Deliverable D5.5
Security monitoring and compliance assurance

<table>
<thead>
<tr>
<th>Editor(s):</th>
<th>Niall Burns, Ewan Dawson, Graham Davidson</th>
</tr>
</thead>
<tbody>
<tr>
<td>Responsible Partner:</td>
<td>Symphonic</td>
</tr>
<tr>
<td>Status-Version:</td>
<td>Final</td>
</tr>
<tr>
<td>Date:</td>
<td>28/12/2017</td>
</tr>
<tr>
<td>Distribution level (CO, PU):</td>
<td>CO</td>
</tr>
<tr>
<td><strong>Project Number:</strong></td>
<td>GA 727301</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------</td>
</tr>
<tr>
<td><strong>Project Title:</strong></td>
<td>SHIELD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Title of Deliverable:</strong></th>
<th>Security monitoring and compliance assurance (M7-M30, Lead: Symphonic, Participants: TECNALIA, AIMES)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Due Date of Delivery to the EC:</strong></td>
<td>23/12/2017</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Workpackage responsible for the Deliverable:</strong></th>
<th>5.5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Editor(s):</strong></td>
<td>Phil Scott (Symphonic)</td>
</tr>
<tr>
<td><strong>Contributor(s):</strong></td>
<td>Niall Burns, Ewan Dawson</td>
</tr>
<tr>
<td><strong>Reviewer(s):</strong></td>
<td>Ed Conley, Xabier Larrucea</td>
</tr>
<tr>
<td><strong>Approved by:</strong></td>
<td>All partners</td>
</tr>
<tr>
<td><strong>Recommended/mandatory readers:</strong></td>
<td>All WP’s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Abstract:</strong></th>
<th>This task will identify the requirements of and implementation of systemic assurance both for authorised accesses and denied access where attempts have been made and do not meet established criteria. Assurance will be considered from both entitlements to access, and effected access.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keyword List:</strong></td>
<td>Consent, consent management, trust, policy, governance</td>
</tr>
<tr>
<td><strong>Disclaimer:</strong></td>
<td>This document reflects only the author’s views and neither Agency nor the Commission are responsible for any use that may be made of the information contained therein</td>
</tr>
</tbody>
</table>
### Document Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Modifications Introduced</th>
<th>Modified by</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1</td>
<td>26/09/2017</td>
<td>First draft ToC version</td>
<td>Niall Burns, Ewan Dawson (Symphonic)</td>
</tr>
<tr>
<td>V0.2</td>
<td>29/11/2017</td>
<td>Second Draft</td>
<td>Phil Scott</td>
</tr>
<tr>
<td>V0.3</td>
<td>01/12/2017</td>
<td>Third Draft</td>
<td>Phil Scott</td>
</tr>
<tr>
<td>V0.4</td>
<td>05/12/2017</td>
<td>Review draft</td>
<td>Phil Scott, Niall Burns</td>
</tr>
<tr>
<td>V0.5</td>
<td>20/12/2017</td>
<td>Draft following review Ed Conley and Xabier Larrucea</td>
<td>Phil Scott, Derick James</td>
</tr>
<tr>
<td>V0.6</td>
<td>21/12/2017</td>
<td>Post Review Modifications</td>
<td>Derick James</td>
</tr>
<tr>
<td>V1.0</td>
<td>28/12/2017</td>
<td>Final version</td>
<td>Xabier Larrucea</td>
</tr>
</tbody>
</table>
## Table of Contents

Table of Contents ........................................................................................................................................ 4
List of Figures .................................................................................................................................................. 5
List of Tables ................................................................................................................................................ 5
Terms and abbreviations ............................................................................................................................... 5
Executive Summary ....................................................................................................................................... 6
1 Introduction .................................................................................................................................................. 7
    1.1 Objective and Scope ............................................................................................................................ 7
2 Background ................................................................................................................................................ 7
3 Analysis ...................................................................................................................................................... 9
4 System Architecture .................................................................................................................................. 10
5 SHiELD Use Cases .................................................................................................................................. 13
6 Static Analysis of policies ........................................................................................................................ 14
    6.1 Analyses ............................................................................................................................................... 14
        6.1.1 Conflict Analyser ........................................................................................................................ 14
        6.1.2 No policy is redundant ...................................................................................................................... 15
        6.1.3 Data access respects privacy levels .................................................................................................. 15
    6.2 Analysis refinement .............................................................................................................................. 15
    6.3 Model Querying ................................................................................................................................... 16
7 Auditing ..................................................................................................................................................... 17
    7.1 Realtime Refinement Audits ................................................................................................................. 17
    7.2 Auditing of External Services ................................................................................................................ 18
    7.3 Self Audit ............................................................................................................................................. 18
8 Integration .................................................................................................................................................. 19
9 References .................................................................................................................................................. 20
List of Figures

FIGURE 1. INFORMATION FLOW PROCESS .................................................................................. 12
FIGURE 2. SUPPORTING COMPONENTS FROM THE SHIELD DoD (DESCRIPTION OF WORK) .......... 14
FIGURE 3. REFINEMENT PROCESS ........................................................................................ 17
FIGURE 4. MODEL QUERYING PROCESS ................................................................................. 17
FIGURE 5. EXTERNAL SERVICE AUDIT .................................................................................... 18
FIGURE 6. SELF- AUDIT DECISION PROCESS ......................................................................... 19

List of Tables

TABLE 1 EXISTING CORE COMPONENTS .................................................................................. 10
TABLE 2. EXISTING OPEN SOURCE COMPONENTS ................................................................. 10
TABLE 3. PLANNED CORE COMPONENTS ................................................................................. 11
TABLE 4. PLANNED OPEN SOURCE COMPONENTS ................................................................. 11
TABLE 5. PRIVACY BY DESIGN VS. USE CASES .................................................................. 13

Terms and abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE</td>
<td>Decision Engine</td>
</tr>
<tr>
<td>NCP</td>
<td>National Contact Point</td>
</tr>
<tr>
<td>PEP</td>
<td>Policy Enforcement Point</td>
</tr>
<tr>
<td>SAT</td>
<td>SATISFIABILITY or Propositional Satisfiability Problem</td>
</tr>
<tr>
<td>XACML</td>
<td>Xtensible Access Control Markup Language</td>
</tr>
</tbody>
</table>
Executive Summary

According to the DoW, this document is focused on requirements for security monitoring and compliance assurance from a consent perspective. In this sense, consent management is a crucial aspect to be discussed. In fact, this report presents the designs for incorporating capabilities to assess a security assessment of a transaction and to modify access permission patterns to correctly represent the intended security posture of the parties involved in data access, both target and requestor. The factors which will be taken into account, from a security monitoring perspective will be meta data generated at design time and made available by other project work packages so that a dynamic evaluation of security characteristics can be effected at run-time. In this way, an up to date monitoring of known security characteristics. This document will be updated with further requirements and the second release will be a prototype against which other SHiELD architectural elements may be tested.
1 Introduction

1.1 Objective and Scope

This document is focused on requirements for security monitoring and compliance assurance from a consent perspective. Consent management is treated within D5.3, but at this stage, we are focused on providing some security mechanisms based on policies to enhance consent management. Therefore, we are not providing overall security monitoring and compliance assurance tools as a whole. Instead, we are just focused on these aspects from a consent perspective. This means we present security assessment tools for policies designed in the context of consent management. Metadata is being made available at design-time by other work packages, and the tools discussed in this report will also run at design-time and before deployment, using the metadata to formally reason about privacy properties on the policies. In this way, we help enforce privacy-by-design. In addition, we provide retroactive auditing of services based on security assessments produced at design-time.

2 Background

Consent is one of the two major issues [1] in healthcare systems. Consent management [2] is traditionally captured as a piece of paper. But these contexts (e.g., healthcare systems) require to capture the informed consent from users in an explicit way that it cannot be a simple signed piece of paper [3]. Consent management systems must define a specific consent architecture [4]. Sometimes a kind of “teleconsent” [5] is required when users are not physically present, and they should capture and represent semantic models [6] for consent management. Sometimes, consent requires the access to patient data [7], and physicians must deal with ethics aspects [8]. The treatment of health information among different actors (e.g., peer to peer) [9] is the other major issue [1], and this is especially relevant in emergency contexts [10] or even in the IoT (Internet of Things) [11]. Consent and policies are tightly related between them. In fact, according to [12]:

"the patient's consent has a pivotal role in granting or removing access rights to subjects accessing patient's medical data. Depending on the context in which the access is being executed, different consent policies can be applied."

Policy frameworks such as [13] are useful to enhance and provide trust to users. In this sense, our approach based on [14] is to provide an integrated set of tools that supports and enables the creation of a formal structure for abstraction, governance and implementation of trust relationships and security policies. Working across multiple disparate organisations and technologies, it provides a standardised trusted mechanism between all parties for sharing data, whilst maintaining strict conformance to the strongly defined trust framework. Our approach consists of outlining a number of core components (requirements), each of which can operate as stand-alone products or can integrate with existing systems. These are:

TRUST framework

- Define Trust Framework
  - Allows the definition of the components of the Trust Framework including Users, Roles, Organisations, Identity Providers, Attribute Providers, Trust Levels, Relationships, etc.
• **Build Policy Rules**
  o Creates strongly typed policies for access to data or services, tightly coupled to the trust framework
  o Enabling policies to account for basic elements such as user and target alongside customisable elements such as device type, geo-location, IP address, etc with highly customisable logic covering all of these elements
  o Queries, reports and provides visualisation of the complete trust eco-system.

**GOVERNANCE**

• **Validate Policy Rules**
  o Resolves issues with rule sets, including overlaps, shadows, anomalies and conflicts

• **Optimise Rules**
  o Patent pending technology optimises analysis and decision processes producing a highly efficient decision table

• **Decision Engine**
  o Highly scalable and distributable decision engine supports advanced attribute retrieval and request processing.

**GATEWAY**

• **Real-time policy enforcement**
  o Gathers claims from users’ access requests and resolve required attributes to support the decision engine – ultimately providing a permit or deny along with explanatory reasons and advice.

• **Trusted data access gateway**
  o A portal/proxy to the real source of data, acts as the trusted gate-keeper to the data, permitting access only to requests which conform to the pre-defined policies and trust framework.

**AUDIT**

Underpinning these core components is a comprehensive auditing tool giving access to fine-grained audit and logging features.

• **Audit Policy Management**
  o Creates a detailed audit trail of all trust framework and policy changes.

• **Audit Access Rights**
  o Allows exploration of access rights granted/denied by the policies.

• **Audit Access Requests**
  o Creates an audit log of all user access requests made through the governance engine.
3 Analysis

The purpose of the analysis tools is to provide extremely strong guarantees that policies meet and reflect an organisation’s security requirements before they are ever deployed. In the context of SHIELD, this helps to fulfil the requirement of privacy by design. While these tools can be used to identify very general errors in policy design, they will additionally query repositories that store the outputs from WP5.2 and automatically identify potential data sensitive breaches. These tools will be described in detail in Section 5.

Additionally, we propose to use the outputs from our analysis tools to drive security auditing tools, both for real-time security monitoring and for auditing past logs. Security auditing will be described in detail in Section 6.

For background, every policy implemented by Symphonic is capable of accepting an access control request targeted at a particular resource in a context which identifies the agent making the request and their privilege levels. This information is used to evaluate the policy to one of four access values, asserting that the access request should permit, deny or not apply, or in the fourth case, that an error occurred. In addition, a policy may produce a set of obligations which must be fulfilled by a Policy Enforcement Point (PEP), such as a requirement for sensitive data to be redacted from the requested resource.

Access requests are evaluated by the Decision Engine component against multiple policies, with decisions combining according to a set of combining algorithms defined over the four possible decision values. The structure of the Decision Engine’s computations, with inputs coming from the data in the access request and provided by external services (Policy Information Points), can additionally be modelled efficiently in propositional logic, following proposals [15] found for the XACML access control language[16]. As we shall explain, this allows us great opportunities to perform policy analysis.

Propositional logic is a language used for proving and refuting claims concerning finite, but often very large, data sets. For instance, it is possible to use it to verify or find errors in the behaviour of digital circuitry used for large computations. Of particular note is that well understood, robust and efficient software exists to perform these verifications and to find these errors in the form of SAT solvers, in a way that is guaranteed to eventually produce a correct verdict.

It is not expected that all aspects of policy decision making can be fully modelled in this way. For instance, the details of how an XPath query is used to extract data from an XML document is not straightforwardly a propositional logic problem. Here, the analyser will err on the side of caution and may report false-negatives where it thinks, incorrectly, that constraints have been violated. Work on this project will try to keep these limitations to a minimum in the context of privacy and consent enforcement. The software is already capable of modelling policies based on arithmetic calculations, by converting them to binary arithmetic and modelling each bit as a proposition, connected together in a binary adder. This technique, commonly known as “bit-blasting”, can be easily extended to handle other forms of calculation.

When bit-blasting is inadequate, false positives can be propagated to other constraint solvers, which may understand what is and what is not possible in an XPath query, or which may understand, say, when two date ranges cannot be jointly satisfied. They then update the SAT solver with the new constraint, and the loop continues with the SAT solver trying to find another constraint violation. This technique falls under the umbrella of another popular technique in formal analysis known as "Satisfiability Modulo Theories". 
4 System Architecture

The components of our proposed system are divided into:

- existing Symphonic core components;
- existing components that will be licensed open source;
- planned components that will be licensed open source;
- planned Symphonic core components;
- third-party open source components.

These components are summarised in the following tables:

### Table 1 Existing core components

<table>
<thead>
<tr>
<th>Existing core components</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trust Framework</td>
<td>Allows users to define the basic vocabulary of policies in terms of users, roles, organisations, identity providers, trust levels and relationships. Additionally, allows users to define the external services that provide additional data in the form of well-typed attributes and transformations on those attributes.</td>
</tr>
<tr>
<td>Policy Manager</td>
<td>Allows users to define the conditions under which different policies permit or deny, and how these policy decisions combine into higher-level policies, as well as what obligations will need to be met by the enforcement point.</td>
</tr>
<tr>
<td>Decision Engine</td>
<td>Highly scalable and distributable system which supports advanced attribute retrieval and request processing in order to evaluate a request to an access decision and its associated obligations.</td>
</tr>
<tr>
<td>Core Translator</td>
<td>Translates internal trust framework and policy model into an openly specified data structure suited to static analysis.</td>
</tr>
</tbody>
</table>

### Table 2. Existing open source components

<table>
<thead>
<tr>
<th>Existing open source components</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAT Translator</td>
<td>Translates the open policy representation into a formal model in propositional logic, performing bit-blasting for numerical computations.</td>
</tr>
<tr>
<td>Conflict Analyser</td>
<td>Uses the core language and the propositional logic model to build propositional logic problems that answer whether or not policies can conflict on a decision request.</td>
</tr>
<tr>
<td>Planned core components</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Model Refinement</td>
<td>Augments the propositional model of policies with user supplied assumptions that cannot be inferred from the Trust Framework, such as assumptions about the behaviour of external services.</td>
</tr>
<tr>
<td>SAT Interpreter</td>
<td>Translates output from the SAT solver back into request/decision pairs in the language of the Trust Framework.</td>
</tr>
<tr>
<td>SMT Component</td>
<td>Takes solutions from the SAT solver and, with the help of the SAT interpreter, attempts to refute them based on identified Trust Framework constraints, augmenting the logic problem to provide more accurate solutions.</td>
</tr>
</tbody>
</table>

Table 4. Planned open source components

<table>
<thead>
<tr>
<th>Planned open source components</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query Engine</td>
<td>Takes user supplied queries in the form of partially specified request/decision pairs in the language of the trust framework. If the pairs are valid according to the policy, possible solutions for the missing fields are provided.</td>
</tr>
<tr>
<td>Evaluators</td>
<td>Takes a request and uses the open policy representation to evaluate it to a decision. A second evaluator will use the propositional model.</td>
</tr>
<tr>
<td>Data Sensitivity Analyser</td>
<td>Uses the core language and the propositional logic model to build propositional problems that verify that data privacy and consent requirements are met.</td>
</tr>
<tr>
<td>Auditing</td>
<td>Aggregates information about the security assessment of the policies and monitors real world request/decision pairs at runtime or as they are stored in logs.</td>
</tr>
</tbody>
</table>
### Third-party open source components

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAT Solver</td>
<td>The sat4j SAT solver library [17]</td>
</tr>
</tbody>
</table>

Information flow between these components is shown in Figure 1 below.

![Figure 1. Information Flow process](image-url)
5 SHiELD Use Cases

The Security Monitoring and Compliance Assurance tools will allow validation of information requests to be monitored by allowing policies for “assured” access to be defined, and by examining the patterns of access against these policies. In this context, the Security Monitoring methodology outlined in WP6 and shown below in Table 5 will be undertaken to assess the rightness of accesses against the policies and associated metadata used to signal security compliance.

Table 5. Privacy by Design vs. Use Cases

<table>
<thead>
<tr>
<th>Data protection and privacy tools</th>
<th>Mobile devices security</th>
<th>Data sensitivity analysis</th>
<th>Consent management</th>
<th>Data hiding</th>
<th>Security monitoring and compliance assurance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use case 1: «break glass circumstance»</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Use case 2: “surgical operation + follow up”</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Use case 3: “chronic condition”</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

In each use case, the consent model will be checked to ensure compliance with expressed patient consent. In addition, metadata from other work packages, where made available at run-time may be checked (according to policy) so that a complete analysis of relevant security aspects can be undertaken.

This metadata may include (according to policy):

**Use Case 1 (Break Glass circumstance)**
- No further metadata required

**Use Case 2 (Surgical Operation + follow up)**
- Data Sensitivity Analysis metadata from Task 5.2

**Use Case 3 (Chronic Condition)**
- Data Sensitivity Analysis metadata from Task 5.2

At this stage of the design, the metadata available from the above work-package is summarised as follows:
Task 5.2 Sensitivity Analysis Metadata

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataClassification</td>
<td>Describes the classification of a specific data identifier.</td>
</tr>
<tr>
<td>DataStoreClassification</td>
<td>Describes the classification of an entire data store. Consists from a map from data identifier to its classification.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensitivityDegree</td>
<td>Defines the data’s sensitivity degree. How much information the specific piece of data reveals.</td>
</tr>
<tr>
<td>confidence</td>
<td>Defines how much the tool is confident in the sensitivity degree it provides.</td>
</tr>
<tr>
<td>categories</td>
<td>A list of categories the data belongs to (such as personal info, health, email). Each category is attached with its own confidence value.</td>
</tr>
</tbody>
</table>

Figure 2. Supporting components from the SHIELD DoD (Description of Work)

As part of the next phase of the project, we will ensure that new tools are aligned with and are additive to the existing OpenNCP constructs (Figure 2).

6 Static Analysis of policies

Static analysis happens with minimal user intervention when policies are designed and before they are deployed. Their purpose is to enforce privacy by design, providing assurance that permitance and denial of access by the decision engine meets privacy requirements.

6.1 Analyses

6.1.1 Conflict Analyser

Existing literature [18], [19] has generally assumed that the individual policies which contribute to a higher level decision should not overlap in such a way that they give conflicting access decisions (one policy saying PERMIT while the other says DENY). The analyser carries this out by
first finding all sets of policies that are combined under a single combining algorithm. Then, for each set of policies, we propositionally encode the assertion that there are two members of the set, both of which are reached during policy evaluation, and one of which evaluates to PERMIT while the other evaluates to DENY.

6.1.2 No policy is redundant

A standard form of static analysis in programming code involves determining unused variables and dead-code. The analyser implements a very general scheme where every subpolicy is checked to ensure that there is at least one decision request where it is evaluated and makes a definite contribution to the policy decision.

To allow us to model redundancy checking, every policy is translated into the core intermediate language by introducing a set of input flags that allow arbitrary policies to be turned on or off at will. Given a policy $P$, we introduce a flag $P_f$ and restructure the policy tree so that $P$ is replaced by

\[
\text{IF } P_f \text{ THEN } P \text{ ELSE NOT\_APPLICABLE}
\]

We can now check whether the policy $P$ is redundant by determining whether we can satisfy

\[
P(f \rightarrow \text{true}) = P(f \rightarrow \text{false})
\]

That is, whether we can distinguish the policy tree from the version of itself where we have switched off the particular subpolicy controlled by $f$.

In the context of consent management, we might consider the first use-case of WP6. Here, the access control policy is expected to allow a “break-glass” scenario, where suitably authorised requests can retrieve data in an emergency situation. This scenario will be captured by one or more policies. If analysis flags these policies up as redundant, the policy administrator will know that there is something very wrong with the policy design: either break-glass is impossible, or it is unnecessary since the other policies are already allowing free access to sensitive data in non-emergency situations!

6.1.3 Data access respects privacy levels

Metadata provided from WP5.2 assigns two values to data entries: a privacy level and a confidence level in the judgement.

Now suppose we have a data entry $D_1$ with privacy level $P_1$ and confidence $C_1$, and another data entry $D_2$ with privacy level $P_2$ and confidence $C_2$.

1. If $P_1 \leq P_2$, $C_1 \geq C_2$ and we permit access to $D_2$, then we cannot deny access to $D_1$.
2. If $P_1 \leq P_2$, $C_2 \geq C_1$ and we deny access to $D_1$, then we cannot permit access to $D_2$.

The intention of rule 1 is that if we permit access to some private data, then we should not deny access to data that is known to be less private. Similarly, the intention of rule 2 is that if we deny access to some private data, then we should not permit access to data that is known to be more private.

Users who receive a warning that either of these conditions have been violated will have strong evidence that there is something wrong with their policy design. Conversely, eliminating such warnings will increase their confidence that they have maintained privacy by design.

6.2 Analysis refinement

As mentioned in Section 2, analysis is expected to find false negatives in real world policies, cases where it believes there is a security constraint violation when there is none. These will occur due
to analysis missing certain assumptions about various services that are not part of the trust framework. For instance, analysis might complain that a conflict arises between policies based on a particular external service query resolving in a way that is known by the policy administrator to be impossible in practice. This must be dealt with so that administrators can be confident that their policies do indeed meet their security requirements. The eventual goal should be for the administrator to have policies for which analysis reports no issues. This is not expected to be possible without intervention.

False negatives are reported to users by providing an access request and a list of service resolutions that lead to the violation. The policy administrator can either dismiss false negatives individually, or else generalise them to logical rules that must hold between the access request and service resolutions. In either case, assumptions will be added to the SAT model to disregard the false negative. This act on the part of the policy administrator is potentially error prone, and must be subject to auditing. There must be a record of exactly what assumptions the administrator provided, with the possibility for others to sign off on those assumptions. Additional auditing will be described in Section 6.1.

6.3 Model Querying

The SAT model affords us the ability to test arbitrary assertions of policies, and for the use cases given in WP6, we wish to provide policy administrators with a simple mechanism to verify that their policies correctly implement their intended consent requirements without having to rely on batteries of unit tests Figure 3.

To that end, the policy administrator will be able to enter a partial request/decision pair, stipulating only parts of a request target and specifying partial constraints on the values that may be resolved from external services during policy evaluation. The query is translated into a Boolean propositional formula and added to the policy model. The SAT analyser then attempts to satisfy the resulting model, and finally, if a satisfying instance is found, the interpreter translates it into a complete request and set of service resolutions. In addition, satisfied results are logged for later auditing.

This query system must be sufficient for users to explore possible ways that consent properties might be violated, and confirm that the only way they can be violated is in intended scenarios such as “break glass.” Additionally, query results may be general enough to be fed as assumptions to model refinement.
7 Auditing

7.1 Realtime Refinement Audits

In Section 6.2, we described facilities that will allow the policy administrator to specify arbitrary assumptions about their policy model which allow static analysis checks to pass. This is error prone and must be subject to testing.

The auditing component (Figure 4) will check all user-supplied model refinements by running the model evaluator against the existing log of request/decision pairs to ensure that the assumptions are satisfied in practice. In addition, the evaluator will be used throughout the operation of the decision engine with live requests to ensure that the assumptions are met going forward.
7.2 Auditing of External Services

The security properties we are interested in for SHiELD concern privacy violations. Additionally, we must consider circumstances such as the ones described in scenario 3 from WP6, which emphasise that multiple health care institutions will sometimes be contacted in order to grant access to medical data. This raises an issue of how security constraints can be guaranteed by isolated NCPs: what assumptions can be made about the interactions between the distributed systems?

In this case (Figure 5), a policy administrator can model an external service using the Symphonic policy language. Symphonic will expose an Audit Service over a secure REST endpoint which allows an external service auditor to send its log of request/decision pairs to be run against an embedded decision engine. If the embedded decision agrees with the logs, the Audit Service notifies the auditor that the logs are consistent with the audit services model of the external service.

![Diagram of external service audit](image)

7.3 Self Audit

For maximum assurance in the formal modelling provided by Symphonic, past logs produced by the decision engine can be rerun as policies are updated to ensure that changes are conservative over the old policies. The span of logs will be rerun through the policy to ensure that the request/decision pairs are repeated exactly. Three versions of tests will be performed:

- the policy as understood by the decision engine, tested via an embedded decision point
- the policy as formalised in the core intermediate language, tested via a core evaluator
- the policy as modelled for the SAT solver, tested via a model evaluator.

By testing against all three interpretations of the policy, we additionally prove that the formal models used by policy analysis are consistent with the supplied history of request/decisions pairs.
8 Integration

We assume that communication between the Symphonic components and the components from work packages WP5.2 will consist in GET requests over suitably secured REST endpoints. The Local Analysis Result Repository will be exposed over a REST service for the other NCPs, under an access control policy implemented internally by the Symphonic policy framework, according to bespoke privacy needs of the policy administrators from different NCPs (as opposed to privacy needs of patients).
9 References


