predict
prioritise
prevent

TRE$_S$PASS

Technology-supported Risk Estimation by Predictive Assessment of Socio-technical Security

Deliverable D2.4.1

TRE$_S$PASS information system

Project:	TRE$_S$PASS
Project Number:	ICT-318003
Deliverable:	D2.4.1
Title:	TRE$_S$PASS information system
Version:	1.0
Confidentiality:	Public
Editor:	Mike Osborne
Cont. Authors:	M. Osborne, A. Tanner, L. Fichtner, W. Pieters
Date:	2016-10-31

SEVENTH FRAMEWORK PROGRAMME
Part of the Seventh Framework Programme
Funded by the EC-DG CONNECT

Members of the TRE$_S$PASS Consortium

1.	University of Twente	UT	The Netherlands
2.	Technical University of Denmark	DTU	Denmark
3.	Cybernetica	CYB	Estonia
4.	GMV Portugal	GMVP	Portugal
5.	GMV Spain	GMVS	Spain
6.	Royal Holloway University of London	RHUL	United Kingdom
7.	itrust consulting	ITR	Luxembourg
8.	Goethe University Frankfurt	GUF	Germany
9.	IBM Research	IBM	Switzerland
10.	Delft University of Technology	TUD	The Netherlands
11.	Hamburg University of Technology	TUHH	Germany
12.	University of Luxembourg	UL	Luxembourg
13.	Aalborg University	AAU	Denmark
14.	Consult Hyperion	CHYP	United Kingdom
15.	BizzDesign	BD	The Netherlands
16.	Deloitte	DELO	The Netherlands
17.	Lust	LUST	The Netherlands

Document History

Authors		
Partner	Name	Chapters
IBM	Mike Osborne	all
IBM	Axel Tanner	2, 3, 4
TUD	Wolter Pieters, Laura Fichtner	5

Quality assurance		
Role	Name	Date
Editor	Mike Osborne	2016-10-31
Reviewer	Dieter Gollmann	2016-10-15
Reviewer	Rolando Trujillo	2016-10-15
Task leader	Mike Osborne	2016-10-31
WP leader	Mike Osborne	2016-10-31
Coordinator	Pieter Hartel	2016-10-31

Circulation	
Recipient	Date of submission
Project Partners	2016-09-30
European Commission	2016-10-31

Acknowledgement: *The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 318003 (TRESPASS). This publication reflects only the authors' views and the Union is not liable for any use that may be made of the information contained herein.*

Contents

List of Figures

Management Summary

This document describes the prototype TRE$_S$PASS information system that supports the TRE$_S$PASS process and tools in modeling risk across complex social-technical environments. These tools and process require an information repository and management framework that allows complex scenarios to be iteratively analysed in such a way that the intermediate and final results are auditable. Put another way, the TRE$_S$PASS information system provides the glue between the Attack Navigator Map (ANM) tool, the data, model and analytics tools. The task of developing an information system that provides the different TRE$_S$PASS tools with the combination of data that they require is complex. During the early project periods it was planned that the information system be based on traditional relational data schemes implemented in database technology. It became apparent that this approach had a number of drawbacks, in particular the use of static structures in heterogeneous scenarios, the lack of built-in versioning for model iterations and finally the lack of simple techniques to add programmable logic.

The final strategy was to use a 100% text based approach. For example JSON and XML are used for data persistence, a source code repository tool is used for versioning and the simplification of data exchange and finally the use of the text based Python scripting language for combining data with programmable logic. The latter allows practitioners to persist their SME knowledge.

Key takeaways:

- The TRE$_S$PASS information system is a storage platform for data gathered through the tools and approaches developed in WP2
- The system provides a set of APIs for interfacing and supporting configuration through the Attack Map Navigator.
- The system provides a set of APIs for supporting the extension and decoration of TRE$_S$PASS models and constructs.
- A method is presented that allows domain specific knowledge to be captured and combined with data in programmable logic.
- The system provides a set of APIs for supplying global, domain and instance data to analytics tools.
- Finally, the system uses a distributed code repository technology for simplifying the the versioning and the sharing of data.

1. Introduction

1.1. Goals

The focus of this deliverable is to address the topic of creating an information system used to store the social and technical data gathered during a TRE$_S$PASS risk analysis and provide interfaces between the TRE$_S$PASS tools and components that require different perspectives at different parts of the TRE$_S$PASS process. Requirements for the information system are defined in The TRE$_S$PASS Project, D2.1.1 (2013).

1.2. Choices made

We chose to use a source code versioning system for the base management platform and not a database platform. This gave us:

- a simple way to versioning model iterations
- a simplified way to share and merge data and logic patterns
- built-in access control features
- built-in automation features.

We chose to move all data and logic to text based solutions for compatibility with source code versioning systems.

We chose the Python scripting language for the text based addition of programmable logic and because it is widely known. This is used for the attack pattern augmentation and annotation logic used by the Attack Pattern Library (APL).

We chose to provide all functionality through RESTful APIs rather than database interfaces.

We provide support for three different categories of data like generic, domain specific and instance specific.

We chose to clearly differentiate between source data and derived or generated data.

We used the RDF data format for persisting social and technical background knowledge.

1.3. Foreground and background

The TRE$_S$PASS Information System is completely in foreground IP with the exception of:

- the Python programming language,
- the Python libraries CherryPy[1], RDFLib[2] and Dulwich[3] .

1.4. Document structure

Chapter 2 gives an overview of the TRE$_S$PASS Information System, while Chapter 3 gives details by an examplary walkthrough through the system and the related resources. Chapter 4 describes how the TRE$_S$PASS Information System was evolved and validated. An alternative approach to handling data related to socio-technical cyber risk is discussed in Chapter 5. We close with conclusions in Chapter 6.

Furthermore, Appendix A explains how to gain access to the prototype, Appendix B gives details of the API to the TRE$_S$PASS Information System with example calls and responses for reference.

[1] CherryPy http://cherrypy.org
[2] RDFLib https://github.com/RDFLib/rdflib
[3] Dulwich https://www.dulwich.io

2. The TRE$_S$PASS Information System

This chapter describes the TRE$_S$PASS Information System that enables the storage of the data relevant for a specific TRE$_S$PASS modeling instance into one common data space.

2.1. Overview

The TRE$_S$PASS process consists of many different steps and tools as shown and explained in The TRE$_S$PASS Project, D5.4.2 (2016).

For a specific instance of this process, i.e., a specific instance of a scenario or use case, it is advisable to keep all involved source, intermediate and result data in a common repository for grouping and managing its life cycle, also in order to support traceability throughout the process.

As the data and evaluations essentially are iterative, as they profit from progressive modelling and refinement, input and result data will exist in different iterations during the life cycle of a risk analysis in the TRE$_S$PASS process. Ideally, these iterations should still be comprehensible (and possibly auditable) in hindsight.

A similar problem exists in software engineering where versioning systems support developers in keeping the history (and attribution) of pieces of code. Using a version control system therefore seems to be very suited also to use in the TRE$_S$PASS process.

Using a standard version control system also means that information ideally should be in text form, like source code, as this allows, at least in principle, easy comprehension by humans, efficient storage of different versions as well as powerful highlighting of changes between versions.

2.2. Implementation

Base inputs to the working of the TRE$_S$PASS Information System include the socio-technical background data, but can also include specific technical data like model excerpts generated automatically (as e.g., described in The TRE$_S$PASS Project, D2.2.2 (2015) for the cloud use case).

Fig. 2.1 gives a summary and overview of the Information System, its content and interactions with the TRE$_S$PASS tools.

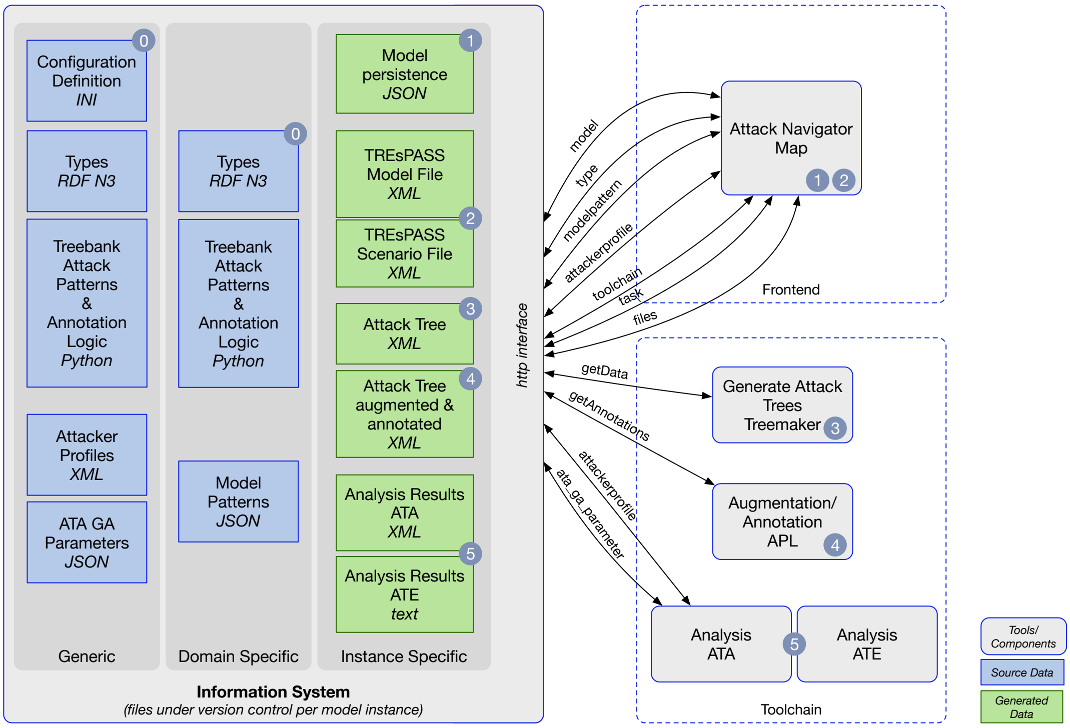

Figure 2.1.: Overview of the TRE$_S$PASS Information System, its content and interactions with the TRE$_S$PASS tools.

In multiple source files, shown in blue in the figure, the background knowledge is captured and made available through the TRE$_S$PASS Information System's http-based interface to the various tools (for details of the API calls see Appendix B):

Types (RDF N3) Social and technical background knowledge is captured in the form of RDF (W3C, 2004) statements. This is used as base stating the available types during the model building with the Attack Navigator Map (ANM) ('types' call) as well as for background queries by Treemaker ('getData' calls). This background knowledge can be combined from different files, i.e., for generic data and domain specific data.

Treebank Attack Patterns & Annotation Logic (Python) Data for the augmentation of generated attack trees as well as the logic to annotate specific attack steps is stored in the form of Python patterns and code. These files contain functions for augmentation and annotation that will be used by the Attack Pattern Library (APL) calls ('getAnnotations' call). More background for the APL can be found in The TRE$_S$PASS Project, D5.3.2 (2015). These functions can be combined from multiple source files so the corresponding knowledge can be differentiated into generic and domain specific parts.

Attacker Profiles (XML) The various available attacker profiles are captured in XML form in this source file. These can be used for selection in the ANM as well as for analysis

in the Attack Tree Analyzer (ATA). ANM also has the possibility to create and store new attack profiles through the 'attackerprofile' call.

ATA GA Parameters (JSON) Source for various collections of parameters for the genetic algorithm part of the Attack Tree Analyzer (ATA), accessible via the 'ata_ga_parameter' call to the TRE$_S$PASS Information System.

Model Patterns (JSON) During the creation of models, patterns of components might be helpful as higher-level building blocks for models. Through 'modelpattern' call to the TRE$_S$PASS Information System these patterns can be queried and stored.

These files are described in more detail in Section 3.2.

A typical process flow will involve the TRE$_S$PASS Information System and create intermediate and result files (shown in green in Fig. 2.1) in the following way:

1. The *Attack Navigator Map* is used to create a model system: during the initialisation of a model the Attack Navigator Map (ANM) will interact with the TRE$_S$PASS Information System to first establish a new model context defined by a unique *modelID* ('model' call). At that time, generic and domain-specific source files will be copied into a model-specific container (=directory with version control) to keep all data for a specific model instance together as context. The ANM can use the defined types ('types' call), model patterns ('modelpattern' call) and attacker profiles ('attackerprofile' calls). The TRE$_S$PASS Information System also serves as a persistence container for the browser-based ANM frontend UI ('model' call, stored in a *Model persistence* JSON file). Toolchains can be selected and started via the 'toolchain' call, status and result (or error information) will be available through the corresponding 'task' call.

   The ANM will export the created model scenario into the *TRE$_S$PASS Model File* and *TRE$_S$PASS Scenario File* as required for input to Treemaker.

2. Treemaker reads model and scenario files and creates an attack tree for the given scenario. To get more information for specific model components, Treemaker can use the 'getData' call to the TRE$_S$PASS Information System, e.g. to understand whether a specific component is a physical component (i.e., persistent, so could be stolen by carrying away the component) or a virtual component (i.e., non-persistent and therefore could be stolen by cloning). The result of running Treemaker is the *Attack Tree* file.

3. This attack tree is augmented and annotated by the APL, using the 'getAnnotations' call to the TRE$_S$PASS Information System, creating as a result an augmented and annotated version of the attack tree ('Attack Tree augmented & annotated').

4. Subsequent analysis tools like the Attack Tree Analyzer (ATA) and Attack Tree Evaluator (ATE) use this augmented and annotated attack tree and analyse it for the most important potential attack vectors, storing their results again in (XML or plain) text files.

5. The ANM then reads these and visualises the results of the analysis.

6. The security professional using the system can then decide whether the current results are satisfactory or whether to refine the model or change parameters for a new analysis cycle.

All of the relevant information is kept in the form of text files, to allow simple capture, versioning and sharing. The versioning specifically allows to keep track of all developments and changes of the model instance, allowing to trace results back to the source data used at the time.

The version control system *git*[1] is chosen as it is distributed and therefore does not require a central server while still allowing sharing and cloning via a central git repository if desired (e.g., with a gitlab[2] server). Such a master directory easily supports sharing and dissemination while supporting sensitive information through strong access control functionality. It is also available for free on all platforms and well-established in the developer community.

2.3. Summary

The TRE$_S$PASS information system serves as a container repository gathering all input and output data for a specific TRE$_S$PASS modelling instance, making it available to the required tools in the modelling and analysis process through http-based API calls, while keeping track of the version history of source and generated data.

This allows to keep the data for a modelling instance in a central place together with its dependency chain and history, allowing managing, branching and roll-back while providing traceability and auditability during the unfolding TRE$_S$PASS process.

[1] Git distributed version control system `https://git-scm.com`
[2] GitLab `https://gitlab.com`

3. Walkthrough using the TRE$_S$PASS Information System by example

This chapter describes the functionality of the TRE$_S$PASS Information System (TIS) using an example model of the ATM use case. For more details about the background to this use case, see The TRE$_S$PASS Project, D7.4.2 (2016).

As the TIS is a backend system that is tied to the Attack Navigator Map (ANM) as a frontend for the risk analysis in TRE$_S$PASS, we use the ANM as the context to describe the functionality. For the detailed information about the ANM itself, we refer to the manual that is available online following the 'manual' link after loading the ANM (see Fig. 3.2 lower left) or directly at https://docs.google.com/document/d/1Qp8nJgdvDespq1Q5zQcAT1SSTKK23m2XKMUKKmoKYQU/edit.

ANM and TIS are accessible online at https://trespass.itrust.lu/attack-navigator-map/index.html (for details about access and login see Appendix A).

The walkthrough consists of:

- Initial setup of the demonstration model system
- Discussion of the configuration files that support the creation of the model scenario
- Discussion of the files generated during the risk analysis with the TRE$_S$PASS tools.

3.1. Initial setup of the demonstration model system

Please follow these steps to prepare the demonstration scenario in the ANM

1. Download the ATM demo model from https://trespass.itrust.lu/tkb/tkb/ATM_Demo_Model
2. Access the ANM at https://trespass.itrust.lu/attack-navigator-map/index.html
3. You will find the ANM with an empty model like shown in Fig. 3.1
4. Use the button 'Create new map' on the right hand side of the ANM
5. Enter a new map title of your choice
6. Use the button 'Import model file' on the right hand side of the ANM
7. Select the just-downloaded file 'atm-model-file.xml'

8. If you are asked

   'A model with this id exists already: id-SJfWe72p-model Do you want to overwrite the existing one?'

   press 'Cancel', then for the following

   'Would you still like to load the file, but use a new id instead?'

   press 'OK'

9. ANM loads the model file and instantiates this model - see Fig. 3.2

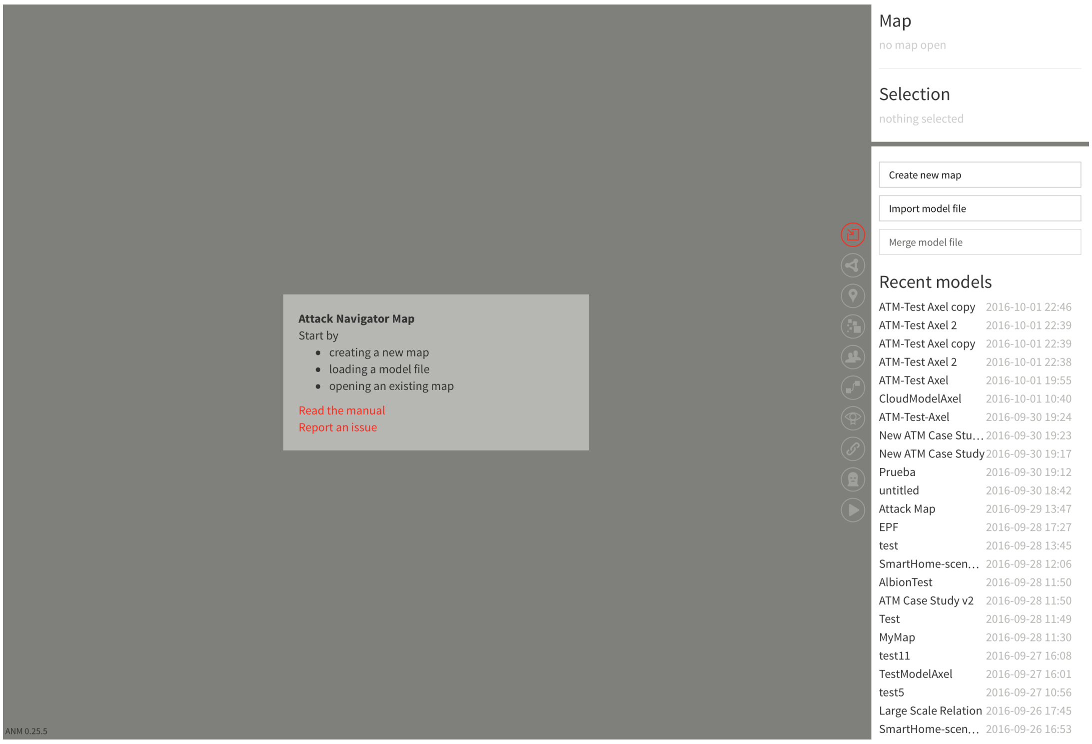

Figure 3.1.: View of the ANM after initial access - yet no model loaded.

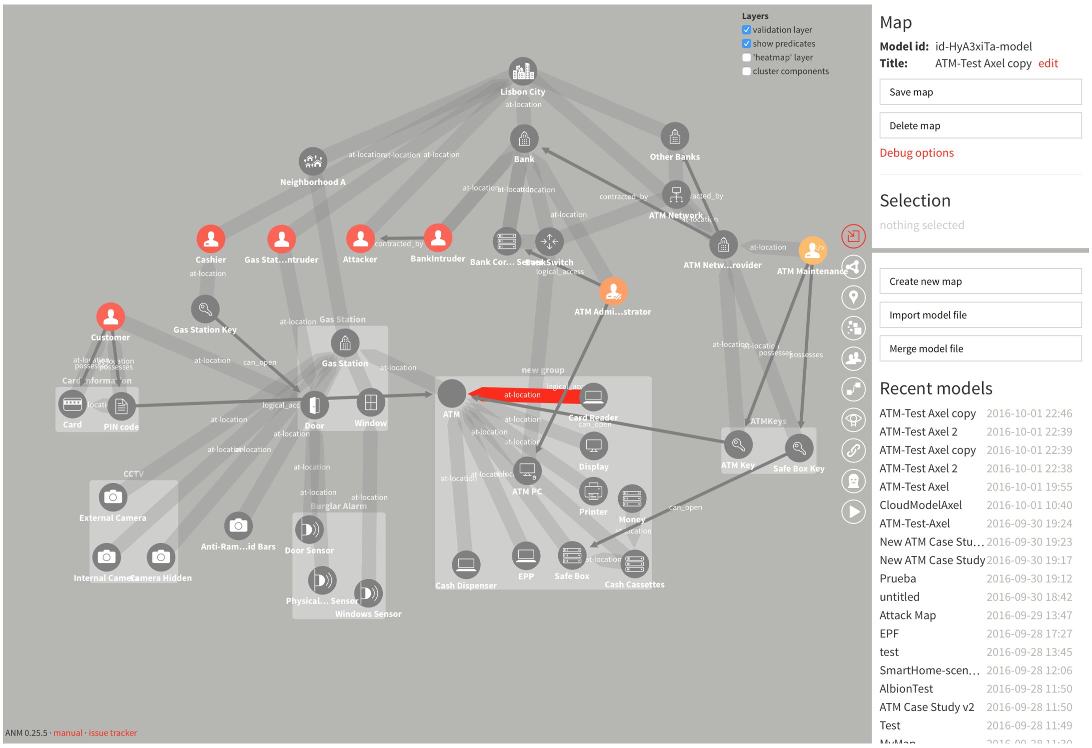

Figure 3.2.: View of the ANM after loading the ATM demonstration model.

3.2. Discussion of the configuration files

The configuration files for the configuration of the TRE$_S$PASS risk analysis tool are accessible easiest by clicking on 'Debug options' in the top right of the ANM (see Fig. 3.2), then clicking on 'edit knowledgebase files'.

3.2.1. Defining the base files - `config_source_file.ini`

This file, as shown in Fig. 3.3, defines for the information system the list of additional configuration files used.

All sections will be described in more detail in the following sections 3.2.2–3.2.7 (with exception of the `toolchains.json` that is used only internally for the definition of the toolchains that are exposed in the ANM).

Two sections allow the definition of multiple files, namely the `annotations` and the `data_types` sections as for these it seems especially appropriate to allow splitting the information into different domains.

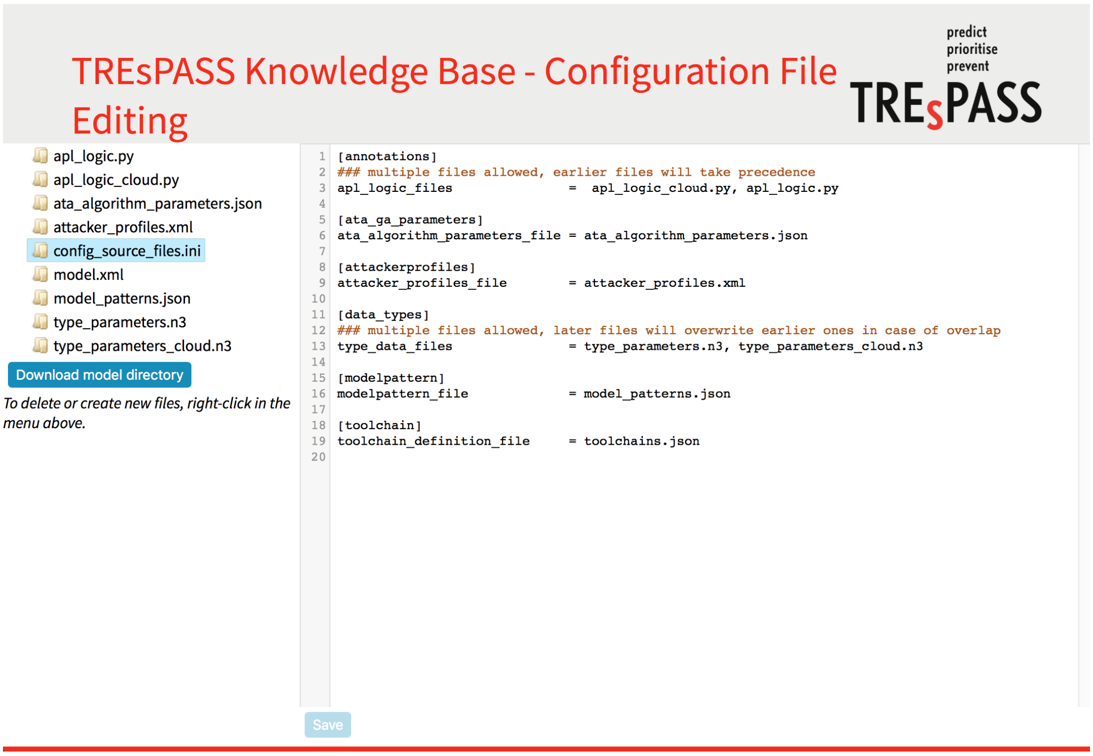

Figure 3.3.: View of the default `config_source_file.ini` file.

3.2.2. Defining used types, extend with additional data - `types_parameters.n3`

The part shown in Fig. 3.4 represents the type definition of a door. The file is semantically structured as RDF (W3C, 2004) information in N3 (W3C, 2011) notation.

The type `tkb:door` herein is marked to be a `tkb:type` - all such elements will be represented by the ANM for selection of new map elements. This definition also contains a set of attributes (marked through the `tkb_has_attribute` predicate) that the ANM is presenting to fill or select, like here the 'Name' and the 'Burglar Resistance Class'. Every such attribute then has a set of possible values.

This can be seen in the screenshot of the ANM in Fig. 3.5.

Additional types can be defined in this file and will be available in the ANM after using the button 're-fetch knowledgebase data' in the 'Debug options' of the ANM.

This file also serves as container for additional type-related properties that can be defined for use in Treemaker and/or the APL logic.

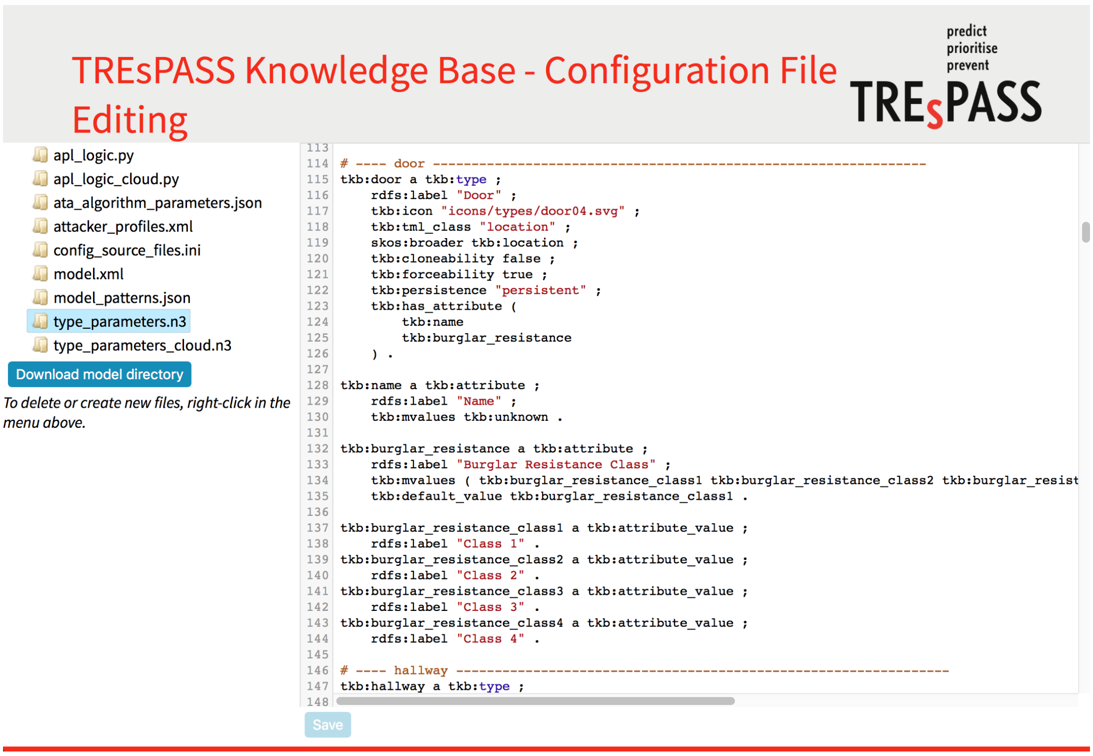

Figure 3.4.: View of the default `types_parameters.n3` file.

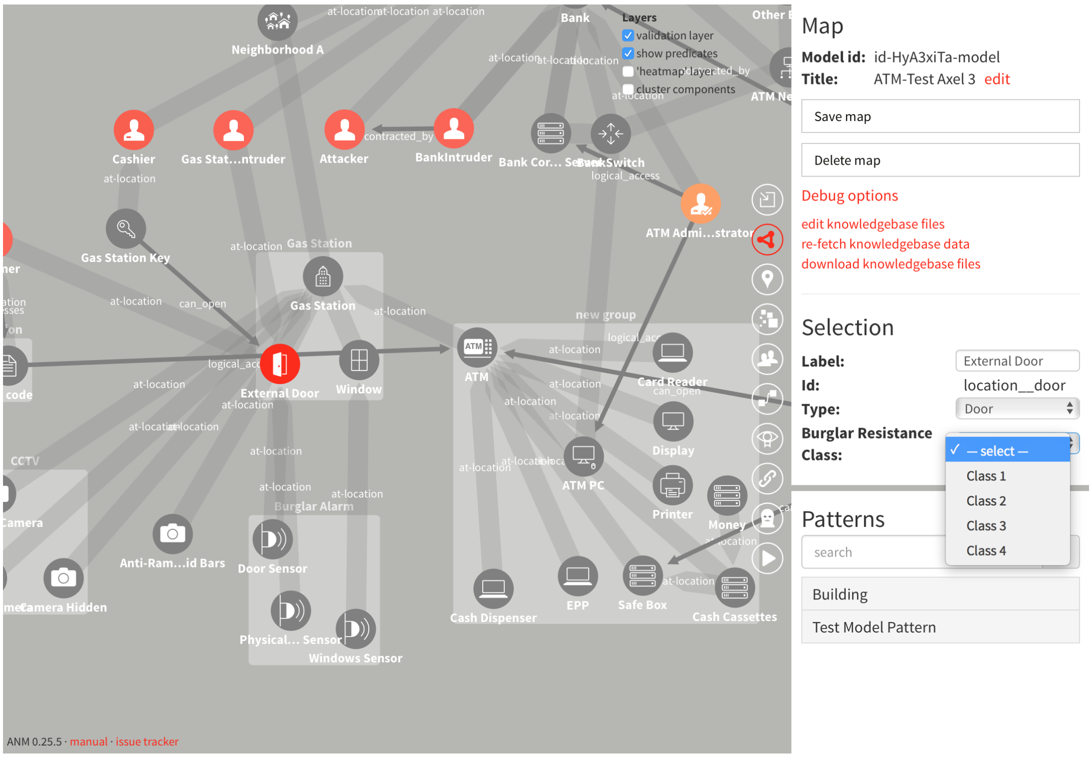

Figure 3.5.: Corresponding ANM representation to the type door as defined in the default `types_parameters.n3` file.

3.2.3. Capture of pre-made model patterns - `model_patterns.json`

Fig. 3.6 shows the default for the `model_patterns.json` file. This file captures ready made patterns that can be used inside the ANM as building blocks. It is best edited by saving existing maps as model patterns (per context menu of the ANM) rather than direct editing of this file.

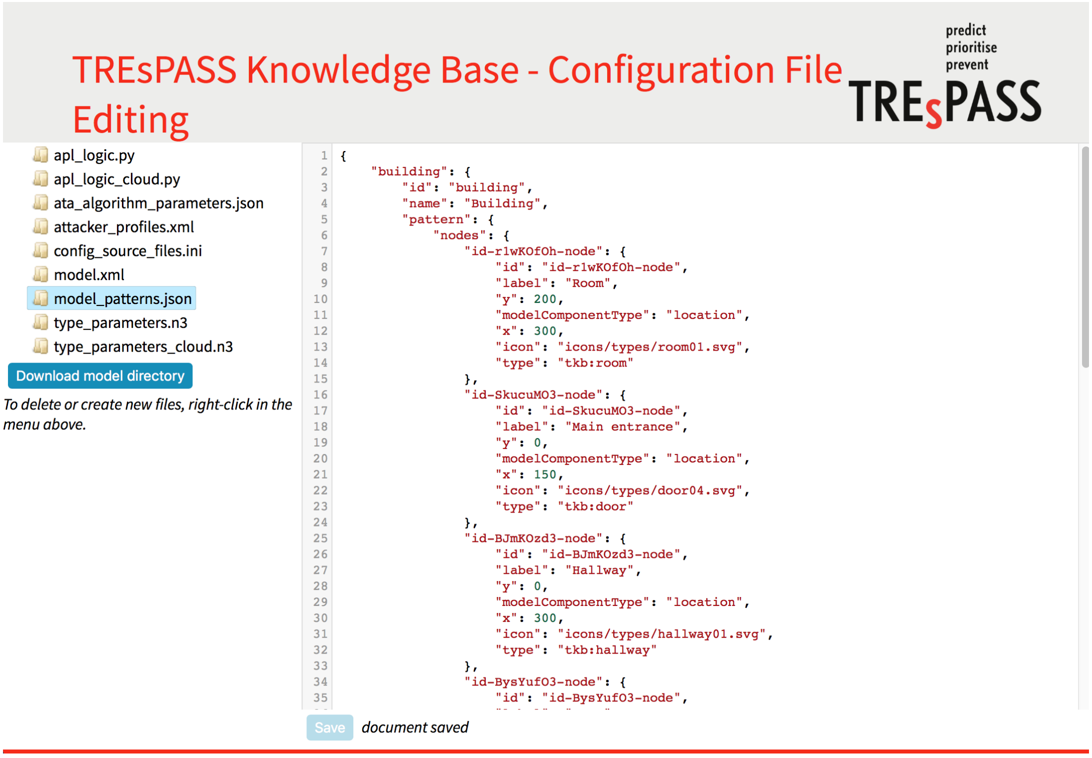

Figure 3.6.: View of the default `model_patterns.json` file.

3.2.4. The central model file - `model.xml`

This file represents the full information about the current state of the model system. It contains the model components in the main XML structure that is in turn used by the following analysis by Treemaker, but also contains the UI-state of the ANM to allow an easy sharing of a model.

It is usually not advisable to edit this file directly, rather to use the ANM to create the file in the correct syntax and semantic.

3.2.5. Defining the logic of augmentations and annotations - `apl_logic.py`

The APL part of the analysis consists of two steps:

- augmentation: after Treemaker creates the initial attack tree for the model, it is possible to extend this generated tree by refining the contents of the leaf labels, e.g., for domain-specific attack definitions.
- annotation: after the augmentation step, during the annotation values like probability of attack, required time and budget etc are attached to all leaf nodes as required for the subsequent analysis tools.

As these are very open steps, which cannot be predefined in generality and then merely configured, to allow for the required flexibility it was decided that these steps would be driven by modular programs with one of the easier to use scripting language. Due to its broad availability, large user-base and support Python was chosen for this task.

This is one of the cases where a potential domain-modularity is useful, therefore multiple APL logic scripts can be used — as to be configured in the `config_sources_file.ini` (see Sec. 3.2.1). During the call to the APL by the backend, the scripts configured here will be called in sequence for all leaf nodes of the generated attack tree, once for augmentation and once for annotation.

A snippet for one of the demo scripts can be seen in Fig. 3.7. An array of augmentator functions is defined here (and can be extended), here a simple `default_augmentator`. All functions defined here would be called in turn for every leaf node of the generated attack tree until the first one matches, i.e., returns a non-`None` result. Here the `default_augmentator` is defined to look through an array of potentially matching leaf-label patterns, then returning a corresponding augmented XML snippet that would replace the given leaf node.

Similarly, Fig. 3.8 shows an example for annotation handling by the APL script. Here two annotator functions are defined, one handling a class of '`MAKE ...`' attack leaves representing social engineering steps, one handling the default case when no more specific pattern has been found, thereby setting the default annotations (as annotations are required for the use of the later analysis tools).

In cases where no fitting annotation for a certain attack tree leaf-label is found, the APL will return an error stating all leaf-labels, for which no annotator is found. As an example, see the error messages shown during the tools run in Fig. 3.9, when the `default_annotator` is removed from the `annotator_functions` in Fig. 3.8, i.e., when using

```
annotator_functions = [make_in_annotator]
```

instead in `apl_logic.py`. In case of such errors, the APL script(s) need to be extended to also cover the leaf-labels shown in the error message.

A more complex example how to use available vulnerability data in APL decisions during annotation can be found as part of the default `apl_logic_cloud.py`.

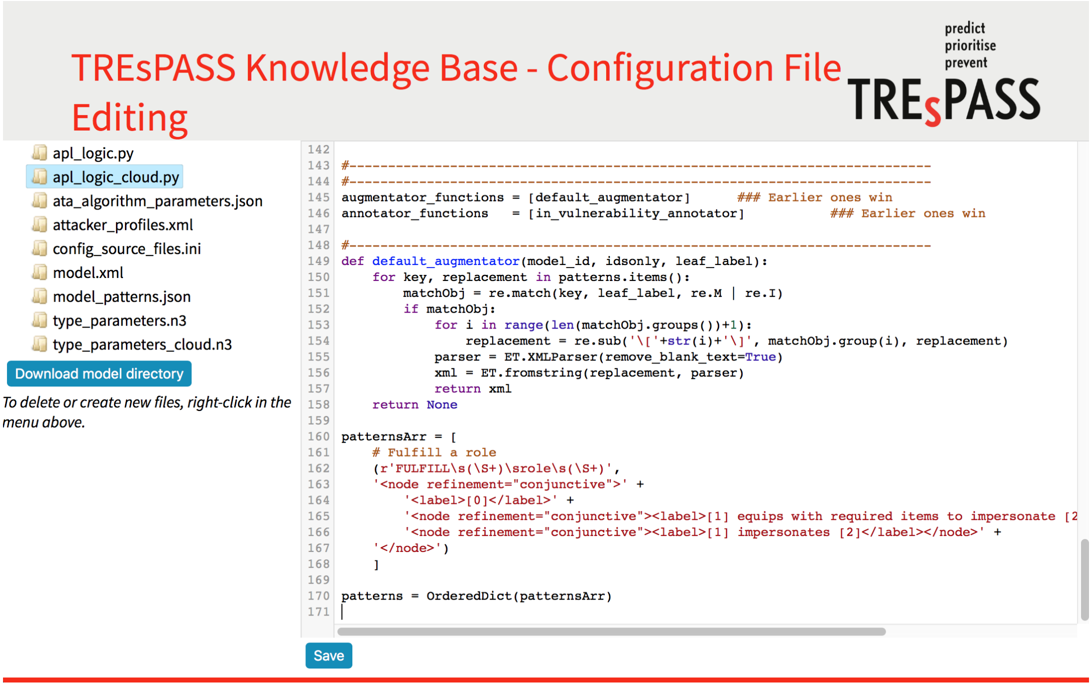

Figure 3.7.: Snippet of the default `apl_logic_cloud.py` file as example for augmentation handling.

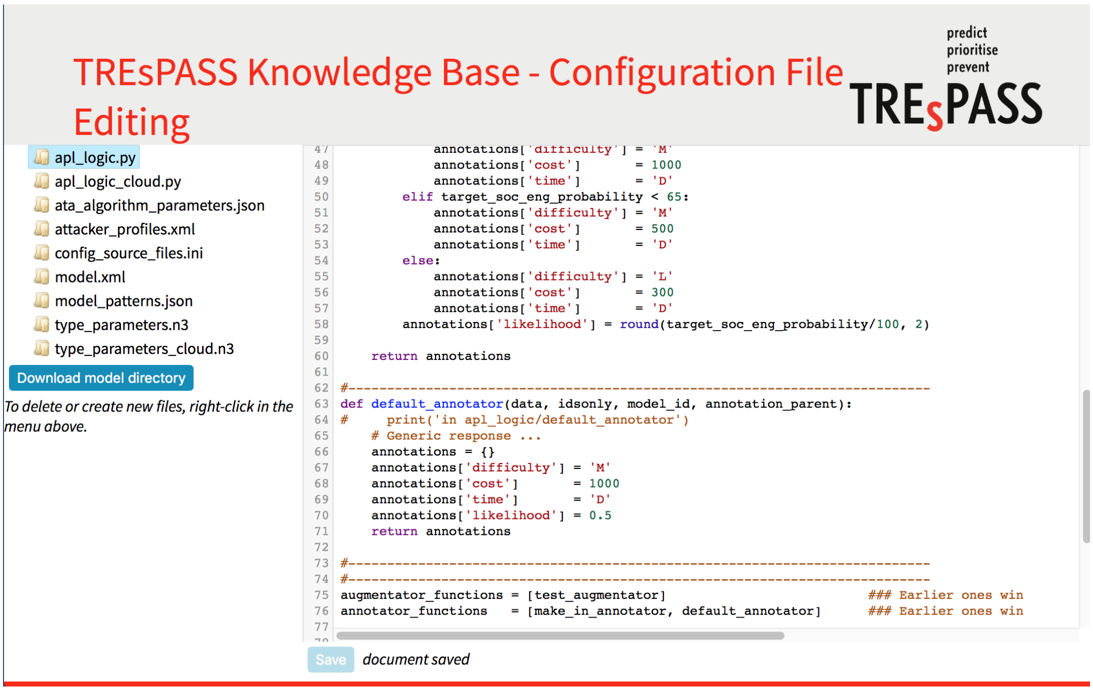

Figure 3.8.: Snippet of the default `apl_logic.py` file as example for annotation handling.

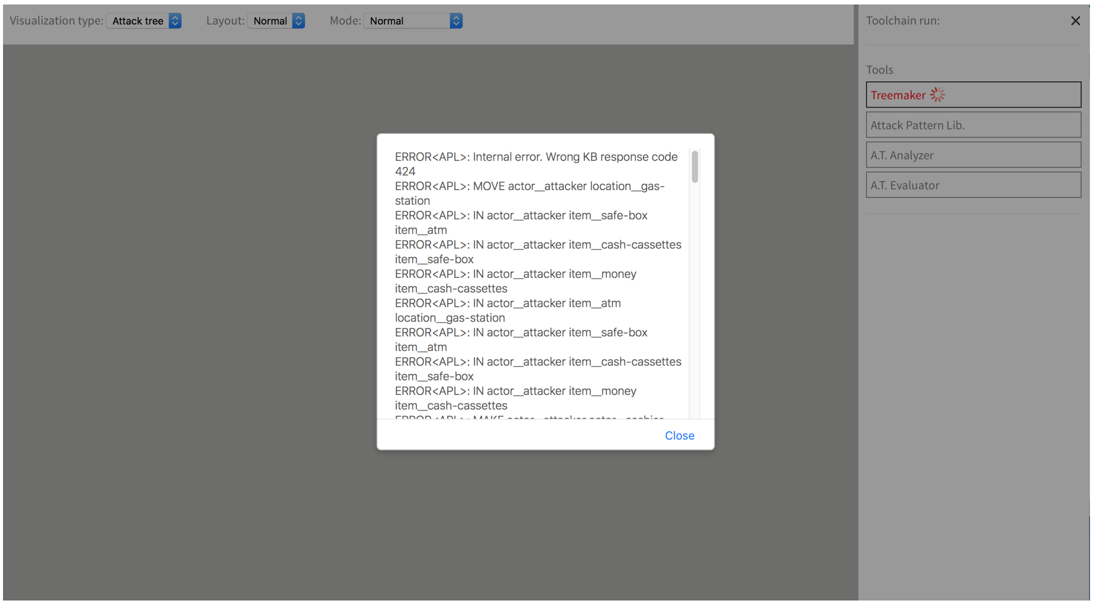

Figure 3.9.: When no fitting annotation can be found for certain attack tree leaf-labels, error messages for these labels will be shown when the APL is run as part of the toolchain.

3.2.6. Defining attacker profiles - `attacker_profiles.xml`

The current set of defined attacker profiles follows Intel's threat agent risk assessment classification (Rosenquist, 2009), but can be extended either via this file or the ANM UI (see Fig. 3.11).

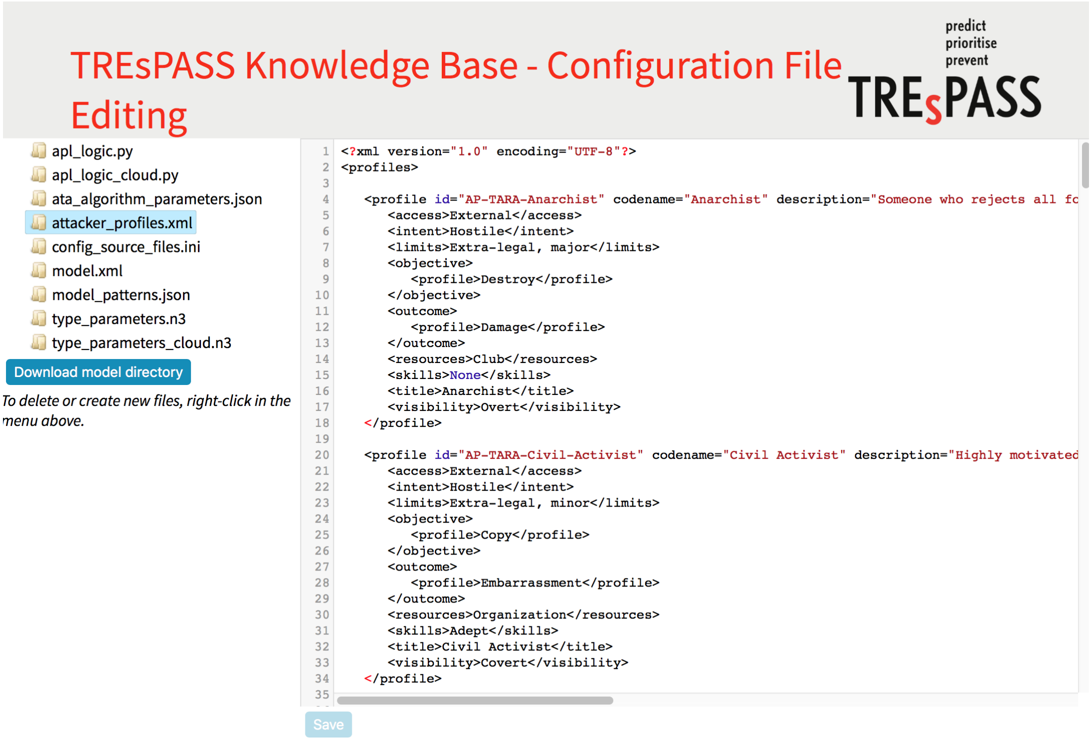

Figure 3.10.: View of the default `attacker_profiles.xml` file.

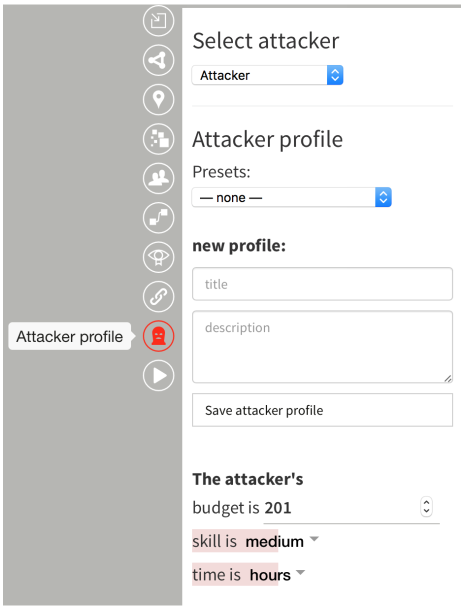

Figure 3.11.: The ANM allows to modify the attacker profiles in its UI.

3.2.7. Additional parameter sets for running the ATA analysis tools - `ata_algorithm_parameters.json`

This file, as shown in Fig. 3.12, specifies sets of parameters especially required by the run of the Attack Tree Analyser (ATA) analysis tool by partner Cybernetica (CYB).

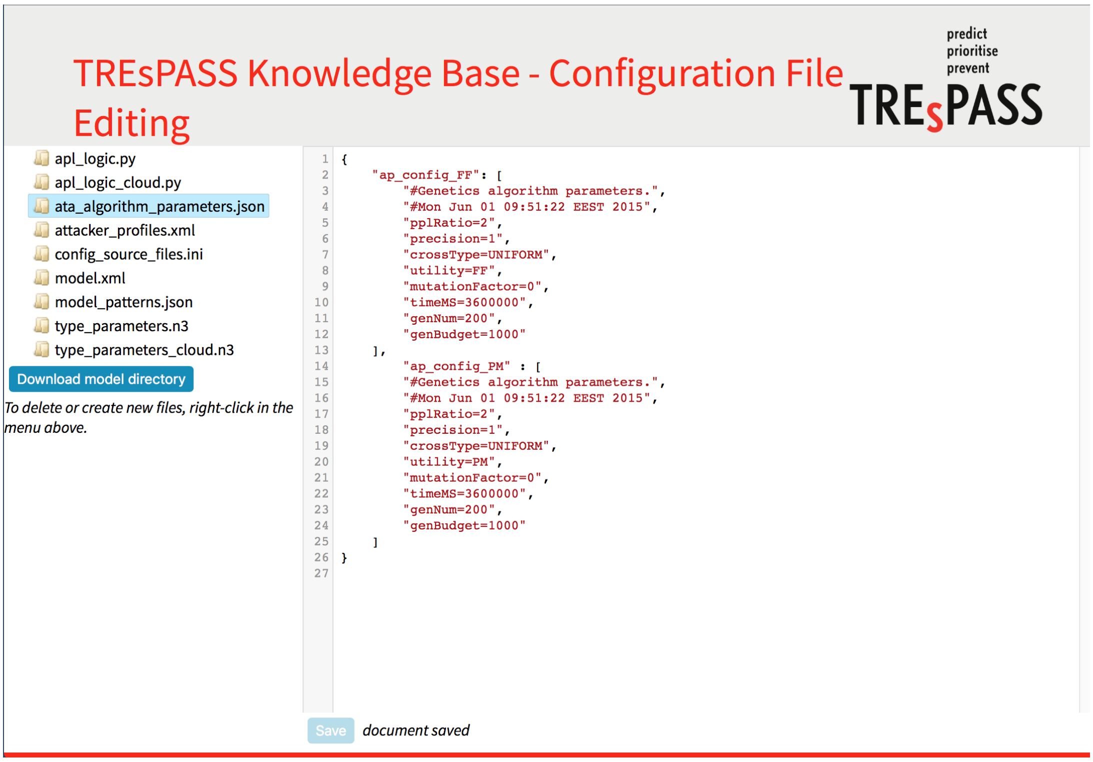

Figure 3.12.: View of the default `ata_algorithm_parameters.xml` file.

3.3. Discussion of the generated files

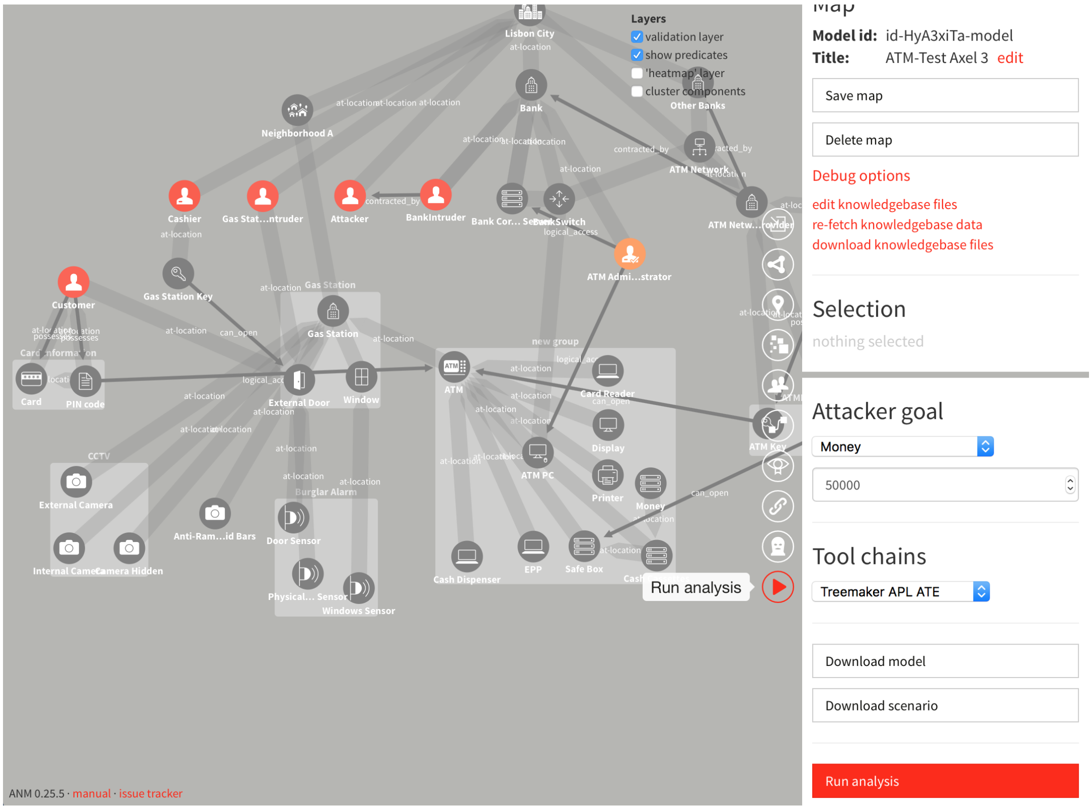

Figure 3.13.: 'Run analysis' section of the ANM.

A run of the different tools can be started in the 'Run analysis' section of the ANM (see Fig. 3.13). The resulting analysis results will be shown like in Fig. 3.14.

During these toolruns a set of new intermediate and result files are generated, as can be seen in Fig. 3.15 (and the overview in Fig. 2.1):

1. TREsPASS Model & Scenario file: before the toolrun start the model as captured in ANM is translated to the required input files for Treemaker, namely `model.xml` and `scenario.xml`.
2. Attack tree generated by Treemaker as file `treemaker_output_combined.xml`
3. Attack tree generated by augmentation and annotation by the APL, as shown in Fig. 3.15 (`apl_output.txt`)
4. Analysis results from Attack Tree Analyser ATA (`ata_output.zip`) and Attack Tree Evaluator ATE (`ate_output.txt`)

All of these files are part of the model directory, which is kept under version control in the TIS.

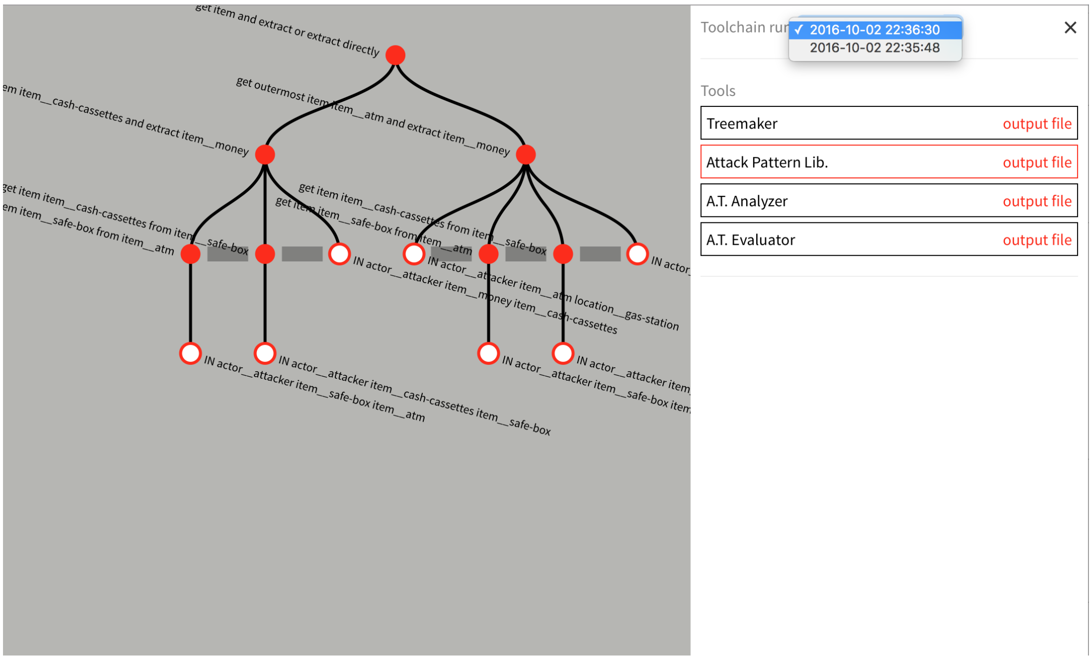

Figure 3.14.: Analysis results as seen in the ANM.

The complete directory including versioning information can be downloaded via the button 'Download model directory' (see Fig. 3.15). As this directory is a normal git repository, all git tools are available to access the version info, like the git command line tools, the free SourceTree[1] (as shown in Fig. 3.16) or the free GitHub Desktop[2].

[1]SourceTree https://www.sourcetreeapp.com/
[2]GitHub Desktop https://desktop.github.com

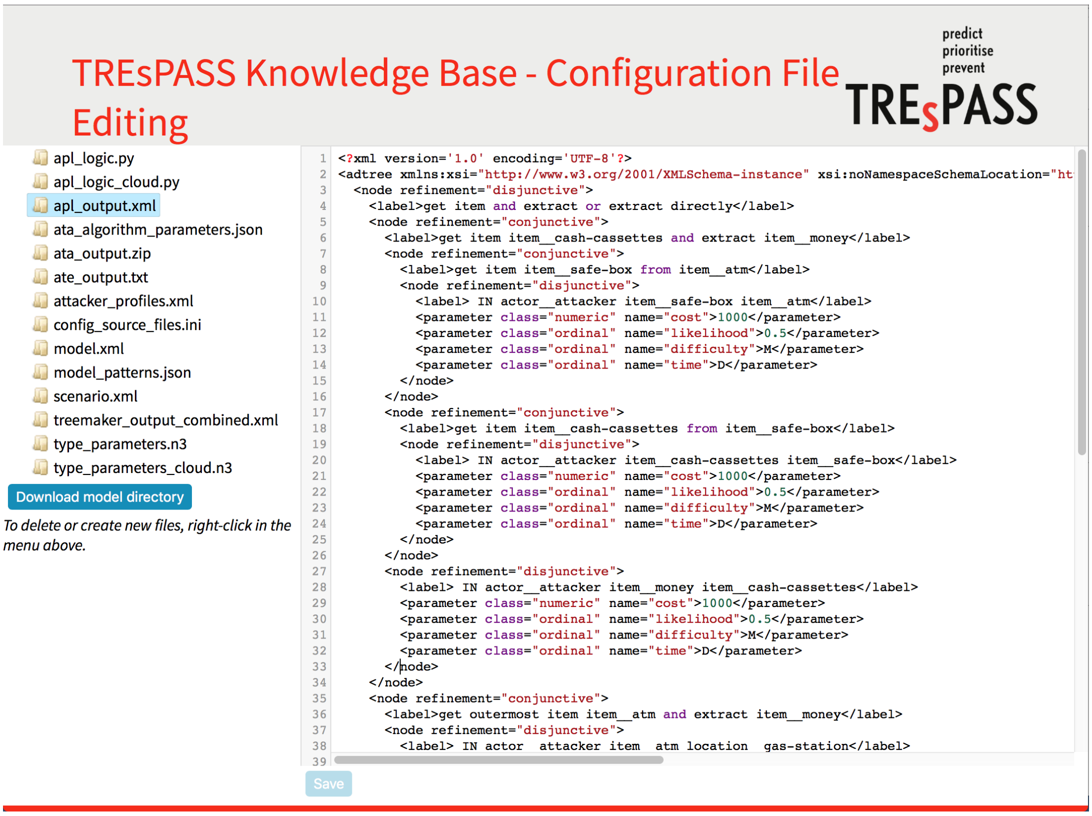

Figure 3.15.: View of the model files after an analysis run. Content of the generated `apl_output.xml` file.

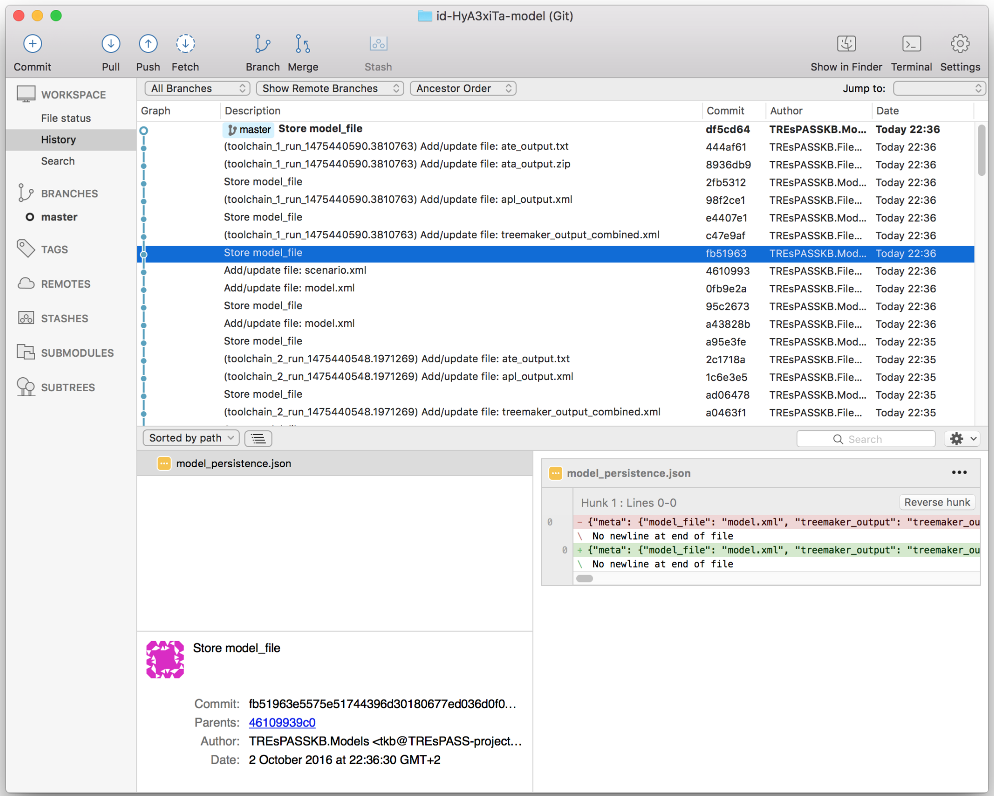

Figure 3.16.: After downloading the model directory, all files are available locally including its version history. This is a normal git repository, so all normal git tools are able to show the history of changes - here through the free application SourceTree.

4. Evolution and validation

The TRE$_S$PASS Information System was created and evolved together with the tools using it, namely the Attack Navigator Map (ANM) as the user interface supporting the creation of new models for the risk analysis, Treemaker for automatically creating attack trees from the model, the Attack Pattern Library (APL), as well as further analysis tools like the Attack Tree Analyser (ATA) and Attack Tree Evaluator (ATE).

Being in step with the development of the tools ensured its suitability to fulfil the requirements driven by the tools and the overall TRE$_S$PASS process. This happened through direct interaction with the tool owners on the one hand, and through the issue tracker shared with ANM for handling findings by the general user/tester community.

The TRE$_S$PASS Information System is available online as the backend accessed by the ANM at https://trespass.itrust.lu/attack-navigator-map/index.html.

5. Alternative approach: Agent-based modelling

In addition to the TRE$_S$PASS information system outlined in the previous chapters, the project has studied alternative ways of handling data related to socio-technical cyber risk. One of these is agent-based modelling and simulation described in this chapter.

Within the TRE$_S$PASS model, actors play a key role. These are both attackers and defenders, who either make decisions based on information available to them (the best possible attack), or act probabilistically (the probability that a defender falls for a social engineering attempt). As such, the actors in the TRE$_S$PASS model have many features in common with the agents in what is called agent-based modelling. In the final stages of the project, an attempt has been made to investigate the suitability of agent-based modelling for analysing socio-technical cyber risk. Although agent-based modelling has been used in a wide variety of application domains, the use in cyber security and associated risk is innovative compared to the state of the art.

Agent-based modelling is characterised by the simulation of interactions between different agents, whose behaviour can be subject to decision-making rules as well as randomness. Agents can observe parts of the environment, and use this to make decisions that contribute to their goals (utility). In this sense, agent-based modelling is related to game theory, but (a) it starts from actual rather than optimal behaviour, (b) it may involve a larger number of agents, and (c) it solves the problem by a large number of simulation runs rather than analytically.

Security risk can be modelled in such an approach by enabling agents to execute actions that may harm other stakeholders. It can then be investigated under what conditions the number of such actions (attacks) and/or their impact becomes higher or lower. Obvious candidate agents in the cyber security space include attackers and defenders. However, one may also consider legislators, vulnerability discoverers, etc.

Two master's thesis projects at TU Delft, supervised by Wolter Pieters, have designed agent-based models for specific aspects of cyber risk. In the first (Slangen, 2016), it was investigated which aspects of the behaviour of attackers and defenders influence the cyber risk incurred by a defender, in terms of the number of attacks against this defender. In particular, the effect of the control strength deployed by the defenders and their investment strategy were studied. The following 5 key insights were derived (quoted):

1. When fewer defenders decide to heavily invest in cyber-security, the group of defenders with a low control strength incurs many successful cyber-attacks. When more defenders decide to heavily invest in cyber-security, the number of successful

cyber-attacks incurred by defenders with a high control strength more or less stays the same.

2. When more threat agents become more skilful and resourceful, this will increase the number of successful cyber-attacks incurred by a defender organisation with a relatively high control strength, not the number of successful cyber-attacks of a defender organisation with a low control strength.
3. When defender organisations follow a reactive strategy with very high investment for a longer period, a defender organisation that invests quite a lot in cyber-security by default, is likely to incur more successful cyber-attacks.
4. Longer spells of a reactive strategy are more effective than short spells of a reactive strategy with respect to lowering the number of successful cyber-attacks incurred by a defender.
5. When threat agents become less rational, the number of successful cyber-attacks incurred by a weakly protected defender decreases whereas the successful cyber-attack incurred by a well protected defender increases.

In the second (Breukers, 2016), the vulnerability discovery rate and number of attacks on certain software and its users (defenders) were investigated. The agent-based model included behavioural characteristics of software vendors, vulnerability discoverers, attackers and defenders. One of the key results is that when vendors have long patch development times, but also release patches continuously, this significantly increases the number of attacks on their software. At the same time, users of the software can reduce their risk by deploying the patches quickly in their organisation. Both master's theses are judged to be publishable as scientific papers.

Compared to the TRE$_S$PASS Information System, agent-based modelling has a different approach to data. The data representation in agent-based models takes the form of:

- Inputs, represented by behavioural rules of the agents
- Outputs, represented by the distribution of the variables of interest (such as number of attack) over the runs in the simulation.

Agent-based models can be used to test whether assumed behavioural rules make sense compared to a reality of observed outputs (when such data is available). Conversely, when data on real-world attacks is scarce, simulated data can be generated from reasonable assumptions on the expected behaviour of the agents.

Compared to the TRE$_S$PASS information system, the current results on agent-based modelling exhibit (a) a more advanced representation of the agents in the system and their behaviour, but (b) a more abstract representation of the systems and attacker under consideration. We expect that the integration of agent-based simulations with the fine-grained system representations (maps) of TRE$_S$PASS is a fruitful area for further research. In such an approach, attackers would make decisions for each attack step rather than for a rather abstract attack as a whole, enabling more detailed analyses of the contribution of controls to the reduction of cyber risk. Agent-based modelling would also enable the TRE$_S$PASS models to work with populations of attackers and defenders.

6. Conclusions

With the TRE$_S$PASS Information System we have created an approach that fulfils the requirements of the TRE$_S$PASS process and tools, namely the need of a backend containing all required configuration files for the risk analysis, as well as responding to requests by ANM, Treemaker and APL calls.

The deliberate choice to move from a database approach to a system based on configuration by text-based files makes the system extendable and allows data to be shared in a simple manner between different practitioners. Using Python as a generic language for integration allows practitioner SME input to be used in arbitrary complexities with programmable logic.

References

Breukers, Y. (2016). *The vulnerability ecosystem: Exploring vulnerability discovery and the resulting cyberattacks through agent-based modelling* (Unpublished master's thesis). TU Delft.

Rosenquist, M. (2009, December). *Prioritizing information security risks with threat agent risk assessment.* "http://www.communities.intel.com/servlet/JiveServlet/download/4693-1-3205/Prioritizing_Info_Security_Risks_with_TARA.pdf". Intel Corporation.

Slangen, R. (2016). *Understanding cyber-risk by investigating the behaviour of defender and threat agent organisations: Why a complex adaptive systems perspective contributes to further understanding cyber-risk* (Unpublished master's thesis). TU Delft.

The TRE$_S$PASS Project, D2.1.1. (2013). *Initial requirements for the data management process.* (Deliverable D2.1.1)

The TRE$_S$PASS Project, D2.2.2. (2015). *Data extraction from virtualized infrastructures.* (Deliverable D2.2.2)

The TRE$_S$PASS Project, D5.3.2. (2015). *Best practices for model creation and sharing.* (Deliverable D5.3.2)

The TRE$_S$PASS Project, D5.4.2. (2016). *The integrated TRE$_S$PASS process.* (Deliverable D5.4.2)

The TRE$_S$PASS Project, D7.4.2. (2016). *Final report case study c.* (Deliverable D7.4.2)

W3C. (2004). *RDF - Semantic Web Standards.* http://www.w3.org/RDF. (Online; accessed 2015-10-26)

W3C. (2011). *Notation3 (N3): A readable RDF syntax.* http://www.w3.org/TeamSubmission/n3. (Online; accessed 2015-10-26)

A. How to access the prototype

To access the prototype described in this deliverable, a one-time registration is required:

1. Connect to the TRE$_S$PASS portal at https://trespass.itrust.lu/login.
2. Click on Sign-up, you will receive a confirmation email, you need to click on it to acknowledge the registration. The itrust ICT administrator will have to personally validate your account.
3. Once you receive the validation email, you will be able to access the Attack Navigator Map at https://trespass.itrust.lu/attack-navigator-map/index.html (and other tools).

B. API Overview and Examples TRE$_S$PASS Information System

This appendix describes the API of the TRE$_S$PASS Information System with examples. The most recent version of the API description can be found online at https://trespass.itrust.lu/tkb/tkb.

B.1. Annotations

- POST adtree as XML, receive augmented and annotated adtree as XML, in context of model with model_id:

```
curl -i -X "POST" "http://localhost:8080/tkb/getAnnotations?model_id=default_model" \
    -H "Content-Type: application/xml" -H "Accept: application/xml" -d '<?xml version="1.0" encoding="UTF-8"?>
<adtree>
  <node refinement="disjunctive">
    <label> IN terry DATA fileX fileX ITEM entity_vim.VirtualMachine_vm-51 entity_vim.VirtualMachine_vm-51</label>
  </node>
</adtree>'
```

  results in

```
HTTP/1.1 200 OK
Date: Fri, 26 Feb 2016 14:37:45 GMT
Content-Type: application/xml
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Server: CherryPy/5.0.1
Content-Length: 2812
Access-Control-Allow-Origin: *

<?xml version='1.0' encoding='UTF-8'?>
<adtree xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://research.cyber
  <node refinement="disjunctive">
    <label>IN terry DATA fileX ITEM entity_vim.VirtualMachine_vm-51</label>
    <node refinement="conjunctive">
      <label>terry obtains fileX from the running entity_vim.VirtualMachine_vm-51</label>
      <node refinement="conjunctive">
        <label>terry obtains administratorobtains administrator privileges</label>
        ...
      </node>
    </node>
  </node>
</adtree>
```

- POST adtree as XML, containing leaves for which no annotations are available, receive list of labels with missing annotations as XML, in context of model with model_id:

```
curl -i -X "POST" "http://localhost:8080/tkb/getAnnotations?model_id=default_model" \
    -H "Content-Type: application/xml" -H "Accept: application/xml" -d '<?xml version="1.0" encoding="UTF-8"?>
<adtree>
  <node refinement="conjunctive">
    <label>terry obtains fileX from the drive of entity_vim.VirtualMachine_vm-51 directly</label>
    <node refinement="disjunctive">
      <label>missing mounts the HD of entity_vim.VirtualMachine_vm-51 and inspects it</label>
    </node>
    <node refinement="disjunctive">
      <label>terry missing fileX in the file system of entity_vim.VirtualMachine_vm-51</label>
    </node>
  </node>
</adtree>'
```

results in

```
HTTP/1.1 424 Missing annotations
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Length: 239
Server: CherryPy/5.0.1
Content-Type: application/xml
Access-Control-Allow-Origin: *
Date: Tue, 23 Feb 2016 14:34:53 GMT

<?xml version='1.0' encoding='UTF-8'?>
<leaves>
  <label>missing mounts the HD of entity_vim.VirtualMachine_vm-51 and inspects it</label>
  <label>terry missing fileX in the file system of entity_vim.VirtualMachine_vm-51</label>
</leaves>
```

B.2. Attackerprofiles

- GET full list of attacker profiles in context of model with model_id (as json)

```
curl -i "http://localhost:8080/tkb/attackerprofile?model_id=default_model" -H "Accept: application/json"
```

results in:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
Date: Tue, 23 Feb 2016 17:10:37 GMT
Server: CherryPy/5.0.1
Content-Length: 2552
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept

[
  {
    "id": "AP0001",
    "description": "Current highly qualified employee with harmless intent. Theft of IP, PII, or business data.",
    "budget": "10000",
    "codename": "Employee Trained",
    "time": "D",
    "skill": "H"
  },
  ...
]
```

- GET full list of attacker profiles in context of model with model_id (as xml)

```
curl -i "http://localhost:8080/tkb/attackerprofile?model_id=default_model" -H "Accept: application/xml"
```

results in:

```
HTTP/1.1 200 OK
Date: Wed, 24 Feb 2016 14:56:36 GMT
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Length: 2182
Content-Type: application/xml
Server: CherryPy/5.0.1
Access-Control-Allow-Origin: *

<?xml version='1.0' encoding='UTF-8'?>
<profiles>
  <profile budget="10000" skill="H" codename="Employee Trained" id="AP0001" time="D" description="Current highly q
  ...
</profiles>
```

- GET specific attacker profile in context of model with model_id (as json)

```
curl -i "http://localhost:8080/tkb/attackerprofile/AP0007?model_id=default_model" \
    -H "Accept: application/json"
```

results in:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
Date: Tue, 23 Feb 2016 17:06:54 GMT
Server: CherryPy/5.0.1
Content-Length: 234
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept

{
  "id": "AP0007",
  "description": "Organized crime organization with significant resources. Theft of IP, PII, or business data; vio
  "budget": "60000",
  "codename": "Organized Crime Group",
  "time": "D",
  "skill": "H"
}
```

- GET specific attacker profile in context of model with model_id (as xml)

```
curl -i "http://localhost:8080/tkb/attackerprofile/AP0007?model_id=default_model" \
    -H "Accept: application/xml"
```

results in:

```
HTTP/1.1 200 OK
Date: Wed, 24 Feb 2016 14:55:42 GMT
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Length: 245
Content-Type: application/xml
Server: CherryPy/5.0.1
Access-Control-Allow-Origin: *

<?xml version='1.0' encoding='UTF-8'?>
<profile budget="60000" skill="H" codename="Organized Crime Group" id="AP0007" time="D" description="Organized cri
```

- PUT: create new attacker profile in context of model with model_id

```
curl -i -X PUT -H "Content-Type:application/json" "http://localhost:8080/tkb/attackerprofile/AP_test?model_id=defa
    -d '{"budget": "5000", "codename": "Test AP", "time": "D", "id": "AP_test", "description": "Test Attacker Prof
```

results in:

```
HTTP/1.1 200 OK
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE
Content-Length: 0
Date: Wed, 09 Mar 2016 16:24:24 GMT
Access-Control-Allow-Origin: *
Server: CherryPy/5.0.1
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Type: application/json
```

- DELETE specific attacker profile in context of model with model_id

```
curl -i -X DELETE -H "Content-Type:application/json" \
    "http://localhost:8080/tkb/attackerprofile/AP_test?model_id=default_model"
```

results in:

```
HTTP/1.1 200 OK
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE
Content-Length: 0
Date: Wed, 09 Mar 2016 16:25:03 GMT
Access-Control-Allow-Origin: *
Server: CherryPy/5.0.1
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Type: application/json
```

B.3. ATA_GA_Parameters

- GET full list of ATA GA Parameter sets in context of model with model_id

```
curl -i "http://localhost:8080/tkb/ata_ga_parameter?model_id=default_model"
```

results in

```
HTTP/1.1 200 OK
Content-Length: 406
Access-Control-Allow-Origin: *
Server: CherryPy/5.0.1
Date: Fri, 26 Feb 2016 11:53:24 GMT
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Type: application/json
```


```
{
  "ap_config_FF": "#Genetics algorithm parameters.\n#Mon Jun 01 09:51:22 EEST 2015\n
  "ap_config_PM": "#Genetics algorithm parameters.\n#Mon Jun 01 09:51:22 EEST 2015\n
}
```


- GET specific set of ATA GA Parameter in context of model with model_id

```
curl -i "http://localhost:8080/tkb/ata_ga_parameter/ap_config_FF?model_id=default_model"
```

results in

```
HTTP/1.1 200 OK
Content-Length: 171
```

```
Access-Control-Allow-Origin: *
Server: CherryPy/5.0.1
Date: Fri, 26 Feb 2016 11:52:52 GMT
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Type: text/plain;charset=utf-8

#Genetics algorithm parameters.
#Mon Jun 01 09:51:22 EEST 2015
pplNr=200
precision=1
crossType=UNIFORM
utility=FF
mutationFactor=0
timeMS=3600000
genNum=200
genBudget=1000
```

B.4. Types

- GET full list of types in context of model with model_id (as json)

  ```
  curl -i "http://localhost:8080/tkb/type?model_id=default_model" -H "Accept: application/json"
  ```

  results in:

  ```
  HTTP/1.1 200 OK
  Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
  Server: CherryPy/5.0.1
  Content-Length: 7356
  Content-Type: application/json
  Access-Control-Allow-Origin: *
  Date: Thu, 25 Feb 2016 17:22:08 GMT

  [
    {
      "tkb:forceability": true,
      "tkb:persistence": true,
      "tkb:cloneability": false,
      "@label": "Actor",
      "skos:broader": {
        "tkb:forceability": true,
        "tkb:cloneability": false,
        "@label": "Human Being",
        "tkb:persistence": true,
        "@id": "tkb:human",
        "tkb:soc_eng_probability": 50,
        "@type": "tkb:generalised_type"
      },
    ...
  ]
  ```

- GET specific type in context of model with model_id (as json)

  ```
  curl -i "http://localhost:8080/tkb/type/tkb:actor?model_id=default_model" -H "Accept: application/json"
  ```

results in:

```
HTTP/1.1 200 OK
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Server: CherryPy/5.0.1
Content-Length: 1677
Content-Type: application/json
Access-Control-Allow-Origin: *
Date: Thu, 25 Feb 2016 17:23:49 GMT
```


```
{
  "tkb:forceability": true,
  "tkb:persistence": true,
  "tkb:cloneability": false,
  "@label": "Actor",
  "skos:broader": {
    "tkb:forceability": true,
    "tkb:cloneability": false,
    "@label": "Human Being",
    "tkb:persistence": true,
    "@id": "tkb:human",
    "tkb:soc_eng_probability": 50,
    "@type": "tkb:generalised_type"
  },
  "tkb:tml_class": "actor",
  "@id": "tkb:actor",
  "tkb:has_attribute": [
    {
      "@type": "tkb:attribute",
      "@label": "Name",
      "@id": "tkb:actor_name",
      "tkb:values": [
        " "
      ]
    },
    ...
  ],
  "@type": "tkb:type"
}
```


B.5. Data

- GET with id as string in context of model with model_id, receive data as json (default w/o Accept header)

```
curl -i "http://localhost:8080/tkb/getData?model_id=default_model&id=entity_vim.HostSystem_host-19"
```

results in

```
HTTP/1.1 200 OK
Date: Mon, 25 Jan 2016 11:56:30 GMT
Content-Length: 134
Content-Type: application/json
Server: CherryPy/3.8.0
```


```
{
  "forceability": "false",
  "label": "nuc2",
  "type": "HostSystem",
  "persistence": "persistent",
```

```
  "cloneability": "false"
}
```

- GET with id as string in context of model with model_id, receive data as xml

```
curl -i "http://localhost:8080/tkb/getData?model_id=default_model&id=entity_vim.HostSystem_host-19" \
    -H "Accept: application/xml"
```

results in

```
HTTP/1.1 200 OK
Date: Mon, 25 Jan 2016 11:56:57 GMT
Content-Length: 200
Content-Type: application/xml
Server: CherryPy/3.8.0

<?xml version="1.0" ?>
<node>
  <forceability>false</forceability>
  <label>nuc2</label>
  <type>HostSystem</type>
  <persistence>persistent</persistence>
  <cloneability>false</cloneability>
</node>
```

B.6. Model

- Create a new model with id 'new_model'

```
curl -i -X PUT -H "Content-Length:0" "http://localhost:8080/tkb/model/new_model"
```

results in

```
HTTP/1.1 200 OK
Content-Length: 0
Server: CherryPy/3.8.0
Date: Thu, 11 Feb 2016 22:33:40 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Type: application/json
```

- Query model with id 'new_model'

```
curl -i "http://localhost:8080/tkb/model/new_model"
```

results in

```
HTTP/1.1 200 OK
Content-Length: 2
Server: CherryPy/3.8.0
Date: Thu, 11 Feb 2016 22:35:35 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Type: application/json

{}
```

- List models

```
curl -i "http://localhost:8080/tkb/model"
```

results in

```
HTTP/1.1 200 OK
Content-Length: 17
Server: CherryPy/3.8.0
Date: Thu, 11 Feb 2016 22:36:16 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Type: application/json

[
  "new_model"
]
```

- Create item in model

```
curl -i -X PUT -H "Content-Type:application/json" "http://localhost:8080/tkb/model/new_model/door1" \
    -d '{"id":"door1", "class":"class1"}'
```

results in

```
HTTP/1.1 200 OK
Content-Length: 0
Server: CherryPy/3.8.0
Date: Thu, 11 Feb 2016 22:42:47 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Type: application/json
```

- Query item

```
curl -i  "http://localhost:8080/tkb/model/new_model/door1"
```

results in

```
HTTP/1.1 200 OK
Content-Length: 41
Server: CherryPy/3.8.0
Date: Thu, 11 Feb 2016 22:43:58 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Type: application/json

{
  "id": "door1",
  "class": "class1"
}
```

- Rename item

```
curl -i -X POST -H "Content-Length:0"  "http://localhost:8080/tkb/model/default_model/door1?rename_to=renamed_door
```

results in

```
HTTP/1.1 200 OK
Content-Type: application/json
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Server: CherryPy/6.0.1
Date: Tue, 21 Jun 2016 21:23:13 GMT
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Origin: *
Content-Length: 40
```

```
{
  "id": "door1",
  "class": "class1"
}
```

- Delete item

  ```
  curl -i  -X DELETE "http://localhost:8080/tkb/model/new_model/door1"
  ```

  results in

  ```
  HTTP/1.1 200 OK
  Content-Length: 0
  Server: CherryPy/3.8.0
  Date: Thu, 11 Feb 2016 22:45:17 GMT
  Access-Control-Allow-Origin: *
  Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
  Content-Type: application/json
  ```

- Delete model

  ```
  curl -i  -X DELETE "http://localhost:8080/tkb/model/new_model"
  ```

  results in

  ```
  HTTP/1.1 200 OK
  Content-Length: 0
  Server: CherryPy/3.8.0
  Date: Thu, 11 Feb 2016 22:45:58 GMT
  Access-Control-Allow-Origin: *
  Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
  Content-Type: application/json
  ```

B.7. Schemas

- GET http://localhost:8080/tkb/schemas/TREsPASS_model.xsd
- GET http://localhost:8080/tkb/schemas/TREsPASS_scenario.xsd

B.8. FileStore

- List all files in context of model with model_id

  ```
  curl -i "http://localhost:8080/tkb/files/list?model_id=default_model"
  ```

  results in

  ```
  HTTP/1.1 200 OK
  Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE
  Server: CherryPy/6.0.1
  Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
  Content-Type: text/html;charset=utf-8
  Access-Control-Allow-Origin: *
  Date: Thu, 14 Jul 2016 07:52:21 GMT
  Content-Length: 632
  ```

```
[
  {
    "text": "apl_logic.py",
    "id": 1
  },
  ...
  {
    "text": "type_parameters_cloud.n3",
    "id": 11
  }
]
```


- List all commits in context of model with model_id

  ```
  curl -i "http://localhost:8080/tkb/files/listcommits?model_id=default_model"
  ```

  results in


```
HTTP/1.1 200 OK
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Type: text/html;charset=utf-8
Server: CherryPy/6.0.1
Date: Thu, 14 Jul 2016 08:17:57 GMT
Content-Length: 2708
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE

[
  {
    "timestamp": 1468484273,
    "path": "model_persistence.json",
    "commit_id": "91cee7c0fab6017ad294a5172b8bfb7613065cb0",
    "message": "Store model_file",
    "file_id": "9bed19878516872a6d3a8f03cf6960fd5746431b"
  },
...
  {
    "timestamp": 1468484273,
    "path": "type_parameters_cloud.n3",
    "commit_id": "86075664e4131b183ff07fd60fc49c05773a1b65",
    "message": "Add base files",
    "file_id": "4da61c0878199d35173faaddc6a610814726ba1c"
  }
]
```


- Read a file in context of model with model_id

  ```
  curl -i "http://localhost:8080/tkb/files?model_id=default_model&filename=attacker_profiles.xml"
  ```

  results in


```
HTTP/1.1 200 OK
Last-Modified: Tue, 01 Mar 2016 10:20:17 GMT
Content-Type: application/xml
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Date: Tue, 01 Mar 2016 15:33:38 GMT
Content-Length: 2187
Access-Control-Allow-Origin: *
Accept-Ranges: bytes
Content-Disposition: attachment; filename="attacker_profiles.xml"
Server: CherryPy/5.0.1

<?xml version="1.0" encoding="UTF-8"?>
<profiles>
```

```
    <profile id="AP0001" codename="Employee Trained" description="Current highly qualified employee with harmless
    ...
</profiles>
```

- Read a file from git in context of model with model_id

```
curl -i "http://localhost:8080/tkb/files?model_id=dummy_model&file_id=fbe9620145dedd078f0d7b48d527367df438f452"
```

results in

```
HTTP/1.1 200 OK
Server: CherryPy/6.0.1
Date: Thu, 14 Jul 2016 09:18:26 GMT
Accept-Ranges: bytes
Access-Control-Allow-Origin: *
Content-Disposition: attachment; filename="fbe9620145dedd078f0d7b48d527367df438f452"
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE
Last-Modified: Thu, 14 Jul 2016 09:18:28 GMT
Content-Type: text/html;charset=utf-8
Content-Length: 300
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept

{"meta": {"epoch-modified": 1468487889.296554, "epoch-persisted": 1468487889.296591, "epoch-created": 1468487889.2
```

- Store a file in context of model with model_id

```
curl -i -X PUT "http://localhost:8080/tkb/files?model_id=default_model&filename=new_file.xml"  \
    -H "Content-Type: application/xml" -d '<?xml version="1.0" encoding="UTF-8"?><root/>'
```

results in

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: text/html;charset=utf-8
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Date: Tue, 01 Mar 2016 15:38:35 GMT
Access-Control-Allow-Origin: *
Server: CherryPy/5.0.1
```

- Store model/scenario file in context of model with model_id: currently filetype is limited to model_file|scenario_file

```
curl -i -X PUT "http://localhost:8080/tkb/files?model_id=default_model&filename=new_file.xml&filetype=scenario_fil
    -H "Content-Type: application/xml" -d '<?xml version="1.0" encoding="UTF-8"?><root/>'
```

results in

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: text/html;charset=utf-8
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Date: Tue, 01 Mar 2016 15:40:06 GMT
Access-Control-Allow-Origin: *
Server: CherryPy/5.0.1
```

- Delete a file in context of model with model_id

```
curl -i -X DELETE "http://localhost:8080/tkb/files?model_id=default_model&filename=new_file.xml"
```

results in

```
HTTP/1.1 200 OK
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Length: 0
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE
Content-Type: text/html;charset=utf-8
Date: Thu, 03 Mar 2016 16:51:37 GMT
Access-Control-Allow-Origin: *
Server: CherryPy/5.0.1
```

- Edit a file in context of model with model_id

  ```
  curl -i "http://localhost:8080/tkb/files/edit?model_id=default_model&filename=attacker_profiles.xml"
  ```

  results in a webpage to edit this file.

- Get zip of model directory with model_id (including the git information)

  ```
  curl -i "http://localhost:8080/tkb/files/zip?model_id=default_model"
  ```

  results in the download of the zipped model diretory.

B.9. Toolchain

- Get list of toolchains

  ```
  curl -i "http://localhost:8080/tkb/toolchain?model_id=dummy_model"
  ```

  results in

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Length: 3011
Date: Fri, 18 Mar 2016 12:34:56 GMT
Server: CherryPy/5.0.1
Content-Type: application/json
```


```
[
  {
    "name": "Treemaker APL ATA ATE",
    "description": "treemaker_apl_ata_ate",
    "tools": [
      {
        "publisher": "T. University of Denmark",
        "hasInput": true,
        "types": [
          "model"
        ],
        "resources": {
          "Input file for testing": "https://trespass.itrust.lu/data/treemaker.zip"
        },
        "name": "Treemaker",
        "description": "This tool generates an attack tree in the XML format that can be used for analyses, visual
        "id": 15
      },
      {
        "publisher": "Cybernetica",
        "hasInput": true,
        "types": [
```

```
          "analysis"
        ],
        "resources": {
          "Input file for testing": "https://trespass.itrust.lu/data/apl.xml"
        },
        "name": "Attack Pattern Lib.",
        "description": "The Attack Pattern Library (APL) is intended to promote the reuse of modular elements to i
        "id": 8
      },
      {
        "publisher": "Cybernetica",
        "hasInput": true,
        "types": [
          "analysis"
        ],
        "resources": {
          "Input file for testing": "https://trespass.itrust.lu/data/approxtree.txt"
        },
        "name": "A.T. Analyzer",
        "description": "The attack tree computation tool can be used to calculate optimal attack vector (from the
        "id": 3
      },
      {
        "publisher": "T. University of Denmark",
        "hasInput": true,
        "types": [
          "analysis"
        ],
        "resources": {
          "Input file for testing": "https://trespass.itrust.lu/data/approxtree.txt"
        },
        "name": "A.T. Evaluator",
        "description": "The attack tree computation tool can be used to calculate a pareto front of attack vectors
        "id": 33
      }
    ],
    "id": 1
  },
  ...
]
```


- Start a toolchain

```
curl -i -X POST "http://localhost:8080/tkb/toolchain/1?model_id=dummy_model&attackerprofile_id=AP0007&delay=0"
```

results in


```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Length: 2
Date: Fri, 18 Mar 2016 12:36:07 GMT
Server: CherryPy/5.0.1
Content-Type: application/json

{
  "task_url": "http://localhost:8080/tkb/task/0d178007ae7044fdb3de5b204ce94e36"
  "task_id": "0d178007ae7044fdb3de5b204ce94e36"
}
```

B.10. Tasks

- Get list of tasks

```
curl -i "http://localhost:8080/tkb/task"
```

results in

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Length: 2
Date: Fri, 18 Mar 2016 12:36:07 GMT
Server: CherryPy/5.0.1
Content-Type: application/json

{}
```

- Get specific task

```
curl -i "http://localhost:8080/tkb/task/0d178007ae7044fdb3de5b204ce94e36"
```

results in

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Length: 2
Date: Fri, 18 Mar 2016 12:36:07 GMT
Server: CherryPy/5.0.1
Content-Type: application/json
{
  "task_id": "91bf413a4edf44e8af09c2caa1bc2155",
  "status": "done",
  "end-epoch": 1458303114.895991,
  "start-epoch": 1458303112.930813,
  "end-date": "2016-03-18T13:11:54.895991+01:00",
  "tool_status": [
    {
      "status": "done",
      "end-epoch": 1458303113.999195,
      "id": 15,
      "start-epoch": 1458303112.931555,
      "result_file_url": "http://localhost:8080/tkb/files?model_id=dummy_model&filetype=treemaker_output",
      "result_filetype": "treemaker_output",
      "end-date": "2016-03-18T13:11:52.931555+01:00",
      "name": "Treemaker",
      "start-date": "2016-03-18T13:11:52.931555+01:00"
    },
    ...
  ],
  "start-date": "2016-03-18T13:11:52.930813+01:00"
}
```

B.11. Modelpattern

- GET full list of model patterns in context of model with model_id (as json)

```
curl -i "http://localhost:8080/tkb/modelpattern?model_id=default_model" -H "Accept: application/json"
```

results in:

```
HTTP/1.1 200 OK
Content-Length: 84
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE
Date: Mon, 21 Mar 2016 17:26:41 GMT
Content-Type: application/json
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Server: CherryPy/5.0.1
```


```
[
  {
    "name": "Test Model Pattern",
    "id": "test_mp",
    "pattern": {}
  }
]
```


- GET specific model pattern in context of model with model_id (as json)

```
curl -i "http://localhost:8080/tkb/modelpattern/test_mp?model_id=default_model" -H "Accept: application/json"
```

results in:

```
HTTP/1.1 200 OK
Content-Length: 70
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE
Date: Mon, 21 Mar 2016 17:27:20 GMT
Content-Type: application/json
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Server: CherryPy/5.0.1
```


```
{
  "name": "Test Model Pattern",
  "id": "test_mp",
  "pattern": {}
}
```


- PUT: create new model pattern in context of model with model_id

```
curl -i -X PUT -H "Content-Type:application/json" \
    "http://localhost:8080/tkb/modelpattern/dummy_mp?model_id=default_model" \
    -d '{"name": "Test Model Pattern", "id": "dummy_mp", "pattern": {}}'
```

results in:

```
HTTP/1.1 200 OK
Content-Length: 0
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE
Date: Mon, 21 Mar 2016 17:29:01 GMT
Content-Type: application/json
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Server: CherryPy/5.0.1
```

- DELETE specific model pattern in context of model with model_id

```
curl -i -X DELETE -H "Content-Type:application/json" \
    "http://localhost:8080/tkb/modelpattern/dummy_mp?model_id=default_model"
```

results in:

```
HTTP/1.1 200 OK
Content-Length: 0
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE
Date: Mon, 21 Mar 2016 17:29:35 GMT
Content-Type: application/json
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Server: CherryPy/5.0.1
```