D6.2 – Implementation of the Assurance Framework Toolkit

Security Assurance Framework for Networked Vehicular Technology

Abstract

SAFERtec proposes a flexible and efficient assurance framework for security and trustworthiness of Connected Vehicles and Vehicle-to-I (V2I) communications aiming at improving the cyber-physical security ecosystem of "connected vehicles" in Europe. The project will deliver innovative techniques, development methods and testing models for efficient assurance of security, safety and data privacy of ICT related to Connected Vehicles and V2I systems, with increased connectivity of automotive ICT systems, consumer electronics technologies and telematics, services and integration with 3rd party components and applications. The cornerstone of SAFERtec is to make assurance of security, safety and privacy aspects for Connected Vehicles, measurable, visible and controllable by stakeholders and thus enhancing confidence and trust in Connected Vehicles.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

DX.X & Title:	D6.2 Implementation of the Assurance Framework Toolkit
Work package:	WP6
Task:	Task 6.2 and 6.3
Due Date:	31 January 2020
Dissemination Level:	PU
Deliverable Type:	R

Authoring and review process information	
EDITOR Angelos Stamou / ICCS	**DATE** 12 February – 12 April 2020 – 14 July 2020
CONTRIBUTORS Panagiotis Pantazopoulos / ICCS	**DATE** 8 April – 20 April 2020 – 27 July 2020
REVIEWED BY Mattia Zeni / TomTom Claudio Griglione / Swarco	**DATE** 22 April 2020 22 April 2020
LEGAL & ETHICAL ISSUES COMMITTEE REVIEW REQUIRED?	
NO	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

Document/Revision history

Version	Date	Partner	Description
V0.1	12/02/2020	ICCS	First draft
V0.2	25/02/2020	ICCS	Inputs to Sections 2 and 3
V0.3	04/03/2020	ICCS	Inputs to Sections 4
V0.4	10/03/2020	ICCS	Inputs to Section 5
V0.5	19/03/2020	ICCS	Inputs to Section 6 and 7
V0.6	26/03/2020	ICCS	Refinement of Section 5
V0.7	08/04/2020	ICCS	Introduction, Conclusions
V0.8	22/04/2020	TomTom	Internal review
V0.9	22/04/2020	Swarco	Internal review
V1.0	23/04/2020	ICCS	Final version
V1.1	28/07/2020	ICCS	Revised to include: • An introductory paragraph explaining and justifying the framework's changes and relevant implications (Section 1). • Emphasis on the availability of the AFT source code and the relevant link (Executive Summary, Section 1, 3 and Conclusions) • Explanations on the functionality of the "Migrations" class (Subsection 3.3) • Extensions of Section 3 including new explanatory figures. • Justifications for the non-explicit presence of usability evaluation results (Appendix A2)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

Table of Contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

Table of Figures

List of Tables

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

Acronyms and abbreviations

Abbreviation	Description
ADV	Development (CC evaluation class acronym)
AFT	Assurance Framework Toolkit
ANSSI	Agence nationale de la sécurité des systèmes d'information
API	Application Programming Interface
ASE	Security Target Evaluation (CC evaluation class acronym)
ATE	Tests (CC evaluation class acronym)
CC	Common Criteria
CLI	Command Line Interface
E-R	Entity Relationship
GIN	Generalized Inverted Index
GIT	Git version control system
GUI	Graphical User Interface
HTTP	Hyper Text Transport Protocol
HTTPS	Hyper Text Transport Protocol Secure
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
JWT	JSON Web Token
MVC	Model View Controller
NG	Angular framework
OEM	Original Equipment Manufacturer
PBKDF2	Password-Based Key Derivation Function 2
PP	Protection Profile
PSQL	PostgresSQL database
RDBMS	Relational Database Management System
REST	Representational State Transfer
SAR	Security Assurance Requirements
SFR	Security Functional Requirements
SPA	Single Page Application
ST	Security Target
TOE	Target of Evaluation
UI	User Interface

Table 1: List of Abbreviations

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

Executive Summary

The software requirements and relevant architecture for the SAFERtec Assurance Framework Toolkit (AFT) specified in the previous WP6 deliverable serves as a basis for the development of the modular and extendable AFT application. The document at hand details the way the AFT specifications, architecture and identified technologies/standards are used to implement a platform-independent toolkit that provides assistance for the application of the SAFERtec (or more generally, the Common Criteria) security assurance framework.

The AF Toolkit is a cross-platform, open source [1] software instance that can significantly assist the security evaluation of automotive products covering at the moment two important evaluation classes; the definition of the Security Target (ASE evaluation class) for the product (ToE) and the compilation of evaluation inputs related to the design specifications (ADV evaluation class).

The document highlights the AFT software components, describes the exposed interfaces and justifies the way that the earlier-identified software requirements are met by the AFT implementation. Testing cases to confirm the appropriate operation of AFT are also provided.

Importantly, the toolkit receives further features to extend its functionality to other evaluation classes while the commitment of involved partners to provide updates and regular maintenance ensure the significant AFT potential for high visibility and positive acceptance.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

1 Introduction

The SAFERtec project objective was to introduce a security assurance framework for connected vehicles [2]. Towards that end, SAFERtec has relied on the most credible yet generic approach to security assurance i.e., the Common Criteria standard [3]. A number of reasons justify this decision: Common Criteria is the most recognised (by more than 30 countries worldwide) and credible relevant standard implying increased usability for the SAFERtec proposal. It also provides (up to) the highest possible level of assurance needed particularly for the case of the ever-increased automation of connected vehicles. Towards that end, Common Criteria requires the gathering of an exhaustive set of data *i.e.*, inputs to eight different assurance classes and comes with increased costs.

Implications of the Common Criteria selection: The SAFERtec proposal as expressed in WP3 considerably enhances the Common Criteria (CC) introducing a tailor-made framework of *direct* and *cost-efficient* applicability to the connected vehicles paradigm. The reliance on that standard suggests that the Common Criteria processes are respected and the rich spectrum of the corresponding evaluation tasks over the evaluated system (i.e., Target of Evaluation) is being followed/enhanced. Along this line one main implication affecting AFT is the fact that the assurance level of any evaluation is not 'computable' but predefined by the standard, essentially determined by the depth of the evaluation tasks that the standard suggests. Therefore, functionality as the one expected by an inference engine that could realise similar computations is no longer relevant/required. On the contrary, the aim for AFT is to include all the necessary software structures/tools (e.g., a knowledge base) to accurately reproduce the involved evaluation tasks and ensure the minimization of both time and effort required. In particular, AFT which is a specialized software designed and specified in deliverable D6.1, provides the necessary online functionality to ease the gathering, organization, management of SAFERtec (or CC) evaluation data as well as the efficient compilation of the relevant outputs (e.g., Security Targets, Architecture specifications of the ToE).

AFT has been developed as a *platform-independent* and *open-source* [1] toolkit to facilitate the cost-efficient compilation of the required evaluation inputs for the security target evaluation (ASE class), the architecture and functional specification reviews (ADV class). The former is based on the ST document that specifies the functions under evaluation and the assurance requirements to be met; it also determines the rest of the required evaluation inputs; the latter includes the evaluation of design documents and detailed design specs describing the TOE interfaces and their relation to the SFRs.

The document at hand describes the way the toolkit implementation has been carried-out relying on the SPA concept (i.e., the front-end is a Typescript while the back-end is a C# implementation). It discusses the way that technologies and standards already identified in D6.1 leverage the development of AFT modules and are used to achieve the required functionality. The AFT exposed interfaces to facilitate external communications are presented and the involved software deployment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

and development is described. Furthermore, the deliverable describes testing cases to verify AFT features as well as the way that the software requirements -identified in D6.1- are met.

Most importantly, the toolkit features welcome expandability properties. Its functionality is already under extension to cover another evaluation class (ATE) which includes the functional tests i.e., to support the ToE developer in describing a test plan and the associated tests scenarios or test scripts.

Thus, the AFT nicely complements the theoretical SAFERtec work providing the means to support the effective application[1] of the introduced security assurance framework; even further, AFT can be used to bring-about significant gains to the CC application compared to unassisted evaluations.

1.1 Purpose of the Document

This document aims to describe in detail the work undertaken in the context of Tasks 6.2 and 6.3. That is mainly the implementation and developed functionality of the AFT as well as demonstrate how the requirements identified in D6.1 are met.

1.2 Intended readership

Besides the project reviewers, this deliverable is addressed to any interested reader as it is of Public dissemination level.

1.3 Inputs from other projects

No input from other projects was considered during the compilation of this deliverable.

1.4 Relationship with other SAFERTEC deliverables

This document builds-upon and expands deliverable 6.1, entitled 'Reference Architecture of the Assurance Framework Toolkit'. Furthermore, it will largely offer the content to be presented in the demo deliverable D6.3

[1] Most of the herein presented functionary will be showcased in the follow-up D6.3 demo. Some screenshots of the application appear in the Appendix A 1: Screenshots of the AFT operation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

2 The AFT High-level Description

2.1 The CC Security Target

In the Common Criteria for Information Technology Security Evaluation (CC) standard [3], the Security Target (ST) is a key-document that adopts a certain structure (describing the ToE assets, threats, assumptions etc.) and terminology to formally define the involved security functional requirements (SFRs) and security assurance requirements (SARs). The ST refers to one specific ToE implementation.

The relevant ASE class [3] includes the evaluation (by the employed evaluator) of the ST that the ToE developer has earlier complied.

2.2 AFT Entities

The different entities that the expert user can input into the system represent the different parts of the Security Target (e.g., assets, assumptions, threats, security objectives etc). Entities are separated into types which define the way that they can be related to each other. For each ToE, the AFT expert (user) must define the various entities (e.g., threats, SFRs and so on) and also create the connections between them in line with CC definitions (see Figure 8).

2.3 Security Target Editing

The main AFT functionality amounts to the support of the ToE developers in compiling the ST. This is achieved in the following way: after choosing a Product the toolkit presents the appropriate entities like Assets, Threats etc in each section of the ST, thus guiding them through the process and thus, save time and resources. In case the ToE developer needs an entity not already defined in the AFT knowledge base (i.e., a relevant threat does not exist in the initial data coming from the SAFERtec modular PP [8]) then they can author a special one for use to cover specific needs.

2.4 Design and Functional Specifications Editing

Another central evaluation class relates to the examination of the ToE's design documents. Those documents are expected to include detailed design specs and source code reviews as well as inputs describing the interfaces of the ToE and the relation of the (earlier identified within the ASE class) SFRs to each interface.

The relevant ADV class [3] includes the evaluation (by the employed evaluator) of the design documents that the ToE developer has earlier complied. AFT seeks to help the ToE developer by introducing an automated yet comprehensive ToE description.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

3 The AFT Software Components

Software products, especially those designed to scale, need to be organized into different logical sections such as folders, namespaces etc. This section presents the basic components of the AFT source code, available for download in [1], along with a brief explanation of the functionality that each one realizes.

3.1 Startup

This class basically performs two functions. It sets up the application to receive requests and then enters a loop during which it serves any requests which arrives. Towards that end, it creates a webhost builder [4] and then runs the host produced. Initial configuration is done by passing a startup class. This class contains two basic methods: *ConfigureServices* and *Configure* (Figure 1).

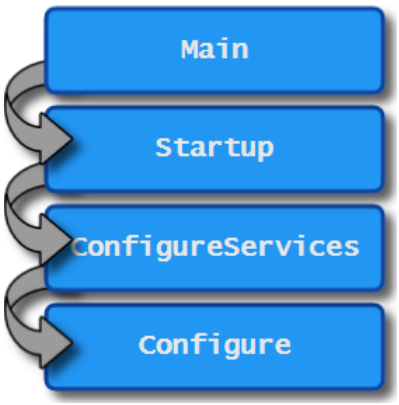

Figure 1 Startup class and included methods

The first one sets up the necessary application services while the method uses these services to set up the actual HTTP pipeline. This is where a set of AFT features are initialized; some are presented in the list below:

- SPA static files serving
- Database connection pool
- JWT authentication mechanism
- Response compression
- Reverse proxying
- Routing
- User authorization
- Other custom services used by the controllers

The last one is where the main AFT application logic resides.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

3.2 Configurations

This folder contains some custom configuration options used by the application. Specifically, the authentication scheme, the authorization policy and some details of the JWT validation parameters. It is not necessary for these to be kept separately, but this helps group some configuration details to prevent clutter in the main application files.

3.3 Migrations

As previously mentioned, the AFT uses a code-first approach to the data models. This means that any time a persistent structure is required, it is created as a class in the application and that every time a change is warranted it is applied to one of those classes (rather than design a database first and then create the classes). Subsequently, there is a need to translate those classes and their relationships into RDBMS [14] constructs such as tables, columns, keys and indices.

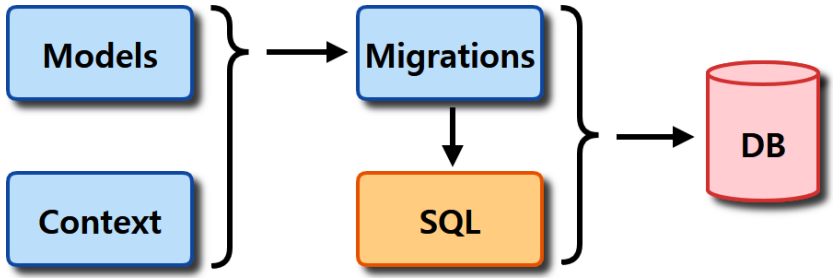

Figure 2 The usage concept of the migrations class

This functionality is served by the Migrations folder which hosts a number of scripts triggering a series of changes to the database such as table creation and alteration (Figure 2). The user can generate these using the Entity Framework tool and apply them sequentially to any database. Thus, any database can be brought in line with any point in the project's history, either forward or backwards, with the minimum possible data loss. This also serves as an additional documentation and can be further customized according to any AFT requirements which might developed in the future.

3.4 Controllers

All requests are routed to the controllers which are responsible for processing them. Those controllers constitute the software that enables the interaction with the application. Any number of similar

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

actions can be grouped in a single controller to help streamline the interface with the AFT which contains the following controllers:

- Authentication which handles things like user login and password changes
- User management for adding users and giving them rights
- Entity management for populating CC components
- Relationship management for connecting entities
- Target management for constructing Security Targets

Figure 3 AFT Controllers and Services interplay

Controllers also make use of services (Figure 3). The way they obtain those services is through constructor injection. Each controller specifies in its constructor declaration which interfaces to services they require. The framework, through reflection makes sure that these services are available when each controller is instantiated, by constructing a single instance of them.

3.5 Services

Services perform the actual work of the application so most of the core logic of the AFT is implemented in an appropriate service. Most custom services used are scoped, which means there is a single instance of them created for each request. A relevant requirement suggests that each service must implement an interface. While not a hard requirement, it is useful in making the program modular and at the same time facilitates unit testing by decoupling involved components.

3.6 Interfaces

As mentioned above, all services are implemented against interfaces. In statically typed object-oriented languages, interfaces are essentially abstract types which define a set of attributes that the implementing type must adhere-to. This is akin to a protocol which the implementations must follow. Their use is important since it decouples the implementation of different components, in this case controllers and services and allows for easy unit testing.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

3.7 ViewModels

These classes define the objects which are required by the REST API [5] as well as the objects returned by the same. They are slim objects with little or no functionality, but they have the purpose of concretely defining the serializable models of the API.

3.8 ClientApp

This is where the client-side of the single-page application resides. It holds a significant amount of functionality since the complexity of the subject requires it. These files are not served as-is; instead they are transpired when the application is compiled, and the result is downloaded to the client upon request.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

4 The Implemented Functionality

4.1 Authentication

The JSON Web Token (JWT) is a standardized way to exchange claims between parties [6]. They are composed of three sections. These sections are separated by dots for simplicity and are base-64 encoded for transmission over the web (see Figure 4).

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI
6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.XbPfbIHMI6arZ3Y922BhjWgQzWXcXNrz
0ogtVhfEd2o
```

Figure 4 JSON token encoded data format

The first section describes the type of token and the algorithm used for the signature. The second section is the list of claims. These claims can come from a predefined list of custom and may include things like the username, the issuer and an expiration date for the claim. The final part is a hash of the previous two parts combined with a secret that is private to the issuer (see Figure *5*). This is how the token is validated.

```
{
  "alg": "HS256",
  "typ": "JWT"}.
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022}.
HMACSHA256(
  base64UrlEncode(header) + "." + base64UrlEncode(payload), secret)
```

Figure 5 *Payload of AFT JSON token for user identification*

This is the token that the AFT expects in order to identify the user in all relevant requests that require it. The process to produce the token is simple. The client sends the credentials, username and password, and the AFT retrieves the previously stored password hash (implemented by PBKDF2 [13]) from the database (see Figure 6 for an overview). It hashes the password received and then compares the two hashes; in case they match the user is verified and receives the AFT access in the coming response.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

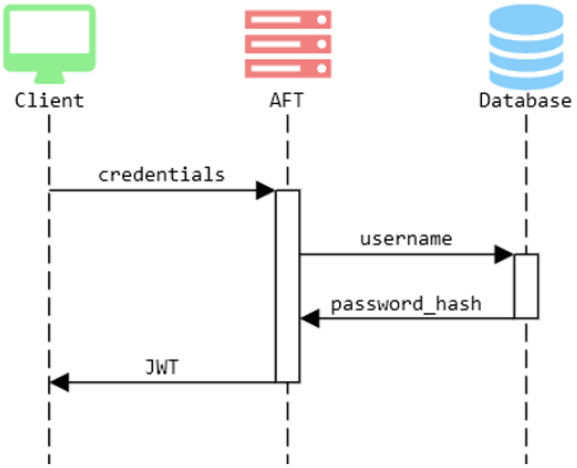

Figure 6 The AFT log-in process

4.2 Search feature

The AFT can potentially have to manage a very large amount of data. Since there are many products with multiple modules, assets, threats and other related entities, there is a need for the toolkit to not only keep all that data stored but also provide the ability for an easy search by the user. This requires some extra consideration so that this potentially large amount of data remains accessible in a user-friendly way.

More specifically, the user is expected to search for a particular entity by entering text and receiving results based on partial text matches on the title of the entities. The most efficient means of handling these searches is to push the relevant query to be performed on the database which means that optimizing the indices will be crucial to the application. A good way to handle these searches which are textual is to create a Generalized Inverted Index (GIN) with trigrams [7]. This means that every word in the selected columns will be broken to three letter combination and those will be used to construct the index. Although it has a small penalty during insertion it compensates by a relatively fast search.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

```
Seq Scan on "Entities"  (cost=0.00..4620.26 rows=20 width=84)
                (actual time=383.082..383.082 rows=0 loops=1)
    Filter: ("Name" ~~* '%filter%'::text)
    Rows Removed by Filter: 202901

Planning time:    0.863 ms
Execution time: 383.099 ms
```

```
Bitmap Heap Scan on "Entities"  (cost=52.15..126.53 rows=20 width=84)
                          (actual time=0.274..0.274 rows=0 loops=1)
    Recheck Cond: ("Name" ~~* '%filter%'::text)
    ->  Bitmap Index Scan on "IX_Entities_Code_Name"
                                            (cost=0.00..52.15 rows=20 width=0)
                                       (actual time=0.272..0.273 rows=0 loops=1)
        Index Cond: ("Name" ~~* '%filter%'::text)

Planning time:  1.228 ms
Execution time: 0.339 ms
```

Figure 7 Comparison of seek times related to the usage of index

The advantage of an AFT item search performed with the index in terms of time is depicted in Figure 7. Our results suggest a large improvement compared to a search without an index.

4.3 Data Models (ASE evaluation class)

The entities presented in deliverable 6.1 were implemented in the database using two simple tables. One has been implemented to hold the entities and the other holds the relationships. This made for a simple implementation which can easily be expanded in the future if there is a need.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

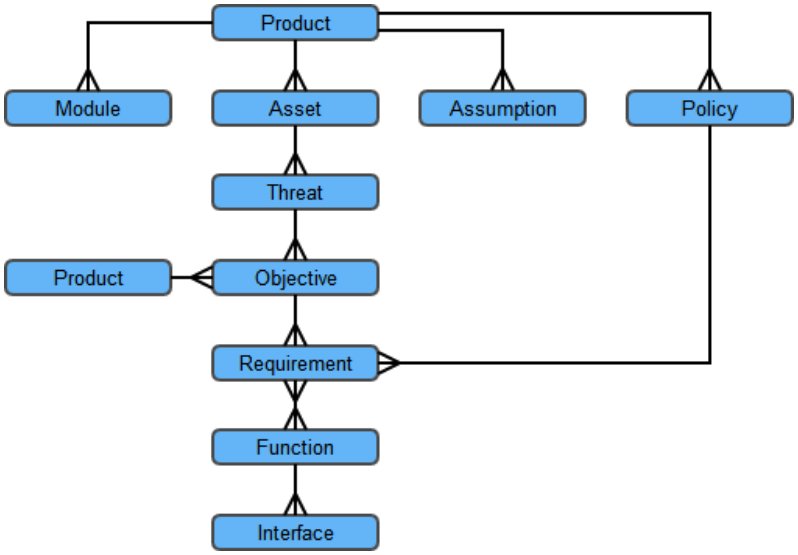

Figure 8 The AFT schema (E-R diagram) to support the Security Target compilation (links appear in line with CC definitions)

The first table holds the data for each entity including the kind of entity it is. Each one has an integer identifying it, which is ensured by the database to be unique. That essentially provides the means to realize the needed AFT *knowledge base* that includes the data and metadata for different modules of each considered ToE in the connected vehicle system.

Column	Type	Nullable	Storage
Id	integer	not null	plain
Code	text		extended
Name	text		extended
Description	text		extended
Kind	integer	not null	plain

Table 2 An AFT data structure to keep entities data

A relationship is a link between two elements and in our case, it must be able to express a many-to-many type of connections in order to support the various links between the entities involved in the ST compilation (see Figure 8). Note here that both CC and the SAFERtec framework define an implementation-independent syntax (i.e., Protection Profile) and subsequently, one that refers to a specific ToE implementation (i.e., Security Target) for systematically specifying the security requirements to be met [8].

The software entities that seek to accurately represent the various security target entries are represented in the E-R diagram of Figure 8 with the core object being the Product (i.e., the ToE).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

Table 3 holds two references to the first table, representing the relationship (between entities). A foreign key [9] to the first table is used for each column making sure that each entry corresponds to an actual entity. Additionally, the table also implements a constraint which is that the integer on the first column must be smaller than the integer on the second column. This way each relationship can only be expressed in one way and self-referencing relationships are prohibited since no number is greater than itself.

Column	Type	Nullable	Storage
LesserEntityId	Integer	not null	plain
GreaterEntityId	integer	not null	plain

Table 3 An AFT data structure to implement relational constraints

Relying on the above data structures, AFT is enabled to assist the ToE developer in compiling the requested security target (ASE evaluation class); the identification of assets (i.e., some data have been already entered in the first table), the involved threats, assumptions and security objectives as well as the relevant SFRs for his product can be carried-out in a cost-effective way.

4.4 Graphical tool (ADV evaluation class)

AFT has been enhanced to provide assistance to the ToE developer in order to automate (as much as possible) the thorough description of the ToE architecture, functional specifications and interfaces in line with the CC suggestions [3]. A simple graphical tool (see Figure 19) that allows the accurate description of the ToE's design and interface specifications has been developed and incorporated in the AFT. Towards that end, a diagramming client-side library has been used. The AFT graphical environment has been carefully designed to 'force' the ToE developer to provide the CC mandatory elements and justify the associated relationships.

Thus, AFT is expected to ease both the ToE developer and the evaluator (i.e., with the availability of more harmonised evaluation inputs) in the context of the ADV class.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

5 The AFT Exposed Interfaces

The web interface is how the AFT communicates with external entities. Currently the single-page application implements the API to provide a graphical UI but it is extendable and can potentially become compatible with a mobile application or any sort of middleware.

Path	Method	Controller	From	Input Parameters	Returns
api/auth/login	POST	AuthController	Body	LoginRequest	JWT
api/auth/changePassword	POST	AuthController	Body	ChangePasswordRequest	Success
api/entity/list	GET	EntityController	Query	Kinds, Filter, Free, Skip, Limit, ForKind	Entity list
api/entity/create	POST	EntityController	Body	Entity	Entity ID
api/entity/update	PUT	EntityController	Body	Entity	Success
api/entity/delete	DELETE	EntityController	Query	Entity ID	Success
api/entity/details	GET	EntityController	Query	Entity ID	Entity
api/relationship/create	POST	RelationshipController	Body	Entity ID, Entity ID	Success
api/relationship/update	PUT	RelationshipController	Body	Entity ID, Entity ID	Success
api/relationship/delete	DELETE	RelationshipController	Query	Entity ID List	Success
api/target/create	POST	TargetController	Body	Target	Entity ID
api/target/update	PUT	TargetController	Body	Target	Success
api/target/delete	DELETE	TargetController	Query	Target ID	Success
api/target/details	GET	TargetController	Query	Target ID, Page	Target
api/target/addEntity	POST	TargetController	Body	Target ID, Entity ID	Success
api/target/removeEntity	DELETE	TargetController	Query	Target ID, Entity ID	Success

Table 4 AFT functionality exposed through programming interfaces

The result that these methods return depends on what is needed. Whenever a resource is created the integer identifier of the created resource is returned to the client. Whenever the client makes a request for data, the requested object type is returned.

In what follows we briefly describe the involved API parameters.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

5.1 Paths

All paths are organized according to the resource they wish to get data for and need to manipulate. Further organizing is done according to the method. All paths follow the template "***api/[controller]/[action]***" where action corresponds to a method on the controller. The names match without that being necessary.

5.2 Methods

The HTTP method assigned to each request follows the standard REST conventions:

1. GET for requesting data
2. POST for creating resources
3. PUT for updating resources
4. DELETE for deleting resources

5.3 Rights

Except for the login method, all other requests require that the user is authenticated via the JWT token. However, each action requires that the user has the appropriate rights. For viewing or changing the entities resources the user needs to have management access to them and the same goes for the target resources, since these two functionalities target different audiences. The implementation of these access rights is minimal but remains quite extensible.

5.4 Responses

The responses contain two main parts: the response code and the content of the response. In case the JWT token is missing or somehow malformed, that means that the user can't be verified and the AFT responds with "401 – Unauthorized". In case the user tries to access a resource that he is not allowed the response is "403 – Forbidden". In most cases whereby something unexpected happens the server responds with "500 – Internal Server Error". Otherwise, if the request is successful the response is "200 – OK" and it is potentially accompanied with an object response for the client. When a resource is created then the only information required is the ID of the newly created entity. Finally, in the case of a GET request the appropriate object, or list of objects is returned.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

6 Development and Deployment

6.1 Development toolchain

Thanks to the multiplatform nature of all technologies used, the AFT development can be possible in all major platforms: Windows, Linux of macOS. A development environment with an IDE, a database and a web server can be set up in all the above platforms with no issue. An important part of the process is a version control system. The Git distributed version control system [10] was selected for its robustness, reliability and wide adoption. Additionally, a Gitlab server was used for repository management purposes. The AFT development approach is depicted in Figure 9.

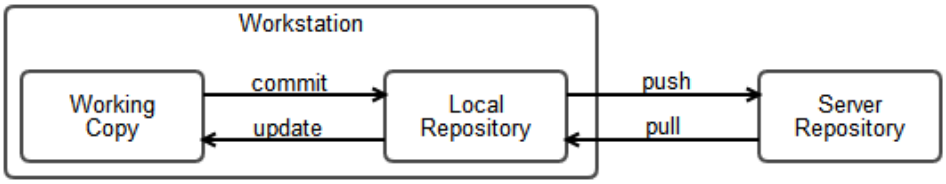

Figure 9 Development process of the AFT software

6.2 Deployment

The AFT has been deployed on a Linux virtual machine. The code is downloaded from the central repository in Gitlab. A PostgreSQL server is installed on the VM which if needed is updated to the schema required by the application. The C# code (i.e., AFT back end) is compiled, and the Typescript code (i.e., AFT front end) is compiled into JavaScript. The application is then served using kestrel, a cross-platform web server [11]. The application is not exposed directly on the internet, instead requests are proxied using a Nginx web server which is used as a reverse proxy and which also holds the HTTPS certificate (see Figure 10).

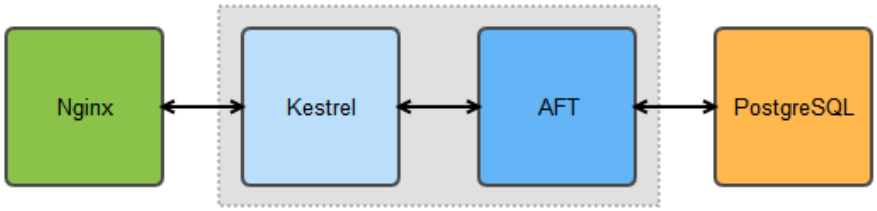

Figure 10 The AFT code deployment chain

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

7 Fulfilment of Software Requirements

What follows is an overview of the correspondence between the requirements presented in deliverable D6.1 and the features of the implementation of the AFT. As justified in the below bullet points, some requirements were fulfilled by specific choices in architecture and others are features of the underlying platforms:

	SPA	.NET	PSQL	MVC	Git	Future
Adaptability	x			x		
Auditability					x	x
Backup						x
Deployment	x	x				
Documentation						x
Extensibility				x		
Interoperability	x			x		
Maintainability	x	x				
Performance		x	x			
Reliability		x	x			
Scalability		x				
Security	x	x	x			
Testability	x			x		

Table 5 AFT architecture and technologies enable the requirements' fulfilment

- The SPA architecture is modular and thus it helps when adding new features and when testing. Also, its reliance on HTTP requests allows for interoperability with other technologies. Additionally, since Angular is accompanied by CLI tooling and IDE integration it offers easy deployment and good maintainability. Authentication was also straightforward to add using service injection.
- The .NET platform has also a CLI toolset and a good IDE integration helping deploy and maintain the toolkit. Since it is a mature and widespread platform it offers good performance in a reliable yet scalable fashion.
- The PostgreSQL RDBMS [14] is also mature, stable, secure and has many features to increase performance like trigram indices.
- The MVC architecture is language-agnostic and has enabled the project to keep clean interfaces and allow for changes and additions in requirements. Finally, it allows for inversion of control making tests easier.
- The git distributed versioning software helped with tracking changes to the source code and keeping everything project-related auditable.
- Future developments will include data auditing, backup management and documentation.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

8 The AFT Software Testing

Unit testing consists of authoring tests for parts of the code in isolation. Usually each class is subjected to a few tests (see an indicative summary in Figure 11) to make sure that its behaviour is the intended one for all different inputs. This has a number of benefits like increasing confidence in the software under test, allowing for easier changes and faster defect detection. In addition to that, designing code for easier testing leads to better and more modular code with clear separation of concerns and reusability. The two basic types of code tested were controllers and services.

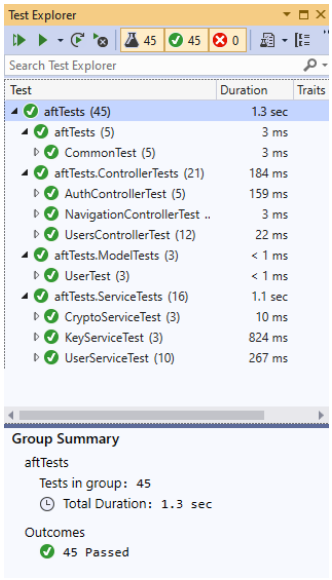

Figure 11 A summary of test results

8.2 Testing the AFT controllers

Testing the controller modules isolated from everything else was the focus here. What these tests examined was how the system was supposed to function with different users and differently formed requests. Furthermore, the way AFT should function with regards to the underlying services was examined. A relevant testing code example appears in Figure 12.

```
[Fact]
public void TestUnknownUser()
{
    var model = new LoginRequest() {
        Email = "unknown@domain.test",
        Password = "password"
    };
    var result = controller.Login(model);
    Assert.IsType<BadRequestResult>(result);
}

[Fact]
public void TestWrongPassword()
{
    var model = new LoginRequest() { Email = userEmail, Password = "wrong" };
    var result = controller.Login(model);
    Assert.IsType<BadRequestResult>(result);
}
```

Figure 12 An example of controllers' test

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

8.2 Testing the AFT services

These tests examined the core functionality of the toolkit. Typically, they are more varied than the controller tests which all tend to be somewhat similar. A relevant testing code example appears in Figure 13.

```
[Fact]
public void TestSameKeyReturned()
{
    var firstResult = service.GetJwtSecurityKey();
    var secondResult = service.GetJwtSecurityKey();
    Assert.Equal(firstResult.Key, secondResult.Key);
    Assert.Equal(1024, secondResult.KeySize);
}

[Fact]
public void TestUpdateSecurityKey()
{
    var firstResult = service.GetJwtSecurityKey();
    service.UpdateSecurityKey();
    var secondResult = service.GetJwtSecurityKey();
    Assert.NotEqual(firstResult.Key, secondResult.Key);
}
```

Figure 13 Example of code for AFT services testing

9 Future Directions and AFT Implications

There are numerous future directions related to the development of AFT. Those are mainly caused and further fuelled by the fact that the relevant automated tools or document templates are scarce [12].

One key-point in the future activities is the population of the AFT knowledge base with relevant data. Clearly, the more the available data (in terms of examples, references, explanations) the higher the AFT usability and efficiency. Enhancing AFT with such data will help both the ToE developers to provide the required inputs more efficiently (e.g., much better understanding of input requirements without spending time resources for their identification) and the evaluators (e.g., higher quality of inputs implying more easy evaluation). A relevant point is the addition of certain tools (e.g., pop-ups, info-buttons) that will further improve the usability and AFT user-friendliness.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

Importantly, all above additions and improvements relate to the toolkit's maintenance and further investments. The involved project partners (ICCS as the developer and Oppida as the consultant) seek to invest effort on the AFT establishment as an effective means for CC-based security evaluations. Both parties share a similar vision (for their own purpose) that makes them commit to further development of the toolkit beyond the SAFERtec lifetime and promotion to relevant evaluators groups/associations.

In terms of the current/short-term improvements, the toolkit is already under process for the addition of features that support the ATE evaluation class [3]. That one includes the evaluation of functional and independent tests whereby the ToE developer is asked to provide a test plan including tests scenarios or test scripts. Detailed information for each test scenario such as prerequisites, operations, and expected results are also needed. The provided documentation and results should justify that they are sufficient to cover every interface related to ToE security function, earlier identified along the ADV class. An AFT automated verification of this coverage and the associated rationale is the targeted feature. Finally, the AFT structured information (i.e., assets, interfaces, SFRs) can serve as a basis for an efficient test-plan definition.

The implications of the introduced toolkit are broad. AFT, when populated with all relevant data, can be clearly used by security evaluators as well as benefit a large part of the automotive industry supply chain such as OEMs and Tier-1 providers. The generated Security Targets of various involved products (e.g., on-board units, etc.) include details (e.g., SFRs) that can become relevant for security engineers of the ITS industry. Importantly, the more the available data the higher the AFT efficiency.

Finally, on a more visionary note, an interesting implication points to the AFT's standardization potential (see also the Deliverable D7.5); as evaluation input data need to respect the strict CC/SAFERtec framework structure it is not easy for an automotive product developer to know if the information he provides is in the appropriate structure/format. AFT, if standardized, would require inputs that meet both CC/SAFERtec framework and automotive technologies constraints. That would help automotive developers produce evaluation inputs, ease the load of the evaluators and further lower the involved costs.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

10 Conclusions

To support the application of the SAFERtec assurance framework (as well as any other CC-based evaluation) and minimize relevant costs in terms of time and resources, we have introduced a platform-independent open-source on-line toolkit (AFT) available for download in [1].

In this document we have relied on the earlier-presented software design and described the development details of the AFT toolkit. The way it is implemented to facilitate the efficient evaluation-data collection and compilation of required entries is described. Two demanding CC evaluation classes are currently supported but the AFT bears some extensions to support a third one.

The toolkit will be offered as an open-source software to OEMs and most notably to security evaluators aiming to fill-in a significant technology gap and thus, contribute to the increase of trust in connected vehicle.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

References

[1] The AFT code and documentation appears at: https://isense-gitlab.iccs.gr/user

[2] P. Pantazopoulos et al., "*Towards a Security Assurance Framework for Connected Vehicles*," in IEEE 19th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), Chania, 2018, pp. 01-06.

[3] ISO/IEC 15408 part 1/2/3:2005. 1996.Information technology, Security techniques, Evaluation criteria for IT security. Retrieved April 12, 2020 from https://www.commoncriteriaportal.org/cc/ Tech. Rep., v3.1, Release 5

[4] .NET API The WebHostBuilder Class : https://docs.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.hosting.webhostbuilder?view=aspnetcore-3.1

[5] The REST API tutorial https://www.restapitutorial.com/

[6] Jones, Michael, Brain Campbell, and Chuck Mortimore. "JSON Web Token (JWT) profile for OAuth 2.0 client authentication and authorization Grants." May 2015. [Online]. Available: https://tools.ietf.org/html/rfc7523

[7] Optimizing Pattern Searching Queries in PostgreSQL https://medium.com/@paramsingh96/optimizing-pattern-searching-queries-in-postgresql-84087d9bac8c

[8] K. Maliatsos, C. Lyvas, P. Pantazopoulos, C. Lambrinoudakis, A. Kanatas, M. Gay and A. Amditis, "*Standardizing Security Evaluation Criteria for Connected Vehicles: A Modular Protection Profile*". In 2019 IEEE Conference on Standards for Communications and Networking (IEEE CSCN), pp. 1–7

[9] Foreign Key Constraint https://en.wikipedia.org/wiki/Foreign_key

[10] The Git open source distributed version control system Available [Online] : https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

[11] Kestrel is an open-source, event-driven, asynchronous server used to host ASP.NET applications on any platform: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel?view=aspnetcore-3.1

[12] Angelos Stamou, Panagiotis Pantazopoulos, Sammy Haddad and Angelos Amditis, "*An Online Toolkit for Efficient Common Criteria Cyber-security Evaluation of Connected Vehicles*", March 2020, submitted for a conference publication

[13] Internet Engineering Task Force, "The Password-Based Key Derivation Function 2 (PBKDF2)", January 2011 https://tools.ietf.org/html/rfc6070

[14] Open source object-relational database system https://www.postgresql.org/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

Appendices

A 1: Screenshots of the AFT operation

Figure 14 Landing page of the SAFERtec AFT

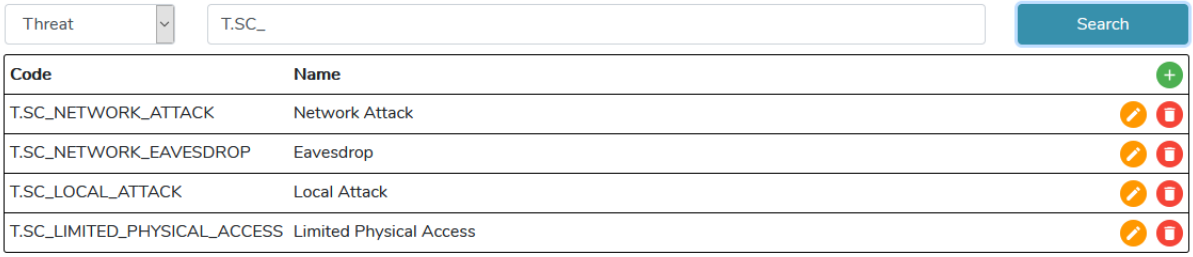

Figure 15 An AFT search for a threat

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

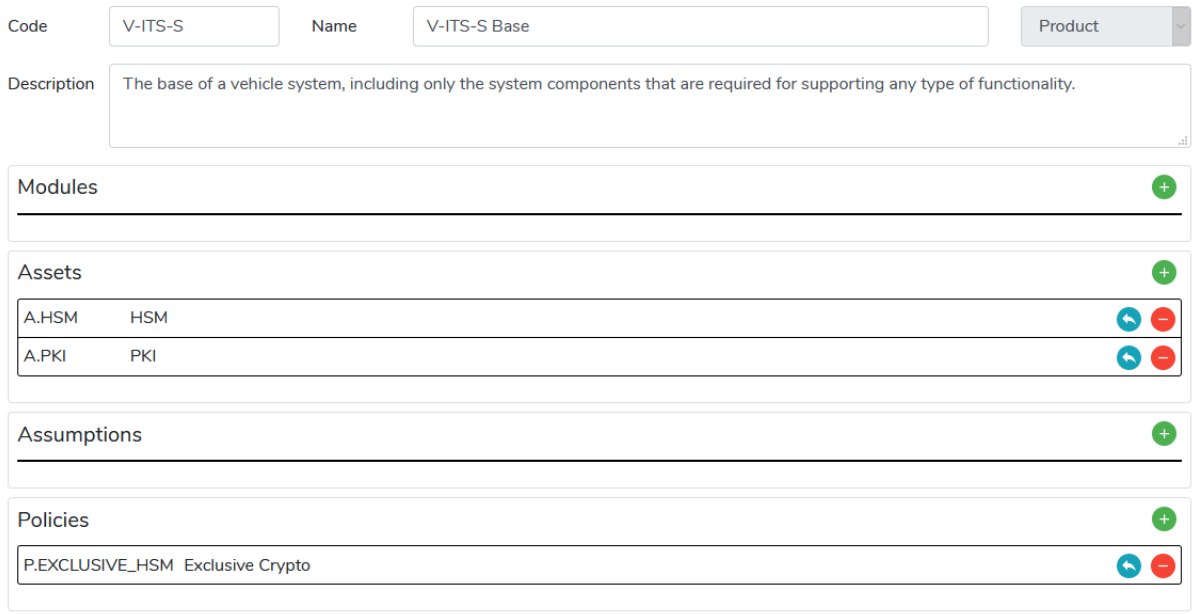

Figure 16 A snapshot of a product (ToE) available entries

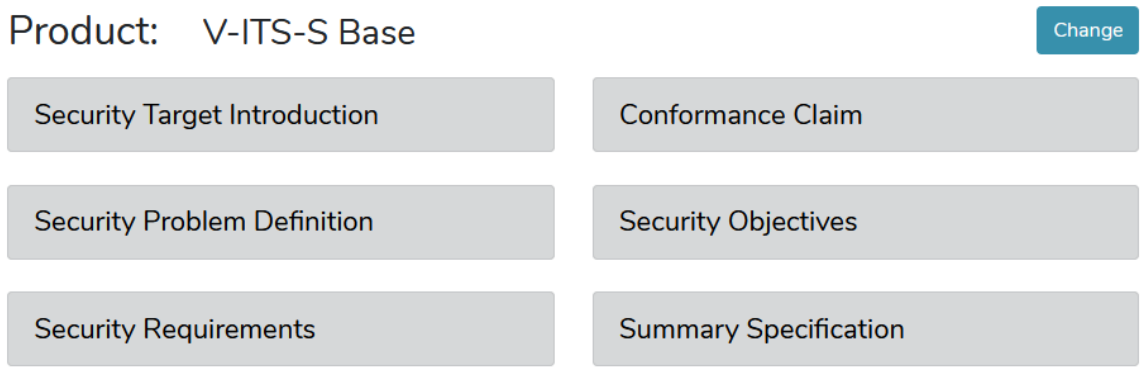

Figure 17 A Security Target main menu

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

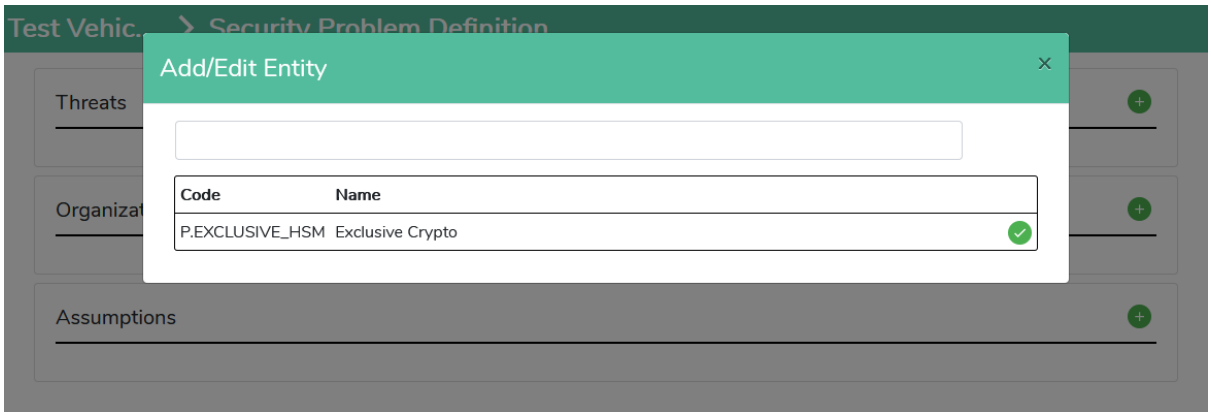

Figure 18 AFT editing of an entity

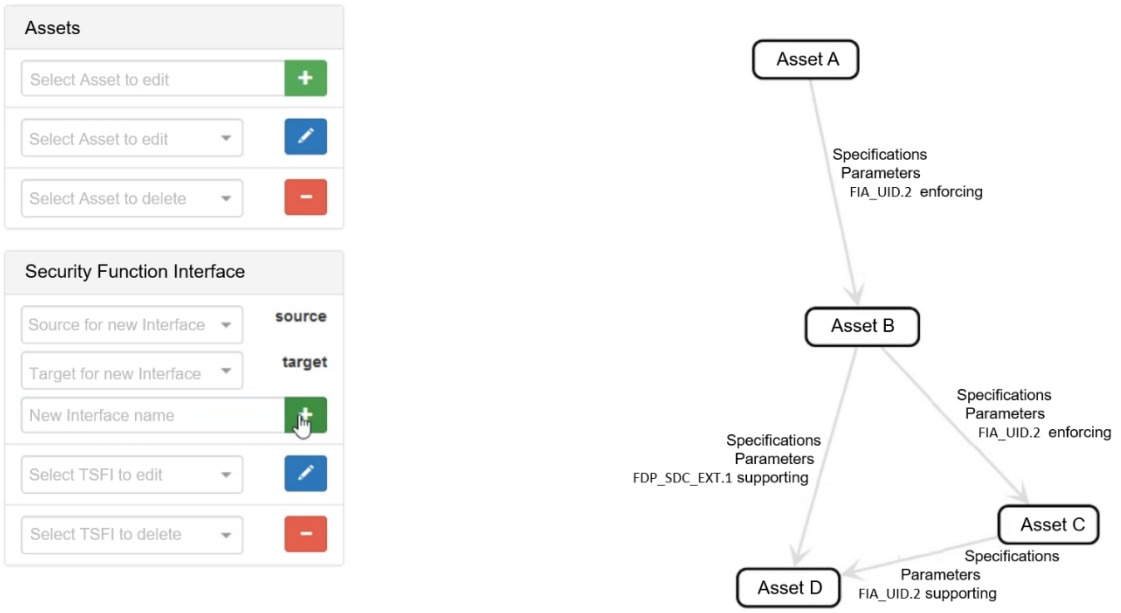

Figure 19 The AFT graphical tool to support evaluation tasks of the ADV class

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319

A 2: AFT user-friendliness and usability

The AFT user-friendliness has been carefully considered along the toolkit's design and implementation phases. A number of comments received by the toolkit testers and the consortium partners, have led to improvements along the development and evaluation tasks. Further to them, a set of main characteristics that are listed in the following points promote the toolkit's user-friendliness:

- Two interfaces were developed in unison in order to support users of both roles i.e., the typical user/Toe developer and the expert user; thus, to promote user friendliness.
- Modern UI design principles have been adopted for a responsive and clean design. Typescript has been used for the UI logic
- A clear distinction between the evaluation classes supported by the tool is provided through a relevant menu, as part of the UI functionalities. Furthermore, numerous information messages have been added using appropriate pop-up messages (e.g., "State conformance to relevant standards ") with relevant pointers to useful external documents in order to guide the developer in providing the needed security evaluation data.
- Reliable and concise software documentation has been prepared to ensure the toolkit's user-friendliness. A read-me file appears also in the online repository of the AFT source code [1]. Finally, a more complete familiarization of users can be achieved by the broad usage of the AFT video demonstrator (D6.3).

Regarding the AFT usability, a number of evaluation comments were received by the (only) consortium partner that can serve as a potential user of the toolkit (i.e., being an accredited security evaluation lab). Those were carefully considered by the WP6 partners to improve the AFT functionality. Further comments from the relevant community (including also authorities such as ANSSI) were sought after but a number of reasons hinder their systematic analysis and presentation.

Evaluation comments of the AFT usability were expected at the project final event. A draft video of the AFT scope and capabilities was prepared for the event and presented to the participants. The expected feedback was planned to be considered for limited improvements (due to time constraints) and most notably for the AFT future extensions/upgrades. Unfortunately, the event was shifted from a physical meeting to an online session due to COVID-19 measures; the interaction with the invited experts was therefore limited while the attendance of highly relevant audience to AFT (i.e., security assurance evaluation experts) was very low.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732319