats don’t exist in isolation. Typically the threats in a system interact, but we need to be careful and make a distinction between interactions that are in a sense automatic, and those that require human input.

3.2.3.1 Loss of Trustworthiness vs. Automatic Chaining

Some threats cause misbehaviours that automatically trigger other threats. For example if a webservice uses a database, then any threat that causes loss of availability of the database will also likely cause the webservice to be unavailable. This cascade of effects is automatic. It does not require any action on the part of an attacker, it is simply a direct consequence of the dependency of the webservice on the database. Effects that are not directly caused by a threat are called secondary effects, and System Modeller can automatically compute the cascade of secondary effects triggered by a primary threat.

The threat in Figure 3.7 is different. Here the effect is a loss of trustworthiness of the users on the host. This may have no automatic consequences. What happens next may depend upon further actions of the attacker now that they have access to the host.

If a system model contains an instance of the threat in Figure 3.7, then System Modeller recomputes the likelihood of any threat that has user trustworthiness of the threatened host as an entry point. This may then cause the trustworthiness levels of other assets to change, and System Modeller calculates this cascade of loss of trustworthiness through the system model. This cascade causes a corresponding change in the likelihood of the threats in the model, and a resulting change in the misbehaviour risk levels.

In general, if a threat causes loss of trustworthiness for some attribute (Table 3.2) of the threatened asset, the new trustworthiness level (for that attribute) is computed using the inverse of the likelihood of the threat (as defined in Table 3.1). Actually, to be precise, it is the inverse of the likelihood of the most likely cause of that loss of trustworthiness misbehaviour, as several threats could have the same effect, and the most likely one is the one that dominates. Moreover, we need to take into account the current trustworthiness level, as the analyst may have asserted that this is lower than the computed value. This occurs for example for user trustworthiness on the internet. The fact that there are threats to hosts connected to the internet, and instances of the threat in Figure 3.7 could lead to loss of trustworthiness of the users of those hosts, and
eventually to loss of trustworthiness of the users on the internet, is actually immaterial, as the internet has by default Very Low user trustworthiness.

\[
\text{trustworthiness} = \min(\text{trustworthiness}, \text{inverse}(\max(\text{likelihood})))
\]

Other threats that have the affected trustworthiness attribute as an entry point must be updated, as their likelihoods are a function of the trustworthiness levels of their entry points. If the misbehaviours caused by these threats are also loss of trustworthiness on some other asset, then the change in threat likelihoods could trigger yet more changes to trustworthiness levels in the model. Obviously this could trigger more changes in likelihoods and trustworthiness levels etc.

One might therefore ask if this cascade of changes to likelihoods and trustworthiness levels will ever end. Does the computation converge? A moments thought show that this is indeed the case because likelihood levels only ever increase, and trustworthiness levels only ever decrease, and both scales are discrete and bounded above and below. Thus no threat likelihood or attribute trustworthiness level can change forever. In fact we only have 5 levels (Table 3.1) so each value can change at most 4 times.

### 3.2.3.2 Primary Root Causes

The cascade effect described in Section 3.2.3.1 propagates the effects of a threat forward, but we can ask a different question. Given a particular risk, i.e. a particular misbehaviour of an asset with its corresponding likelihood and impact, what are the possible root causes? In fact given that System Modeller computes the likelihood of all the threats in a system, we can ask a stronger question, which are the most likely root causes of this risk? These are the primary root causes.

This capability is a powerful one. The primary root causes of the highest identified risks are precisely the threats that have the biggest potential impact on the system. These are the threats that must be addressed first. The workflow an analyst would follow is discussed in more detail in Section 4.2.4.

### 3.2.3.3 Control Strategy Trustworthiness

The threat in Figure 3.7 has five possible control strategies shown in Figures 3.8 through 3.12. However, as already discussed in Section 3.2.2.3 these possible control strategies have different strengths. This is modelled by assigning to each control strategy a trustworthiness level. This level represents the degree to which a particular control strategy can be relied upon to protect against the threat.

For example, the control strategy in Figure 3.8 will protect against all known software vulnerabilities for which patches are available, however it will not protect against zero-day exploits. This control strategy is therefore assigned a lower trustworthiness level than formal verification of the software (Figure 3.10), which establishes rigorous evidence that the software is free of defects.

If an analyst selects a particular control strategy for a threat, then the analyst is making an assertion that in the deployed system the threat will be treated through the use of those controls. System Modeller then uses the corresponding trustworthiness level of that control strategy to reduce the likelihood level of that threat by an appropriate amount.

To be more precise the dominant control strategy for a threat is the strongest control strategy selected (by the analyst) from those available to treat the threat i.e. the selected control strategy with the highest trustworthiness level. The inverse (using Table 3.1) of the trustworthiness level of the dominant control strategy forms an upper bound on the likelihood of the threat:

\[
\text{likelihood} = \min(\text{likelihood}, \text{inverse}(\max(\text{trustworthiness})))
\]

The threat likelihood cannot be greater than this, however it can be lower if the trustworthiness levels of its entry points are high enough. In other words, if the threat is very unlikely, then the control strategy may
be unnecessary as the controls may not be strong enough to bring the likelihood any lower than it is for the untreated threat. However if the threat is extremely likely, most control strategies, no matter how weak, will have some effect on the threat likelihood.

This change to the threat likelihood as a result of the selection of one or more of the control strategies associated with that threat is then propagated throughout the system model as described in Section 3.2.3.1.

By focusing on the primary root causes of the most serious risks in the system (Section 3.2.3.2), and applying the strongest feasible controls to those threats, the overall risk level of the system can be reduced.

3.2.4 Threat Pattern Coverage

For the approach discussed above to be truly effective, the threat pattern catalogue within a domain model must embody knowledge of all possible threats within the chosen domain.

Initially this sounds like a daunting task. There are many projects and organisations that aim to catalogue known vulnerabilities, for example CVE\(^2\), and weaknesses, for example CWE\(^3\), and these catalogues are vast and forever changing as new vulnerabilities are discovered.

Our approach works on a different level. Each threat pattern can represent many specific vulnerabilities (Section 3.2.2), and so what is important is that every vulnerability corresponds to a threat pattern in the domain model. That way we can be sure that we capture all threats within a domain.

3.2.4.1 Threat Patterns vs. Vulnerabilities and Weaknesses

If we compare a threat pattern like Figure 3.7 with the entries in a catalogue like CVE or CWE we notice some striking differences.

An entry in CVE is a specific vulnerability in a specific piece of software, for example a buffer overflow vulnerability, or an SQL injection vulnerability, in a specific application. A corresponding entry in CWE might be “CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer”, or “CWE-89: Improper Neutralization of Special Elements in an SQL Command (‘SQL Injection’)”.

Neither of these operates at the same level as a threat pattern. If we think of an instance of the threat pattern of Figure 3.7 in a given system model, then this will correspond to a specific host in the model, connected to a specific network, via a network path discovered by System Modeller. The threat is then that an attacker on the specified network can exploit a vulnerability of the specified host, in this case to acquire a shell.

There may be many known ways to do this, corresponding to different entries in the CVE database for remotely exploitable vulnerabilities in the software running on the host. We could consider creating a threat pattern for each one. In such a threat pattern the host asset would only match instances that corresponded to hosts running the software to which the CVE corresponds. A control strategy for this CVE based threat pattern might be to apply the specific patch for that vulnerability. This is weaker than the control strategy in Figure 3.8 which is to apply all known patches for all software running on the host. We can envisage such a threat pattern for each CVE that could causes this instance of the threat in Figure 3.7, each with a control strategy to apply the specific patch for that vulnerability. Mathematically we could consider the instance of the threat in Figure 3.7 as some kind of limit of the corresponding instances of the CVE threat patterns. To address each of the CVE threats we would need to apply all the patches corresponding to those CVEs. Figure 3.8 captures that in a single control strategy.

The entries in CWE correspond to weaknesses rather than vulnerabilities. When compared to threat patterns, we see that a threat pattern like Figure 3.7 typically requires that at least one asset has a weakness, but we do not specifically care what that weakness is, only that it is exploitable i.e. a threat. Thus threat patterns

---

\(^2\) Common Vulnerabilities and Exposures: [https://cve.mitre.org/](https://cve.mitre.org/)

\(^3\) Common Weakness Enumeration: [https://cwe.mitre.org/](https://cwe.mitre.org/)
can be thought of as representing weaknesses, but not in the sense of CWE where they are weaknesses in the software implementation, but rather weaknesses in the pattern of interactions between assets.

### 3.2.4.2 OWASP Top 10 - 2017

The OWASP Top 10 - 2017 represents an alternative approach to CVE and CWE. Here the top 10 risks to web application security are identified from a combination of empirical data and expert feedback. The entries are clearly CWE like, in that they represent weaknesses, but they represent general categories of weaknesses, and are ranked in order of risk level.

Like the entries in CWE, the risks in the OWASP Top 10 do not include details of the interactions between assets that may lead to a threat actually existing, unlike our threat patterns. As a result the risk levels assigned in the OWASP Top 10 do not take into account the trustworthiness of a threat’s entry points (Section 3.2.2.2), nor the business impact levels.

The OWASP Top 10 also makes a distinction between risks that we might identify. For example, the entries:

- **A1:2017 - Injection**
- **A4:2017 - XML External Entities (XXE)**
- **A8:2017 - Insecure Deserialization**

are all variations on injection attacks, with the latter two being new (to the 2017 edition of the Top 10) specific variants. A threat pattern would instead make a distinction between which assets were involved in undertaking the injection attack, and what the resulting misbehaviour was. Specific threat patterns would only be created for the different forms of injection attack if different security controls were required to protect against them. Generic control strategies like “sanitize inputs”, or “only accept input from trusted sources” are equally applicable to all of them.

### 3.2.4.3 The RestAssured Domain Model

The RestAssured domain model is one of the outputs of WP7. At this stage of the project the domain model captures the asset class and their relationships required to model the use cases, and a collection of generic threat patterns that have evolved from previous projects (Section 3.2.1.1).

During the remainder of the project the RestAssured domain model will be refined, and the threat pattern catalogue expanded to include threats, vulnerabilities, and weaknesses specific to the RestAssured domain. In particular, Task 7.3 will investigate new and emerging mitigation techniques, and these will be incorporated within the RestAssured domain model as control strategies.

### 3.2.5 Risk Calculation

#### 3.2.5.1 Impact Levels and Risk

Risk is a function of likelihood and impact. If a particular misbehaviour (of an asset) has no impact, then there is no risk. Table 3.3 details the impact levels defined in RestAssured. It is important to note that an impact level is associated with each possible misbehaviour of each asset, and that the impact levels within a system model are asserted by the analyst. System Modeller is unable to compute impact levels as these are very much system specific, and only the system designer or operator has the contextual information to specify the correct impact level for each misbehaviour. This being said, System Modeller can provide default
(low) values for impact levels on the assumption that an asset is a supporting asset, leaving the analyst to specify higher levels for those assets identified as primary assets.

<table>
<thead>
<tr>
<th>Impact</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very High</td>
<td>This misbehaviour in this asset will be fatal to key business interests. Very few misbehaviour sets will be at this level, but those that are must be prevented.</td>
</tr>
<tr>
<td>High</td>
<td>This misbehaviour in this asset has a serious impact on the business, and will be fatal if not addressed quickly.</td>
</tr>
<tr>
<td>Medium</td>
<td>This misbehaviour in this asset can be tolerated for a short time, but will become serious if not addressed quickly.</td>
</tr>
<tr>
<td>Low</td>
<td>This misbehaviour in this asset can be tolerated, but it does degrade business function or efficiency.</td>
</tr>
<tr>
<td>Very Low</td>
<td>This misbehaviour in this asset has negligible impact on the business.</td>
</tr>
</tbody>
</table>

Table 3.3: Misbehaviour impact levels

ISO 27005 specifies that risk is a function of likelihood and impact, but does not specify what that function should be. In RestAssured we have adopted the mapping in Table 3.4. This has been carefully chosen to be consistent with the interpretations of likelihood and impact levels defined in Tables 3.1 and 3.3. System Modeller uses Table 3.4 to perform the final mapping from the computed asset misbehaviour likelihood levels, to risk levels, using the impact levels asserted by the analyst. The output is for each asset, and each misbehaviour that can arise at that asset (as a result of a threat pattern instance), a corresponding risk level.

<table>
<thead>
<tr>
<th>Impact</th>
<th>Very Low</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Low</td>
<td>Very Low</td>
<td>Very Low</td>
<td>Very Low</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Low</td>
<td>Very Low</td>
<td>Very Low</td>
<td>Low</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Medium</td>
<td>Very Low</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>High</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>Very High</td>
<td>Very High</td>
</tr>
<tr>
<td>Very High</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>Very High</td>
<td>Very High</td>
</tr>
</tbody>
</table>

Table 3.4: Mapping misbehaviour likelihood and impact to risk level

3.2.5.2 Detailed Mathematical Formulation

In this section we give a more precise mathematical formulation of the risk calculation performed by System Modeller. The material in this section can be safely skipped by those who are not mathematically inclined. Table 3.5 defines the symbols used in the rest of this section.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>The set of asset classes in a system model.</td>
</tr>
<tr>
<td>M</td>
<td>The set of possible misbehaviours of members of A.</td>
</tr>
<tr>
<td>T</td>
<td>The set of threat instances in a system model.</td>
</tr>
<tr>
<td>TWA</td>
<td>The set of trustworthiness attributes defined in Table 3.2.</td>
</tr>
<tr>
<td>CS</td>
<td>The set of control strategies in a system model.</td>
</tr>
<tr>
<td>DC</td>
<td>The set of threats that are the direct causes of misbehaviour m at asset a.</td>
</tr>
<tr>
<td>EP</td>
<td>The set of entry points to the threat t. Recall that an entry point is a pair consisting of an asset a and a trustworthiness attribute twa of a (Section 3.2.2.2).</td>
</tr>
<tr>
<td>SEC</td>
<td>The set of secondary effect conditions of t. A secondary effect condition is a pair ((m, a)) consisting of a misbehaviour m of an asset a. If all the secondary effect conditions of t are active then t (may) be automatically triggered (Section 3.2.3.1).</td>
</tr>
<tr>
<td>TCS</td>
<td>The set of control strategies selected for threat t.</td>
</tr>
<tr>
<td>(L_{(m,a)})</td>
<td>The likelihood of misbehaviour m at asset a.</td>
</tr>
<tr>
<td>(L_t)</td>
<td>The likelihood of threat t.</td>
</tr>
<tr>
<td>(TW_{(twa,a)})</td>
<td>The trustworthiness level of trustworthiness attribute twa at asset a.</td>
</tr>
<tr>
<td>(TW_{cs})</td>
<td>The trustworthiness level of control strategy cs.</td>
</tr>
<tr>
<td>(l_{twa})</td>
<td>The misbehaviour (in M) corresponding to loss of trustworthiness of trustworthiness attribute twa (Section 3.2.3.1).</td>
</tr>
</tbody>
</table>

**Table 3.5: Risk calculation symbols and definitions**

**Asset Misbehaviour Likelihood:**
The likelihood of asset a experiencing misbehaviour m is given by the likelihood of the most likely threat to directly cause m at a. If a direct cause is at the end of a chain of threats from some primary root cause, then its likelihood already incorporates the likelihood of that root cause:

\[
L_{(m,a)} = \max\{L_t \mid t \in DC(m, a)\}
\]

**Loss of Trustworthiness Misbehaviours:**
In Section 3.2.3.1 we explained that for threats that cause loss of trustworthiness, the trustworthiness level of the affected trustworthiness attribute is bounded by the inverse of the most likely threat that would cause that misbehaviour. In other words, the trustworthiness level is bounded by the inverse of the likelihood of that misbehaviour:

\[
TW_{(twa,a)} \leq L^{-1}_{(l_{twa},a)}
\]

**Control Strategy Trustworthiness:**
In Section 3.2.3.3 we explained that the trustworthiness level of a control strategy imposes an upper bound on the likelihood of a threat, and that the strongest selected control strategy has the dominant effect:

\[
L_t \leq \max\{TW_{cs} \mid cs \in TCS(t)\}^{-1}
\]
Likelihood of Primary Threats:
The likelihood of a primary threat is given by the trustworthiness levels of its entry points (Section 3.2.2.2):

\[ L_t = \max\{TW(twa,a) \mid (twa,a) \in EP(t)\}^{-1} \]

However, we must then take into account any control strategies selected for that threat:

\[ L_t = \min\{\max\{TW(twa,a) \mid (twa,a) \in EP(t)\}^{-1}, \max\{TW_{cs} \mid cs \in TCS(t)\}^{-1}\} \]

Likelihood of Secondary Effects:
A secondary effect requires all the secondary effect conditions to be present, which means the likelihood of a secondary effect is given by the least likely secondary effect condition:

\[ L_t = \min\{L(m,a) \mid (m,a) \in SEC(t)\} \]

Similarly to primary threats, we also need to take into account any selected control strategies:

\[ L_t = \min\{\min\{L(m,a) \mid (m,a) \in SEC(t)\}, \max\{TW_{cs} \mid cs \in TCS(t)\}^{-1}\} \]

Overall Risk Level Calculation:
To compute the risk levels it is sufficient to compute the likelihood of each \((m,a)\), where \(m\) is a misbehaviour that can occur at asset \(a\). From this, Table 3.4 can be used directly, in order to compute the risk level of \((m,a)\) from the impact level of \((m,a)\) specified by the analyst.

The value of \(L(m,a)\) for any \((m,a)\) depends on the likelihood of the direct causes of \((m,a)\), but these in turn depend upon the likelihood of other threats, which in turn may depend upon the likelihood of other misbehaviours at other assets.

We must therefore solve for all the \(L_t\), \(L(m,a)\), and \(TW(twa,a)\) simultaneously. System Modeller employs an iterative algorithm that starts from initial values for the trustworthiness levels (a combination of default values from the domain model, and initial values asserted by the analyst), and initial likelihood levels of Very Low. The algorithm then loops through a cycle of:

1. calculating likelihoods,
2. updating trustworthiness levels,

until a stable solution is reached. As stated in Section 3.2.3.1 this is guaranteed to converge as:

1. likelihoods can only ever increase,
2. trustworthiness can only ever decrease,
3. the sets of trustworthiness levels and likelihood levels are discrete and finite (therefore bounded above and below).

To be precise, the calculation is performed with a fixed set of selected control strategies. The analyst selects the control strategies in System Modeller and then pushes a button to recompute the risk levels. For trustworthiness levels the only active rule is the lowering of some \(TW(twa,a)\) as a result of threats to loss of trustworthiness. For likelihoods:

1. The likelihood of primary threats can increase as a result of reductions in trustworthiness levels, but can never decrease, as the computation contains no rule that increases trustworthiness levels.
2. The likelihood of an asset misbehaviour could only decrease if the likelihood of the most likely threat that directly caused it decreased. From the point above, this would have to be a secondary effect, as the likelihood of primary threats can never decrease.

3. The likelihood of a secondary effect could only decrease if the likelihood of one of its secondary effect conditions decreased, and from above that would have to be as a result of the decrease in the likelihood of some other secondary effect. As there is no way to initiate this chain, the likelihoods of secondary effects (and thus asset misbehaviours) can never decrease.

3.3 Mapping from ReAs-CSAP to Secure System Modeller

This section describes a mapping from the different elements of the ReAs-Cloud System Analysis Pattern (ReAs-CSAP) to the corresponding constructs of the System Security Modeller (SSM). This mapping is outlined in Table 3.6. Furthermore, the different ReAs-CSAP- and/or SSM-elements are assigned to the corresponding risk assessment concepts that are represented by the different elements.

- **Goals** consider the objectives and restrictions regarding the desired and/or required level for the security and privacy of assets. These objectives can be, for example, specified by laws, regulations or contracts.

- **Assets** represent any item that has value to an organisation or a person. Assets can be classified into primary assets and supporting assets. Primary assets include business processes as well as information and data. Supporting assets support the provision of business processes. In this context, they can be responsible for processing, storing and transferring the relevant data. Examples for supporting assets are software, hardware or network components.

  A risk analysis is performed regarding to the security and privacy properties of assets. Here, the value of an asset is assessed. In this assessment the impact for the case that security or privacy properties of an asset get compromised is considered. Furthermore, the likelihood for such security or privacy incidents are assessed. Based of the asset value and the likelihood for a security or privacy incident the risk level for an asset is determined.

- **Threats** are potential events that can lead to a loss of the security and privacy properties of assets. In the context of a risk analysis, the likelihood for an actual occurrence of a threat is assessed. Examples for threats are attacks of hackers, system failures or incorrect configurations of cloud components.

- **Mechanisms** are countermeasures for supporting the implementation of the goals regarding the security and privacy properties of assets. Here, mechanisms shall reduce the likelihood that relevant threats lead to security or privacy incidents.
Table 3.6: Mapping from ReAs-CSAP-elements to System Security Modeller-elements

<table>
<thead>
<tr>
<th>Risk Assessment Concept</th>
<th>ReAs-CSAP-element</th>
<th>ReAs-CSAP-instance type</th>
<th>SSM-element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal</td>
<td>Indirect Stakeholder</td>
<td>Legislator</td>
<td>Impact Level</td>
</tr>
<tr>
<td></td>
<td>Indirect Stakeholder</td>
<td>Contract</td>
<td>Impact Level</td>
</tr>
<tr>
<td></td>
<td>Indirect Stakeholder</td>
<td>Domain</td>
<td>Impact Level</td>
</tr>
<tr>
<td></td>
<td>Indirect Stakeholder</td>
<td>Assessor</td>
<td>Asset: Stakeholder</td>
</tr>
<tr>
<td>Asset</td>
<td>Cloud Element</td>
<td>Service</td>
<td>Asset: Process</td>
</tr>
<tr>
<td></td>
<td>Cloud Element</td>
<td>Pool</td>
<td>Asset: Data Centre</td>
</tr>
<tr>
<td></td>
<td>Cloud Element</td>
<td>IaaS</td>
<td>Asset: Process</td>
</tr>
<tr>
<td></td>
<td>Cloud Element</td>
<td>PaaS</td>
<td>Asset: Process</td>
</tr>
<tr>
<td></td>
<td>Cloud Element</td>
<td>SaaS</td>
<td>Asset: Process</td>
</tr>
<tr>
<td></td>
<td>Cloud Element</td>
<td>Resource</td>
<td>Asset: Host</td>
</tr>
<tr>
<td></td>
<td>Cloud Element</td>
<td>Hardware</td>
<td>Asset: Host</td>
</tr>
<tr>
<td></td>
<td>Cloud Element</td>
<td>Software</td>
<td>Asset: Code</td>
</tr>
<tr>
<td></td>
<td>Cloud Element</td>
<td>Location</td>
<td>Asset: Space</td>
</tr>
<tr>
<td></td>
<td>Cloud Element</td>
<td>Cloud Software Stack</td>
<td>Asset: Software Stack</td>
</tr>
<tr>
<td></td>
<td>Cloud Element</td>
<td>Development Env. / API</td>
<td>none[^1]</td>
</tr>
<tr>
<td></td>
<td>Cloud Element</td>
<td>Software Product</td>
<td>Asset: Code</td>
</tr>
<tr>
<td></td>
<td>Cloud Element</td>
<td>Data</td>
<td>Asset: Data</td>
</tr>
<tr>
<td></td>
<td>Cloud Element</td>
<td>Sticky Policy</td>
<td>Control</td>
</tr>
<tr>
<td>Actor</td>
<td>Direct Stakeholder</td>
<td>Data Subject</td>
<td>Asset: Human</td>
</tr>
<tr>
<td></td>
<td>Direct Stakeholder</td>
<td>Data Controller</td>
<td>Asset: Stakeholder</td>
</tr>
<tr>
<td></td>
<td>Direct Stakeholder</td>
<td>Cloud Provider</td>
<td>Asset: Stakeholder</td>
</tr>
<tr>
<td></td>
<td>Direct Stakeholder</td>
<td>External RestAssured Provider</td>
<td>Asset: Stakeholder</td>
</tr>
<tr>
<td></td>
<td>Direct Stakeholder</td>
<td>Data Consumer</td>
<td>Asset: Stakeholder</td>
</tr>
<tr>
<td></td>
<td>Direct Stakeholder</td>
<td>Online Service Developer</td>
<td>Asset: Stakeholder</td>
</tr>
<tr>
<td></td>
<td>Direct Stakeholder</td>
<td>Cloud Administrator</td>
<td>Asset: Stakeholder</td>
</tr>
<tr>
<td></td>
<td>Direct Stakeholder</td>
<td>IaaS Operator</td>
<td>Asset: Stakeholder</td>
</tr>
<tr>
<td></td>
<td>Direct Stakeholder</td>
<td>SaaS Operator</td>
<td>Asset: Stakeholder</td>
</tr>
<tr>
<td></td>
<td>Direct Stakeholder</td>
<td>Cloud Support</td>
<td>Asset: Stakeholder</td>
</tr>
<tr>
<td>Mechanism</td>
<td>Cloud Element</td>
<td>RestAssured Platform</td>
<td>Control</td>
</tr>
<tr>
<td></td>
<td>Cloud Element</td>
<td>Data Protection Component</td>
<td>Control</td>
</tr>
<tr>
<td>Threat</td>
<td>none</td>
<td>none</td>
<td>Threat</td>
</tr>
</tbody>
</table>

[^1]This is not need in the current phase of the project, but we may look at this again in the second half of the project.
4 Modelling Tools

4.1 CSAP-tool

The CSAP-tool represents a graphical editor for the creation and instantiation of Cloud System Analysis Patterns (CSAP) (see also Sec. 3.1.2). To this, the tool provides two editors in form of the Designer-editor and the User-editor (see Fig. 4.1). These editor are explained in the following sections.

**Figure 4.1: Editor modes of the CSAP-tool**

4.1.1 The Designer-editor

The Designer-editor of the CSAP-tool enables the users the creation of user-specific Cloud System Analysis Patterns (CSAP). Here, the users can build a user-specific CSAP on the basis of

- a CSAP that is oriented towards the original CSAP ([?]) (see Fig. 4.1) or
- an empty CSAP (see Fig. 4.2).

The empty CSAP contains only an Indirect Environment, a Direct Environment and a Cloud (see Fig. 4.2).

Users can now create their own CSAP by adding Indirect Stakeholders, Direct Stakeholders and Cloud Elements to the CSAP. Furthermore, associations between Direct Stakeholders and Cloud Elements as well as between Direct Stakeholders among each other can be created. If the original CSAP is used as base for creating a user-specific CSAP, already existing CSAP-elements can be modified and/or deleted.

The Figure 4.3 shows the addition of a new Cloud Element to an empty CSAP. After adding the Cloud Element, its name can be defined (see Fig. 4.4). In the Designer-editor this name of a CSAP-element defines also its instance type (see Fig. 4.5).
Figure 4.2: Creation of a user-specific CSAP on basis of an empty CSAP

Figure 4.3: Addition of a Cloud Element to a user-specific CSAP
Figure 4.4: Defining the name of an added Cloud Element

Figure 4.5: Equality regarding the name and instance type of an added Cloud Element
The adding of Indirect Stakeholders and Direct Stakeholders follows the same procedure as the adding of Cloud Elements.

### 4.1.2 The User-editor

In the User-editor any defined CSAP can be instantiated. The instantiation of a CSAP regarding the ReAs-CSAP is described in detail in Section 3.1.2.2. The Figure 4.6 shows the instantiation of a Cloud Element of the instance type Data during the instantiation of a ReAs-CSAP. Because this Cloud Element-instance shall specify the data of customers, it gets the name Customer Data. The instantiation of the Cloud Element is marked by displaying the type name Data surrounded by angle brackets under the name of the Cloud Element (see Fig. 4.7). For a subset of the properties of the instantiated Cloud Element the corresponding values can be specified in the according dialog during its instantiation (see Fig. 4.6). For all properties, except of the instance type, the corresponding values can be assigned in a property panel (see Fig. 4.8).

After the instantiation of the Cloud Element Customer Data, its incoming and outgoing associations are displayed with dashed lines (see Fig. 4.7). This dashed presentation indicates that the associated Cloud Elements and Direct Stakeholders have not been instantiated yet. In Figure 4.9 it can be seen that the association RestrictAccessTo is displayed again with a pulled through line, after the associated Cloud Element of the instance type Sticky Policy has been instantiated.

The instantiation of Indirect Stakeholders and Direct Stakeholders follows the same procedure as the instantiation of Cloud Elements.

The Indirect Environment, the Direct Environment and the Cloud can not be instantiated. For the Cloud its type (public, private, hybrid, community) has to be defined if the default value public does not fit (see Fig. 4.10).
Figure 4.7: Representation of the instantiated Cloud Element of the instance type Data in a ReAs-CSAP

Figure 4.8: Definition of Properties of the instantiated Cloud Element of the instance type Data
Figure 4.9: Representation of the association between the instantiated Cloud Elements of the instance types *Data* and *Sticky Policy*

Figure 4.10: Specification of the type of the Cloud
4.2 System Security Modeller

The System Security Modeller is a web-based graphical tool which allows a system designer or analyst to perform a design-time risk assessment of a system. This section describes the tool, and the extensions for RestAssured.

4.2.1 Architecture

The System Modeller UI is a web-based user interface for a system of back-end components. Figure 4.11 gives an overview of the high-level architecture of the constituent services and underlying triple store. The services are accessed through a REST API, either using the System Modeller UI, or some other application. The interface exposes selected parts of the models, such as collections of assets or threats (read only) or update methods to allow users to edit models. These calls are sanitised and passed to the underlying querier and validator components. A low-level semantic store component allows access to the actual triple store.

As shown in figure 4.11, the store contains one core model, and potentially multiple domain and system models (Section 3.2.1). All models are stored in separate graphs, allowing for easy import and export.

4.2.2 User Interface

Figure 4.12 shows the canvas of System Modeller with a validated model loaded. This is the end stage of developing a system model. To get here a series of steps are required:

- add assets to the model, giving them meaningful labels, and setting cardinalities (some assets by design are singletons e.g. the internet)
- connect the assets using the allowed relationship types from the domain model (depends on the types of the assets being connected)
- validate the model to generate inferred assets (these are implicitly defined through the relationships between other assets)
- set trustworthiness levels for asset attributes, and impact levels for misbehaviours
- calculate risk levels to determine which threats pose the highest risk
Figure 4.12: The System Modeller canvas

Figure 4.13: The Threat Editor window
The threat editor, as shown in Figure 4.13, shows the details of a threat, including the pattern (see Section 3.2.2), causes and (primary and secondary) effects and control strategies. The user can propose controls or entire control strategies from here, or mark a threat as acceptable. The secondary effects tab (which is collapsed in the figure as it contains many secondary effects) contains a link to the Misbehaviour Explorer (Figure 4.14) for each misbehaviour. This connection between threats and misbehaviours allows the user to explore the cause and effect chain as explained in Section 3.2.3. This can help to prevent a threat further down the chain by managing the risks for a threat that could potentially cause it. If for instance a threat causes a web service to be unavailable because the data it uses is unavailable, it might help to protect the data so that the web service never becomes unavailable for that reason.

Figure 4.14: The Misbehaviour Explorer
4.2.3 RestAssured Extensions

As mentioned in section 3.2.1, the model hierarchy and algorithms are based on previous projects. The same is true for System Modeller and its components. RestAssured expands our models by adding support for risk level calculation. In previous projects, controls applied to assets were a binary concept:

1. all controls existed and a control strategy’s requirements were satisfied leading to a threat being blocked

    OR

2. some controls were missing so the threat stayed a vulnerability or had to be accepted

In reality, it’s much more complex to gauge the risks to a system. As explained in section 3.2.3, threats can influence each other and controls can be more or less effective. This depends on a multitude of other factors such as other threats to the same asset, the trustworthiness of the control, the strength of a control when applied to an asset of that type, the likelihood a threat applies, etc.

To address this, we have introduced a number of new concepts, that capture information on a domain and system level and allow to infer risk levels and ultimately assess the risk to the system as a whole. The concepts we introduced are:

- **Likelihood**
  As explained in Section 3.2.3, this concept describes the likelihood of something “going wrong”, i.e. not as planned or expected. This is used for misbehaviours and threats, describing how likely they are to occur. Likelihood is a scale, currently covering five different levels (“Very Low” to “Very High”). As shown in Table 3.1, likelihood maps directly to trustworthiness.

- **Trustworthiness**
  This concept is the inverse of Likelihood, meaning that if something is unlikely to go wrong, its Trustworthiness is high (Table 3.1). This applies to multiple concepts:
  - Mitigation levels capture the effectiveness of a particular Control in preventing a Misbehaviour when applied to the same Asset.
  - Control Strategies have a “Blocking Effect”, which encodes how well the control strategy blocks a threat. This now takes into account that some Control Strategies might be more effective and preferable to others. It provides a starting point for the potential ordering of Control Strategies to minimise the number of Controls required or maximise the blocking effect using only a given number of Controls.
  - Assets have Trustworthiness Attributes that are asserted by the user and pre-populated using default values from the domain model. They capture the reliability of assets or the actors and ultimately allow the tool to make assumptions about the propensity for threats to arise in the system.
  - Controls can have a strength, which can be used to determine the strength of a Control Strategy.

- **Impact**
  This is asserted by the user of System Modeller, describing the impact a misbehaviour at a particular asset would have on the business interests (Table 3.3). As with Trustworthiness, this concept allows filtering of threats by importance: high impact threats would have fatal consequences for a business and have to be addressed immediately, while lower impact threats may even be tolerated.

- **Risk**
  This is a purely inferred value, obtained through complex calculations involving the other concepts.
above. A much more detailed description of the calculation can be found in section 3.2.5. The Risk Level represents a classification mechanism for threats which can be used to infer the overall risk level of the system.

4.2.4 Workflow

System Security Modeller defines a workflow to be followed by the system designer or risk analyst:

1. First the system model is created using the system designer or risk analyst’s expert knowledge.

2. Then the system model is validated. Validation assigns default values to the trustworthiness levels for the entry points of the threats discovered during validation (Section 3.2.2.2), and default impact levels for the misbehaviours caused by those threats. These default values are specified in the domain model.

3. The default assignment of impact values does not take into account which are the primary assets in the system, and which are supporting assets. The system designer must identify the primary assets and adjust the impact levels accordingly. Figure 4.16 shows how this is done in the Misbehaviours tab for an Asset.

4. The default assignment of trustworthiness levels also cannot know all the details of the system. For example it does not automatically know whether the users of an asset should be regarded as trustworthy or not. The system designer or risk analyst can enter this information by changing the assigned values as shown in Figure 4.17.

5. Changing impact values and trustworthiness levels invalidates the computed threat and misbehaviour likelihoods, and the resulting risk levels. This is shown by the indicator turning from green to red as shown in Figure 4.15. Clicking the button then starts the recalculation of likelihoods and risk levels. Once the calculations have finished, the UI is updated with the latest data and can be explored as before. The overall risk level is displayed in the “Highest risk” widget to the right of the risk calculation button.

6. At this stage System Security Modeller has identified the threats to the system and the corresponding risk levels. The system designer or risk analyst must then decide whether the risk levels are acceptable. If not, they must select from the available control strategies (Section 3.2.2.3) the most appropriate controls. To do this the system designer can explore the root causes of each misbehaviour (Figure 4.14) and select either:

   • control strategies that eliminate the root cause threats, or
   • control strategies for intermediate threats that block the effect propagation.

After selecting controls the likelihoods and risk levels must be recomputed (Figure 4.15).

7. This process is repeated until the overall risk level is acceptable.
Figure 4.16: Updating the Impact of a Misbehaviour

Figure 4.17: Updating the level of a Trustworthiness Attribute of an Asset
5 Conclusion

In this deliverable we have reported the outcomes from WP7 during the first reporting period of RestAssured. The major achievements were:

- Development of the theoretical underpinnings for two design-time risk assessment tools.
- Development of prototypes of those two tools: CSAP and System Security Modeller.

In Section 3 we describe in detail the threat models that underpin our two approaches to risk modelling. The first is a high-level context-oriented approach that employs patterns to identify the relationships between the system and the stakeholders (Section 3.1). Here stakeholders include Data Subjects, Data Controllers, and Cloud Providers.

The second approach employs graph-based models to model the assets of a system and the relationships between them (Section 3.2). A domain specific catalogue of threat patterns captures the possible threats within a domain, and through pattern matching the specific threats in a system model can be identified. Moreover, by specifying the trustworthiness levels of certain attributes of system assets, the likelihood of those threats can be computed. This includes two mechanisms by which the effects of threats can be propagated: automatic secondary effect chaining, and loss of trustworthiness effects (Section 3.2.3.1). Finally, by specifying the impact levels for primary asset misbehaviours, risk levels can be computed.

The two approaches have been implemented in the CSAP (Section 4.1) and System Security Modeller (Section 4.2) tools respectively. These tools are extensions of tools developed in previous projects, and as a consequence they are now reasonably mature. The RestAssured enhancements to System Security Modeller have, for example, been demonstrated to an organisation outside the RestAssured consortium, which generated a lot of internal interest within that organisation.

In Table 3.6 we have defined a mapping between the constructs from CSAP and the corresponding constructs in System Security Modeller. However, as described in Chapter 4 of deliverable D3.2, during the first phase of the project we have realised that much tighter integration of CSAP and System Security Modeller would be beneficial. We propose therefore to continue the development of the tools within Task 7.3, with a view to developing a tool suite that can address risk assessment within supply chains for cloud based systems. Specifically, we initially envisaged a methodology with the sequential application of the tools. First the context of a system is defined in CSAP, and then the details are refined in System Security Modeller. However, OCC, one of the use case partners, identified some additional requirements to support the situation where software providers supply pre-tested software components. In such a scenario the software supplier would provide a detailed model of the System Security Modeller and this would then be imported into CSAP by the service operator, before the resultant composite system in CSAP is then further refined in System Security Modeller. (See Chapter 4 of D3.2 for more details of this proposed methodology.)

To address these additional requirements further development of CSAP and System Security Modeller will be undertaken. One of the key technical challenges is the refinement of components based upon imported sub-models (Section 4.3 of D3.2), and the degree to which the relationships between assets within sub-models and assets within the parent model can be inferred. We plan to investigate what information can be inferred and what must be specified in the sub-models by the component suppliers, and to extend CSAP and System Security Modeller to provide the necessary mechanisms for the development of composite models.

In Section 2.3 we outlined the relationship between run-time risk assessment and Adaptation. The run-time risk assessment fuses both the design-time system model from System Security Modeller, and run-time information from the Adaptation components. The APIs required for this will be developed in Task 7.3. Initial work has already started, and this will continue as the risk assessment tools and Adaptation components are further developed. Task 7.3 will also investigate new and emerging privacy enhancement technologies. These represent possible mitigation techniques for threats to privacy. In the RestAssured
domain model these will be incorporated as control strategies for privacy threats (Section 3.2.2.3). Future work in Tasks 7.1 and 7.3 will build on this to look at cost parameters for security controls and security assurance metrics.