This project is funded
by the European Union

Extract of report delivered for EU H2020 Project Nº 883188

Digital Security and privacy for citizens and Small and Medium Enterprises and Micro Enterprises

Spyderisk Risk Assessment Software and its Knowledgebase

IT Innovation Centre, University of Southampton

authors:

Stephen C. Phillips
Mike Surridge
Steve Taylor
Ken Meacham
Panos Melas

reviewed by Maya Anderson, IBM Research

Background and Explanation

This 70-page document is an unedited extract of deliverable D4.2 prepared for the EU CyberKit4SME Project. All of the work done in this document was completed by members of the IT Innovation Centre at the University of Southampton prior to November 2023 and reviewed by IBM Research, also a CyberKit4SME project partner. Being an extract, some references in this document are not provided and other context is missing.

The motivation for publishing this extract is that is describes key details of the Spyderisk Risk Assessment Software, published as open source during the course of the CyberKit4SME project.

Table of Contents

Table of Figures

List of Tables

Table of Acronyms and Definitions

Acronym	Definition
API	Application programming interface
AuthN	Authentication
AWS	Amazon web services
CAPEC	Common attack pattern enumeration and classification
CEP	Complex event processing
CISO	Chief information security officer
CJA	Customer journey analysis
CJML	Customer journey modelling language
CMDB	Configuration management database
COTS	Commercial-off-the-shelf
CPE	Common platform enumeration
CSV	Comma-separated variable
CVE	Common vulnerability and exposures
CVSS	Common vulnerability scoring system
CWE	Common weakness enumeration
DNS	Domain name service
DoS	Denial of service
EHR	Electronic health record
EPS	Events per second
ESP	Extensible authentication protocol
EVTX	Microsoft Windows event log format
GDPR	General data protection regulation
HDFS	Hadoop distributed file system
HORM	Human and organisational risk modelling
HTTP	Hypertext transport protocol
HTTPS	Hypertext transfer protocol secure
I/O	Input/output
IaaS	Infrastructure as a Service
IaC	Infrastructure as code
IoC	Indicator of Compromise
IoT	Internet of things
IP	Internet protocol
IS	Information system
ISO	International Standards Organisation
IT	Information technology
ITIL	Information technology infrastructure library
JSON	JavaScript object notation
K8s	Kubernetes
KPI	Key performance indicator
LAN	Local area network
LDAP	Lightweight directory access protocol
ME	Medium enterprise

Acronym	Definition
MISP	Malware information sharing platform
MS	Microsoft
NVD	Network vulnerability database
OTP	One-time password
OWASP	Online web application security project
PaaS	Platform as a service
PME	Parquet Modular Encryption
PSK	Pre-shared key
REST	REpresentational State Transfer
RTP	Risk Treatment Plan
SDS	Secure data service
SIEM	Security information event management
SIM	Subscriber information module
SL	Service Ledger
SME	Small to medium enterprise
SMS	Simple message service
SNMP	Simple network management protocol
SoA	Statement of Applicability
SOC	Security operation centre
SQL	Standard query language
SSH	Secure shell
SSM	Spyderisk System Modeller
SSP	Secure simple pairing
STIX	Structured threat Information eXpression
TCP	Transmission control protocol
TLS	Transport layer security
UDP	User datagram protocol
URL	Uniform resource locator
USB	Universal serial bus
VM	Virtual machine
VXLAN	Virtual extensible LAN
WP	Work-package
XML	eXtensible markup language

II. RISK ANALYSIS MODELS AND TOOLS

II.1. Introduction

Since the previous version of this deliverable (D4.1[1]), there has been a significant change in the exploitation plan for the System Security Modeller, now known as "Spyderisk". Rather than planning to commercialise the work in a spin-out company, the team at the University of Southampton have open sourced the software, knowledgebase and related tools[2] and are building a sustainable open community around the work for researchers and businesses, with the intention of fostering trust and security in complex systems by supporting both by-design and risk assessment during operations in ways that explain how harm can arise before it happens.

The exploitation plan will be described in D6.7, but the shift to open source is mentioned here to explain the name change and also to explain the need for some of the reported activities (for instance, see Section II.6).

II.2. Goals

The mission of CyberKit4SME is the democratisation of cyber-security: providing cyber-security tools powerful enough to protect SMEs' information systems but which are simple enough to use by non-experts. Spyderisk's long history has given it a wealth of capabilities but the research previous to this project was not focussed on simplicity, which is itself a big challenge.

The primary project objective from the Grant Agreement relating to this work is Objective #1:

Develop tools allowing SMEs and MEs to analyse their information networks and identify and assess security risks, without frequent recourse to expensive consultants.

> Result: a set of tools and templates allowing SMEs and MEs to create high-level models of their information assets and networks, identify threats and assess associated risk levels, and obtain guidance on the security measures that could be used to mitigate these risks. The tools will cover both technical and human/organisational risk factors and mitigation measures. They will be understandable by non-specialist staff working for SME and ME.

Spyderisk developments in CyberKit4SME have been focussed on meeting this goal. Extensive work on exposing and simplifying the computer attack graphs, as well as additions to the user interface to display additional information all make it easier for non-specialist staff to understand the Spyderisk analysis. The knowledgebase has been extended to include more on GDPR compliance, additional help relating to potential modelling errors, and a revamped set of threats around authentication and data encryption to support further scenarios including the SDS component from IBM in WP5. User documentation and knowledgebase documentation have been improved and two training courses published to support users.

In addition, we have worked to address Objective #3:

Develop tools allowing SMEs and MEs to monitor their security, forecast risks and anticipate the need for technical or organisational protection measures.

> Result: a set of tools allowing SMEs and MEs to use advanced security monitoring and risk analysis models (see Obj #1) to monitor and forecast cyber security risk levels,

[1] CyberKit4SME D4.1: Initial Risk Analysis Monitoring and Forecasting Tools

[2] Spyderisk on Github: https://github.com/Spyderisk

and hence initiate appropriate countermeasures. The focus will be on making sophisticated SIEM technology more accessible to SMEs and MEs (in terms of cost, time and skills needed), and using shared security intelligence (see Obj #4) to provide more accurate forecasts of possible incidents.

Various integration points between Spyderisk and the SIEM (Keenaï) have been developed to meet this goal, primarily around the state report management and recommendation system. In addition, a simple entity API has been developed to increase the speed of operation of the Spyderisk service and provide an integration point for 3rd-party tools.

The tool feeds into the validation work (Objective #5) and dissemination (Objective #6).

Finally, we have created training and dissemination material to address Objective #6:

Disseminate the results from CyberKit4SME and develop training to support their use, promoting awareness of and creating a demand from SMEs and MEs for sophisticated tools and approaches to cyber security.

As well as four joint papers with SINTEF (reported below), IT Innovation have authored two other papers.

1. An overview paper for Spyderisk, outlining how it works and validating it against a known cyber-attack. This paper has been submitted, but a pre-print is available:

   Phillips, Stephen; Taylor, Steve; Boniface, Michael; Surridge, Mike (2023). *Automated Knowledge-Based Cybersecurity Risk Assessment of Cyber-Physical Systems*. Submitted for publication in IEEE Access. TechRxiv preprint. https://doi.org/10.36227/techrxiv.24061590.v1

2. A study into technology adoption and acceptance:

   Pickering, B., Phillips, S.C., Surridge, M. (2022). *Tell Me What that Means to You*: Small-Story Narratives in Technology Adoption. In: Kurosu, M. (eds) Human-Computer Interaction. Theoretical Approaches and Design Methods. HCII 2022. Lecture Notes in Computer Science, vol 13302. Springer, Cham. https://doi.org/10.1007/978-3-031-05311-5_19

To assist with training, IT Innovation have created two training courses, both available free online:

1. Cybersecurity Risk Assessment & Modelling: Core Principles
2. Getting Hands on with Spyderisk

Both courses are hosted at https://training.spyderisk.org and the second course includes free use of a Spyderisk deployment so that people can experiment with the tool without needing to install it.

II.3. Spyderisk Overview

II.3.1. Software Components

The Spyderisk runtime software has three separable parts:

- The Spyderisk System Modeller service which is a Java Spring[3] application running in Tomcat.
- The Spyderisk System Modeller web user interface which is a React[4] application.

[3] Java Spring: https://spring.io/

[4] React: https://react.dev/

- The System Modeller Adaptor written in Python, supporting more experimental ideas, and providing an easy integration point for 3rd party tools.

The Spyderisk System Modeller service makes use of:

- KeyCloak[5] for authentication.
- MongoDB[6] to store some authorisation data.
- Apache Jena TDB[7] to store the knowledgebase and system models.

II.3.2. Spyderisk Concepts

The following sections necessarily must refer to various concepts used in Spyderisk. To aid understanding, we provide here an explanation of the terms and how they map onto terms in the ISO 27000 series which are the standards used for information security risk management.

Figure 1 illustrates the core ontology used in Spyderisk to model the generic risk assessment method, including cause and effect.

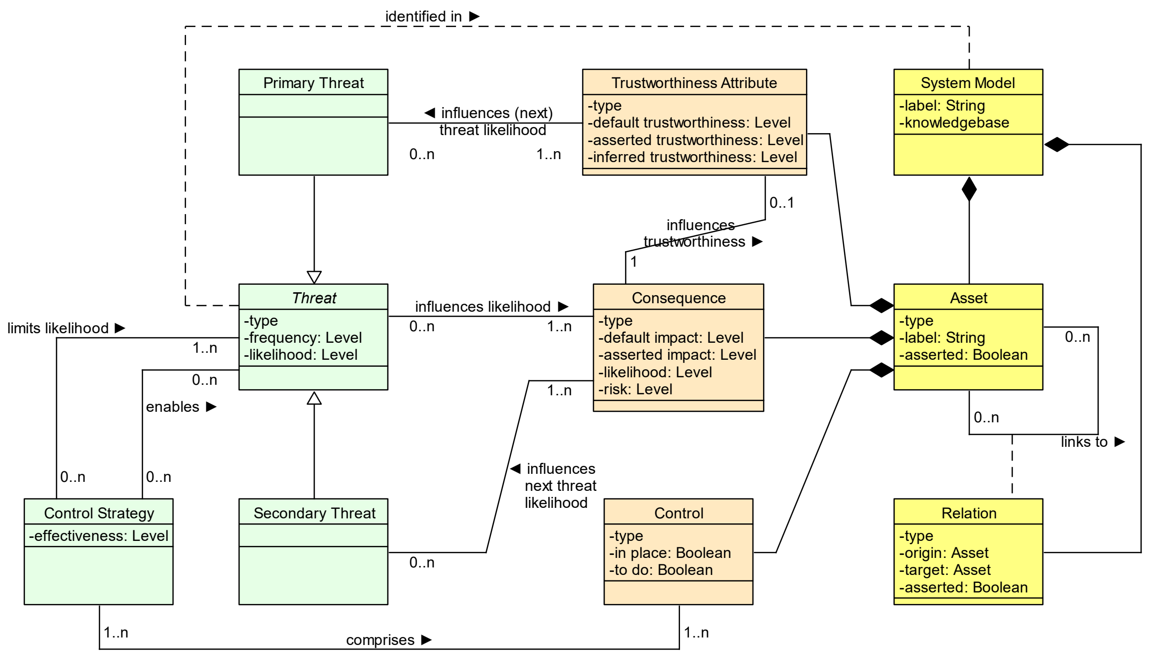


Figure 1. Core Spyderisk Ontology

In summary, the key Spyderisk concepts are as follows:

- The System under examination is a cyber-physical system of different types of **Assets** and their **Relationships**.
- **Threats** attack Assets by exploiting vulnerabilities (often represented in Spyderisk as lowered **Trustworthiness Attributes** of different types). A Threat has a **frequency** determined by intrinsic factors such as its inherent difficulty, and extrinsic factors such as the motivations of actors to attack via this Threat. Threats have a **Likelihood**

[5] Keycloak: https://www.keycloak.org/

[6] MongoDB: https://www.mongodb.com/

[7] Apache Jena TDB: https://jena.apache.org/documentation/tdb/

which is the result of an interactive calculation, taking into account initial conditions, other Threats and any **Control Strategies** which are in place.

- Assets have **Consequences**, which represent undesirable effects of Threat attacks on Assets. Examples of Consequences include Loss of Confidentiality, Integrity or Availability for data. A Consequence has a **Likelihood**, which is determined by the highest Likelihood of any causing Threat.
- A Consequence has an **Impact** level, set by the analyst, that represents the severity of the type of Consequence on the affected Asset.
- A specific Consequence on an Asset has an associated **Risk Level**. The Risk Level is determined by the Consequence's Impact (determined by judgement) combined with its Likelihood (determined by the likelihood of its causing Threats).
- Threat Likelihood is limited by **Controls** applied at Assets, indicated as Control-Asset pairs. Spyderisk has the additional notion of a collection of such Control-Asset pairs, known as a **Control Strategy.** The Threat Likelihood limitation effect is specified per Control Strategy, meaning that all Control-Asset pairs need to be present to achieve the Threat Likelihood limitation effect.

These concepts are mapped to ISO 27000[8] and ISO 27005[9] vocabulary as follows:

Asset *"An asset is anything that has value to the organization and which, therefore, requires protection. For the identification of assets, it should be borne in mind that an information system consists of more than hardware and software"* [ISO 27005].

Consequence *"Outcome of an event affecting objectives"* [ISO 27000]. The event is typically an Information Security Event or Incident (defined below) and the effect referred to is usually a negative effect against the objectives of the system's owners.

Control *"Measure that is modifying risk"* [ISO 27000]. The aim of Controls is typically limitation of risk likelihood or impact. For Spyderisk, Controls aim to limit likelihood of Consequences.

Impact: from Consequence Criteria *"Consequence criteria should be developed and specified in terms of the extent of damage or loss, or harm to an organization or individual resulting from the loss of confidentiality, integrity and availability of information. […] Consequence criteria define how an organization categorizes the significance of potential information security events to the organization."* [ISO 27005]. Other sources make analogies between Impact and Severity – i.e. the measure of loss should the Consequence occur.

Likelihood: "*Chance of something happening"*. [ISO 27000]

Risk *"Effect of uncertainty on objectives"* [ISO 27000]. Also **Risk Level** (**Level of Risk)** *"magnitude of a risk expressed in terms of the combination of consequences and their likelihood"* [ISO 27000]. For Spyderisk purposes, Risk Level is the combined Impact and Likelihood of a Consequence.

System: defined as **Information System** *"set of applications, services, information technology assets, or other information-handling components"* [ISO 27000].

Threat *"potential cause of an unwanted incident, which can result in harm to a system or organization"* [ISO 27000]. The actual manifestation of Threat in the System is **Information Security Event** *"identified occurrence of a system, service or network state indicating a*

[8] ISO/IEC 27000:2018. Information technology — Security techniques — Information security management systems — Overview and vocabulary https://www.iso.org/standard/73906.html

[9] ISO/IEC 27005:2022. Information security, cybersecurity and privacy protection — Guidance on managing information security risks https://www.iso.org/standard/80585.html

possible breach of information security, policy or failure of controls, or a previously unknown situation that can be security relevant" [ISO 27000]. There is also **Information Security Incident**, which represents the manifestation of a series of Threats: *"single or a series of unwanted or unexpected information security events that have a significant probability of compromising business operations and threatening information security"* [ISO 27000].

II.4. Software Implementation Details

II.4.1. Attack Graph

In the Spyderisk risk assessment methodology, a Threat may increase the likelihood of a Misbehaviour (also known as a "Consequence") which may go on to increase the likelihood of further Threats. In this way, a directed graph ("attack graph") of Threats and Misbehaviours is built up, sometimes branching, sometimes joining. The chains of Threats and Misbehaviours model both the steps an attacker may take through a system (where deliberate Primary Threats are involved) and secondary effect cascades (which model the inevitable knock-on effects using Secondary Threats).

Attack graphs (and "attack trees") are not a new way to represent potential threats to a system, and there are many existing styles and variations[10]. One key aspect of the latest iteration of the Spyderisk attack graphs it to explicitly include "AND" and "OR" nodes in the graph to maximise comprehension.

It is useful to consider an attack graph which begins with one or more "root causes" (a Threat) and ends with a Misbehaviour that has a high risk level (often combining a high business impact and a high likelihood). The attack graph demonstrates how the eventual undesirable Misbehaviour can be caused and is the starting point for when a risk analyst is trying to understand what controls to use to reduce the likelihood.

In the System Modeller web interface, the attack graph has for a long time been accessible through the Threat and Consequence explorer interfaces, but these interfaces are limited in that they only let the user navigate to the root cause or a previous or next step in a graph and we have found in the validation work that users can easily get lost and even go round in loops when navigating the graph in this manner.

In CyberKit4SME we have been developing better interfaces to help the user navigate and now have a visual representation and a summary list built into the web interface as well as a stand-alone analysis tool[11].

Creating a plot of the whole attack graph is not difficult, and such a plot was presented in the previous deliverable (D4.1[1]). The challenges were to extract the most useful summary from the underlying data and to present it in a clear manner. The general strategy adopted was to extract the highest likelihood, shortest paths through the full attack graph, as it is these routes that make the most sense to address first. The concept of a "shortest path" is not uniquely defined in these graphs and various attempts have been made to achieve a result that is useful for users.

When considering a "shortest path" it is necessary to define what the distance is which is being minimised. The most obvious approach is to count the number of steps through the graph from the root cause, but even that is not simple as there can be many possible routes to get to a node. Initially though the smallest distance from the root cause was used as a metric.

[10] Lallie, H.S., Debattista, K., Bal, J., 2020. A review of attack graph and attack tree visual syntax in cyber security. Computer Science Review 35, 100219. https://doi.org/10.1016/j.cosrev.2019.100219

[11] Plot-attack-graph software: https://github.com/Spyderisk/plot-attack-graph

After further examination of the graphs we switched to counting the smallest number of primary threat nodes which needed to be passed through to get to a node as a better metric. This metric essentially ignores all secondary threats in considering “distance”. In many cases in the attack graphs, there will be a primary threat followed by a chain of many secondary threats which (a) are not deliberate steps, they are automatic consequences, (b) rarely have any controls and (c) can be arbitrary modelling artifacts in terms of their quantity. If we do not ignore them therefore, one path with 1 primary threat and 10 secondary ones will be considered “longer” than a path with 10 primary threats in a row.

The following Figure shows a shortest path attack graph including just the primary threats (red squares) and the two target misbehaviours (orange circles) and with each node numbered. In the left figure the numbering is according to the minimum primary threat distance. Note that after the OR merge the number does not change, but after the AND merge the minimum primary threat distance of 10 is found by merging both the routes prior to the AND. In the right figure the numbering shows the minimum and maximum node distances from the root cause, demonstrating (a) in general how there are many longer routes which the plot is not showing (hence the difference between the minimum and maximum) and (b) how many secondary threat nodes there can be (for instance see before and after the OR node, jumping from 9 to 16).

Algorithms to find the shortest path attack graph have been implemented in the stand-alone tool and in the System Modeller service. The System Modeller service has an API endpoint to request the attack graph data in JSON format:

`GET /models/{modelID}/threatgraph`

With parameters:

- **`riskMode`**: CURRENT or FUTURE (the mode of the risk calculation)
- **`allPath`**: true or false (whether to display all paths, or just the shortest)
- **`normalOperations`**: true or false (whether to display “normal operation” threats)
- **`targetURIs`**: a list of the target Consequences, e.g., `system%23MS-LossOfConfidentiality-e0fe457b`

The response to this call provides an attack graph for each target consequence. Within each graph there is a list of nodes with their distance from the target consequence, sorted into node types: threats, consequences and trustworthiness attribute sets. All the links between nodes are also listed. The response format is:

```
{
  "graphs": {
    <target consequence short URI>: {
      "threats": {
        <threat short URI>: <distance from target consequence>,
        ...
      },
      "consequences": {
        <consequence short URI>: <distance from target consequence>,
        ...
      },
      "twas": {
        <trustworthiness attribute set short URI>: <distance from target consequence>,
        ...
      },
      "links": [
        [
          <source short URI>,
          <target short URI>
        ],
        ...
     ]
    },
    ...
  },
  "uriPrefix": "http://it-innovation.soton.ac.uk/ontologies/trustworthiness/"
}
```

This endpoint is used from the web interface to present a flattened list summary view of the entire attack path to any target Consequence (see Figure 3). The web interface also has a button to retrieve a graphical representation of the attack graph (see Figure 4).

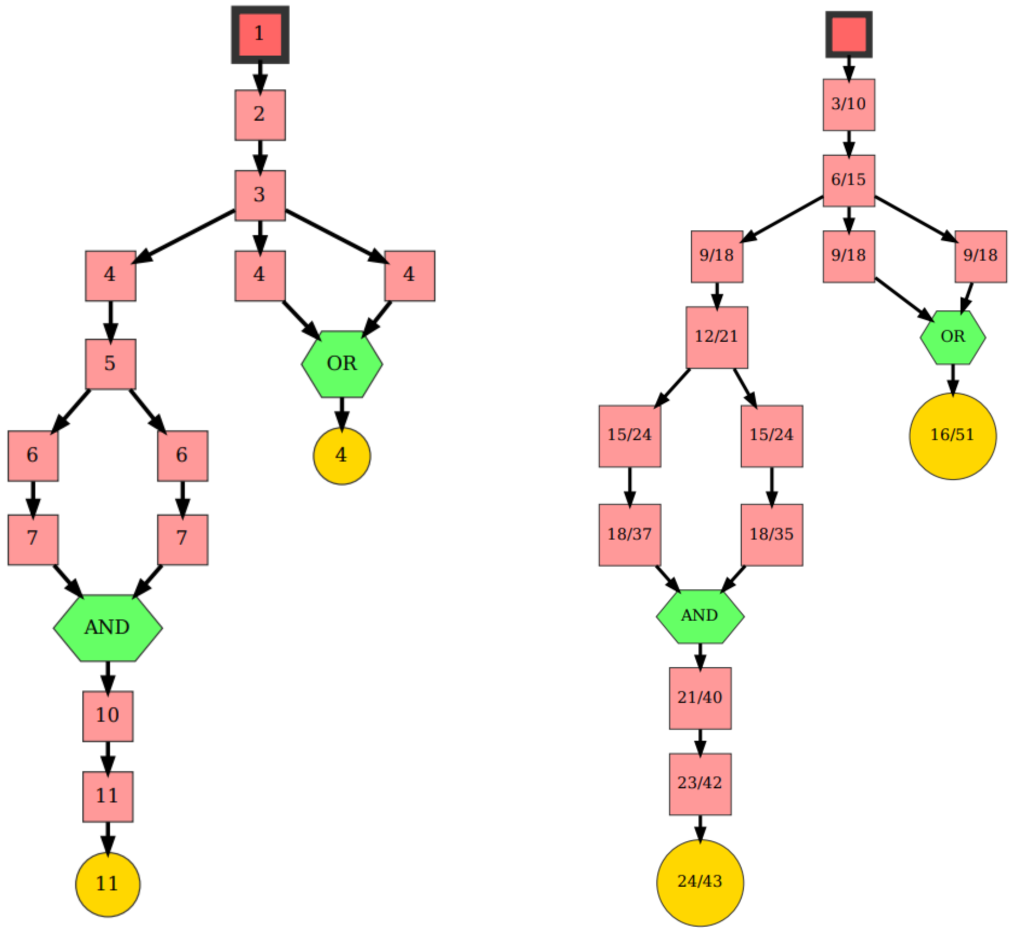
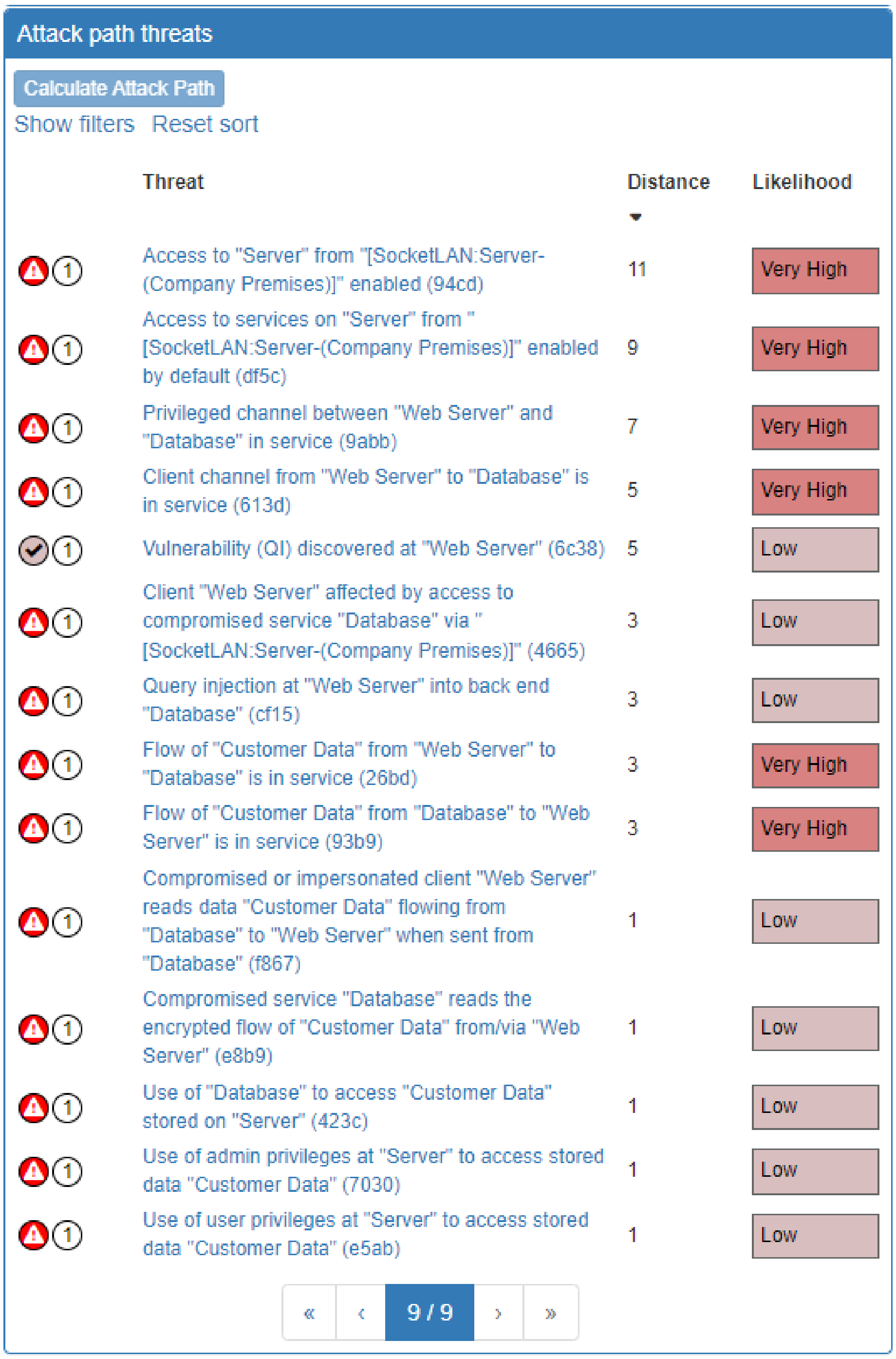


Figure 2. Minimum primary threat distances (left); Minimum/Maximum distances from the root cause, including secondary threats and misbehaviours (right).


Attack path threats

Calculate Attack Path

Show filters Reset sort

Threat	Distance	Likelihood
Access to "Server" from "[SocketLAN:Server-(Company Premises)]" enabled (94cd)	11	Very High
Access to services on "Server" from "[SocketLAN:Server-(Company Premises)]" enabled by default (df5c)	9	Very High
Privileged channel between "Web Server" and "Database" in service (9abb)	7	Very High
Client channel from "Web Server" to "Database" is in service (613d)	5	Very High
Vulnerability (QI) discovered at "Web Server" (6c38)	5	Low
Client "Web Server" affected by access to compromised service "Database" via "[SocketLAN:Server-(Company Premises)]" (4665)	3	Low
Query injection at "Web Server" into back end "Database" (cf15)	3	Low
Flow of "Customer Data" from "Web Server" to "Database" is in service (26bd)	3	Very High
Flow of "Customer Data" from "Database" to "Web Server" is in service (93b9)	3	Very High
Compromised or impersonated client "Web Server" reads data "Customer Data" flowing from "Database" to "Web Server" when sent from "Database" (f867)	1	Low
Compromised service "Database" reads the encrypted flow of "Customer Data" from/via "Web Server" (e8b9)	1	Low
Use of "Database" to access "Customer Data" stored on "Server" (423c)	1	Low
Use of admin privileges at "Server" to access stored data "Customer Data" (7030)	1	Low
Use of user privileges at "Server" to access stored data "Customer Data" (e5ab)	1	Low

« ‹ 9 / 9 › »

Figure 3. Example flattened attack path summary.

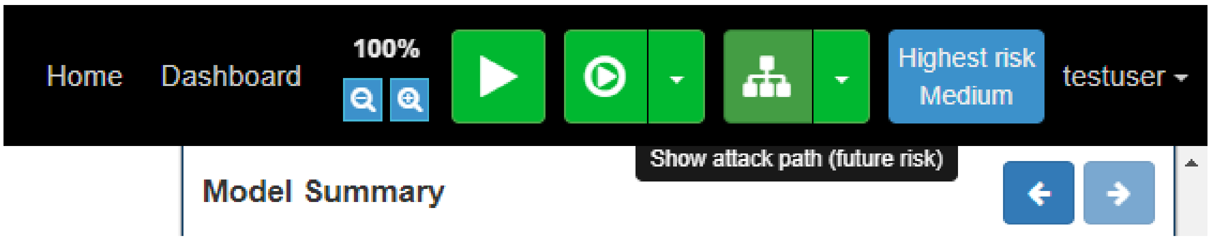


Figure 4. Spyderisk toolbar showing the new attack path button (with tooltip).

As well as providing useful information to the user of the Spyderisk interface, the short attack paths were also generated with the integration with HORM in mind (see Section **Error!**

Reference source not found.) as they represent summaries of steps through the system that could usefully be described in a simple graphical form.

II.4.2. Recommendations

Work on providing recommendations was initially done in ProTego (https://protego-project.eu/) and then continued jointly in FogProtect (https://fogprotect.eu/) and CyberKit4SME, with further development and re-engineering being performed just in CyberKit4SME.

Spyderisk provides a risk assessment of the Threats and Consequences in a system model and the calculated risk level can often be reduced by adding Control Strategies to the model. Often a Control Strategy is just a single Control on an Asset, but sometimes multiple Controls are needed to reduce the likelihood of a Threat. For instance, to reduce the likelihood of the Threat of a host being impersonated, you could use the “X509” Control Strategy which comprises both the “X509 Certificate” control at the host and the “X509 Verifier” at the client end of a client-service communication.

The Spyderisk web interface allows a risk analyst to try adding different Control Strategies and see how they affect the risk assessment. The web interface also proposes Control Strategies that may be used to address each Threat examined. The web interface does not provide recommendations (yet) and providing recommendations is considered a requirement when the Spyderisk web interface is not used and Spyderisk is instead integrated with external tools such as SIEMs (e.g., Keenaï) where there will be an operator using the system and requiring decision support.

The approach we have taken to providing recommendations is to build on the attack graph described above by computing expressions describing how to reduce the likelihood of each Threat in the shortest path graph. For any Threat node we can either apply a Control Strategy which directly addresses the Threat (there may be zero to many of these), or reduce the likelihood of all the immediate causes of the Threat (the nodes immediately preceding it in the graph - either a single node or connected by an OR or and AND).

This recursive approach is illustrated in Figure 5. Each red Threat node show the single Control Strategy which addresses it directly (though there may be zero or many in the general case): for instance “T1: CSG1”. Beneath the first line is a logical expression showing how the node’s own Control Strategy can be combined with the expression for the node(s) feeding into the Threat. Threat “T2” therefore can be addressed by either (“OR”) using CSG1 to reduce the likelihood of T1 and hence of T2, or directly using CSG2.

The expression for Threat T4 is complicated because T4 can be caused by either T2 or T3 and so *both* must be prevented (hence the inner “AND”). The expression simplifies down to just “OR(CSG3, CSG2)”, revealing that it doesn’t matter whether CSG1 is applied or not.

The expression for the final target Consequence (in yellow) comes from combining the expressions for T4 and T5 with an “OR”. This is because the Consequence requires both T4 and T5 to be likely and so addressing either is sufficient.

By using such an analysis of the computed attack graph, we arrive at a logical expression providing the options we have to reduce the likelihood of the target Consequence. In this case, the expression naturally arose as an “OR” of three options, but it may not, so within the software we rearrange the expression into disjunctive normal form (DNF), or “an OR of ANDs”.

Creating the DNF form which addresses the target Consequence is just the first step. The software will try each option in the DNF expression in turn, recomputing the risk levels. It may be that the resulting risk level is not low enough and so another iteration, providing a new DNF expression with potentially more than one further option must be taken. The software recurses through the options building up a tree of Control Strategies applied to the system model to explore the possibilities and finally report on the options which reduce the risk level to a target value.

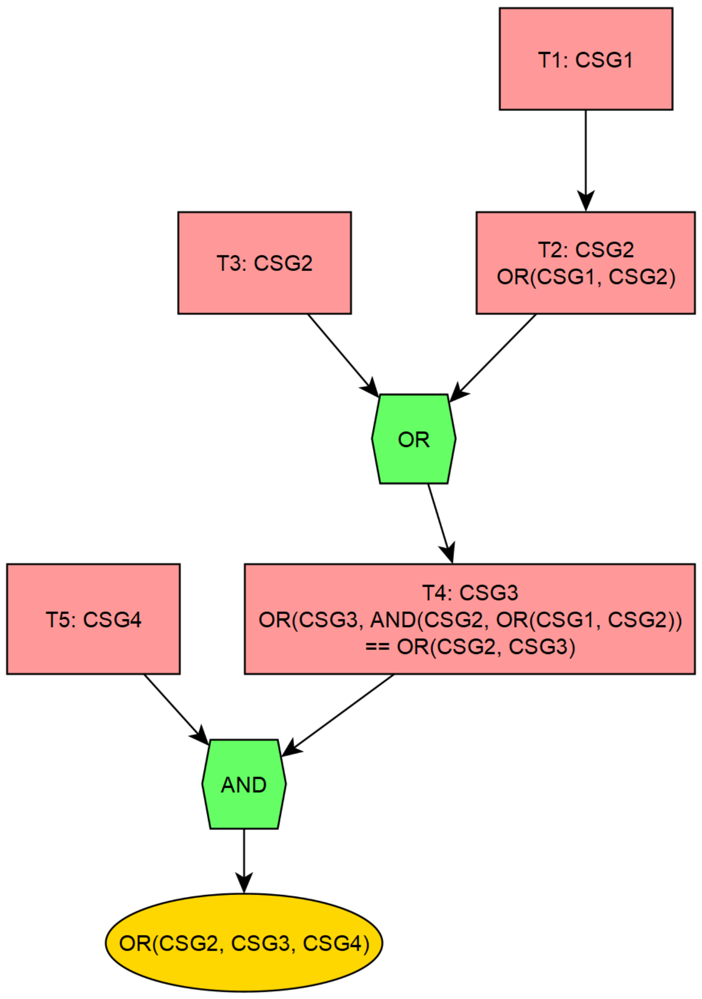


Figure 5. An illustration of the logical expressions computed to address each Threat node of an attack graph.

The recommendation algorithm was first implemented in the (Python) System Model Adaptor and subsequently reimplemented in the (Java) Spyderisk System Modeller service as a core function. See Section II.4.5.c for details of the API.

II.4.3. Simple Entity API

The Spyderisk Java “system modeller” service provides various RESTful API endpoints, mostly written to directly support the Spyderisk web user interface. As a result, although most information about the system model and underlying knowledgebase can be obtained, many API endpoints are quite specific to the requirements of the web interface and can be hard or inefficient to use by other client software, as they may return copious amount of data regarding linked entities in a response when only a small specific piece of data is required.

The need to improve the speed of some operations in the web interface, combined with the requirements arising out of integration work with SINTEF have led us to implementing a new set of API endpoints to query entities in system models and their linked knowledgebases.

The new API, implemented in the Java service, is designed to respond to queries for all instances of an entity type or a single instance. Response documents include URIs pointing to further entities which can be queried separately if the client requires further information (in contrast to the previous API which tended to embed such linked entities). The new API is in use in parts of the System Modeller web interface and the System Modeller Adaptor. We plan to also implement methods to create and update attributes of appropriate entities.

The new API endpoints all incorporate the system model ID in the URL. The endpoints come in pairs: one to get all entities of a type and one to retrieve a single entity. Where a single entity is requested, it is defined by appending the entity's URI to the endpoint URL (following the standard design of RESTful APIs). In these cases, and in the response documents, we are using a shortened version of the full URI using just the URI fragment, so that instead of encoding e.g., "`http://it-innovation.soton.ac.uk/ontologies/trustworthiness/system#MS-LossOfAuthenticity-7e6aa2f`" we just require "`system#MS-LossOfAuthenticity-7e6aa2f`". The omitted part of the URI can be understood if necessary from the context of the model so no information is lost.

The new endpoints are described below.

II.4.3.a. System Model Endpoints

`GET /models/{modelId}/entity/system/trustworthinessAttributeSets`

`GET /models/{modelId}/entity/system/trustworthinessAttributeSets/{uri}`

A JSON representation of a trustworthiness attribute set object.

Example Response:

```
{
  "uri": "string",
  "type": "string",
  "id": "string",
  "trustworthinessAttribute": "domain-model-TWA-uri",
  "locatedAt": "asset-uri",
  "assertedLevel": "string",
  "inferredLevel": "string",
  "externalCause": true,
  "causedThreats": [
    "threat-uri"
  ],
  "minOf": "string",
  "maxOf": "string",
  "hasMin": "string",
  "hasMax": "string"
}
```

`GET /models/{modelId}/entity/system/threats`

`GET /models/{modelId}/entity/system/threats/{uri}`

Retrieves a JSON document describing a specific system model threat.

Example Response:

```
{
  "uri": "string",
  "type": "string",
  "id": "string",
  "label": "string",
  "description": "string",
  "parent": "domain-model-threat-uri",
  "threatens": "asset-uri",
  "appliesTo": "string",
  "frequency": "string",
  "entryPoints": [
    "twas-uri"
  ],
  "secondaryEffectConditions": [
    "misbehaviourSet-uri"
  ],
  "misbehaviours": [
    "misbehaviourSet-uri "
  ],
  "blockedByCSG": [
    "controlStrategy-uri"
  ],
  "mitigatedByCSG": [
    "controlStrategy-uri"
  ],
  "triggeredByCSG": [
    "controlStrategy-uri"
  ],
  "triggered": true,
  "minOf": "string",
  "maxOf": "string",
  "hasMin": "string",
  "hasMax": "string",
  "currentRisk": true,
  "futureRisk": true,
  "secondaryThreat": true,
  "normalOperation": true,
  "risk": "string",
  "directMisbehaviours": [
    "misbehaviourSet-uri"
  ],
  "indirectMisbehaviours": [
    "misbehaviourSet-uri"
  ],
  "indirectThreats": [
    "threat-uri"
  ],
  "causedBy": [
    "threat-uri"
  ],
  "initialCause": true,
  "rootCause": true,
  "likelihood": "string"
}
```

`GET /models/{modelId}/entity/system/misbehaviourSets`

`GET /models/{modelId}/entity/system/misbehaviourSets/{uri}`

This REST method retrieves a misbehavour set (MS).

Example response:

```
{
  "uri": "string",
  "type": "string",
  "id": "string",
  "misbehaviour": "domain-model-misbehaviour-uri",
  "locatedAt": "asset-uri",
  "impactLevel": "string",
  "defaultLevel": "string",
  "risk": "string",
  "externalCause": true,
  "normalOpEffect": true,
  "causedThreats": [
    "threat-uri"
  ],
  "causedBy": [
    "threat-uri"
  ],
  "minOf": "string",
  "maxOf": "string",
  "hasMin": "string",
  "hasMax": "string",
  "likelihood": "string"
}
```

`GET /models/{modelId}/entity/system/controlStrategies`

`GET /models/{modelId}/entity/system/controlStrategies/{uri}`

Retrieves control strategies (CSG) for a specific CSG URI.

Example response:

```
{
  "uri": "string",
  "type": "string",
  "id": "string",
  "label": "string",
  "description": "string",
  "parent": "domain-model-controlStrategy-uri",
  "blockingEffect": "string",
  "coverageLevel": "string",
  "controlSets": [
    "controlSet-uri"
  ],
  "mandatoryCS": [
    "controlSet-uri"
  ],
  "optionalCS": [
    "controlSet-uri"
  ],
  "blocks": [
    "threat-uri"
  ],
  "mitigates": [
    "threat-uri"
  ],
  "triggers": [
    "threat-uri"
  ],
  "currentRisk": true,
  "futureRisk": true,
  "enabled": true,
  "minOf": "string",
  "maxOf": "string",
  "hasMin": "string",
  "hasMax": "string"
}
```

`GET /models/{modelId}/entity/system/controlSets`

`GET /models/{modelId}/entity/system/controlSets/{uri}`

Retrieves control set (CS) for a specific CS URI.

Example response:

```
{
  "uri": "string",
  "type": "string",
  "id": "string",
  "locatedAt": "asset-uri",
  "control": "domain-model-control-uri",
  "proposed": true,
  "coverageLevel": "string",
  "minOf": "string",
  "maxOf": "string",
  "hasMin": "string",
  "hasMax": "string"
}
```

`GET /models/{modelId}/entity/system/assets`

`GET /models/{modelId}/entity/system/assets/{uri}`

Retrieves an asset for a specific Asset URI.

Example response:

```
{
  "uri": "string",
  "type": "string",
  "id": "string",
  "label": "string",
  "description": "string",
  "asserted": true,
  "minCardinality": 0,
  "maxCardinality": 0,
  "population": "string"
}
```

II.4.3.b. Domain Model Endpoints

A system model is linked to a single domain model (knowledgebase) which is shared between multiple system models. Through the following endpoints, aspects of the linked domain model are queried.

`GET /models/{modelId}/entity/domain/trustworthinessAttributes`

`GET /models/{modelId}/entity/domain/trustworthinessAttributes/{uri}`

Retrieves a trustworthiness attribute (TWA) for a specific URI, from the domain model.

Example response:

```
{
  "uri": "string",
  "type": "string",
  "id": "string",
  "label": "string",
  "description": "string",
  "metaLocatedAt": [
    "domain-model-asset-uri"
  ],
  "minOf": "string",
  "maxOf": "string",
  "hasMin": "string",
  "hasMax": "string",
  "visible": true
}
```

`GET /models/{modelId}/entity/domain/misbehaviours`

`GET /models/{modelId}/entity/domain/misbehaviours/{uri}`

This REST method retrieves domain misbehaviour for URI.

Example response:

```
{
  "uri": "string",
  "type": "string",
  "id": "string",
  "label": "string",
  "description": "string",
  "metaLocatedAt": [
    "domain-model-asset-uri"
  ],
  "minOf": "string",
  "maxOf": "string",
  "hasMin": "string",
  "hasMax": "string",
  "visible": true
}
```

`GET /models/{modelId}/entity/domain/levels/{metric}`

`GET /models/{modelId}/entity/domain/levels/{metric}/{uri}`

Retrieves a domain level metric for a specific URI.

Example response:

```
{
  "uri": "string",
  "type": "string",
  "id": "string",
  "label": "string",
  "description": "string",
  "levelValue": 0
}
```

`GET /models/{modelId}/entity/domain/controls`

`GET /models/{modelId}/entity/domain/controls/{uri}`

Retrieves domain model control for a specific control URI.

Example response:

```
{
  "uri": "string",
  "type": "string",
  "id": "string",
  "label": "string",
  "description": "string",
  "metaLocatedAt": [
    "domain-model-asset-urri"
  ],
  "minOf": "string",
  "maxOf": "string",
  "hasMin": "string",
  "hasMax": "string",
  "visible": true
}
```

II.4.4. State Report Management

In CyberKit4SME, Spyderisk needs to integrate with OpenVAS[12] (the open vulnerability assessment scanner) and the Keenaï SIEM to provide up-to-date risk assessments based on the known state of the operational (or “runtime”) system. We also wanted to explore the semi-automatic building of the initial system model (see Section II.4.7). An initial custom integration

[12] OpenVAS: https://openvas.org/

with OpenVAS (via Keenaï) was initially performed based on some work done in two previous projects: ProTego and Fog Protect.

In ProTego, Spyderisk was integrated with OpenVAS, with another partner's component summarising the OpenVAS scan result data and passing the summary to Spyderisk. We developed a methodology for mapping between resources detected by OpenVAS (such as hosts and processes) and those found in the system model by adding additional data to the assets in the system model such as IP addresses or CPE strings. We also developed a method to map from CVEs reported by OpenVAS to appropriate changes in the trustworthiness attributes of the assets in the system model. Both these approaches have been reused in CyberKit4SME, but in CyberKit4SME all the processing of the OpenVAS report (including querying the CVE to obtain CVS v2 metrics) is done by the Spyderisk software.

OpenVAS scan information:

a) reduces the trustworthiness level of asset trustworthiness attributes corresponding to the CVSS v2 metrics AV and AU, which classify vulnerabilities based on the means of access and the need for authentication.
b) reduces the trustworthiness level of asset trustworthiness attributes corresponding to the CVSS v2 metrics A, I and C which classify vulnerabilities based on the consequences of their exploitation for the vulnerable asset availability, integrity or for data handled by the asset, plus (in specific combinations) user or root access to the asset.
c) reduces the trustworthiness level of other asset trustworthiness attributes if exploitation would have consequences for other assets with specific relationships to the vulnerable asset, including query injection to a back-end database, or cross-site scripting attacks on clients of a vulnerable service.

Some similar work was done in ProTego to interpret the scan results from the OWASP Zap tool[13] (a web application scanner) and in Fog Protect to interpret data from Wazuh[14] (an open source SIEM).

In CyberKit4SME we looked at what information could be gained from Nmap[15] (network mapping) scans and from sources of data such as the Azure API[16] (for cloud-based deployments). As with the information regarding active vulnerabilities obtained from sensors such as OpenVAS, we considered the question of how the data could be used to update the system model.

Rather than continue with custom project integrations, all the requirements from the previous work were considered and a unified method for capturing and applying data relating to the state of the system model was designed and implemented. The integration with Keenaï was then updated to match the slight change in the external API.

II.4.4.a. System Model Variants

The Figure below shows the relationship between four different variants of the system model, considering initial model construction, and both design-time and runtime risk assessment and recommendation:

[13] OWASP ZAP: https://www.zaproxy.org/

[14] Wazuh: https://wazuh.com/

[15] Nmap: https://nmap.org/

[16] Azure API: https://learn.microsoft.com/en-us/rest/api/azure/

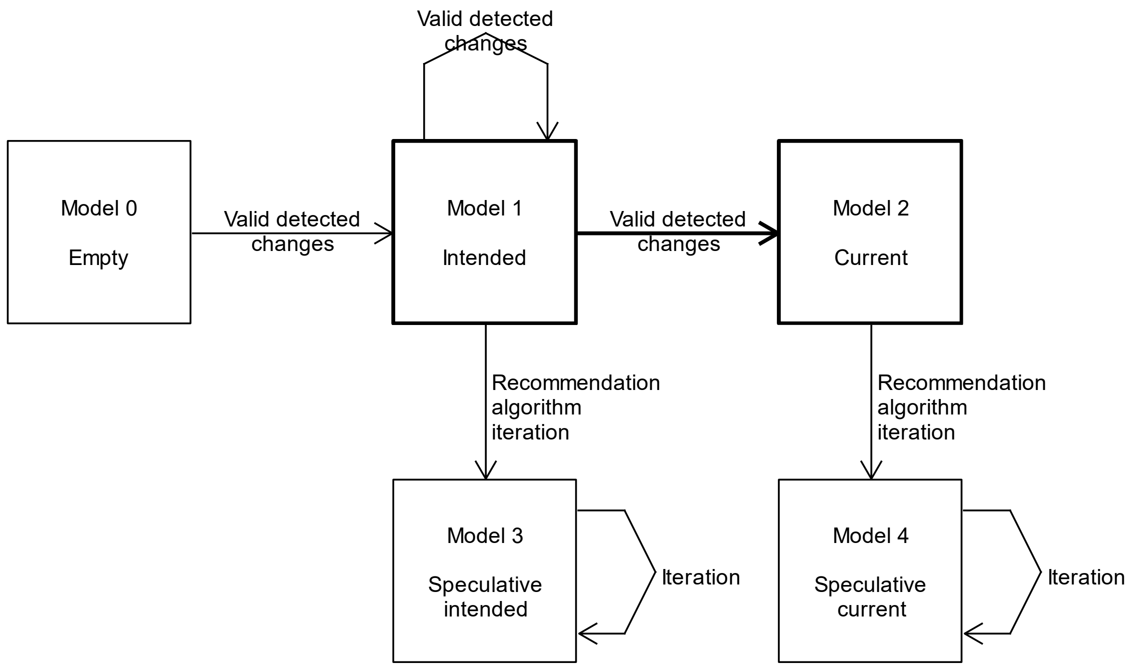


Figure 6. A representation of five different types of system model and the transitions between them.

Table 1. Explanation of the five different system model types.

Model	Type	Risk calculation type	Represents	Purpose
0	Blank canvas	N/A	The starting point	
1	Intended	Future	The intended system	To understand and manage long-term risks
2	Current	Current	The system as it is understood to be now	To understand and manage immediate risks
3	Speculative intended	Future	A trial change in the system	To propose changes that would reduce the future risk level
4	Speculative current	Current	A trial change in the system	To propose changes that would reduce the current risk level

Model 0 to 1 (empty to intended)

This transition represents building as much of the system model as possible in an automated manner, taking in information from scanners such as nmap. This is reported in Section II.4.7.

Model 1 to 3 & Model 2 to 4 (recommendations)

The recommendation algorithm can take a design-time or current system model and try various changes with the aim of reducing the risk level in the system. Various attempts might be made by the recommendation algorithm, either building on previous speculative changes or starting from Model 1 or Model 2. The recommendation algorithm is discussed in Section II.4.2.

Model 1 to Model 1 (intended evolution over time)

Over time the model of the intended system will need to change to match any deliberate evolution of the actual system. Any change in Model 1 would need to be reflected in Models 2, 3 & 4.

As Model 1 is the "intended" model, any changes to it must also be intended. It is therefore not possible to automatically update Model 1 with a new detected host or process for instance if that information has come from some sort of scanner as it would not be possible to know if the change was indeed intentional.

If Model 1 was linked to a configuration management system (CMDB) where deliberate changes to the system were logged, then an automated update to the intended model could be envisaged.

We will assume that Model 1 to Model 1 changes are performed deliberately by a human when necessary to reflect deliberate changes in the system.

Model 1 to Model 2 (intended to current)

This document is primarily concerned with understanding how to create Model 2 from Model 1. That is, how to maintain, or create on demand, a model of the system as it is understood to be now, based on the model of the intended system (Model 1) and a set of one or more changes to the system reported by external sensors (such as OpenVAS or a SIEM).

II.4.4.b. Classes of External Sensor Reports

Different sources of information about the system report on its state in different ways. Not just the message or file format, but the implied meaning of the data. When interpreting sensor data we need to understand what classes it belongs to. Many system sensors will periodically scan (or "sample") part or all of the system.

Type

Some sensors may report just what they consider to be exceptions (and therefore things not reported could be assumed to be fine), and some may report everything they find.

Scope

The scope of the data in the message may pertain to all of the system modelled in the system model or just to part of it. For instance, if a complete system consisted of on-premise and cloud systems, a single sensor (such as an OpenVAS agent) would only be able to report on one part or the other. Knowing the scope is particularly important in the case that the message "type" is to report exceptions. In this case, that anything within the scope that is not mentioned is considered fine so the scope if key.

Expiry

There are common cases where the state change reported in a message should not be considered permanent. The state change may:

- never expire (applicable to sensors used when creating Model 1 from Model 0).
- expire after a certain time period.
- expire after an absolute time; or
- be ignored if there is a newer report about the same scope from the same sensor.

II.4.4.c. Sequencing of "Intended" Model Updates, Sensor Data, Risk and Recommendation Requests

Requirements:

- The "Intended" system model (Model 1) may need updating at any point, when the intended system is itself updated.
- Sensor data about the system (affecting Model 2) may come from multiple sources and at any time.
- Requests from other integrated software or human operators for an updated current risk calculation (based on Model 2) or recommendation (Model 4) may come in at any time.

For a recommendation to be provided, a risk calculation must first be run. At the point of running a current risk calculation we must ensure that we have the best possible model of the current situation based on Model 1 plus any valid detected changes that have been received.

There may be a workflow in place to request an updated risk calculation or recommendation which coincides with a new report from a sensor. For example, a daily OpenVAS scan may be followed by a daily risk calculation, but these things do not have to be dependent. In the initial CyberKit4SME integration (and in previous work) the state update and the risk calculation request were always tied together, limiting flexibility. In this implementation we keep a record of state report updates and apply them to Model 1 to create Model 2 when necessary, to satisfy the requirements above.

The most frequently a new risk calculation could sensibly be requested is every time new sensor data arrives (or expires). If the frequency of the risk calculation is slower than the rate of sensor input, then there is the possibility that high risk events are missed.

II.4.4.d. Implementation

A "State Report Message" is used to describe changes that should be applied to the system model, along with metadata regarding the type of report, its scope and when the data contained within should expire. Here we describe the message formats to communicate the necessary data. The API and architecture are described below in Section II.4.5.

State Report Message

The message has up to 4 fields: "type", "expiry", "scope" and "state". Only "state" is mandatory.

Type

Defines whether the state report message is describing everything in the system or just what the sensor considers to be exceptions.

"**type**": "everything" | "exceptions"

Expiry

If the "expiry" field is not defined, or "null", then the state change is understood as intended to be permanent, updating the baseline model.

An individual message can expire in more than one way: "newest" could go with "period" for instance. The field therefore needs to be a list of expiry data.

Each "**expiry**" list element has a type:

"**type**": "newest" | "period" | "timestamp"

If "type" is "period" or "timestamp" then we also need:

"**time**": number of seconds (for "period") | ISO date-time string

If "type" is "newest" then we also need:

"**label**": a string used to search for reports of the same type where a newer one should overwrite the older one. The label could be just the name of the data source (e.g. "nmap") or, if you had two data sources reporting about the same aspect of some assets, the same label could be used in reports from both sources so that the data from whichever source had reported most recently was used.

Scope

“**scope**”: can be undefined or null if it is not needed.

“**scope.type**”: “all” | “embedded” | “resource”

If “scope.type” is “resource” then we also need:

“**scope.uri**”: a scope resource identifier URI

If “scope.type” is “embedded” then rather than referring to a scope resource, the scope information is embedded in the message using:

“**scope.items**”: an Asset Set Description message.

State

The “state” part of the message is mandatory and describes the configuration state of one or more assets and ultimately will be translated into updates of trustworthiness attribute levels, impact levels and controls on specific assets. If this part of the message is created by an external data source it may be more abstract, describing an asset by its label or CPE for instance, but a more tightly integrated data source could generate a message specifying an asset by ID or even going directly to the trustworthiness attribute set URI, etc.

“state”: a list of state items

Each state item in the list would describe an asset plus one or more of the “trustworthiness”, “impacts” and “controls” fields:

```
{
    "asset": {
        <Asset Description>
    },
    "trustworthiness": [
        {
            "trustworthinessAttribute": "http://.../<TWA>",
            "level": "http://.../<TrustworthinessLevel>",
            "operator": "="
        },
        ...
    ],
    "impacts": [
        {
            "misbehaviour": "http://.../<Misbehaviour>",
            "level": "http://.../<ImpactLevel>",
            "operator": "="
        },
        ...
    ],
    "controls": [
        {
            "control": "http://.../<Control>",
            "enabled": true
        },
        ...
    ],
}
```

The “trustworthiness.operator” and “impact.operator” fields can take the following values:

- “=”: set the level to the specified value;
- “<=”: decrease the level to a value, but not increase the level if it is already below that value;
- “>=”: increase the level to a value, but not decrease the level if it is already above that value.

We do not currently have a use case where we need to change an impact level of a consequence: it is included for completeness.

To apply this message, the specific asset must first be identified in the system model and then the asset’s attributes set.

We can also have a form of this part of the message where the “asset” is null and the other fields refer to trustworthiness attribute set URIs, impact set URIs and control set URIs (that is URIs referring to asset-specific properties).

```
{
    "trustworthiness": [
        {
            "trustworthinessAttributeSet": "http://.../<TWAS>",
            "level": "http://.../<TrustworthinessLevel>",
            "operator": "="
        },
        ...
    ],
    "impacts": [
        {
            "misbehaviourSet": "http://.../<MisbehaviourSet>",
            "level": "http://.../<ImpactLevel>",
            "operator": "="
        },
        ...
    ],
    "controls": [
        {
            "controlSet": "http://.../<ControlSet>",
            "enabled": true
        },
        ...
    ],
}
```

Such a message can be applied directly without identifying the asset.

Note that in these lists, only attributes that the sensor has any opinion about are declared. Other configuration data should be left alone.

Asset Description Format

A variety of ways are provided to identify an asset (or assets) in the system-model, ranging from directly referring to asset IDs to more general descriptions using data known to external sensors. Any “type” parameters should be URIs and interpreted according to the domain model’s class hierarchy. If an Asset Description matches more than one asset then the software should raise an exception. Some examples are described below.

Describe a unique asset if we know the asset ID:

```
{
    "id": "4b4bdbb7"
}
```

Describe a unique asset by its label (given the context we should assume this is an asset label):

```
{
    "label": "IE"
}
```

Describe a unique asset by its label and class:

```
{
    "label": "IE",
    "type": "http://.../ApplicationProcess"
}
```

Describe a unique process asset by label, type and metadata describing the CPE string:

```
{
    "label": "IE",
    "type": " http://.../ApplicationProcess",
    "properties": [
        {
            "key": "process",
            "value": "microsoft:internet_explorer"
        },
        {
            "key": "version",
            "value": "8.0.6001:beta"
        }
    ]
}
```

Describe a host asset by IP address metadata (which you would assume to be unique but may not be if two host assets had mistakenly had the same IP added to their additional data):

```
{
    "type": "http://.../Host",
    "properties": [
        {
            "key": "IP",
            "value": "1.2.3.4"
        }
    ]
}
```

Describe an asset of a certain type that has a relationship to another asset (this may return multiple assets if used in a system model query, but may also describe a single new asset):

```
{
    "type": "http://.../ApplicationProcess"
    "relation": {
        "type": " http://.../isConnectedTo",
        "to": { <description of a network asset> }
    }
}
```

The “relation” field needs a “type” and either “from” or “to” (describing the asset it is related to).

Asset Set Description Format

To implement the scope field, it is necessary to be able to define sets of assets.

The general case is:

```
[
    <Multi-Asset Description>,
    ...
]
```

A “Multi-Asset Description” could be one of the Asset Description messages above. It could also include similar descriptions that will match multiple assets (and this should be permitted). Any “type” parameters should be URIs and interpreted according to the domain model’s type hierarchy. Some examples follow.

Describe assets of a certain type that have a relationship to another asset:

```
[
    {
        "type": "http://.../Host",
        "relation": {
            "type": " http://.../isConnectedTo",
            "to": { <description of a network asset> }
        }
    }
]
```

Describe all assets of a type:

```
[
    {
        "type": "http://.../ApplicationProcess"
    }
]
```

Describe a list of assets, using the asset message format:

```
[
    {
        "id": "4b4bdbb7",
        "type": " http://.../Asset"
    },
    {
        "id": "4b4bd999",
        "type": " http://.../Asset"
    },
    {
        "label": "IE",
        "type": " http://.../ApplicationProcess"
    }
]
```

II.4.4.e. Examples

The following examples demonstrate how the State Message Format is suited to all the identified scenarios.

II.4.4.e.1. Bootstrapping the initial system model

A SIEM such as Keenaï or a network scanner such as Nmap have a lot of information about an existing system which can in theory be utilised to create part of a system model (going from Model 0 to Model 1 in the Figure above).

The “expiry” and “scope” fields can be null. Where possible, assets would be described with known metadata (such as IP address or CPE) and relations could be added (most easily using labels to cross-reference). Creating and ingesting such a document is not without difficulties but the message format can support the information required.

This example message describes a Host hosting an Apache web server process.

```
{
    "type": "everything",
    "state": [
        {
            "asset": {
                "type": "http://.../Host",
                "label": "server",
                "properties": [
                    {
                        "key": "IP",
                        "value": "1.2.3.4"
                    }
                ]
            }
        },
        {
            "asset": {
                "type": "http://.../ApplicationProcess",
                "properties": [
                    {
                        "key": "process",
                        "value": "apache:httpd"
                    },
                ],
                "relation": {
                    "type": " http://.../isHostedBy",
                    "asset": {
                        "label": "server"
                    }
                }
            }
        },
        ...
    ]
}
```

Interpreting such messages has not yet been implemented but the message format is suitable for this purpose.

II.4.4.e.2. Host monitoring

In this example, through monitoring log files, Keenaï notices that there has been no log messages from a host in the last 10 minutes and sends a message to Spyderisk describing the down host by IP address. The message would be repeated every 10 minutes while the host remains down. If the host comes up again, no special message is sent.

To finesse the message slightly, we can set the expiry period to a little more than 10 minutes to ensure that subsequent host down messages are received reliably before the expiry time.

We use the special Control which indicates that an asset is out of service.

```
{
    "type": "exceptions",
    "expiry": [
        {
            "type": "period",
            "time": 620
        }
    ],
    "state": [
        {
            "asset": {
                "type": "http://.../Host",
                "properties": [
                    {
                        "key": "IP",
                        "value": "1.2.3.4"
                    }
                ]
            },
            "controls": [
                {
                    "attribute": "http://.../DisabledHost",
                    "enabled": true
                }
            ]
        }
    ]
}
```

In this case, a tight integration between Keenaï and Spyderisk means that Keenaï can construct the message using knowledge of the appropriate trustworthiness attribute.

II.4.4.e.3. Untrustworthy connection

If a SIEM such as Keenaï detects that there is a connection to a host from a known “bad” IP address (for instance using information supplied through Service Ledger), then it can send a message to that effect. The expiry semantics could be the same as the previous example.

```
{
    "type": "exceptions",
    "expiry": [
        {
            "type": "period",
            "time": 620
        }
    ],
    "state": [
        {
            "asset": {
                "type": "http://.../Host",
                "properties": [
                    {
                        "key": "IP",
                        "value": "1.2.3.4"
                    }
                ]
            },
            "trustworthiness": [
                {
                    "attribute": "http://.../UserTW",
                    "level": "http://.../TrustworthinessLevelVeryLow",
                    "operator": "<="
                }
            ]
        }
    ]
}
```

As with the previous example, Keenaï can construct the message using knowledge of the appropriate trustworthiness attribute.

II.4.4.e.4. Example from Fog Protect

To ensure the generic solution developed in CyberKit4SME satisfies all known requirements we reviewed examples from previous projects where similar (but custom) integration had been performed. In this scenario an external scanner detects an event, and the system model needs to be updated to reflect the event but the single event causes changes in more than one asset. The software integrating with Spyderisk in this case is not expected to understand the detail of the model or the asset attributes.

To reimplement the scenario an adaptor would be created to map from the reported event to a pre-prepared State Report Message stored in a database (with appropriate user interface if necessary). The example State Report Message relating to this scenario reduces the benevolence and reliability of a user and also reduces the confidentiality trustworthiness attribute of a data set (indicating that it is believed to be compromised):

```
{
    "type": "exceptions",
    "state": [
        {
            "asset": {
                "label": "user",
            },
            "trustworthiness": [
                {
                    "attribute": "http://.../benevolence",
                    "level": "medium",
                    "operator": "<="
                },
                {
                    "attribute": "http://.../reliability",
                    "level": "medium",
                    "operator": "<="
                }
            ]
        },
        {
            "asset": {
                "label": "config-file",
            },
            "trustworthiness": [
                {
                    "attribute": "http://.../confidentiality",
                    "level": "low",
                    "operator": "<="
                }
            ]
        }
    ]
}
```

II.4.4.e.5. OpenVAS integration

The items from the OpenVAS report are processed using an adaptor/plugin, converting from CVEs to trustworthiness attributes, and are inserted into a State Report Message template such as:

```
{
    "type": "exceptions",
    "expiry": [
        {
            "type": "period",
            "time": 172800
        },
        {
            "type": "newest",
            "label": "openvas"
        }
    ],
    "scope": {
        "type": "all"
    },
    "state": [...]
}
```

This template uses an expiry combining 48 hours and “newest”. Therefore, the vulnerability data contained within will be applied to the system model for the next 48 hours (but would then be discarded as out of date), but would also be ignored (discarded) if a newer report with the same “openvas” label is received.

The “state” data in the implementation has been created such that the assets are identified using metadata (e.g. IP addresses and CPE strings), meaning the actual assets in the system model are found at the point of applying the report for the purpose of a risk calculation.

II.4.5. System Modeller Adaptor APIs

Whilst the Java “system modeller” service provides most endpoints, it is the “System Modeller Adaptor” (or “ssm-adaptor”) which provides additional functionality to receive and process State Report Messages, apply those reports in advance of a risk calculation or recommendation calculation, and plot attack graphs.

The architecture is shown in the following Figure.

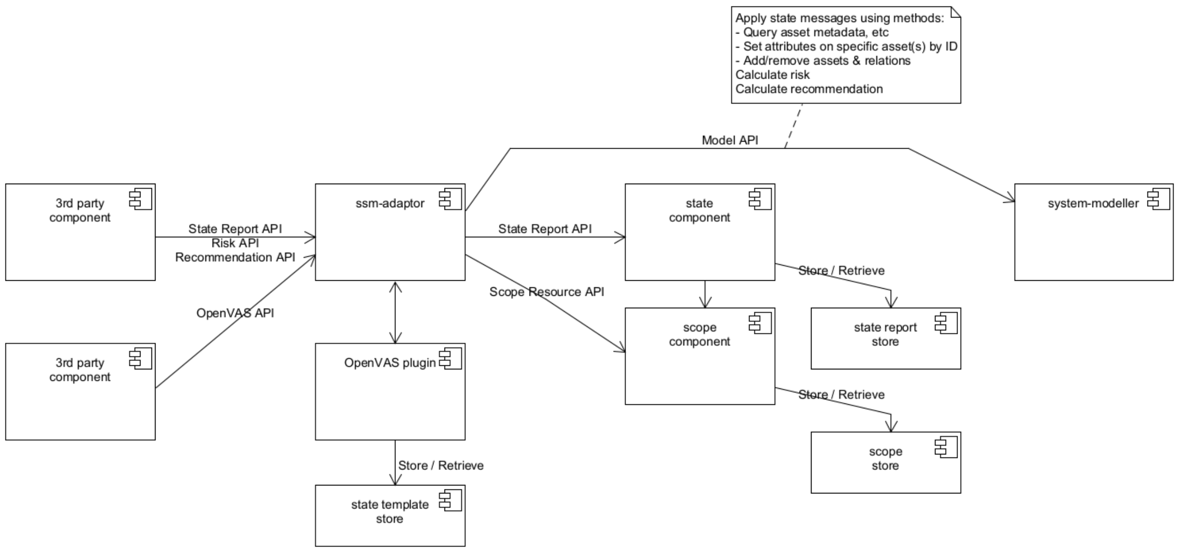


Figure 7. System modeller adaptor architecture.

II.4.5.a. State Report Message API

POST /api/v2/models/{model-webkey}/state/

This endpoint can be used to send State Report Messages to. It responds with a 200 code on successful receipt.

Upon receiving a report, the message along with the time it was received is stored in a database.

This method will be used directly by tightly integrated sensors which are able to construct the necessary message. If necessary, an adaptor/plugin can be used to convert from a sensor's output format to use this API (such as with OpenVAS).

GET /api/v2/models/{model-webkey}/state/

A GET to this endpoint would retrieve the valid list of state messages (oldest to newest) to be applied to the system model.

II.4.5.b. Sensor Notification API

An endpoint to receive notifications of new sensor reports has been created, initially just for OpenVAS reports:

POST /api/v2/models/{model-webkey}/notify/openvas-report

This method is used by Keenaï which orchestrates the timing of OpenVAS scans. A call to this endpoint causes the ssm-adaptor to look for a new report(s) in a pre-configured folder. This folder needs to be shared between Spyderisk and Keenaï.

Keenaï"s OpenVAS configuration creates separate files for different severity levels. Spyderisk need to know about all vulnerabilities. This means Spyderisk requires the high and medium severity files.

There may be more than one OpenVAS scanner per Spyderisk model (for instance, in the case of services in cloud and on-prem). Also Keenaï splits the scans into vulnerability severity classes: high, medium and low. So even if there is only one OpenVAS scanner then more than one file could need reading by the ssm-adaptor.

Therefore, the ssm-adaptor and Keenaï should be configured with the root location on the disc that scan result files are to be placed. Keenaï should place new scan files into a sub-folder of the root location using the Spyderisk model webkey as the folder name. The call from Keenaï to the ssm-adaptor has the webkey as a parameter so the ssm-adaptor will be able to find the sub-folder and will need to manage finding the newest files in there itself (recording which ones it has previously processed).

e.g.

- <openvas-report-root>
    - <ssm-model-webkey>
        - 4530912f-2251-41ca-92a1-54cd9e0c7131-high.xml
        - 4530912f-2251-41ca-92a1-54cd9e0c7131-medium.xml
        - 45c40c99-9e21-4530-b6a4-92d4ad237bac-high.xml
        - 45c40c99-9e21-4530-b6a4-92d4ad237bac-medium.xml

II.4.5.c. Recommendation API

POST /api/v2/models/{model-webkey}/recommendations

Upon receiving a call to this endpoint, the ssm-adaptor responds immediately with a job ID and then, in the background:

1. If it is a current risk calculation
    - Copy the intended system model (1) to the temporary one for current risk calculation (2).
    - Update the state of the temporary model (as described above).
2. Execute the recommendation algorithm, storing the resulting recommendations for later querying.
3. If it is a current risk calculation then delete the temporary model (2).

Parameters in URL:

- model-webkey (string): the webkey of the system model where recommendations are required.

Parameters in query string:

- risk_mode (string): specifies the risk mode calculation CURRENT or FUTURE by enabling corresponding control sets in the model, default value is CURRENT

Responses:

- 202 Successful response

Example value:

```
{

  "jobid": "string",
  "status": "string"

}
```

GET /api/v2/models/{model-webkey}/jobs/{job-id}

This is an auxiliary call to support long-running jobs. It allows the client to check the status of a background task (e.g. risk calculation). The ssm-adaptor will only respond successfully if the “model-webkey” and “job-id” match existing resources.

Parameters:

- model-webkey (string): the webkey of the system model
- job-id (string): the ID of an executing background task

Response code:

- 200 Successful response

Example response:

```
{
  "modelId": "string",
  "status": "CREATED",
  "err_msg": "string",
  "created_at": "2022-03-09T15:48:23.889Z",
  "updated_at": "2022-03-09T15:48:23.889Z",
  "messages": "string",
  "id": "string"
}
```

The state model for the status shown in the following Figure.

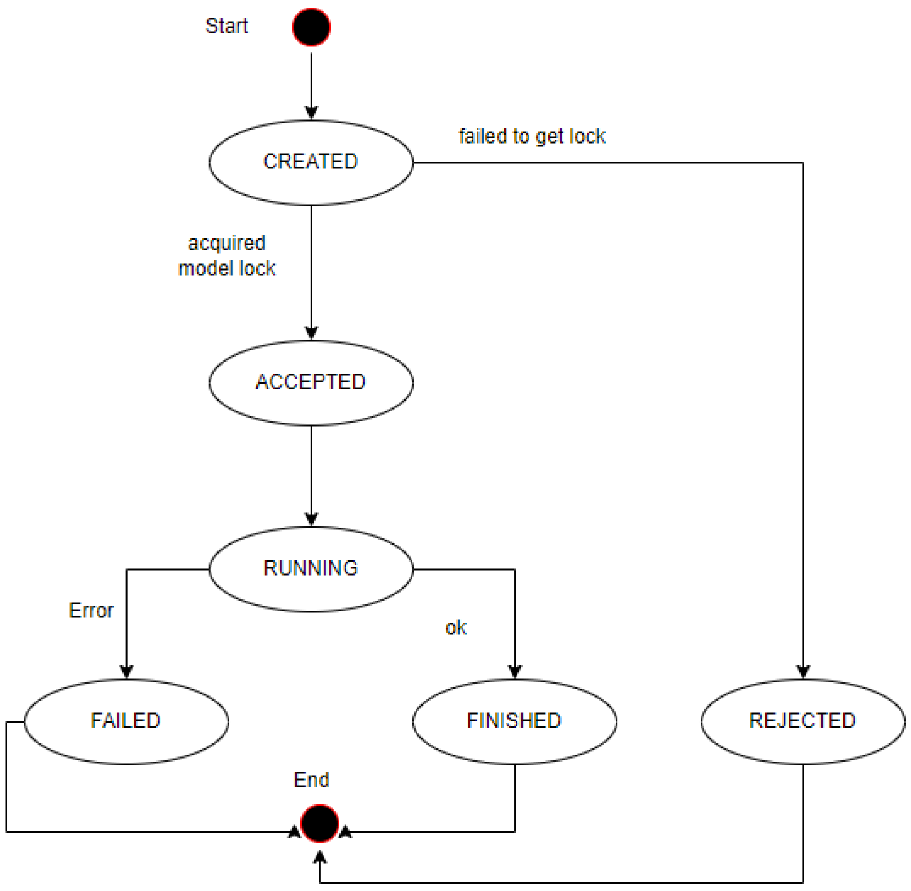


Figure 8. State model for job status.

GET /api/v2/models/{model-webkey}/recommendations/{job-id}

Once a recommendation job has completed, this method can be used to retrieve the results.

Parameters:

- model-webkey (string): the webkey of the system model
- job-id (string): the ID of a completed risk calculation task

Responses code:

- 200 Successful response

Example response:

```
{
    "current": {
        "state": {
            "risk": {
                "overall": "string",
                "components": {
                    "veryHigh": 0,
                    "high": 0,
                    "medium": 0,
                    "low": 0,
                    "veryLow": 0
                }
            },
            "consequences": [
                {
                    "asset": {
                        "label": "string",
                        "type": "string",
                        "uri": "string",
                        "identifier": "string",
                        "properties": [
                            {
                                "key": "",
                                "value": ""
                            }
                        ]
                    },
                    "label": "string",
                    "description": "string",
                    "impact": "string",
                    "likelihood": "string",
                    "risk": "string",
                    "uri": "string"
                }
            ]
        }
    },
    "recommendations": [
        {
            "identifier": 0,
            "category": "string",
            "controlStrategies": [
                {
                    "uri": "string",
                    "description": "string"
                }
            ],
            "controls": [
                {
                    "label": "string",
                    "uri": "string",
                    "asset": {
                        "label": "string",
                        "type": "string",
                        "uri": "string",
                        "identifier": "string",
                        "properties": [
                            {
                                "key": "",
                                "value": ""
                            }
                        ]
                    },
                    "action": "string"
                }
            ],
            "state": {
                "risk": {
                    "overall": "string",
                    "components": {
                        "veryHigh": 0,
                        "high": 0,
                        "medium": 0,
                        "low": 0,
                        "veryLow": 0
                    }
                },
                "consequences": [
                    {
                        "asset": {
                            "label": "string",
                            "type": "string",
                            "uri": "string",
                            "identifier": "string",
                            "properties": [
                                {
                                    "key": "",
                                    "value": ""
                                }
                            ]
                        },
                        "label": "string",
                        "description": "string",
                        "impact": "string",
                        "likelihood": "string",
                        "risk": "string",
                        "uri": "string"
                    }
                ]
            }
        }
    ]
}
```

II.4.5.d. Attack Graph Plot API

GET /api/v2/models/{model-webkey}/path_plot

The method is available for use by any integrated software and is also currently used to plot graphs for display in the main web interface. Upon receiving a call to this endpoint, the ssm-adaptor will:

1. Run the risk calculation for the model.

2. Request the attack graph data.
3. Plot the attack graph.

Parameters in URL:

- model-webkey (string): the webkey of the system model

Parameters in query string:

- risk_mode (string): specifies the risk mode calculation CURRENT or FUTURE by enabling corresponding control sets in the model, default value is CURRENT

Responses:

- 200 Successful response (an SVG document will be returned)
- 404 Item not found
- 422 Validation error

II.4.6. Web Interface Updates

Numerous updates have been made to the visual elements of the web interface over the period with a focus on improving usability, including the attack graph display already described in Section II.4.1). The other, more noticeable updates are described here.

Figure 9. Updated logo and documentation links. Figure 9 and Figure 10 illustrate the tidier top toolbar along with the new branding and updated documentation (linking to https://spyderisk.org). Figure 10 shows the new attack graph button and also shows the highest risk level. Previously, the software indicated the number threats of the highest present risk level (e.g., "Risk: high / 1234 threats". Through the validation process we discovered that this caused the users to focus on reducing the number of threats which should not be the aim and can only really be done by reducing the size of the model. What is important is to reduce the highest risk level, and so now just this is highlighted. It is perfectly acceptable to have thousands of threats in a system if none of them are ever going to occur!

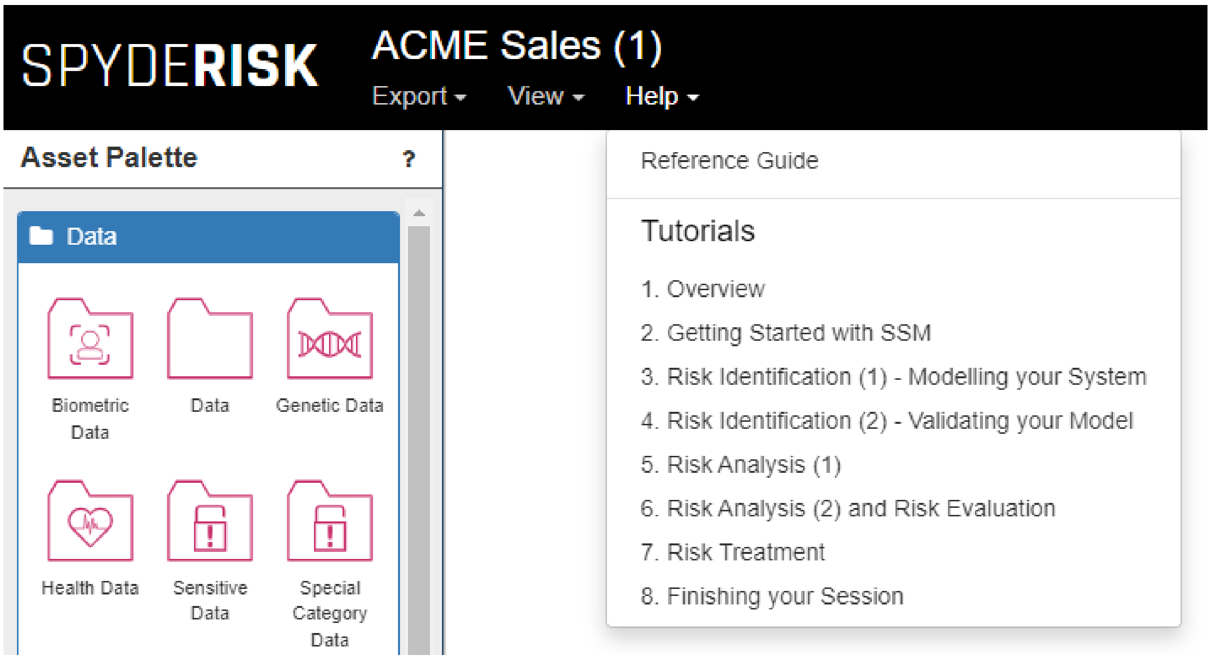


Figure 9. Updated logo and documentation links.

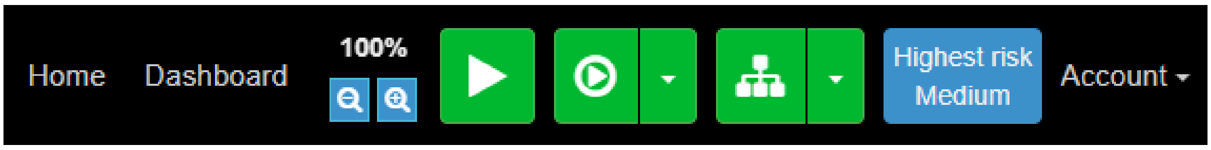
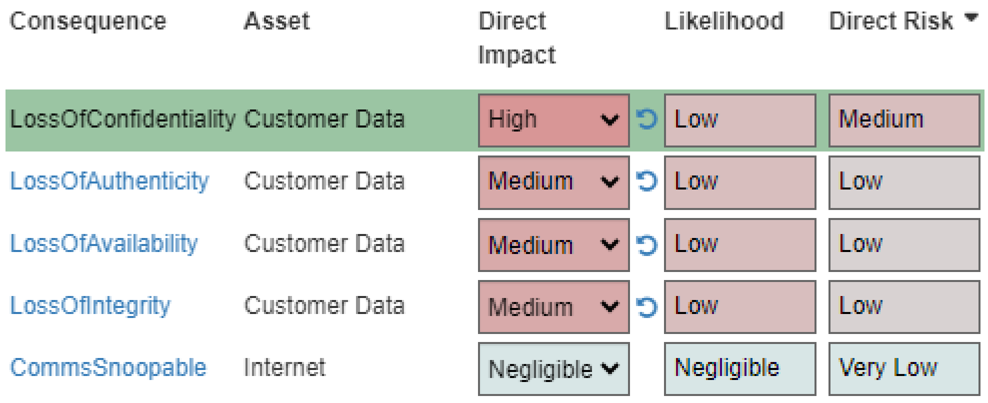


Figure 10. Tidier toolbar, including new attack graph button and risk indicator.

Figure 11 shows a tidier layout and the addition of a ”reset” button which resets the impact level for a Consequence at an Asset back to its default value – a feature that was not previously available.



Consequence	Asset	Direct Impact	Likelihood	Direct Risk
LossOfConfidentiality	Customer Data	High	Low	Medium
LossOfAuthenticity	Customer Data	Medium	Low	Low
LossOfAvailability	Customer Data	Medium	Low	Low
LossOfIntegrity	Customer Data	Medium	Low	Low
CommsSnoopable	Internet	Negligible	Negligible	Very Low

Figure 11. Tidier layout and inclusion of "reset" arrow to reset impact to the default level.

The icon for an adult previously appeared to be wearing a shirt and tie which is generally associated with business men. This icon has been replaced with a non-gendered icon as seen in Figure 12.

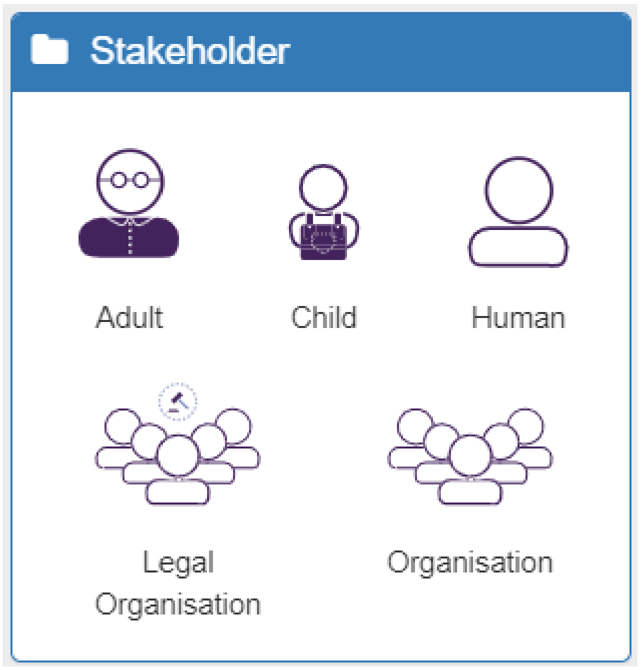
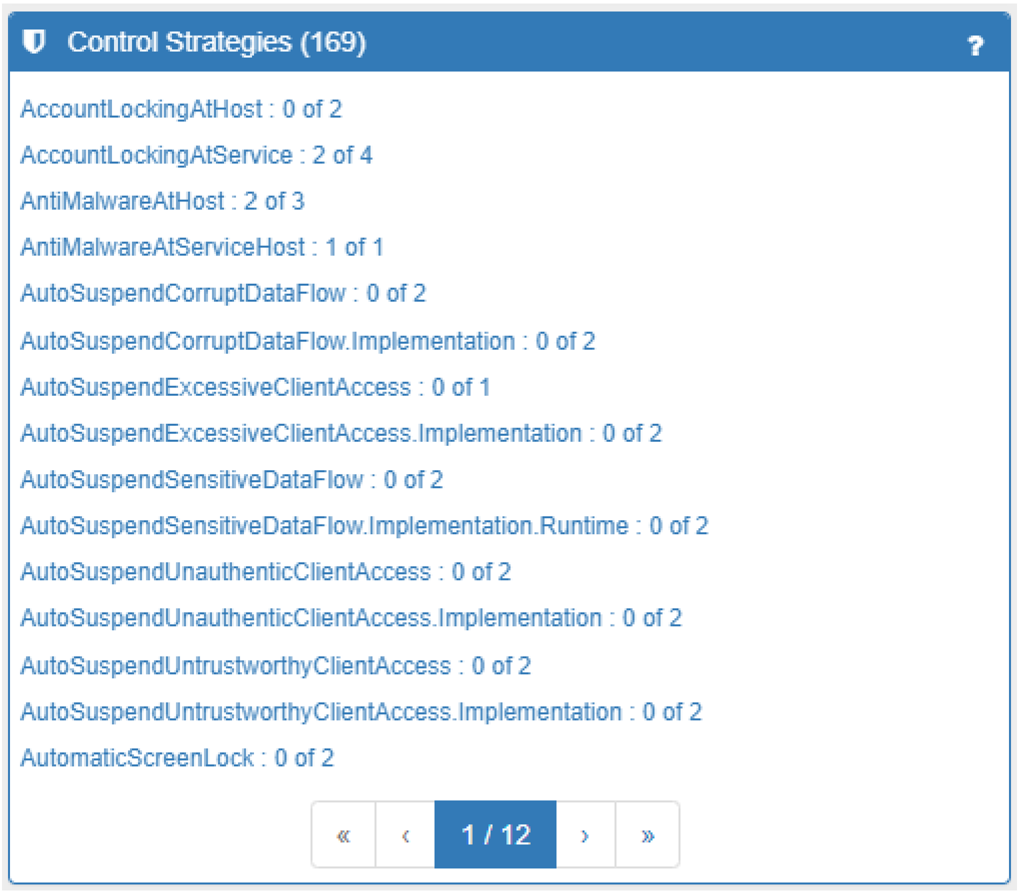


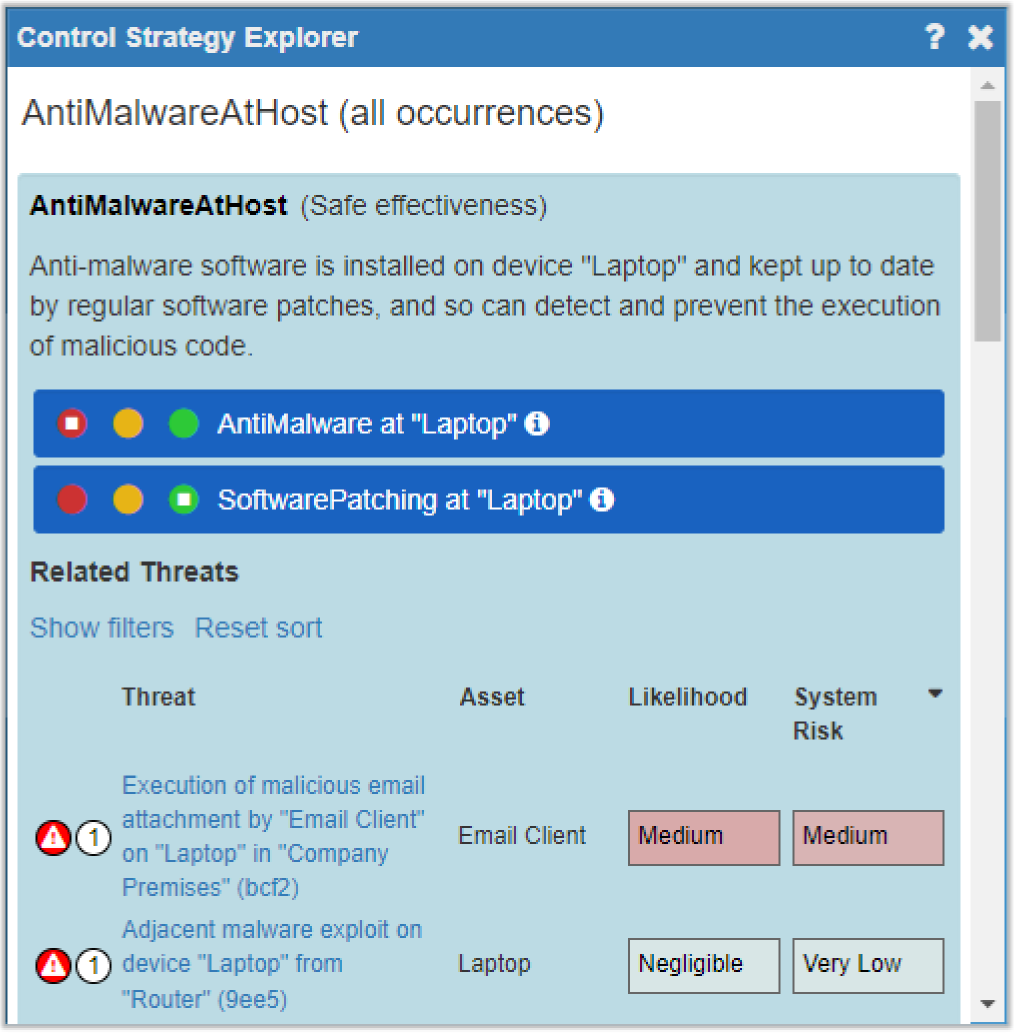
Figure 12. Refreshed "Adult" icon to make it non-gendered.

As previously mentioned, sometimes for a Control to be effective it is necessary to use it in conjunction with one or more other Controls as part of a Control Strategy. Control Strategies have been presented with Threats in the Threat Explorer since the concept was introduced, but looking across the whole model it was previously only possible to browse and interrogate a global list of Controls. We have now added a list of Control Strategies (Figure 13) and a Control Strategy Explorer (Figure 14) to provide direct access independent of Threats. In this way, if you know the Control Strategy you need you can navigate directly there, and it is also possible to spot where a Control Strategy is implemented in some but not all cases which may indicate a mistake. The Control Strategy Explorer also shows all the Threats enabled or reduced in likelihood by the Control Strategy (information which was previously not available).


Control Strategies (169)

AccountLockingAtHost : 0 of 2
AccountLockingAtService : 2 of 4
AntiMalwareAtHost : 2 of 3
AntiMalwareAtServiceHost : 1 of 1
AutoSuspendCorruptDataFlow : 0 of 2
AutoSuspendCorruptDataFlow.Implementation : 0 of 2
AutoSuspendExcessiveClientAccess : 0 of 1
AutoSuspendExcessiveClientAccess.Implementation : 0 of 2
AutoSuspendSensitiveDataFlow : 0 of 2
AutoSuspendSensitiveDataFlow.Implementation.Runtime : 0 of 2
AutoSuspendUnauthenticClientAccess : 0 of 2
AutoSuspendUnauthenticClientAccess.Implementation : 0 of 2
AutoSuspendUntrustworthyClientAccess : 0 of 2
AutoSuspendUntrustworthyClientAccess.Implementation : 0 of 2
AutomaticScreenLock : 0 of 2

Figure 13. The list of Control Strategies for the whole model is now available.



Control Strategy Explorer

AntiMalwareAtHost (all occurrences)

AntiMalwareAtHost (Safe effectiveness)

Anti-malware software is installed on device "Laptop" and kept up to date by regular software patches, and so can detect and prevent the execution of malicious code.

AntiMalware at "Laptop"

SoftwarePatching at "Laptop"

Related Threats

Show filters Reset sort

Threat	Asset	Likelihood	System Risk
Execution of malicious email attachment by "Email Client" on "Laptop" in "Company Premises" (bcf2)	Email Client	Medium	Medium
Adjacent malware exploit on device "Laptop" from "Router" (9ee5)	Laptop	Negligible	Very Low

Figure 14. The new Control Strategy Explorer aids understanding by letting the user explore all instances of a Control Strategy and see the related Threats.

II.4.7. Model Discovery

We previously identified the problem that Spyderisk users had in constructing their system models because there is often a lack of knowledge of what the assets and relations in the real system actually are. Having an up-to-date asset register is one of the basic steps to take to improve cyber-security but it is not unusual to find that an organisation does not know in sufficient detail what they have.

We did some further prototyping work on creating system models from information gained from network scans (for legacy systems) or cloud APIs (for newer infrastructure as code systems) but did not complete this as a working feature as there were other more pressing priorities and the validation partners were able to create their system models without this feature. The new model state API (described in Section II.4.4) has been designed to ingest such state information in the expectation that this work may be continued at a later time.

Having previously looked at interpreting the output of Nmap scans, we investigated the options for creating system models of cloud deployments as these are increasingly common, in particular for SMEs who do not have the capacity to maintain their own physical servers.

Deployments of services in cloud providers can generally be made in two ways: either manually, one resource at a time in a web interface, or automatically by defining the required deployment in a configuration file and submitting the deployment description to the cloud provider. The second option has the advantage that the configuration is well defined in a file (or files) which can be kept in a version control system, and is known as Infrastructure as Code (or IaC).

Different cloud providers have their own configuration files and languages to define a deployment, but there is also the Terraform system[17] which provides a common language which can be then targeted at different cloud providers (though still with specialisations to address the different capabilities and subtleties of the different providers). The multi-cloud aspect of Terraform makes it an attractive target for conversion into a Spyderisk system model, but it does not address those deployments that have not been made via Terraform

Fortunately, there is a second project, Terraformer[18], which interrogates a wide range of cloud providers and creates Terraform configurations files (HCL format) from what it discovers. This makes the HCL format a good choice for conversion into a Spyderisk system model. There is also a Python library, pyhcl[19], which will read in the HCL files into a format which can then be manipulated. It would be possible to build a Python adaptor which read in HCL files and used the new Spyderisk Simple Entity API (as described in Section II.4.3) to create a (partial) system model in the service.

Figure 15 illustrates the proposed toolchain to convert from a cloud deployment to a Spyderisk system model, along with some examples of the alternative configuration formats used by major providers.

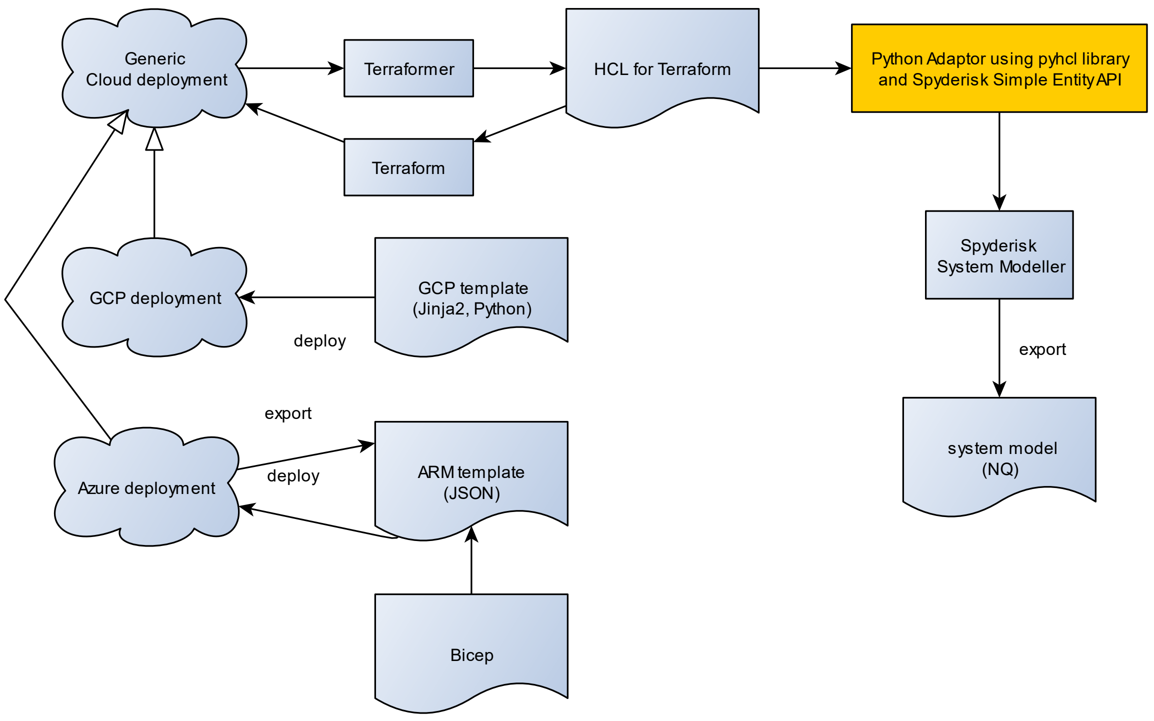


Figure 15. Proposed method to create system models from cloud deployments.

We made a prototype to create partial system models in simple cases, including laying out the assets in the model by using the layout engine of graphviz[20], but creating a comprehensive tool for the general case was a major undertaking which, as a lower priority, could not be

[17] Terraform: https://www.terraform.io/

[18] Terraformer: https://github.com/GoogleCloudPlatform/terraformer

[19] pyhcl: https://github.com/virtuald/pyhcl

[20] Graphviz: https://graphviz.org/

resourced. The research and experimentation to reach this point will hopefully be built upon in the future.

II.4.8. Reporting

A requirement from many potential users of Spyderisk is to have more accessible and actionable reporting options to extract data from the risk analysis. In the information security field, ISO 27001[21] compliance reporting is an often-used document style. The concepts used in Spyderisk were developed for automated machine reasoning of risk, and in Section II.3.2 we already described how they map from Spyderisk concepts to those of the ISO 27000 standards. Here we detail how to report the data held in the Spyderisk analysis using the two key documents for ISO 27001: the Risk Treatment Plan (RTP) and the Statement of Applicability (SOA). This forms part of a larger analysis of reporting conducted for the CyberKit4SME, SYNTHEMA[22] and NEMECYS[23] projects with the implementation to be completed in SYNTHEMA and NEMECYS.

ISO 27001 is part of a family of standards directed at information security risk management, known as the ISO 27000 family. For this discussion, the relevant standards in this family are ISO 27000[8] (Overview and vocabulary), ISO 27001 (Requirements), ISO 27002[24] (Information security controls) and ISO 27005[9] (Guidance on managing information security risks).

The RTP contains the risk identification and assessment - i.e. risks (including impact and likelihood), the associated risk level and the causing threats. It then illustrates how the risks are controlled - i.e. what controls have been selected for each risk, together with the resulting residual risk levels for each risk.

The SOA is a statement of which ISO27001 Annex A controls (which themselves have been derived from) have been selected, which risks they apply to, where they are applied; and if an Annex A control is not selected, a justification why not.

Given that several controls used in the RTP can be those of ISO 27001 Annex A, The SOA is derived from the RTP - i.e. the justifications for why, how and where the ISO 27001 Annex A controls are applied comes from the risk assessment and control selections found in the RTP (the “how” of the SOA), along with the associated residual risk; the residual risk (assuming it is acceptable) is justification of the why.

II.4.8.a. Risk Treatment Plan

The ISO 27000 family implementation guidance (ISO 27003[25]) advises the following for creating a Risk Treatment Plan:

> “*Guidance on formulating an information security risk treatment plan (6.1.3 e))*
>
> *ISO/IEC 27001 does not specify a structure or content for the information security risk treatment plan. However, the plan should be*

[21] ISO/IEC 27001:2022. Information security management systems Requirements. https://www.iso.org/standard/27001

[22] SYNTHEMA: Synthetic haematological data over federated computing frameworks (https://synthema.eu/)

[23] NEMECYS: Cybersecurity of connected medical devices (https://nemecys.eu/)

[24] ISO/IEC 27002:2022. Information security, cybersecurity and privacy protection — Information security controls https://www.iso.org/standard/75652.html

[25] ISO/IEC 27003:2017. Information technology — Security techniques — Information security management systems — Guidance https://www.iso.org/standard/63417.html

formulated from the outputs of 6.1.3 a) to c). Thus the plan should document for each treated risk:

- *selected treatment option(s);*
- *necessary control(s); and*
- *implementation status.*

Other useful content can include:

- *risk owner(s); and*
- *expected residual risk after the implementation of actions.*

If any action is required by the risk treatment plan, then it should be planned indicating responsibilities and deadlines (see also 6.2); such an action plan can be represented by a list of these actions.

A useful information security risk treatment plan can be designed as a table sorted by risks identified during the risk assessment, showing all the determined controls. As an example, there can be columns in this table which indicate the names of the persons responsible for providing the controls. Further columns can indicate the date of implementation of the control, information about how the control (or a process) is intended to operate and a column about the target implementation status."

Given this guidance, the natural form for the Spyderisk output format for ISO 27001 is a table with two major sections:

- **Risk Assessment**, where Threats / Information Security Events / Incidents lead to Consequences. Each Consequence has a Risk Level combined from its Likelihood and Impact (Consequence Criteria)
- **Risk Control,** where Residual Risk showing the effect of Controls on the identified Risk for each Consequence

The focus of the table is on Consequences – this is because each Consequence has an associated Risk Level, which must be assessed and if necessary controlled. Therefore each row in the table corresponds to a specific Consequence, with its associated causing Threats, Impact, Risk Level and Controls.

Table 2 shows the proposed Spyderisk output format for the ISO 27001 Risk Treatment Plan.

Table 2. ISO 27001 Spyderisk Output Format.

		Risk Identification & Assessment							Risk Control						
SSM	**C ID**	**Root Cause Threat**	**Direct Cause Threat**	**Consequence**	**Asset**	**Impact**	**Likelihood**	**Risk Level**	**Control Strategy**	**Control-Asset**		**Residual Likelihood**	**Residual Risk**	(none)	**Implement-ation Status**
ISO 27000 series	**C ID**	**Information Security Event (ISO 27000)**	**Information Security Incident (ISO 27000)**	**Consequence (ISO 27000)**	**Information Asset (ISO 27000) Asset (ISO 27005)**	**Consequence Criteria (ISO 27005)**	**Likelihood (ISO 27000)**	**Level of Risk (ISO 27000)**	[Collection of] **Control (ISO 27000)**	**Control (ISO 27000)**		**Residual Likelihood (ISO 27000)**	**Residual Level of Risk (ISO 27000)**	**Risk Owner**	**Implement-ation Status**
	1	Execution of malicious email attachment by EmailClient on Notebook	Loss of control due to compromised input	Loss of Control	Steel Mill Furnace	High	High	High	Anti Malware At Host	Anti Malware (ISO 27001 8.7 Protection against malware)	Notebook	Negligible	Negligible	Systems Manager	Suggested / Excluded / In Progress / Planned / Implmented
										Software Patching	Notebook				
	2	Execution of malicious email attachment by EmailClient on Notebook	Compromised or impersonated client injects fake content to Furnace Control Interface	Loss Of Authenticity	Control Data	High	High	High	Anti Malware At Host	Anti Malware (ISO 27001 8.7 Protection against malware)	Notebook	Negligible	Negligible	Systems Manager	Suggested / Excluded / In Progress / Planned / Implmented
			Compromised Furnace Control Interface injects fake content into the encrypted flow of Control Data to/via Furnace Control							Software Patching	Notebook				
									Spam Filtering	Spam Filtering	Email Client	Low	Low	Systems Manager	Suggested / Excluded / In Progress / Planned / Implmented

Table 2 has two sets of column headings, firstly showing the Spyderisk concept name and secondly the equivalent ISO 27000 series name. The table has two rows with illustrative examples of Consequences, causing threats, risk levels and Controls. The columns are described as follows.

Each Consequence has an ID - an identifier of risk (denoted **C ID – Consequence ID**). This is simply a free text field intended for any kind of unique identifier specified by the analyst. It is the choice of the analyst using whatever naming convention that they use.

Risk Identification & Assessment

This is the first major section, describing Consequences. Each Consequence occurs at an Asset, has an associated Impact, Likelihood and Risk Levels. Each Consequence may be caused either directly or indirectly by one or more Threats, thus there are sub-rows for Threats associated with the Consequence. There are two columns for the causing Threats, one column describing the Threats that directly lead to the Consequence, and another for any root cause Threats that trigger a chain of Primary and Secondary Threats and Consequences that eventually lead to the Consequence that is the subject of the row.

It may be the case that the same root cause Threat leads to multiple different direct cause Threats for the Same Consequence, and this is illustrated in Consequences 1 and 2. Both are caused by the same root cause, but the direct cause Threats and the Consequences are different.

- **Root Cause** – All Root Cause Threats that lead to this specific Consequence. (Threat ID + Description)
- **Direct Cause** – All Direct Cause Threats that lead to this specific Consequence (ID + Description)
- **Consequence** – The type of the Consequence
- **Asset** – The Asset that Consequence type occurs on
- **Impact** – The Impact (damage, severity) of the Consequence occurring on the Asset
- **Likelihood**- The Likelihood of Consequence occurring on the Asset
- **Risk Level** – The overall Risk Level – determined by combining the Impact and the Likelihood.

Risk Control

Spyderisk may recommend multiple Control Strategies to address a single Consequence. These consist of different Control-Asset pairs and each Control Strategy has its own limitation effect on a Threat's Likelihood (and thus on a Consequence likelihood) resulting from the combined effect of all Control-Asset pairs in the Strategy. Control Strategies are presented as options to the modeller, who is responsible for deciding upon which Strategy to choose. Different options of Control Strategies are illustrated in Consequence 2, where there are two Control Strategies suggested, with differing levels of Risk limitation.

- **Control Strategy** – the name of the collection of Control-Asset pairs that combined limit the likelihood of the Consequence.
- **Control & Assets** - All Control-Asset pairs in the Control Strategy. All must be present for the Control Strategy to be valid and have its limitation on the Consequence likelihood. Some Controls will be those referenced in ISO 27001 Annex A. An example here is: "Anti Malware (ISO 27001 8.7 Protection against malware)".
- **Residual Likelihood** - Residual Likelihood of Consequence that results from the correct application of Control Strategy (i.e. all Control-Asset pairs in Control Strategy).

- **Residual Risk Level** - This is the resulting Residual Likelihood combined with the Consequence Impact.
- **Risk Owner** - This is not included in Spyderisk output, but suggested in ISO 27003, so a placeholder is included for manual entry.
- **Implementation Status** – This can be either of the following options. Each option supports manual entry of any additional text / notes regarding the implementation status (e.g. planned implementation date, reason for exclusion).
  - *Suggested* – this means that the Control Strategy has been suggested by the Spyderisk but no decision has yet been made by the modeller whether it will be implemented.
  - *Excluded* – this means that the Control Strategy has been assessed and a decision has been taken not to implement it. This decision will require justification (a common justification is that an alternative Control Strategy was chosen)
  - *Planned* – this means that the Control Strategy is planned for implementation (i.e. a positive decision has been made) but it is not yet implemented.
  - *In Progress* – the Control Strategy is partially implemented.
  - *Implemented* – this means that the Controls in the Control Strategy have been implemented in the real system under examination.

The rows in the RTP are typically sorted by Residual Risk Level, with the highest risks at the top, so it is clear at a glance what the worst case Risk in the system under examination.

II.4.8.b. Statement of Applicability

The Statement of Applicability (SoA), (ISO 27001 Section 6.1.3 d) concerns how standardised controls described in ISO 27001 Annex A (which are derived from ISO 27002) are used in the Risk Treatment Plan. The Implementation guidance in ISO 27003 has the following guidance on SoA.

> *"Guidance on producing a Statement of Applicability (SoA) (6.1.3 d))*
>
> *The SoA contains:*
>
> — *all necessary controls (as determined in 6.1.3 b) and 6.1.3 c)) and, for each control:*
> — *the justification for the control's inclusion; and*
> — *whether the control is implemented or not (e.g. fully implemented, in progress, not yet started); and*
> — *the justification for excluding any of the controls in ISO/IEC 27001: 2013, Annex A.*
>
> *Justification for including a control in part relies on the effect of the control in modifying an information security risk. A reference to information security risk assessment results and the information security risk treatment plan should be sufficient, along with the information security risk modification expected by the implementation of necessary controls.*
>
> *Justification for excluding a control contained within ISO/IEC 27001:2013, Annex A can include the following:*
>
> — *it has been determined that the control is not necessary to implement the chosen information security risk treatment option(s);*
> — *the control is not applicable because it is outside the scope of the ISMS (e.g. ISO/IEC 27001:2013, A.14.2.7 Outsourced development is not*

applicable if all the organization's system development is performed in-house); and

— *it is obviated by a custom control (e.g. in ISO/IEC 27001:2013, A.8.3.1 management of removable media could be excluded if a custom control prevents the use of removable media).*

NOTE A custom control is a control not included in ISO/IEC 27001:2013, Annex A.

A useful SoA can be produced as a table containing all 114 controls of ISO/IEC 27001:2013, Annex A along the rows plus rows with the additional controls that are not mentioned in ISO/IEC 27001:2013, Annex A, if needed. One column of the table can indicate whether a control is necessary to implement the risk treatment option(s) or can be excluded. A next column can contain the justification for inclusion or exclusion of a control. A last column of the table can indicate the current implementation status of the control. Further columns can be used, such as for details not required by ISO/IEC 27001 but usually useful for subsequent reviews; these details can be a more detailed description of how the control is implemented or a cross-reference to a more detailed description and documented information or policies relevant for implementing the control.

Although it is not a specific requirement of ISO/IEC 27001, organizations can find it useful to include responsibilities for the operation of each control included in the SoA."

[ISO 27003:2017]

Given this guidance, the SoA is presented as an alternative view of the Risk Treatment Plan, focused on the Controls from ISO 27001 Annex A – i.e. the table's columns are re-ordered to present the Controls more prominently than in the RTP, and the rows are sorted so that the Controls appear in the order of ISO 27001 Annex A. Each control in Annex A is included as a row, and the row illustrates the Control's implementation decision and status. Where the Controls are either "Planned", "In Progress" or "Implemented", the row illustrates how the Control contributes to lowering the Residual Risk of the associated Consequence, thus justifying its selection. Where Controls are "Excluded", the modeller must enter a reason for its Exclusion.

An example of the Statement of Applicability format is shown in Table 3. This contains mostly the same fields as for the RTP but has additional rows for the Controls in Annex A that are either not mapped into the Spyderisk's knowledge base or are Excluded from implementation. Two example rows illustrate specifically Exclusion ("7.12 Cabling Security)" and Implemented ("8.7 Protection against malware"). The table also describes other Controls not in Annex A (which Annex A names "Custom Controls") in the form of "Software Patching" and "Spam Filtering").

Table 3. Statement of Applicability.

	Risk Control								**Risk Identification & Assessment**				
SSM	**Control Strategy**	**Control-Asset**		**Residual Likelihood**	**Residual Risk**	(none)	Implementation Status		**Root Cause Threat**	**Direct Cause Threat**	**Consequence**	**Asset**	**Impact**
ISO 27000 series	[Collection of] **Control (ISO 27000)**	**Control (ISO 27000)**		**Residual Likelihood (ISO 27000)**	**Residual Level of Risk (ISO 27000)**	**Risk Owner**	**Implementation Status**	**C ID**	**Information Security Event (ISO 27000)**	**Information Security Incident (ISO 27000)**	**Consequence (ISO 27000)**	**Information Asset (ISO 27000) Asset (ISO 27005)**	**Consequence Criteria (ISO 27005)**
		7.12 Cabling security				Systems Manager	*Excluded* + Reason for Exclusion						
	Anti Malware At Host	Anti Malware (ISO 27001 8.7 Protection against malware)	Notebook	Negligible	Negligible	Systems Manager	In Progress / Planned / Implmented	1	Execution of malicious email attachment by EmailClient on Notebook	Loss of control due to compromised input	Loss of Control	Steel Mill Furnace	High
								2	Execution of malicious email attachment by EmailClient on Notebook	Compromised or impersonated client injects fake content to Furnace Control Interface	Loss Of Authenticity	Control Data	High
	Anti Malware At Host	Software Patching	Notebook	Negligible	Negligible	Systems Manager	In Progress / Planned / Implmented	1	Execution of malicious email attachment by EmailClient on Notebook	Loss of control due to compromised input	Loss of Control	Steel Mill Furnace	High
								2	Execution of malicious email attachment by EmailClient on Notebook	Compromised Furnace Control Interface injects fake content into the encrypted flow of Control Data to/via Furnace Control	Loss Of Authenticity	Control Data	High
	Spam Filtering	Spam Filtering	Email Client	Low	Low	Systems Manager	In Progress / Planned / Implmented	2	Execution of malicious email attachment by EmailClient on Notebook	Compromised or impersonated client injects fake content to Furnace Control Interface	Loss Of Authenticity	Control Data	High

II.4.8.c. Output Format

The chosen format for output is Microsoft Excel. This is because it is widely available, the output can be easily edited manually, and Excel is widely used in ISO 27001 compliance records by users already. Further, once the output is in Excel format, it can be easily translated into other formats (for example tables in MS Word, CSV).

II.5. Knowledgebase Implementation Details

As mentioned, the knowledgebase is the configuration that determines what can be modelled. The knowledgebase has been developed over time to address requirements from many projects. At this point in time we have reached the 5th major iteration of the knowledgebase which will be used in this project's validation which was partly developed in CyberKit4SME. It provides a way to model the real-world system, by which we mean a way to describe a system in sufficient detail that appropriate threats to the system can be found and controlled. A system model as drawn by the user consists of assets and asset to asset relations. In addition, the knowledgebase also describes how to infer additional assets that are required in the model, but which are either not provided to the user or which they may forget to add. The inference of assets that may be forgotten has been expanded in v5 in order to simplify the initial modelling and help less experienced users.

Since the previous report (D4.1) the knowledgebase has been further developed to add features of relevance to CyberKit4SME and also to fix bugs that have been identified through validation.

The main additions are to support the key aspects of GDPR, and to support the modelling of the Parquet encryption technology developed in this project by IBM. In addition, we have begun to look at how to model threats to data privacy which can be caused by the use of machine learning systems (to support the additional work of IBM in this area), though this is being pursued in a different project.

II.5.1. Modelling SDS and PME security

II.5.1.a. Baseline knowledge base limitations

II.5.1.a.1. Data structures

In the baseline Spyderisk knowledge base used as a starting point for these developments, the following models were used to represent data. They are shown as a class diagram in Figure 16.

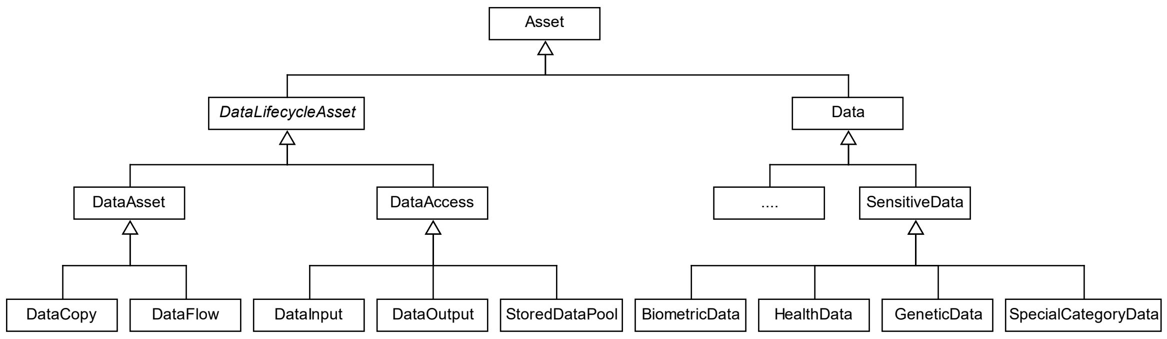


Figure 16. Class diagram for some of the data assets.

The Data asset subclass represents the presence of data in a system. This is an abstract concept used to make it easier for Spyderisk users to define their system models. Subclasses of Data represent data types that are more sensitive (and have higher default compromise impact levels), and/or are subject to regulatory obligations or constraints. Adding a Data asset to a system model signifies that there is at least one copy of that type of data in the modelled system, but it doesn't represent physical copies of the data itself.

Copies of data are represented by inferred assets which are subclasses of the DataLifecycleAsset class and have a relationship to a Data subclass indicating its type. DataLifecycleAsset has further subclasses:

- DataAccess represents a deserialized copy of the Data in the workspace of a running Process, with a relationship to the relevant Process subclass.
- DataAsset represents a serialized copy of the Data, with two subclasses:
    - DataCopy represents a persistent copy of the Data stored on a Host, with a relationship from the relevant Host.
    - DataFlow represents a volatile copy of the Data passed in messages between Processes, with relationships to the source and destination Processes.

Controls like encryption apply to serialized data, i.e., to subclasses of DataAsset, but not to Data (which just represents the data type within the system) nor DataAccess assets. The DataAccess subclass represents both the deserialized data and the ability of the associated Process to serialize/deserialize that data (to or from a DataCopy or DataFlow). Controls representing the possession by a Process of cryptographic keys, or its ability to process data in the encrypted domain apply to DataAccess assets.

Inference rules generate DataAccess in a subclass reflecting the relationship of the Process with the Data:

- DataInput: a subclass of DataAccess indicating that the data is used by the Process, as input to a calculation and/or for display to an interactive user;
- DataOutput: a subclass of DataAccess indicating that the data is created by the Process, as output from a calculation or by obtaining it from a user;
- StoredDataPool: a subclass of DataAccess indicating that the Process serves the Data to other Processes (it provides read and/or write access to other Processes).

The inference sequence creates DataAccess assets first for every Process-handles-Data relationship, then DataCopy assets for every Host-stores-Data relationship, and then DataFlow assets that convey data to or from Processes that access data remotely (i.e., Processes not running on the same Host as a stored copy of the data). Further rules deduce the path taken by each DataFlow, and relationships are added between the DataFlow and processes/networks through which they may pass.

The Data asset cannot be attacked directly because it represents an abstract concept useful because it makes it easier for Spyderisk users to define their system model. The Data asset subclasses DataAsset and DataAccess can be attacked, leading to compromised confidentiality, integrity (including authenticity) and availability. These attacks involve an attacker exploiting access to a Host where data is stored, a Host or Network through which data may pass, or control of a process with access. However, being inferred assets, the DataAsset and DataAccess are not visible in the system model canvas view provided by the system-modeller web user interface. So-called 'surfacing' threats are used to propagate threat effects from DataFlow/DataCopy assets up to the Data asset, so they become visible to the Spyderisk user.

The baseline models assume that Data assets may represent complex structures or even sets of separate data elements, but that these elements have the same relationships with Processes, and the same security attributes and impact levels. Spyderisk users are encouraged to treat disparate data as one Data asset where this is the case. The philosophy is that Spyderisk is a tool for risk analysis, not software design, so application assets (user roles, processes, and data) should be modelled at the highest level possible. Combining data into one asset helps keep in resisting the temptation to model details of the software that are not needed for risk analysis purposes.

However, this means concepts like database tables are modelled as one system data entity, whose elements all have the same security properties. With SDS and PME, this is not true, because fields can be encrypted with different keys, and access controlled independently of the rest of the data.

II.5.1.a.2. Access control mechanisms

In the baseline Spyderisk knowledge base, most access control mechanisms are modelled as Controls:

- Human entry to a Space: enforced by Controls at the Space, representing the strength of the perimeter and means to regulate entry via doors in the perimeter.
- Human access to an Interactive Host: enforced by Controls at the Host, representing user authentication mechanisms (e.g., username/password, biometrics, etc).
- Access to serialized Data stored on its Host: enforced by Controls at the DataCopy asset representing the serialized data (e.g., encryption), or at the Host (e.g., operating system permissions).
- Connection to a Logical Subnet: enforced Controls at the Host (normally a router), representing cryptographic keys or user/device authentication mechanisms.
- Connection by Client processes to a Service process: enforced by Controls at the Service, representing client authentication mechanisms (e.g., username/password, X509, etc).

In these cases, the presence of Controls is assumed to prevent access, except as defined in the system model. The system model represents a 'sunny day', so only legitimate access should be represented. In some cases, the presence of Controls is assumed to prevent even legitimate access unless the asset seeking asset has complementary controls, e.g., if an Interactive Host uses username/password verification, a Human asset representing a legitimate user needs a password. The potential for 'lock out' is then modelled as a side effect threat, which is addressed by the complementary controls. There may also be further 'triggered' threats representing attacks on the control mechanisms, e.g., the threat of a username/password being compromised through user error or via technical means.

For connections to a Logical Subnet or a Service, control may also be achieved by having a Process providing the access control decisions. This is indicated via a 'controls' relationship from the controlling Process to the Logical Subnet or Service to which it controls access. In this case, the connecting device or client must be granted permission by the controlling Process. The Logical Subnet or Service is assumed to have a means to verify the granted permission – no explicit Control is needed because this is implied by the presence of the 'controls' relationship. There is also an inferred interaction between the connecting device or client and the controlling process. This is assumed to be equivalent to the OIDC Implicit Flow (though not necessarily using the OIDC protocols and schema):

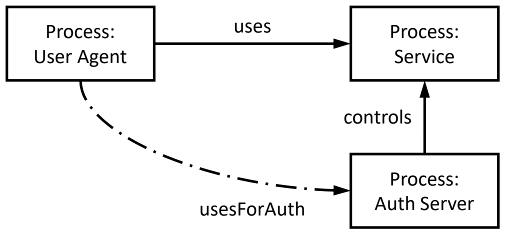


Figure 17. Baseline model for access control provided by an authorization server (implicit flow)

In this pattern, the assets are all Process assets. The 'uses' relationship indicates that the User Agent is a client of the Service, and the 'controls' relationship indicates that the Auth Server controls access to the Service. The inferred (sunny day) interaction is then:

1) User Agent sends a request to Service which redirects to the Auth Server.
2) User Agent authenticates to Auth Server which creates an Access Token.
3) Auth Server sends the Access Token to the User Agent, redirecting it to Service.

4) Service checks the Access Token is signed by the Auth Server.
5) Service establishes a session between User Agent and Service.

The User Agent is therefore a client of the Auth Server, although their interaction cannot be used to exchange data. Their interaction is represented by the 'usesForAuth' relationship, shown as a dashed arrow in Figure 17 to signify that it is inferred to exist. There is no such interaction between the Service and the Auth Server, as this is set up in the Service configuration. The threat of impersonating User Agent to Service does not apply, as the Service is not itself authenticating its clients. The presence of the 'controls' relationship breaks the threat patterns, so those threats are not generated by system-modeller. However, they are replaced by other threats representing other ways to impersonate User Agent to Service, by taking control of the Auth Server, or impersonating User Agent to the Auth Server.

The main issue with this model is that it only covers the OIDC Implicit Flow, in which the token passed via the User Agent has everything needed by the Service. When using SDS with a key Vault, the Vault is protected by a KeyCloak OIDC service, but using the OIDC Authorization Code Flow in which there is an interaction between the Service and the Auth Server. An extension of the existing model is necessary to handle this.

II.5.1.a.3. Cryptographic protection of Data

The baseline model has an Encryption control for every serialized DataAsset, indicating that the data is serialized in encrypted form. If this is asserted, the data is protected from threats to confidentiality and authenticity from an attacker with physical access via a Host or in the network.

However, this also blocks access for Processes that handle the data. That is modelled by a 'side effect' threat, triggered by the presence of an asserted Encryption control, and leading to a loss of availability of the DataAsset. There are two ways to address the side-effect threat:

- Assert controls representing encrypted domain processing, signifying that the Process can process the data without decrypting it.
- Assert the AccessKey control for a DataAccess asset related to the serialized DataAsset, indicating that the associated Process has a key for data decryption or encryption.

The first option doesn't involve a key, so it is not related to the CyberKit4SME knowledge base extensions. It does trigger a side-effect threat representing computational overheads of homomorphic encryption, which may lead via secondary threats to a loss of availability. Where this makes encrypted domain processing inappropriate, the only option is to give the Process a key. But that triggers 'control exploitation' threats representing the fact that an attacker who controls the Process can use its key to access data.

When using SDS access to serialized data is controlled by encrypting the data, but the keys are managed by a separate process (the key Vault). The Process using the data doesn't need to retain its own copy of the key, so the usual side-effect threat (inability to access data) should not apply. What's more, while the SDS does authenticate as itself to access the Vault, the keys provided depend on the rights of a producer/consumer client of the SDS. The client must get an authorization token from Vault, which is protected in turn by KeyCloak. Threats to the data still exist, but these depend on the attacker being able to compromise the Vault or impersonate a legitimate user, in addition to having physical access. The baseline model lacks a means to express the fact that a service manages keys giving access to data. Once this is added, threats in the existing model must be modified so they don't apply where this is the case, and new threats added to model attack paths involving the key management service.

II.5.1.b. CyberKit4SME SDS and PME Extensions

II.5.1.b.1. Access control models

The capabilities used in SDS and PME are provided by an OIDC service (KeyCloak) controlling access to a Vault whose keys allow access to (the Fields in) stored Data. The first step is to extend the existing model so it can also cover the OIDC Authorization Code Flow. This may be needed where keys are supplied direct from the Vault to SDS rather than via the User Agent. The necessary relationships are shown in Figure 18:

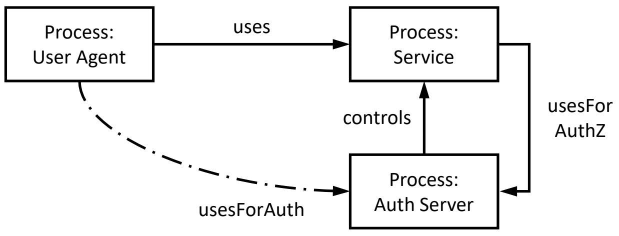


Figure 18. New model for access control provided by an authorization server (Authorization Code Flow)

The inferred interaction in this case is:

1) User Agent sends a request to Service which redirects to the Auth Server.
2) User Agent authenticates to Auth Server which creates an Access Token.
3) Auth Server sends an Authorization Code to the User Agent, redirecting it to Service.
4) Service sends the Authorization Code in a request for an Access Token from Auth Server
5) Auth Server authenticates Service, validates the Authorization Code, and returns the Token.
6) Service establishes a session between User Agent and Service and uses the Access Token to access other resources.

In contrast to the Implicit Flow, here Service is a client of the Auth Server, so this case can be distinguished by adding a new relationship ‘usesForAuthZ’. User Agent is still a client of the Auth Server, with which it must authenticate to access the service, so the ‘usesForAuth’ relationship is still inferred.

The new relationship created some issues with existing knowledge base patterns. The pattern of one service using another and being controlled by it was interpreted to mean that the first service was a proxy to the second, where the proxy forwards credentials sent with connection requests to the back-end service. The back-end service handles authentication, and if successful, the proxy accepts the connection forwards subsequent requests to the back-end service. The back-end service therefore controls whether the proxy accepts connections. In Figure 18, the same pattern appears for the Service and Auth Server, but in this case the Service is not acting as a proxy – it redirects the User Agent but is not involved in the subsequent exchange between User Agent and Auth Server. On the other hand, the ‘usesForAuthZ’ relationship specifies that there is a connection, which could itself be mediated via one or more proxies (this may be necessary in some cloud environments). This issue was solved by adding a new base relationship type for relationships signifying client-service connections and adding ‘usesForAuthZ’ as a sibling rather than a child of the original base type. Existing patterns were then modified as necessary, using the new base relationship type where ‘usesForAuthZ’ should be matched and the original base type where it should be excluded.

The usual client impersonation threats are still negated by the presence of the ‘controls’ relationship, so there was no need to alter those threats. Client impersonation is still possible if the attacker can take control of the Auth Server, impersonate the User Agent to the Auth Server or impersonate the Auth Server to the Service. This last possibility was not previously covered so a new threat was added. Note that the OAuth 2.0 and OIDC specifications address this threat by insisting that controls are used to secure the exchange between Auth Server and Service. In principle the knowledge base must assume the pattern may not be implemented according to the standard, so the new threat should always be created, leaving the system-modeller user to specify whether those controls are present.

II.5.1.b.2. Cryptographic protection of Data

The next extension is to add a model for a Process managing cryptographic keys to control access to stored, serialized data. To do this, the range of the 'controls' relationship was extended to include the DataAsset class, and the 'usesForAuthZ' relationship used as before to represent the key retrieval by the Process from the Vault.

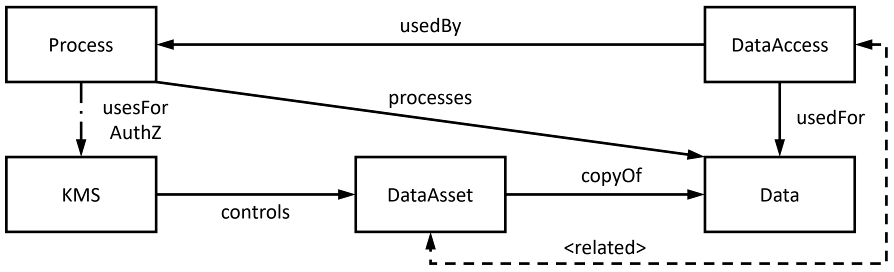


Figure 19. Key management process controlling access to serialized data

The conceptual model is shown in Figure 19, which covers the case where the Process is processing the serialized data (i.e., the data is an input or output for some calculation or user interface). The dotted link between the DataAsset and DataAccess means that the Process must be accessing the DataAsset directly, not via a data service. This is not shown explicitly because the pattern depends on whether the DataAsset is a DataFlow or a stored DataCopy, and whether it is input or output.

Previously, if the 'Encryption' control is asserted at the DataAsset representing the serialized data, it would protect the data but trigger a side-effect threat at the DataAccess unless the Process uses homomorphic encryption or the 'AccessKey' control is asserted at the DataAccess. The presence of the 'controls' relationship from a KMS means the data is encrypted, so threats blocked by the 'Encryption' control should be suppressed when this is present, as should the side effect threat. The assumption is that the Process does not need to retain a key, but it does need access to the data, so it must be a client of the KMS so it can fetch keys when needed. The 'usesForAuthZ' relationship can therefore be inferred, as shown.

If the Process doesn't 'process' the data but only 'serves' it (i.e., enables access to the data by other processes with which it communicates), the situation is a little different. Firstly, unless the data service process selection or update queries against data values, it can access and serve data in encrypted form, leaving its clients to handle encryption or decryption. This is supported in the baseline model, where a data service is not subject to the usual side effect threat and so doesn't need to have an AccessKey. This means if the key is controlled by a KMS, the Process need not have a 'usesForAuthZ' relationship to it. For this reason (in contrast to Figure 19), the 'usesForAuthZ' relationship is not inferred, and must be asserted by the system-modeller user where the service does fetch keys for query processing or so it can exchange data with clients in unencrypted form.

If a data service has no 'usesForAuthZ' relationship with the KMS, it means it exchanges data with clients encrypted by the keys that are managed by the KMS. This means the DataFlows to/from clients are also controlled by the KMS. This inference is added by patterns like the one in Figure 20.

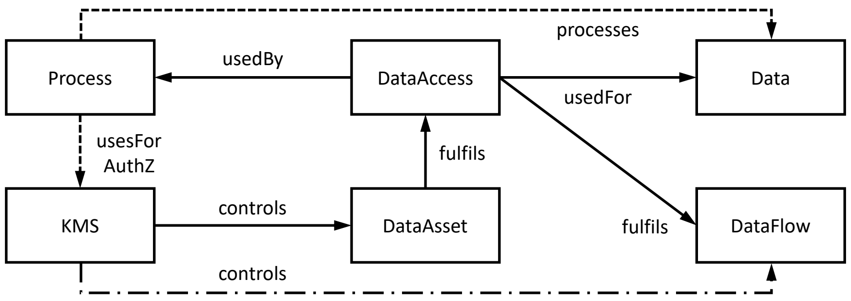


Figure 20. Key management for data flows

This shows a Process serving data obtained from the DataAsset and sending the data to a consumer process as a DataFlow. A similar pattern handles the case where the data flows to rather than from the data service. The relationships shown as dotted lines are ‘prohibited: the prohibited ‘processes’ means the Process serves the data but does not process it, and the prohibited ‘usesForAuthZ’ means the Process does not get keys from the KMS. That being the case, the DataFlow is encrypted with the same keys, so the KMS also controls access to the DataFlow. The pattern in Figure 20 is applied iteratively, so if the DataFlow goes to another data service which forwards it, the forwarded DataFlow is also controlled by the KMS, and so on.

This means if a producer/consumer process has remote access to a stored DataCopy controlled by a KMS via a chain of relaying data service processes, it is possible to specify (by adding an ‘usesForAuthZ’ relationship) which data service (if any) accesses keys from the KMS. DataFlows between the DataCopy and that service are controlled by the KMS, but beyond that service they are not (unless the system-modeller user asserts otherwise by asserting more ‘controls’ relationships). If none of the data services has an ‘useForAuthZ’ the KMS, then all the DataFlows will be controlled by the KMS up to the producer or consumer, where the inference pattern of Figure 19 will be matched, so the producer/consumer will have an inferred ‘useForAuthZ’ relationship with the KMS.

Direct threats to data negated by an ‘Encryption’ control are suppressed by the presence of a ‘controls’ relationship from the KMS. However, there are still some ways an attacker could get access to the unencrypted data:

- If an attacker can take control of the KMS, the data would be compromised if they can access it via the Host of a stored DataCopy, or in the network during transfer of a DataFlow.
- If an attacker can take control of a Process that retrieves keys not on behalf of some client, then they can also get the keys and access the data.

These possibilities were addressed by adding new threats to the knowledge base. Note that the second threat arises only when the Process has a ‘usesForAuthZ’ relationship to the controlling KMS, and only when it is not responding to a client controlled by the KMS.

II.5.1.b.3. Modelling the SDS

In CyberKit4SME, the SDS serves data controlled by the KMS, and the SDS fetches the keys so it can process queries and handle encryption/decryption of the data for the producer/consumer. However, while the SDS authenticates as itself to the KMS, it must pass a token obtained from the KMS via the client, which tells the KMS which data the client is authorized to access. This means before it can access the SDS, the client must login to the KMS (which is protected in turn by a KeyCloak OIDC service) and get a token. In more detail, what happens is as follows:

1) Client logs into the KMS to get a token.
2) Client submits a SQL query to the SDS, with the token from KMS.
3) SDS requests a key to decode table footers from KMS, sending the token with the request.
4) KMS authenticates SDS as itself, checks the token and returns the requested key if the token says the client should have access to the requested key.
5) SDS decrypts the footer to find out which columns are needed, and retrieves result data, still in encrypted form (which it can do because Parquet encryption is used).
6) SDS requests the KMS to return the keys needed to decrypt the result data, sending the token with the request.
7) KMS decrypts and returns the requested keys to which the client should have access.
8) SDS decrypts the result data and returns it to the client (possibly after re-encrypting using a regular AccessKey shared with the client that is not managed by the KMS.

One significant consequence is that no query will be executed unless the client's token is valid, as if it is not, no key is returned by step (4), and the query cannot go ahead in step (5). Even though the client does not authenticate with the SDS, it must login to the KMS before the SDS will run its queries. Hence the sequence is equivalent to (though implemented differently from) an OIDC Authorization Code Flow, in which the KMS controls access to the SDS. To specify this in system-modeller, a 'controls' relationship can be added from the KMS to the SDS. Since the KMS is controlled in turn by a KeyCloak service, the system model should look like the pattern from Figure 21:

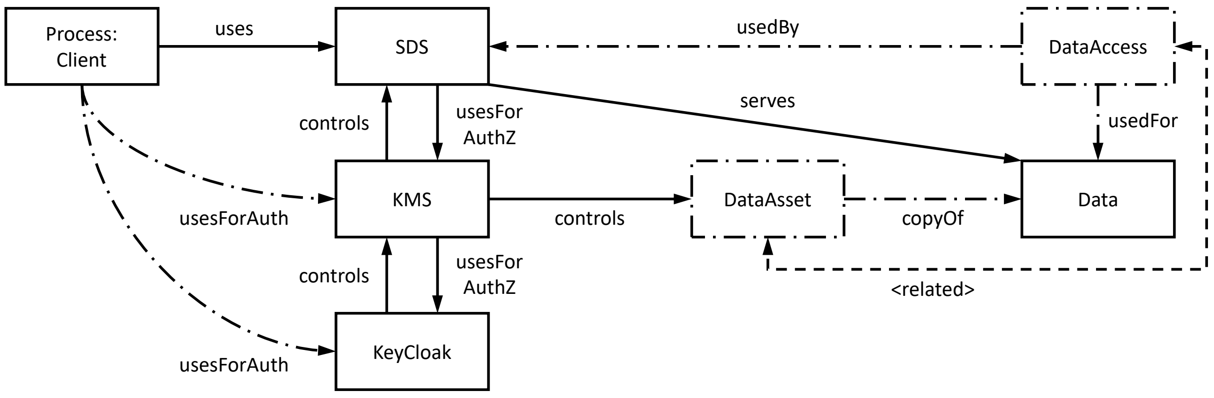


Figure 21. Modelling the SDS in CybeKit4SME

Much of the structure here will be inferred from the 'SDS-serves-Data' relationship, and those between the SDS and Data and their Hosts. If they are on different hosts, then 'KMS-controls-DataAsset' may also be inferred via the pattern from Figure 20.

In this situation, the usual threats to the serialized DataAsset do not apply, but an attacker could get access by compromising the KMS. An attacker who takes control of the SDS would not be able to access the data because the SDS is acting for a client of the KMS. The attacker could use the rights of the SDS to connect to the KMS but would be unable to get an Authorization Code expressing the rights of the client.

The attacker could access data if they can compromise the Client or impersonate the Client to both the SDS and KMS (by impersonating the Client to KeyCloak). This would not give the attacker direct access to the DataAsset used by the SDS, but they would be able to get an Authorization Code and submit requests to the SDS, gaining access to DataFlows between the SDS and the Client. If PME is used, data fields stored in the DataAsset will be encrypted by different keys, and the Client's Authorization Code specifies which keys can be retrieved by the SDS. If the attacker also has control of the SDS, they still cannot access data not contained in the DataFlows to or from the Client, so control of the SDS still poses no additional threat. To model which data is vulnerable and which is still protected, it is necessary to extend the baseline model of data.

II.5.1.b.4. Data structures and PME

To model fully the implications of PME, it is necessary to extend the model of data to include the presence of distinct elements within a Data class. Each such element can now be represented by a new Field asset, and a new relationship Data-containsField-Field used to asset that the Field is part of the data represented by the Data class. In principle, the same Field may arise in more than one Data asset.

The model assumes that a Data asset contains elements, of which zero or more may be asserted as Fields. The Data asset then represents the rest of the Data, so it is only necessary to assert the presence of a Field if it has distinctive properties, i.e., different impact levels if compromised, different relationships to Processes, or being present in more than one Data asset. This means that if a Data asset does contain Fields:

- impact levels for compromise of individual Fields should be added to the Fields;

- impact levels for compromise of the Data should ignore those impacts – so the level indicates the impact of a compromise only on Data not contained in any of its Fields.

Encryption controls are still associated DataCopy or DataFlow assets, including a control type for DataCopy assets indicating the use of PME. If this is present, it is assumed that each Field is encoded using a distinct cryptographic key.

Process-Data relationships encoding the way processes handle (serve or process) data are unchanged, and by default they apply to all elements within the Data. However, if the Data contains a Field, a new Process-ignoresField-Field relationship may be used to indicate that the Field is not handled by the Process. If a Process ignores a Field, then two further assumptions are made:

- the Field will not be included in serialized DataAssets used by that Process (unless required by some other Process accessing the same serialized copy of the data); and
- if PME encryption is applied to such a DataAsset, the Process will not have access to the key for the ignored field.

To determine which Fields are present in which serialized copies of the Data, the following inference sequence is used:

1. Static stored copies of the Data (those not produced as output by any Process) are assumed to contain all the associated Fields.
2. Stored copies of the Data produced as output by a local Process contain Fields not ignored by that Process.
3. Data flows produced as output from a Process contain Fields not ignored by that Process.
4. Where data flows from a producer Process to a stored copy, the data flows and stored copies contain Fields not ignored by the producer Process.
5. Where data flows from a stored copy to a consumer Process, the data flows contain Fields not ignored by the consumer Process.

Steps (4) and (5) are handled by iterative construction rules: step (4) determines which Fields could be present given the sources of the Data, and step (5) determines which of these are present given the destinations. Because (5) terminates at stored copies of the Data, the stored copy will contain a Field if it can be supplied by a producer, but this will be omitted from data flows served to consumers that don't need it. This matches the behaviour of SDS, where if a client is not permitted to access a Field, it causes an error, so the Field is never transmitted (even in encrypted form) to a consumer process that does not require it.

Threats to Fields then arise if they are contained in compromised DataAssets:

- if the confidentiality of a DataAsset is compromised, so are all Fields in the DataAsset;
- if the authenticity/integrity of a DataAsset is compromised, so are all Fields in the DataAsset;
- if the availability of every DataCopy asset containing the Field is compromised, the availability of the Field is compromised.

If a DataAsset is encrypted using PME and served by an SDS, the DataFlows between the SDS and each of its Clients will only contain Fields to which the Client has access. As discussed above, while an attacker with control of the SDS or the data Host would retrieve the serialised DataAsset, they would still only have access to the same Fields. Thus the combination of key management by the KMS, control over client access to the SDS by the KMS, and the modelling of fields and inference as to which are present in each serialized copy of the Data provides an effective model of protection by PME.

II.5.2. GDPR

II.5.2.a. Baseline knowledge base limitations

The baseline Spyderisk knowledge base contains so-called ‘compliance threat’ models corresponding to requirements specified in several articles from the General Data Protection Regulation.

A compliance threat is one that has no causes or effects and no likelihood. It is either present or not, and if present it represents a potential non-compliance with respect to some requirement. This may be a legal requirement imposed by a law or regulation (like the GDPR), a voluntary obligation arising from some declared or undeclared policy, or an aspiration to use a defined set of best practices. The threat has no likelihood, but it may have control strategies representing measures that may be used to meet the compliance requirement given that the potential for non-compliance exists.

The baseline Spyderisk knowledge base focuses on aspects of the GDPR that are directly related to system implementation choices. Since the knowledge base includes physical locations, this includes where the system is implemented (under what jurisdictions) and how it is implemented (with what security or security-related features). This led to the following GDPR compliance threats:

- Article 6: a threat representing a situation where data related to a data subject is processed within the system, with control strategies representing:
  - processing by consent, with controls representing the inclusion of a suitable interface for the subject to give or revoke consent, plus logging data access;
  - processing to protect vital interests, with controls denoting that the process is critical (to life), plus logging access;
  - processing under contract, to fulfil a legal obligation, in the public interest or in pursuit of legitimate interests, with controls denoting this, plus logging access.
- Article 8: a threat triggered by the controls for processing by consent, representing the possibility that the subject is a child, with a control strategy representing:
  - a means to seek parental consent and block processing if this is not obtained.
- Article 9: a threat triggered by vital interest controls when processing special categories of data, representing the need for consent if the subject is able, with a control strategy for:
  - checking if it is possible to obtain consent from the data subject, and having a means to bypass the need for this should the subject be unable to express consent decision;
- Article 9.4: a threat arising if genetic, biometric or health data is transferred across a border, with a control strategy representing:
  - checks on laws and regulations both sides of the border to ensure that the transfer does not breach extra conditions imposed on such data by Member States.
- Article 44: threats arising when data is transferred to or stored in countries not subject to the GDPR, with control strategies representing the presence of adequate safeguards.

Because of the issues with agreements at government level (like Safe Harbour and Privacy Shield), the control strategies addressing the Article 44 threats use controls that apply to the organization operating the assets used to receive or store personal data.

The main limitations of the baseline GDPR model fall into two areas:

- only a subset of articles of most concern to system implementers were covered, and
- usability was poor (see below).

It was decided that extending coverage to more GDPR articles that relate to organizational policy and governance is beyond the scope of CyberKit4SME. It is not possible to do this without also extending the knowledge base to model organizational structures and policies far beyond the IT systems used. It was also considered unfeasible to capture GDPR requirements in these areas sufficiently well that an SME could avoid the need to take legal advice, so the value of such extensions would be low.

The focus was therefore to improve usability of the existing GDPR models that are related to the IT systems. In that regard, the main concerns were:

- the threat models depend on users making relevant assertions which may be overlooked, e.g., the need to include the data subject and their relation to data in the system model;
- the threat models generate false positives which may be confusing to Spyderisk users.

The problem of false positive threat detection was especially problematic where threats are found for data storage or processing under the control of the data subject. These threats are very clearly nonsensical, so they may also undermine confidence in the output of a Spyderisk analysis. For example, the threat used to model Article 6 is shown in Figure 22:

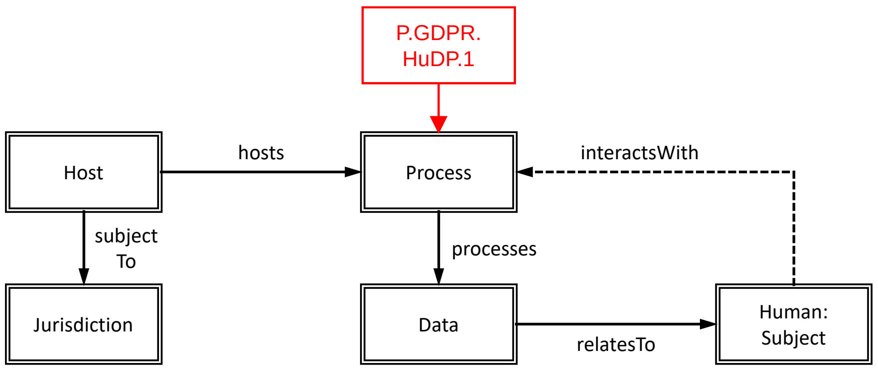


Figure 22. Baseline compliance threat for GDPR Article 6

At that time, this threat was considered adequate. The ‘interactsWith’ relationship is a prohibited link, ensuring that the threat would not be detected where the data is being processed interactively by the data subject. However, the true situation is more complex, as data may be stored remotely from the processing and controlled by someone else, so consent may still be needed. To cover these scenarios, a more complex threat pattern is needed, and more sophistication is needed to exclude false positives without creating false negatives (the threat not being detected when it is applicable).

II.5.2.b. CyberKit4SME GDPR Extensions

II.5.2.b.1. Detecting modelling oversights

The first area of improvement was to introduce modelling errors to the knowledge base. A modelling error is a special type of compliance threat to detects a potential non-compliance with requirements imposed by the knowledge base itself. Two types of modelling error threats are used:

- Modelling errors with no control strategies: these signify the presence of an unresolvable inconsistency or ambiguity in the model of related system assets.
- Modelling errors with control strategies: these signify the presence of an ambiguity or a potential oversight.

The only way to address the first type of modelling error is to change the arrangement of assets and relationships used to model the system to be analysed. Most cover impossible situations like a stack of virtual machines with circular provisioning relationships, or processes communicating when there is no network path between their hosts. This type of error is not normally related to a regulatory compliance issue.

The second type of modelling error can be addressed by using control selections to signal how to resolve the ambiguity. The first GDPR model enhancements were two such modelling errors to detect where Spyderisk users may have forgotten to specify a relationship between personal data and a data subject. These are shown in Figure 23 and Figure 24:

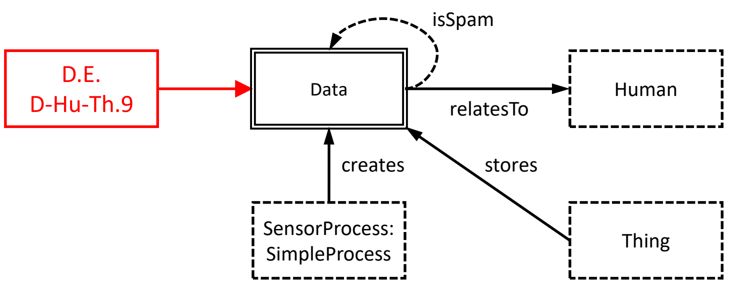


Figure 23. Detecting a missing relationship between personal data and the data subject

Dotted lines signify that an asset or relationship is prohibited so the threat will not be created if they are present. The only required asset is the Data asset, and the threat is triggered if there is no Human to which it is related. The prohibited isSpam relationship suppresses this threat when the Data is an inferred flow of spam from an email client. The other two prohibited assets in Figure 23 prevent the threat being created if the data is an inferred asset associated with an IoT Thing. This case is covered by Figure 24, so must be excluded from Figure 23 to prevent duplicate threats being created (which would degrade rather than enhance usability).

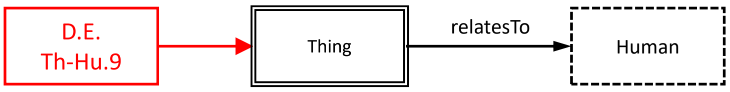


Figure 24. Detecting a missing relationship between a personal IoT device and the data subject

In Figure 24, the data associated with the IoT Thing is inferred to exist, so it won't be visible in the Spyderisk system-modeller user interface. The threat is therefore formulated in terms of the Thing, making it more accessible and user-friendly.

In both cases, the control strategy contains an 'artificial' control whose purpose is to tell the Spyderisk system modeller that the data is impersonal. Strictly speaking, these two threats are not part of the GDPR model because they are relevant to any regulatory framework dealing with personal data. However, they ensure that the GDPR model threats will be generated when appropriate, by signalling to the Spyderisk user that they must specify the relationship to a data subject or specify that the data is impersonal.

II.5.2.b.2. Article 5: need for a data controller

The baseline knowledge base includes some GDPR threats in which the data controller is included as a stakeholder. The data controller is responsible for ensuring compliance with GDPR Article 5, and for implementing data protection measures under Article 24, implying that there must be a controller.

Two threats were added, which work in the same way as the modelling error threats described above, to check that there is a data controller for either storage or processing of personal data. For brevity, only one is shown, related to the controller for data storage:

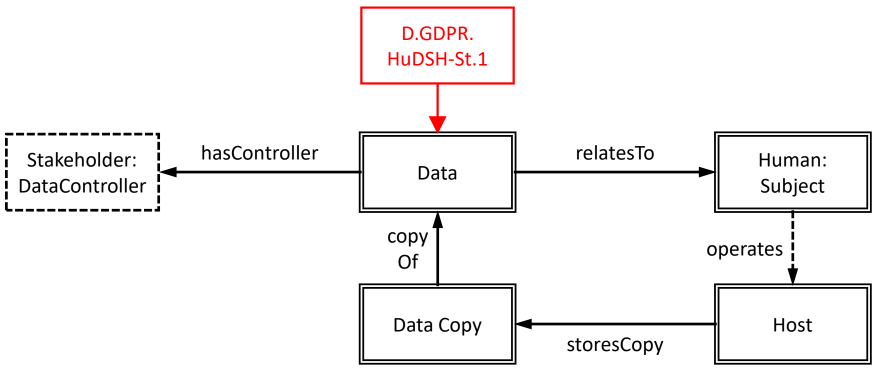


Figure 25. Detecting a missing data controller for personal data storage

Note that this could have been classified as a modelling error, but if GDPR does not apply then there would be no error. Although this threat functions in the same way as a modelling error, it is in fact a compliance threat expressing a GDPR compliance requirement.

II.5.2.b.3. Avoiding false positives

The main area of concern was false positive detection of threats, especially threats representing potential non-compliance with Articles 6, 8 or 9. Their control strategies expressing requirements for technical measures, and so directly affect system design and configuration choices. The main cause of false positives was the failure of the threats from the baseline knowledge base to adequately capture the involvement of the data subject in the storage or processing of their own data.

One improvement is visible in Figure 25 – the inclusion of a prohibited 'operates' relationship between the data subject and the host where the data is stored. This ensures that the missing data controller threat is not triggered where the data subject is in control of their data. The same patterns were used in threats to detect an unspecified jurisdiction where personal data is stored or processed, and the extra prohibited relationship was initially added to suppress false positive detection of those threats, and now applies to the missing data controller threats.

For Article 6, 8 or 9 the situation is more complex because there isn't a single host or process involved. Once this was added to the baseline Article 6 threat shown in Figure 22, it became impossible to avoid false positives using a single threat. During CyberKit4SME, the threat was split into three subcases.

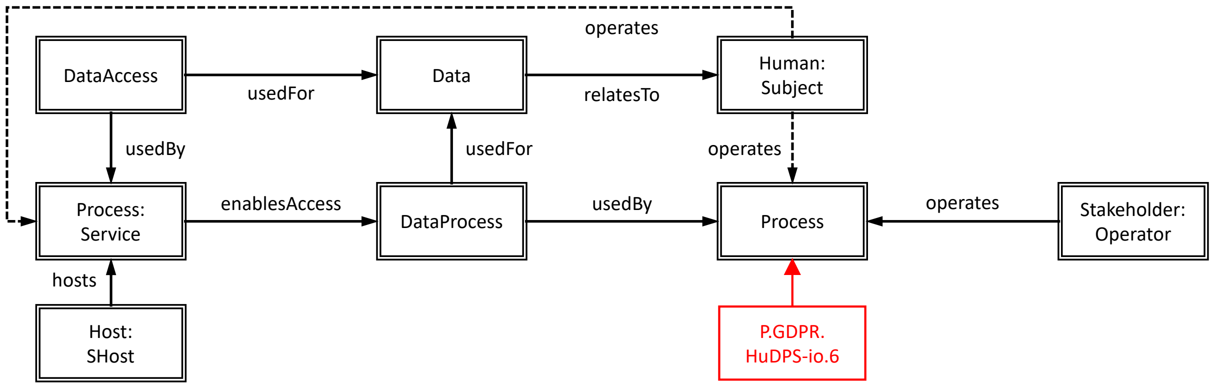


Figure 26. GDPR Article 6: Data subject not involved in personal data processing.

In Figure 26, the data subject is not operating the process using their data, nor the process controlling access to their data. Article 6 applies so a legal basis must be found, one of which is consent by the subject. The simple Process-Data relationship has been replaced by a DataProcess asset (a type of DataAccess representing deserialized data – see above). This is used because it has a relationship with another process (usually a data service) that controls access to serialized data. This construction makes it possible to cover all possible cases: data read from storage on the local host using the rights of the process on its host, data from remote storage arriving in a data flow regulated by a data service, data created remotely and sent to the process but never stored, including in each case whether a key managed by another process is needed.

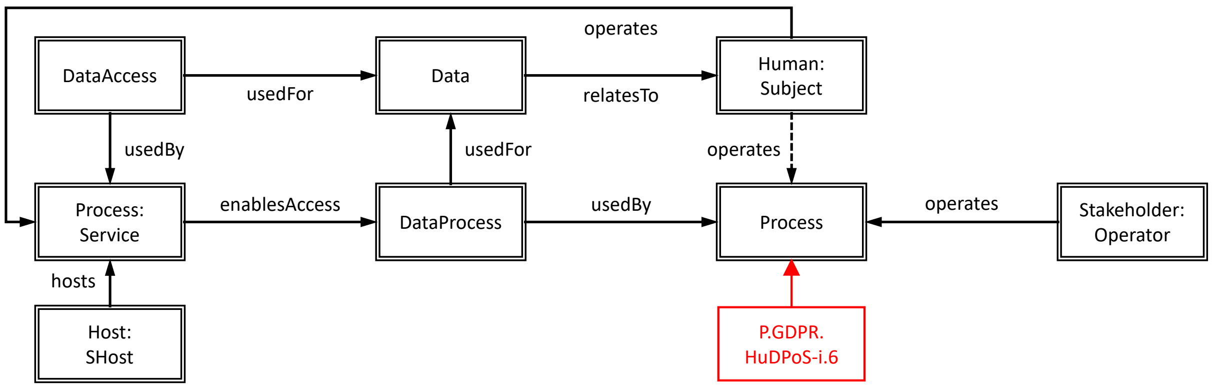


Figure 27. GDPR Article 6: Data subject controls access to their data

In Figure 27, the data subject operates the process that controls access to the data, but not the process that is consuming the data. In this situation, consent is required for the processing, but the technical measures needed to implement this are a little different because some are covered by the fact that the subject regulates access to the data.

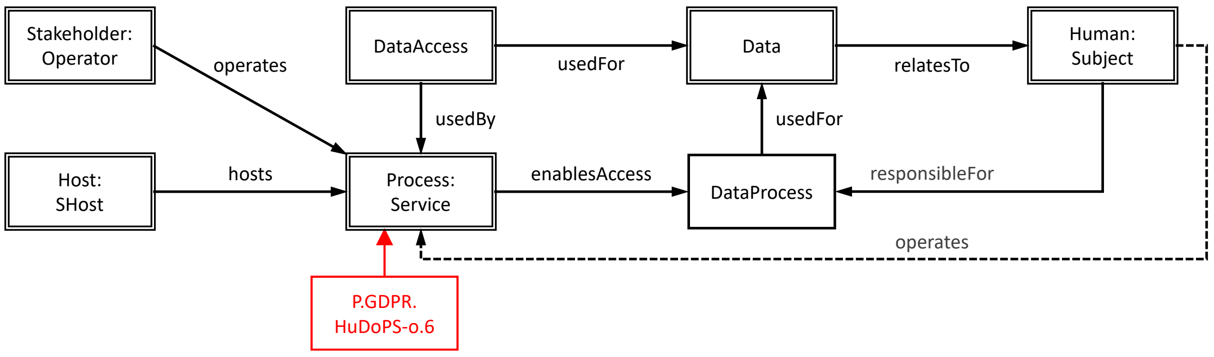


Figure 28. GDPR Article 6: Data subject controls the process using their data

In Figure 28, the data subject operates the data processing but not the storage and access control for their data. The operator of that service needs consent to do this, but again, some of the technical measures that would be needed in Figure 26 are covered by the fact that the data subject is running the process using their data.

The remaining possibility is that the subject operates both the data storage and access control, and the data processing. In that situation, the data subject is the data controller, so no specific measures are needed to comply with GDPR Article 6, so no need to include a threat for this.

Similar improvements were made to the threat models for GDPR Article 8 (triggered when the legal basis under Article 6 is user consent) and Article 9 (triggered when the legal basis is vital interests).

Finally, false positive threats can also be generated where the system spans international borders, and some users are neither citizens nor residents of the EU. This may mean the GDPR does not apply to their data, so any GDPR threats would be false positive compliance threats. To prevent this, all GDPR threats were converted to ‘triggered’ threats where the trigger is a control signifying that the data subject is protected by the GDPR (as an EU citizen or resident).

The only problem with this is that a Spyderisk user may forget to select the control to indicate that a data subject is protected by the GDPR. To avoid that possibility, a new modelling error threat is used for every subject of personal data. The modelling error is itself addressed by any control that asserts the applicable regulatory system. In this way, the Spyderisk user is alerted (by the modelling error) of the need to specify (by selecting a corresponding control) which regulations apply. Once this is done, the relevant set of regulatory compliance threats is generated. At present, the only supported set of regulations is the GDPR, so the modelling error threat has only two control options: GDPR or 'none'. In future, it will be possible to introduce alternative regulatory models, and even model cases where more than one set of regulations may apply.

II.5.3. AI Privacy

The addition of the task around AI Privacy in WP5 led to discussions between IBM and IT Innovation around how to incorporate AI privacy into the Spyderisk knowledgebase and thus into the system models developed by users. A cross-project meeting was also held between IBM, IT Innovation, SBA Research and The European Institute for Innovation through Health Data (IHD):

- IT Innovation, IBM and SINTEF are working in the NEMECYS[23] project to look at risk analysis of connected medical devices where data privacy is paramount.
- SBA, IHD and IT Innovation are working together in the SYNTHEMA[22] project to look at risk analysis of synthetic medical data with SBA developing quantitative metrics to measure the privacy properties of synthetic data (often generated by generative AI systems). IHD lead the data governance work.

In both projects, IT Innovation are further developing the Spyderisk software and knowledgebase.

As incorporating consequences and threats around AI privacy into the knowledgebase was not part of the original plan we decided that work on this topic could not be resourced in CyberKit4SME and extensive work is now underway in both NEMECYS and SYNTHEMA around this, with collaboration between IBM and SBA anticipated via open source software.

II.6. Sustainability

To promote the sustainability of Spyderisk, we have begun creating an open community around the software, tools and methodology. Open sourcing is the first step, but this is more than just open sourcing: the Spyderisk Open Project aims to increase trust globally of complex IT systems through open community, governance, tools and knowledge supporting responsible automated risk assessment. We aim to democratise risk assessment by creating and distributing knowledge and tools suitable for education and industry including SMEs.

More information on the approach can be found in D6.7[26] but the activities are reported here.

II.6.1. Open Source

We have created the Spyderisk organisation on GitHub and open sourced the following repositories:

- system-modeller[27]: Spyderisk web service and web client
  https://github.com/Spyderisk/system-modeller
- system-modeller-deployment[27]: Deployment scripts for Spyderisk System Modeller
  https://github.com/Spyderisk/system-modeller-deployment
- system-modeller-adaptor: Service sitting in front of the system-modeller, providing additional (sometimes experimental) functionality
  https://github.com/Spyderisk/system-modeller-adaptor

[26] D6.7: 2nd report on exploitation actions, final results and sustainability plan

[27] Initially open-sourced in a UKRI Trustworthy Autonomous Systems project.

- system-modeller-docs: Documentation for the Spyderisk System Modeller
  https://github.com/Spyderisk/system-modeller-docs
- plot-attack-graph: Tool to plot attack trees of Spyderisk system models
  https://github.com/Spyderisk/plot-attack-graph
- domain-network[27]: Network domain model
  https://github.com/Spyderisk/domain-network
- domain-csv2doc: Tool to convert from a domain model's CSV serialisation to a documentation website
  https://github.com/Spyderisk/domain-csv2doc
- domain-csv2nq: Convert from domain model CSV source files into NQ artifact
  https://github.com/Spyderisk/domain-csv2nq
- Access-Tool: Tool to create a plain text dump of a Microsoft Access database
  https://github.com/Spyderisk/Access-Tool
- training-risk-assessment: General course on risk assessment
  https://github.com/Spyderisk/training-risk-assessment
- training-spyderisk: Hands on training course for Spyderisk
  https://github.com/Spyderisk/training-spyderisk

The software is all licensed using the permissive, OSI-approved, Apache 2.0 licence and documentation is covered by Creative Commons. The work of open sourcing has included tidying up the repositories, updating 3rd-party dependencies, developing new continuous integration systems, clarifying the embedded documentation and adding meta docs such as a code of conduct for contributors.

II.6.2. Open Documentation

Documentation is key for users and future collaborators:

- The Spyderisk user documentation has been updated, open-sourced, linked to in the software, and published on the website:
  https://spyderisk.org/documentation/modeller/latest/
- Documentation can be generated automatically from the knowledgebase using the csv2doc tool (see tool list above), and documentation for the network knowledgebase has been generated and published on the website:
  https://spyderisk.org/documentation/knowledgebase/network/v6a3-1-4/
- An overview paper has been written and submitted to the open access journal IEEE Access. A pre-print is available:
  https://www.techrxiv.org/articles/preprint/Automated_Knowledge-Based_Cybersecurity_Risk_Assessment_of_Cyber-Physical_Systems/24061590
- A wiki is under development:
  https://wiki.spyderisk.org/

II.6.3. Open Training

We have been developing free and open training materials for CyberKit4SME and beyond:

- Two training courses are publicly available using the open training material published on GitHub: one on general risk assessment and one hands-on tutorial for Spyderisk:
  https://training.spyderisk.org/courses
- A deployment of Spyderisk has been made available to support the training course:
  https://training.spyderisk.org/system-modeller/

The training courses are being promoted in the forthcoming Professional Skills Hub of the University of Southampton Electronics and Computer Science school: a digital resource for undergraduates to develop real-world, domain-specific skills and knowledge. We aim to build on this educational material in the future to help grow the open community around Spyderisk.

II.6.4. Open Tooling

The “Access Tool” mentioned above was developed in the CyberKit4SME project to support our open project goals. It supports a transition away from the proprietary Microsoft Access database currently used to develop the Spyderisk knowledgebases.

Microsoft Access is an obsolete visual application builder tool, still used fairly widely because many organisations rely on legacy Access applications. First released more than 30 years ago, Access was not designed for a modern multiuser world with software source code maintained in Git, and networked SQL databases, or even the internet. Legacy Access applications are increasingly difficult to maintain.

Ideally Access applications would be translated to some other open source system that does not rely on Access, or Microsoft Windows, or any Microsoft products at all. This is a problem that many other people have tried to fix, so we broke it down.

Instead of a single-user untrackable GUI point-and-click development environment, the tool turns Access applications into diffable plain text files that can be stored in Git, and put online using services such as SourceHut, Codeberg or GitHub.

After the files are retrieved from the Git service using a command such as git pull, the tools can then be used to load them into Microsoft Access, forming a complete GUI application in the normal Access way. Microsoft Access and Microsoft Windows are still needed to run the source code, but we can now see what needs to be done to remove them.

In the meantime, Access development has become repeatable, using standard development tools.

The System Modeller Deployment project creates a deployment of Spyderisk using Docker containers. It orchestrates the system modeller, MongoDB, a reverse proxy, the system modeller adaptor and (optionally) Keycloak. It has been reworked to provide two options:

1. a deployment including an insecure version of Keycloak, suitable for deployment on a personal computer with no external access or in a test environment; and
2. a deployment which uses a pre-existing Keycloak deployment, suitable for deployment in a production environment.