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, Chloe Yong

Reviewed by Stefano De Angelis (UoS Cyber); Tiberiu Cocias (Elektrobit)

Date: 2021-10-15 (original document); 2024-09-24 (extract)

Background and Explanation

This document is an unedited extract of deliverable D4.1 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 other CyberKit4SME project partners. 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 (previously known as “SSM”), 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
ESP	Extensible authentication protocol
EVTX	Microsoft Windows event log format
GDPR	General data protection regulation
HDFS	Hadoop distributed file system
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

Acronym	Definition
LAN	Local area network
LDAP	Lightweight directory access protocol
ME	Medium enterprise
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
PSK	Pre-shared key
REST	REpresentational State Transfer
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
SOC	Security operation centre
SQL	Standard query language
SSH	Secure shell
SSM	System Security 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

I. Risk Analysis Models and Tools

I.1. Background

The System Security Modeller (SSM) provides a graphical interface through a web browser for a user to perform a risk-assessment of a complex information system. The tool identifies the threats to the system (including threat cascades and attack paths) and computes the likelihood of the threats. It combines these with the specified business impacts to calculate risks, and provides the user options of what security controls to add to the system to reduce the risks to an acceptable level.

The initial research for this tool was carried out in the FP7 SERSCIS project and continued in H2020 5G-ENSURE (where it was used to analyse trust relationships in 5G mobile networks). The system was further developed in H2020 SHIELD and H2020 RESTASSURED where the shift to supporting the ISO 27005 risk assessment process was introduced. The Innovate UK ASSURED project brought in the concept of modelling compliance (e.g. with regulatory requirements). Recently, the system has been developed in the H2020 Fog Protect[1] and ProTego[2] projects to include aspects relating to IoT and Fog computing and alignment with OWASP[3] and CVSS[4] (these projects are ongoing). Most recently the UK EPSRC-funded SPYDERISK Accelerator project has contributed to help create a ”minimum viable product” (MVP) for commercialisation, adding user-facing improvements rather than fundamental changes.

I.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. The SSM's long history has given it a wealth of capabilities but the research to date has not been focussed on simplicity, which is itself a big challenge and about more than just having an attractive interface.

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.

Various aspects of the SSM are being addressed in CyberKit4SME to meet this goal such as helping the user build a model, making attack paths easier to understand, helping the user understand where to optimally place security controls, reinstating an analysis of GDPR compliance, and adding the ability to model complex systems such as Kubernetes.

---

[1] H2020 Fog Protect: https://fogprotect.eu/

[2] H2020 ProTego: https://protego-project.eu/

[3] OWASP: https://owasp.org/

[4] CVSS: https://nvd.nist.gov/vuln-metrics/cvss

In addition, this work combined with the SIEM described in Section erro**Error! Reference source not found.** addresses the risk analysis aspects of 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, 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 the SSM and the SIEM (Keenaï) are being developed to meet this goal.

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

I.3. Overview

An installation of the SSM service is intended for use by multiple users from the same organisation (or their guests). Each user accesses the service via a web interface, logs in and authenticates using the open-source identify management service KeyCloak[5] (which may be synchronised or federated with other authentication systems such as Open LDAP or Active Directory).

The SSM provides an environment for users to create and analyse models of their information system(s) but also essential model management features in the SSM dashboard, as shown in Figure 1.

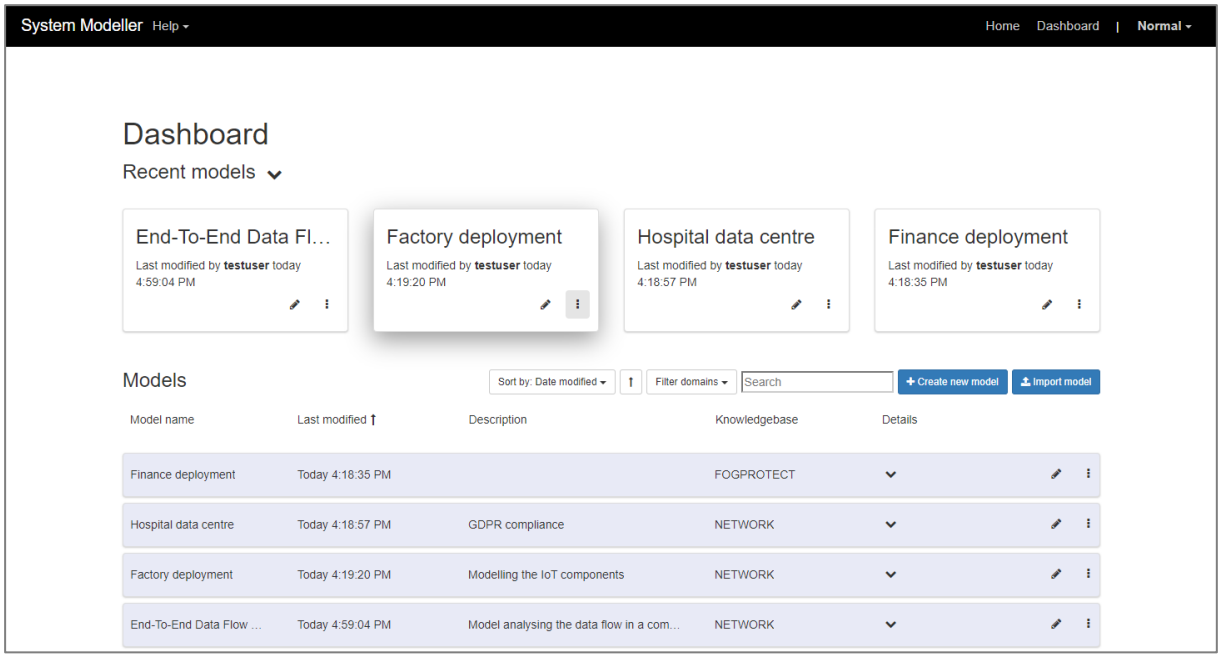

Figure 1. SSM dashboard for managing a user's models.

In the dashboard, user can see their list of models, with recent models highlighted in “cards” at the top. Models can be uploaded, created, deleted, renamed, shared and opened.

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

When a model is opened, the modelling and exploration interface is shown. Multiple models may be opened in different browser tabs. An example system model is shown below in Figure 2. It is drawn in the web interface using drag and drop. A model consists of a set of “assets” such as hosts, processes, data, networks, etc, linked by typed “relationships”: a Server asset may “host” an Application Process asset for instance.

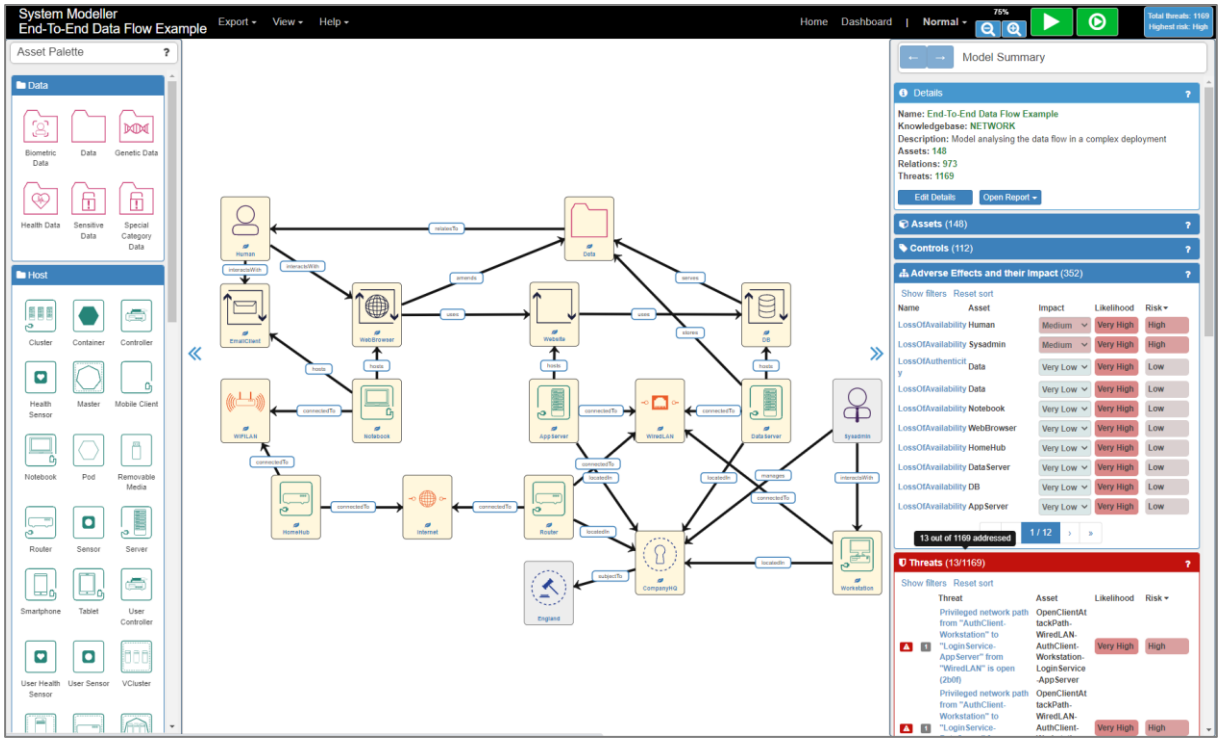

Figure 2. Example SSM system model as viewed in the modelling interface.

To complete the system model, assets must be linked by relationships which have a start and end. Some pairs of asset types do not have any possible link, some have just one and some have many options which are presented to the user for them to choose between upon making the connection.

The large number of (sometimes quite similar) asset types and relationships that the user can choose amongst means that the information system can be described very precisely and the correct threats found along with an accurate risk calculation. The drawback is the complexity for the user.

Once the assets and relationships comprising the model have been drawn, the SSM can find threats to the system. Figure 3 shows part of the list of 1169 threats to this example system and demonstrates how hovering over a threat highlight the assets involved as well as providing a more detailed explanation.

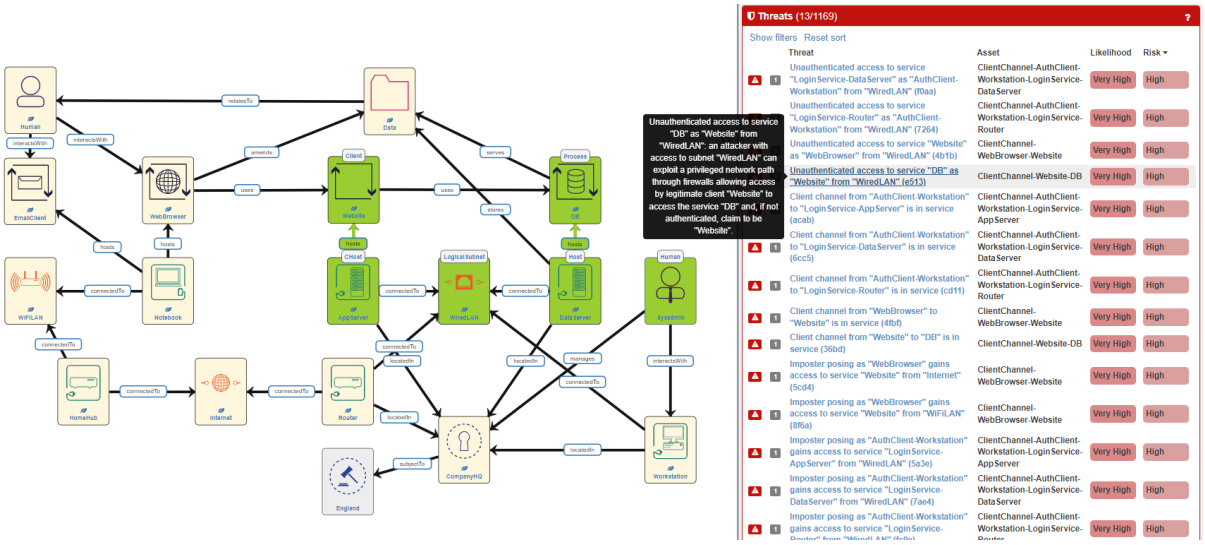

Figure 3. Threats to the system are listed (note, the screenshot also shows the result of the risk analysis).

The risk calculation is based on the threat likelihood and the business impact of any adverse effects caused by the threat. The factors used as input for the threat likelihood computation are the discovered attack paths through the system and the “trustworthiness” of the assets, or how likely they are to deviate from good behaviour. The default values are generally acceptable but of course an interface is provided for when the value needs modification, as shown in Figure 4.

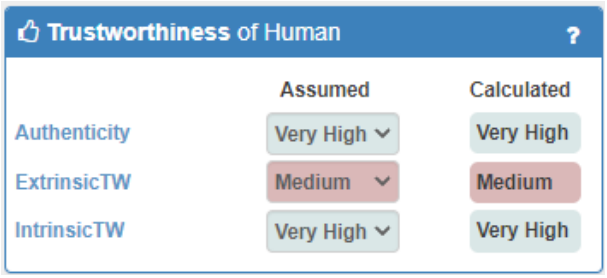

Figure 4. The interface used for adjusting the assumed trustworthiness of a "human" asset.

In the case of a Human asset, the various trustworthiness attributes have commonplace meanings:

- Authenticity: the likelihood that the person is not being impersonated.
- Extrinsic trustworthiness: the likelihood that the person will not be induced to do the wrong thing by an external influence (for instance a phishing attack).
- Intrinsic trustworthiness: the likelihood that the person will behave well in the absence of external influence. For instance, that they are free of malicious motives and unlikely to make errors.

All asset types, not just humans, have “trustworthiness” attributes – a concept that has been extended from everyday use.

The business impact of adverse effects must be specified by the user, again by adjusting sensible defaults. Generally, only the adverse effects on the primary business assets needs to be considered as the relationships between assets is properly considered. For instance, if the availability of a service such as a website was crucial to a business then this should be specified (commonly on the website data asset), but there is no need to then also specify that the process serving the website, or the host storing the website data must also have high availability as that is inferred.

Figure 5 shows the settings for a Data asset which in this example represents the files comprising a website. We can see that the impact of loss of availability of the data set has been set to “very low”, the likelihood has been calculated as “very high” and therefore the risk of this occurring is “low”. A standard matrix describing risk as a function of impact and likelihood is used with 5-point scales for all three factors.

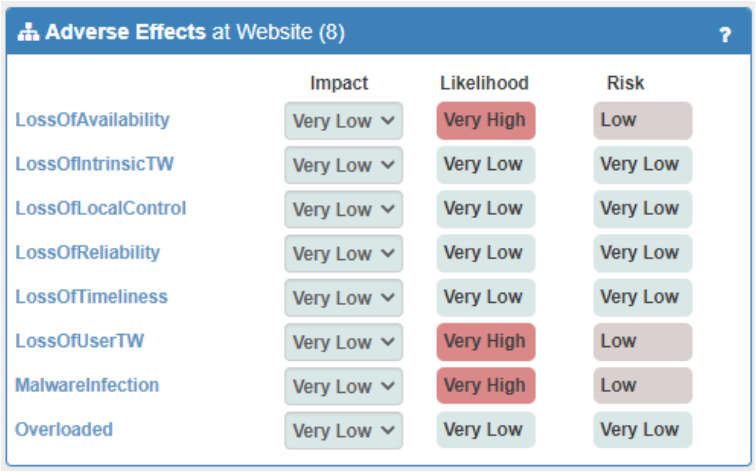

Figure 5. The settings for the various adverse effects that can affect a data asset.

Once the risks have been calculated, it is generally suggested to look at the highest risks across the whole system model. Figure 6 shows an example “top 10”.

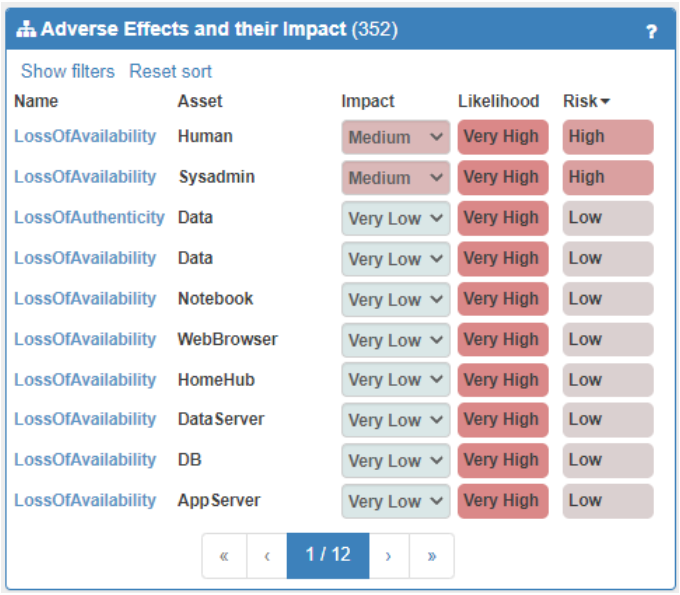

Figure 6. The top 10 adverse effects for the example system.

To examine the cause of the undesired adverse effects, and to understand where best to place what additional security controls, the “threat explorer” and “adverse effect explorer” are used. An example “adverse effect explorer” window showing probable causes of the loss of authenticity of the data asset can be found in Figure 7.

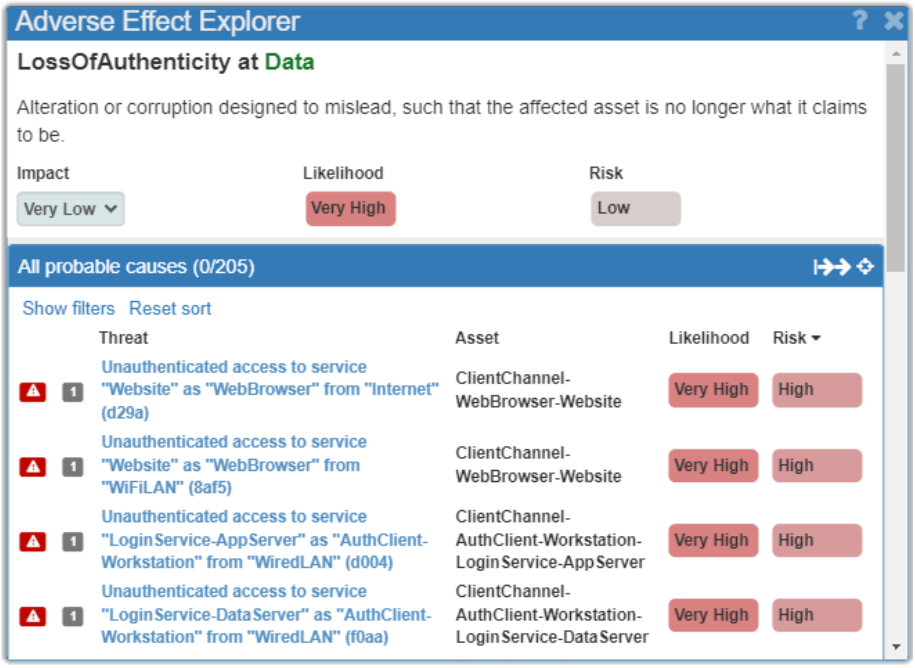

Figure 7. The "adverse effect explorer" shows which threats have caused a problem as well as the threats enabled by the adverse effect.

The “adverse effect explorer” lets the user explore all the threats that have some impact on the chosen adverse effect and provides various filters to aid exploration along the attack paths both preceding and following the adverse effect. The corresponding “threat explorer” opens when a threat is clicked on and fully describes the threat, the adverse effects both before and after it in the discovered attack paths and shows what security controls may be chosen to reduce the threat likelihood. Figure 8 shows an example threat which is a root cause of the loss of authenticity of the data asset, meaning that there is nothing that causes the threat other than what is expected from the external environment.

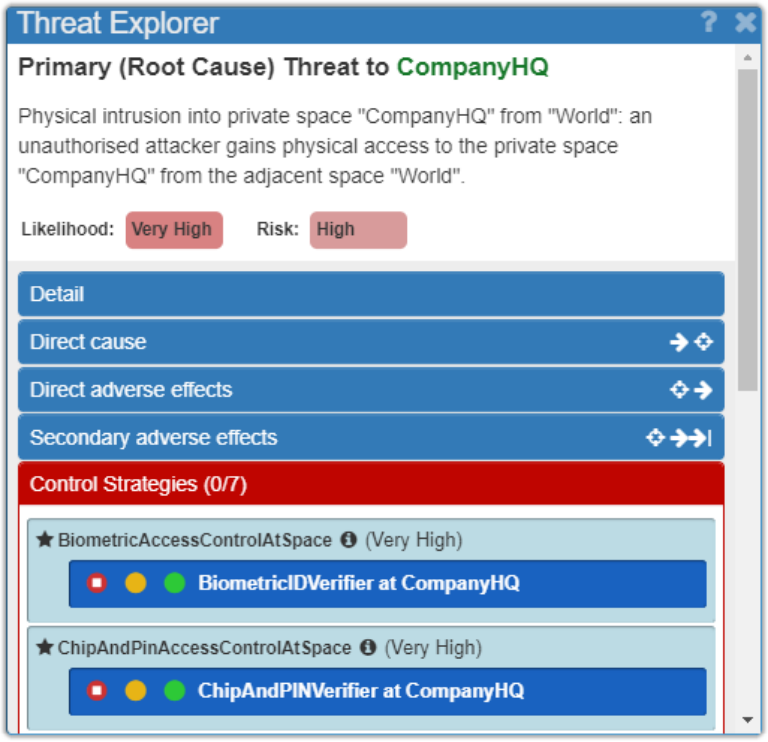

Figure 8. The "threat explorer" showing a root cause of the problem with the data asset as well as two of the possible security controls that may be added to reduce the likelihood of physical intrusion into the office space.

Once the user has explored the problems, added additional security controls and reduced the residual risk to an acceptable level, they may output two reports to help record the situation and implement the necessary changes: a “risk treatment plan” and a “technical report”.

To help the user in understanding the web interface, there are many tooltips provided as well as online documentation with various tutorials available from the “Help” menu. In addition, many locations include small help buttons (“?”) which take the user directly to the appropriate location in a reference document. An example is shown in Figure 9.

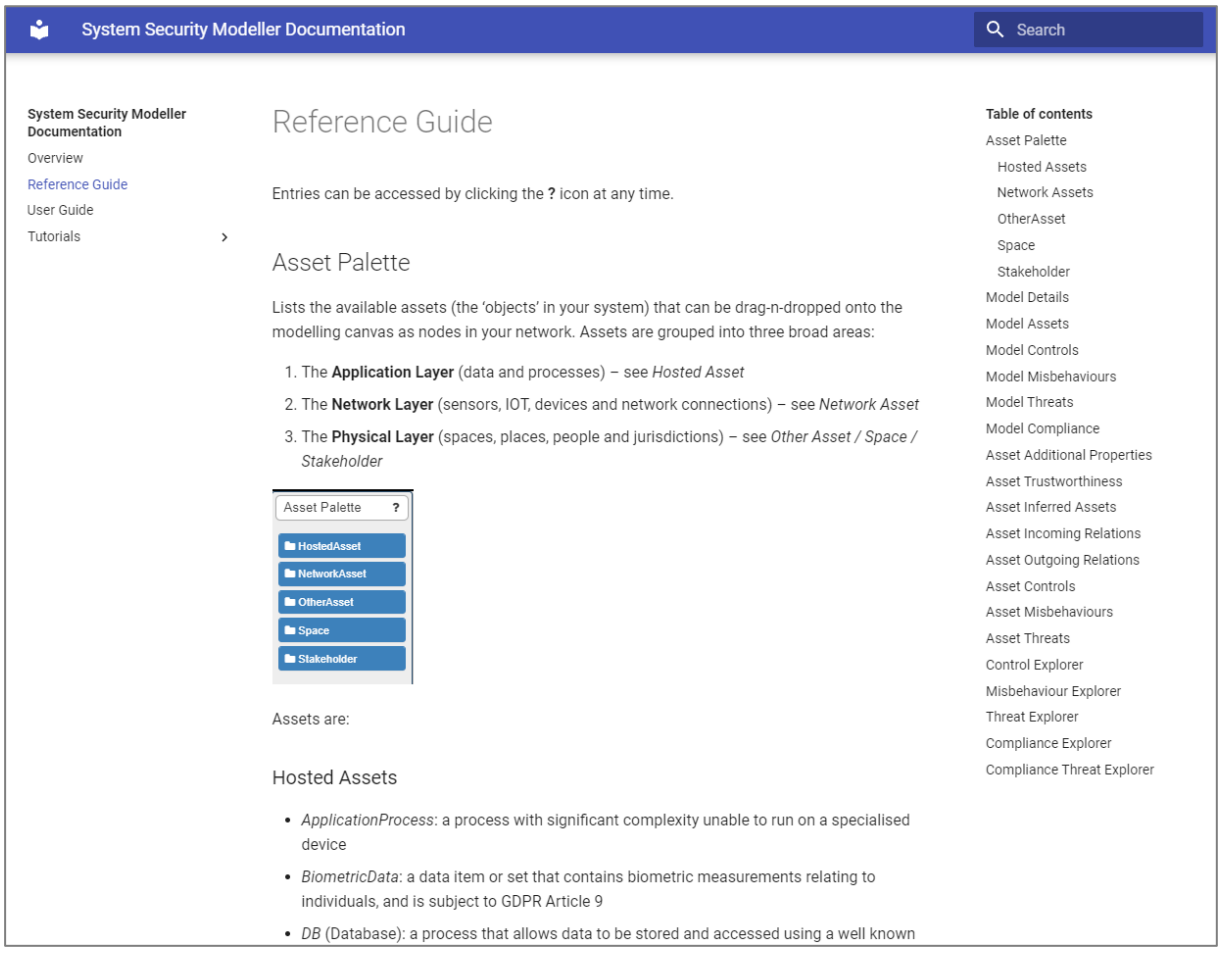

Figure 9. Online reference documentation for the SSM.

I.4. Architecture

The text below is reproduced from CyberKit4SME D2.2 "Technical Specifications" for convenience.

The SSM is a Java application with a REST API for client applications. The primary client application is a web-based user interface which utilises ReactJS[6] for a single-page application.

User authentication is handled by an external KeyCloak[7] component (using the OpenID Connect Authorization Code Flow). Users are identified as having a "user" role and in addition may have the "admin" role. The role(s) permit access by the SSM to different parts of the API accordingly. Authorization data is held by the SSM (not KeyCloak).

Data for the SSM is persisted in two different databases:

1. a MongoDB[8] database storing authorization data (who has what rights on what system model) and model meta-data (last edit date, etc);
2. a JenaTDB[9] database holding the knowledgebases of threats and controls (also known as "domain models") and the system models drawn by the users.

[6] ReactJS: https://reactjs.org/

[7] KeyCloak: https://www.keycloak.org/

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

[9] JenaTDB: https://jena.apache.org/documentation/tdb/

A high-level component architecture of the SSM can be found in Figure 10.

The SSM is deployed in containers using Docker Compose[10] or Kubernetes[11]. The application is split into four containers:

1. A reverse proxy (NGINX[12]) through which the client communicates. The proxy also hosts the user documentation website.
2. The SSM Java application deployed in Tomcat[13]. The service also serves the ReactJS client to the user's web browser. The JenaTDB database is hosted in the same container and communication with JenaTDB is done in-process.
3. KeyCloak deployed in JBoss[14].
4. MongoDB.

In addition, the prototype software to provide a risk analysis of an operational model is currently implemented (in Python[15]) in a separate micro-service called the "SSM-adaptor".

When deployed in Kubernetes, data persistence must be provided by Kubernetes persistent volumes.

A deployment diagram may be found in Figure 11.

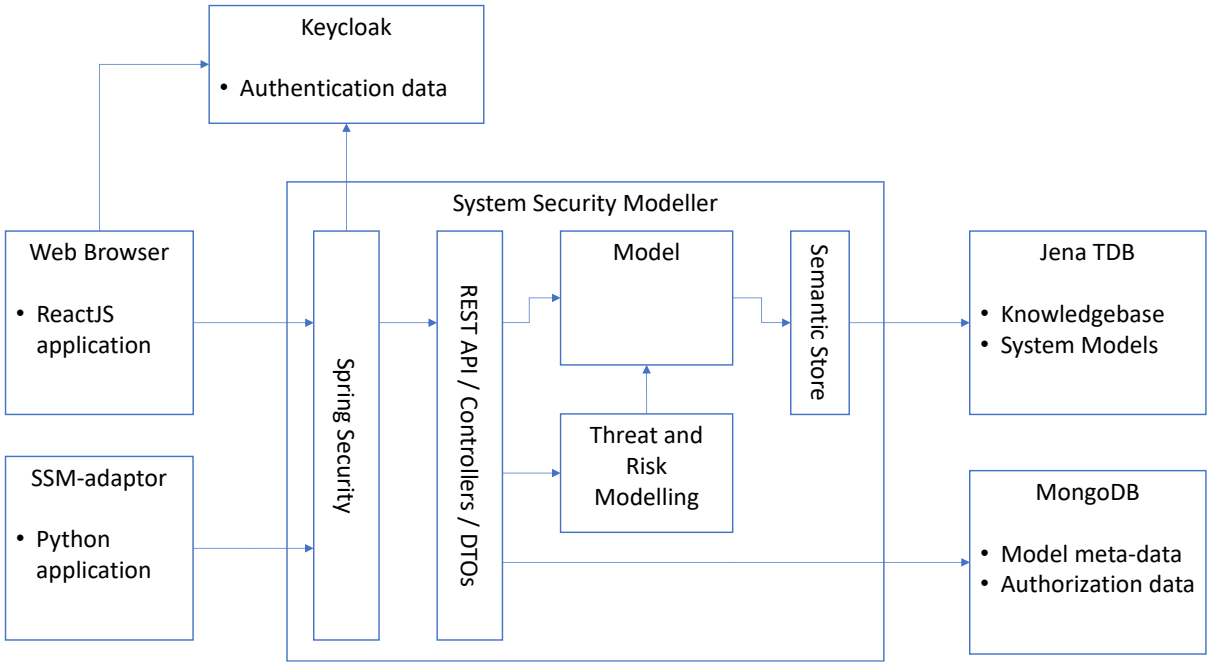

Figure 10. Component architecture of the System Security Modeller.

[10] Docker Compose: https://docs.docker.com/compose/

[11] Kubernetes: https://kubernetes.io/

[12] NGINX: https://www.nginx.com/

[13] Tomcat: https://tomcat.apache.org/

[14] JBoss: https://www.jboss.org/

[15] Python: https://www.python.org/

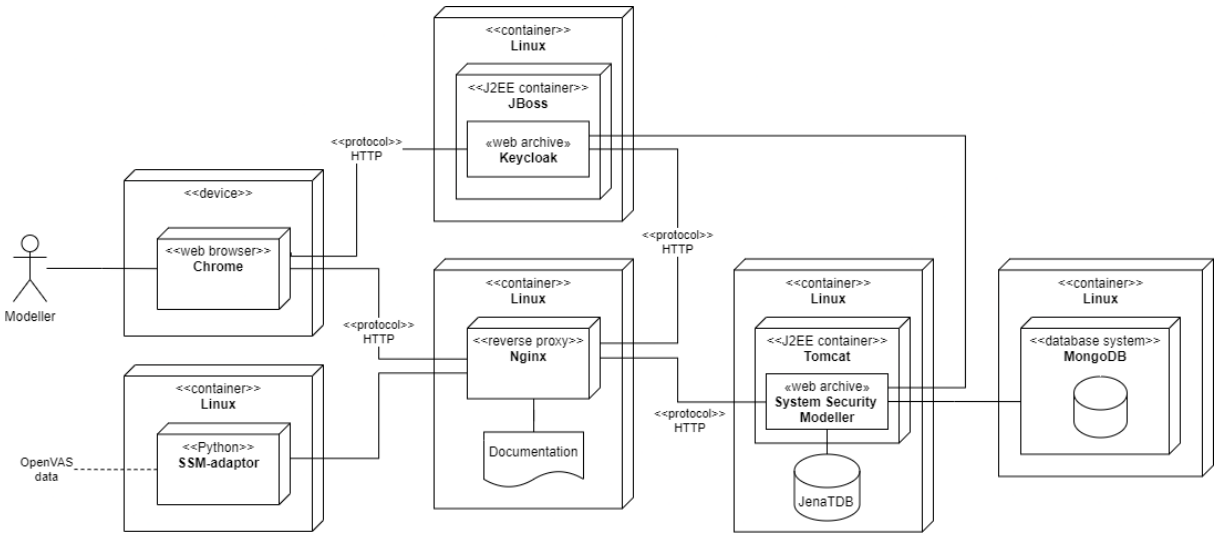

Figure 11. Deployment diagram for a single instance of the System Security Modeller.

I.5. Implementation details

The SSM comprises the software and its configuration. The configuration is variously known as the ”knowledgebase” or ”domain model” and defines the asset types, their potential relationships, the threats and security controls.

I.5.1. Software

The SSM software has three separable parts:

- The SSM service which is a Java Spring[16] application running in Tomcat.
- The SSM web user interface which is a ReactJS application.
- An ”SSM adaptor” written in Python, supporting more experimental ideas such as the adaptation of system models to current vulnerabilities (as found through OpenVAS) and automated control recommendation.

The SSM Service makes use of:

- KeyCloak for authentication.
- MongoDB for some authorisation data.
- JenaTDB for storing the knowledgebase and system models.

The focus on usability in the CyberKit4SME has led to work in this project on how to help the user build the initial model and how to help them navigate the results of the analysis. These two aspects are described below.

I.5.1.a. Model discovery

The SSM has been used in validation exercises in several research and commercial validation exercises and users have consistently had trouble building the initial system model. Sometimes this is because of a lack of understanding about which assets or relations to use (which we are addressing with better documentation and inference of missing parts) but when the user is trying to draw a model of an existing system another problem can occur: that they do not actually know what assets they have and how they are related.

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

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.

To address this issue, we have been investigating in CyberKit4SME what tools and techniques may be used to gather information on an existing system and convert that data into a partial system model. The task is complicated by the wide variety of IT systems in operation today, ranging from services deployed manually directly on servers, through software and infrastructure deployment tools, virtualisation of all kinds, to cloud systems and Kubernetes.

For the infrastructure (networks, hosts) we need to consider physical and virtual servers and networks both local and remote which may have been deployed manually, through some API in a cloud (known as Infrastructure as a Service, IaaS) or through some Infrastructure as Code (IaC) configuration system. Some companies (for example if following ITIL[17]) use a configuration management database or "CMDB" (such as ServiceNow[18]) to keep records of deployments. The use of "cloud" here encompasses on-premise cloud systems (such as OpenStack[19]), private remote clouds, public cloud, hybrid cloud or even cloud edge systems (e.g. AWS Outpost[20]). The various combinations along with potential sources of information are summarised in Table 1.

Table 1. Summary of deployment and query options for infrastructure.

	Physical (generally on premise)	**Virtual (on premise and cloud)**
Deployed manually	The classic "legacy" system where servers, switches and networks have been deployed in a machine room. Sources of information: network scanners; CMDB; SNMP.	Virtual infrastructure created manually on on-premise servers. Cloud infrastructure created manually through APIs or web interfaces (IaaS). Sources of information: network scanners; CMDB; the cloud's API.
Deployed through a tool (IaC)	N/A	Deployment is via generic tools such as Terraform[21] and Heat[22] or cloud-specific configuration definitions and APIs such as AWS CloudFormation[23]. Sources of information: network scanners; CMDB; the cloud's API; the IaC configuration files.

Software may also have been deployed manually or through some API or configuration file and may be hosted on any of the infrastructure options described above or even, in the case of function-as-a-service systems, not permanently hosted at all. There is also the complication of containerisation exemplified by Kubernetes. The options are summarised in Table 2.

[17] ITIL: https://www.itil.org.uk/

[18] ServiceNow: https://www.servicenow.com/

[19] OpenStack: https://www.openstack.org/

[20] AWS Outpost: https://aws.amazon.com/outposts/

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

[22] OpenStack Heat: https://wiki.openstack.org/wiki/Heat

[23] AWS CloudFormation : https://aws.amazon.com/cloudformation/

Table 2. Summary of deployment and query options for software.

	On physical hardware	**On cloud**	**On Kubernetes**
Deployed manually	The classic "legacy" system of software deployed ad hoc. Sources of information: network scanners; CMDB.	Ready-made services deployed through an API or web interface (Platform-as-a-Service, or PaaS). Sources of information: network scanners; CMDB; cloud API.	Possible but unlikely.
Deployed through a tool	Software deployed using e.g. Ansible, Chef or Puppet. Sources of information: network scanners; CMDB; software deployment configuration files.	Software deployed using e.g. Ansible[24]. A cloud provider's services or saved VMs can be provisioned using a cloud's provisioning configuration. Sources of information: network scanners; CMDB; various configuration files.	Kubernetes configuration (often generated from Helm charts[25]) defines what containers to deployed and how they are to be orchestrated. The containers can be off-the-shelf or built for the purpose (manually, or defined through a script, Dockerfile[26], Ansible etc). Sources of information: network scanners; CMDB; Kubernetes config; Kubernetes API; Helm charts.

The whole area is a multi-dimensional space of options and combinations and all will be found in practice. We are prioritising the investigation of what data can be gathered from network scanners (as they can be used in almost any situation) and from cloud APIs (as new deployments are increasingly put into cloud systems and they should provide detailed information).

A review of network scanning software suggests two primary targets: Nmap[27] and OpenVAS[28]. Nmap is a popular open source network scanner which scans IP addresses by sending packets to well-known ports to determine what services are available. It is able to provide information about the operating system and services running on a host. OpenVAS is primarily a vulnerability scanner: it attempts to discover what software processes are running and uses vulnerability databases to find what vulnerabilities may be present (according to the software and version found).

OpenVAS is maintained as open source by Greenbone Networks GmbH and is a fork of the previously open source Nessus[29] scanning tool which evolved into a close-source scanner

[24] Ansible: https://www.ansible.com/

[25] Helm chart: https://helm.sh/docs/topics/charts/

[26] Dockerfile: https://docs.docker.com/engine/reference/builder/

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

[28] OpenVAS: https://www.openvas.org/

[29] Nessus: https://www.tenable.com/products/nessus

maintained by Tenable Inc. Nessus is a very popular proprietary vulnerability scanner and it and others such as the Qualys[30] scanner are also potential integration targets in the future.

A prototype system for taking the XML output of an Nmap scan and drawing a diagram similar to the SSM system model was developed as a proof-of-concept. An example of the output can be found in Figure 12.

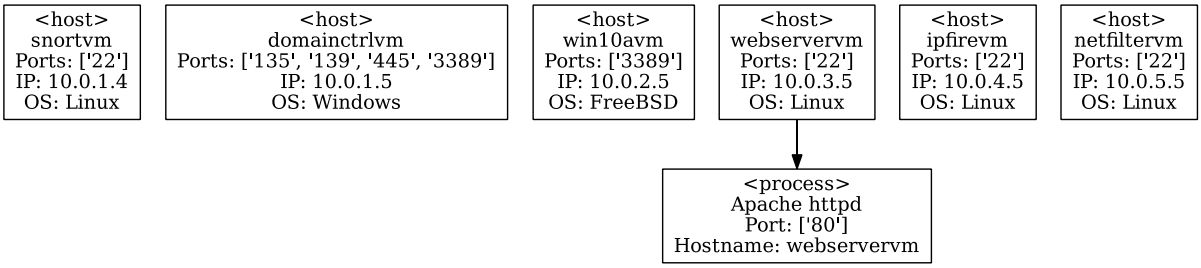

Figure 12. SSM-style system model built from an Nmap scan of the CyberKit4SME testbed.

This example shows “host” and “process” assets discovered through Nmap along with additional meta-data that has been extracted. The additional meta-data is not used in the SSM’s threat and risk calculation but is useful for humans to identify assets and also for integration into automated system that can keep the model up to date (for instance with vulnerability information). This example does demonstrate the fallibility of network scans, in that the “win10avm” virtual machine has been identified as using the “FreeBSD” operating system when it is in fact a Windows 10 host.

Further work is needed on what information Nmap can discover relating to the network infrastructure. Other techniques such as querying network hardware (e.g. via Simple Network Management Protocol, SNMP) may be needed.

Similar work on using the Microsoft Azure API to query cloud resource has begun but is at too early a stage to report on here.

The intention is to create a system that can pull information from a variety of sources and assist the user in the initial model building.

I.5.1.b. Attack path exploration

As described previously, the SSM’s “threat explorer” and “adverse effect explorer” provide an interface to the user to help them explore cause and effect and thus understand cause and effect chains, or “attack paths” in the system. Such understanding is vital as it informs the user as to where to best place additional security controls to reduce the risk.

To take an example, in CyberKit4SME D3.1 “Cyber Security Baseline Analysis Report” a model of the Tiani Spirit system was built. One “medium” risk adverse effect of concern was the loss of confidentiality of the source code. The “medium” risk arose from the combination of the choice of “high” impact and the calculated “low” likelihood. Opening the adverse effect explorer at that point shows 333 threats which are on paths that can cause the problem with that likelihood along with 21 threats that immediately proceed the problem on an attack path.

Examples of threats that directly cause the adverse effect are:

- Interception of “Source Code” by compromised service “SVN Service”
- Snooped flow of data “Source Code” between “IDE” and “SVN Service”
- Interception of “Source Code” by compromised client “IDE”

[30] Qualys: https://www.qualys.com/

All three of these threats are found because the SSM has worked out what data flows there are in the system and therefore where the data is vulnerable. All these threats may be controlled by encrypting the data flow, and that may well be the best idea, but none of them is the root cause: the attacker should not be able to access the data flows at all.

The threat explorer also lets the user examine the “root causes”. In this case there are two:

- Phishing attack fools “Customer” feeding malicious URL to “Customer Browser”
- Physical intrusion into space “Tiani Office”

The advantage of looking at the root causes is that tackling them is very likely to result in the reduction in likelihood of a large number of attack paths and therefore be a cost effective solution. In the adverse effect explorer, the user is able to jump directly to each root cause to examine them. Proposed controls for the root causes are:

- Phishing attack:
    - Spam Filtering for the Customer Email
    - Security Training for the Customer
- Physical intrusion (a physical lock was specified as already being in place):
    - Biometric ID Verification at the Tiani Office
    - Chip and PIN Verification at the Tiani Office

Again, these may well be excellent solutions, but what if we just don’t trust the customers to get things right and cannot influence that part of the “system”? In this case we need to move from the root cause, along the attack paths towards the adverse effect we are concerned about and find a good intermediate point to place an additional control.

At this point the interface becomes problematic. Whilst you can navigate from a threat, to any of the adverse effects it directly causes and from them to any of the threats they enable, there are so many possible attack paths it is easy to get lost, loop back on yourself and end up confused.

In CyberKit4SME we have begun prototyping solutions to this problem: creating diagrams of the attack paths to help the user understand how to get from a root cause to a chosen adverse effect. We will use these diagrams in the validation work to help the SME end users and refine them as necessary.

Figure 13 shows one such diagram. It is included here primarily to help illustrate the combinatorial explosion of paths that we are dealing with even in such a simple case: the detail is not relevant (though can be seen at high zoom level). The Figure shows all the possible attack paths from the phishing attack to the loss of confidentiality of the source code, including both the alternating threats and associated adverse effects on the paths. The shortest paths are likely to be the most useful ones to consider and they are highlighted in bold.

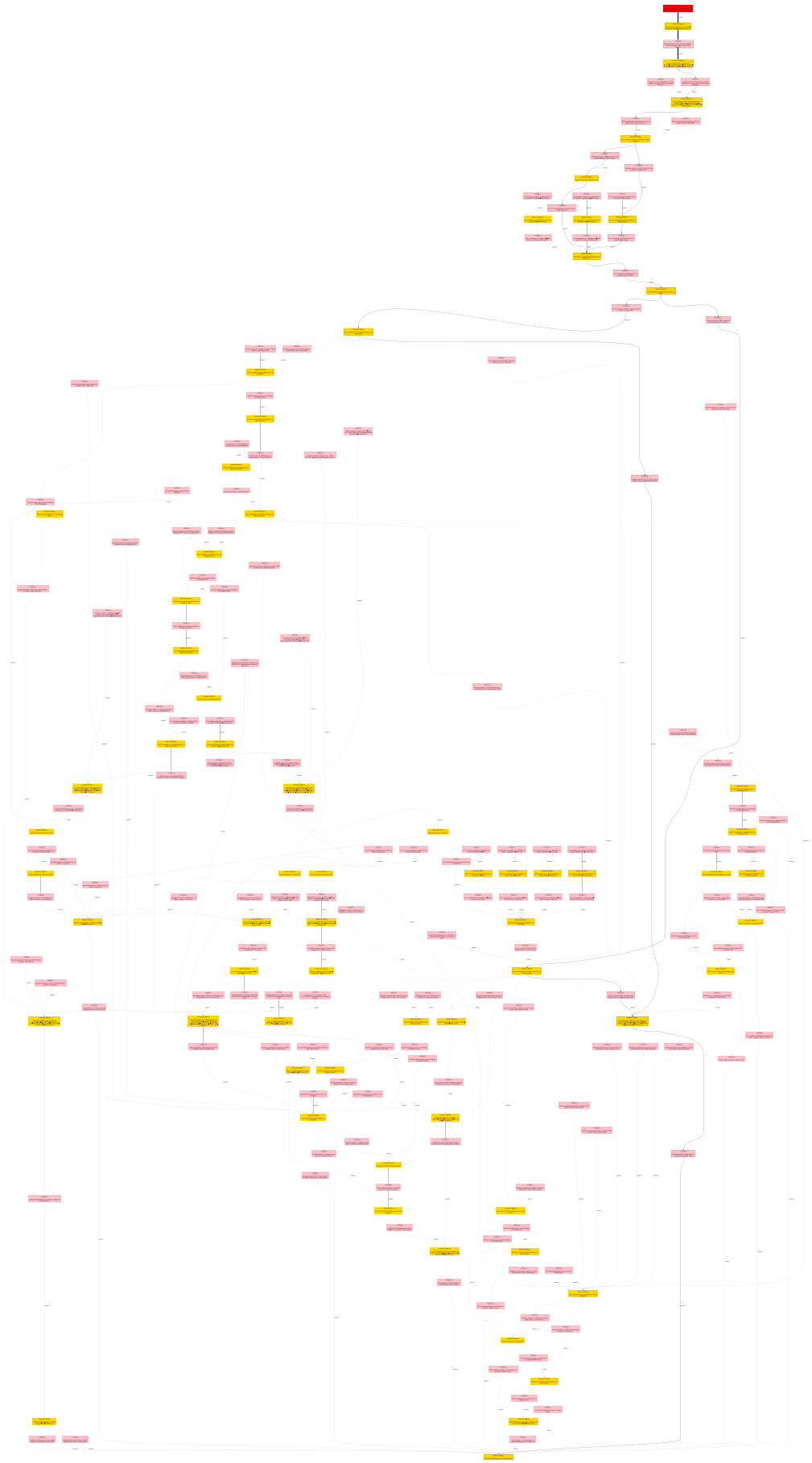

Figure 13. All the attack paths from a phishing attack to the loss of confidentiality of the source code. Shortest paths are shown in bold and links that move further away from the target are dotted.

The second diagram, Figure 14, attempts to illustrate just the shortest attack paths with sufficient context to make the steps understandable. Here the assets which are involved in or attacked in each threat are added and the Figure is arranged with each asset aligned with the threat where it is first involved. This provides an indication of the path through the system that the attack traverses and describes it using just the asserted assets that the user if familiar with from the standard SSM system model view.

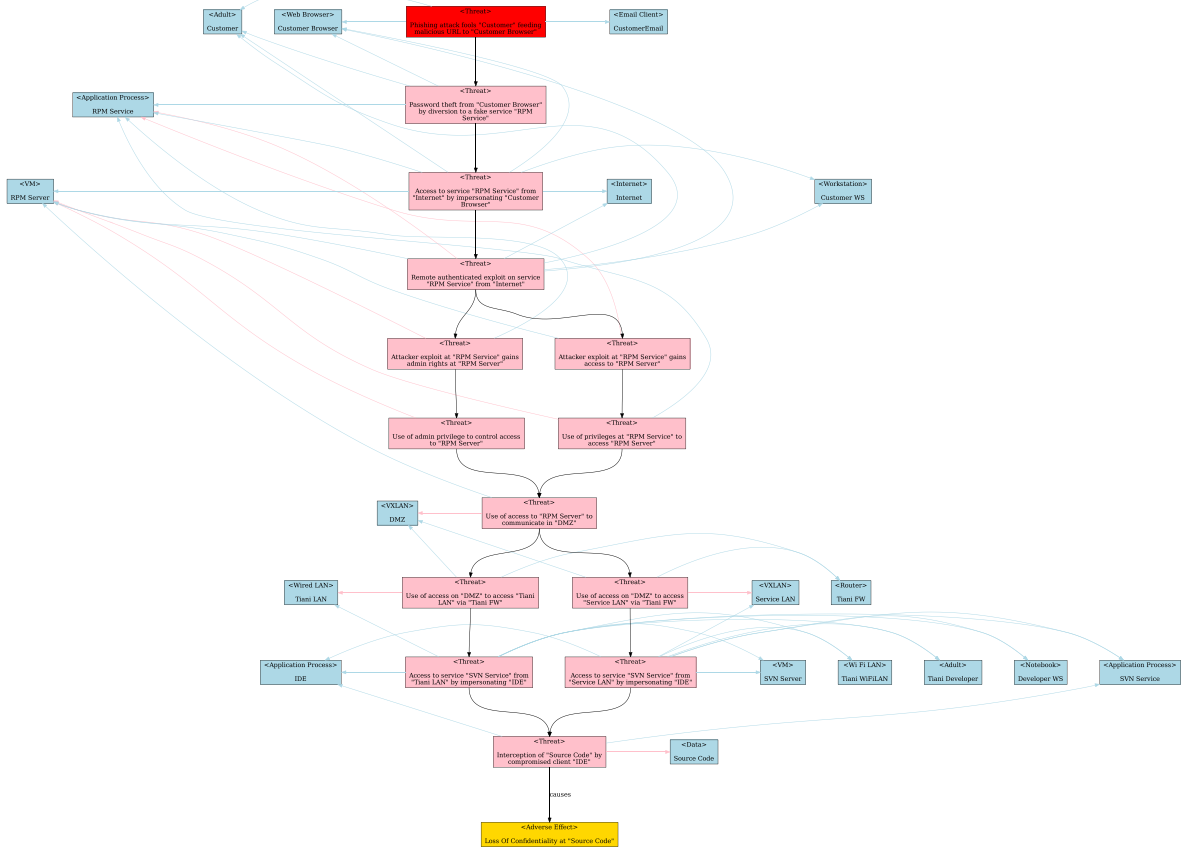

Figure 14. The shortest attack paths from the phishing attack to the loss of confidentiality of the source code. Just the threats are shown along with the assets which are involved with the threat or threatened by the threat (shown with a red link).

Further work on these diagrams will be done based on validation feedback.

I.5.2. Knowledgebase

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.

I.5.2.a. Scope

The asset types available depend on the knowledgebase loaded into the SSM. In the v5 knowledgebase currently being developed for Fog Protect, ProTego and CyberKit4SME, the asset types the user can add fall in several categories and include:

- Data: data sets or parts thereof
    - Data, sensitive data, health data, genetic data, biometric data
- Process: executable (and normally running), networked software
    - Application process, API server, ingress, database, email client, email service, email MX, web client, web browser
- Host: physical and virtual hosts for process and/or data
    - Fixed hosts: server, cluster, workstation, router, base station
    - Mobile hosts: laptop, tablet, smartphone, smartwatch, USB stick
    - Virtual hosts: VM, virtual cluster, virtual router, Kubernetes assets (master, worker, pod, container)
    - IoT hosts: (health) sensor, (health) controller, with and without basic UI
- Network: physical and virtual networks and devices
    - Wired LAN, WiFi LAN, VXLAN, internet, public/private cellular network
- Space: physical spaces where other assets are physically accessible
    - Public space, private space, data centre, vehicle
- Stakeholder: a user (human) or organisation engaged in the system with some role (rights and responsibilities)
    - Organisation, human (adult or child)
- Other: other intangible types of assets
    - Jurisdiction

The knowledgebase covers a large number and wide range of threats, along with associated security controls. The threats fall into the following categories:

- Access and control privileges

  These represent situations where an untrustworthy agent with certain privileges can gain access to further privileges, related to resource access and control

    - Space, devices and networks
        - Management of resources
        - Management of access at a space
        - Access to networks
        - Control and use of devices
    - Privileges, processes and their interactions
        - Control of processes
        - Control of services and clients
    - Access to stored data
        - Access to stored data via a host device
        - Access to stored data via a process
    - Access to flowing data (via a compromised process)
    - Encryption bypass attacks (access to data using a compromised key)

    - Encrypted storage side effects (inability to access encrypted data store)
    - Encrypted data flow side effects (inability to access encrypted flowing data)
- Insider attacks

  These represent situations where a legitimate user or organisational stakeholder performs malicious actions. In most cases, this is modelled by reducing the trustworthiness of processes and devices they are operating, managing or using

    - Organisational corruption (Stakeholder (organisational) influence)
    - Insider attacks on network control and access
        - Insider (human) influence
        - Moderated insider (human) influence
- Exploiting vulnerable software

  These represent situations where an attacker can cause execution of vulnerable code and thereby gain temporary use of privileges

    - Discovery of vulnerabilities
    - Vulnerability exploits on a host (accessing vulnerable code)
        - Local/remote, authenticated/anonymous, including Bluetooth and USB connections; from attackers and worms
    - Vulnerability exploits on services (accessing vulnerable code)
        - Local/remote, authenticated/anonymous; from attackers and worms
    - Process I/O vulnerabilities
    - Host vulnerability exploit consequences
    - Process vulnerability exploit consequences
    - Query injection vulnerability exploitation
    - XSS vulnerability exploits -> loss of authenticity in client
    - Disabling access to or operation of vulnerable assets
- Other malicious attacks

  These represent situations where a malicious attacker exploits a weakness other than a software vulnerability

    - Physical access threats
        - Physical intrusion, access and tampering
        - Side effects of physical access control
    - Access to hosts and networks
        - Remote access to insecure hosts/networks
        - Authentication blocks network access
    - Network spoofing attacks (network impersonation attacks)
    - Network snooping attacks (process communications confidentiality)
    - Service impersonation
    - Client impersonation
        - Client impersonation
        - Client authentication failures
    - Exploiting successful impersonation
    - DoS attacks (Network DoS attacks)

    - Attacks using malicious email (phishing)
- Non malicious threats

  These threats represent the effect of accidents and unintentional errors that could cause problems without provocation by malicious attackers

    - Software bugs (causing process and hosts to crash)
- Secondary threats

  Secondary threats represent mechanisms for threat effects to propagate through the system. Most do not have control strategies because they are inevitable once the cause is present.

    - Device overload and availability
        - Effect of overload
        - Effect of loss of availability
    - Network communications availability (effect of loss of network availability)
    - Process communications
    - Process I/O dependencies and availability
    - I/O dependencies and availability (propagation of issues via served data)
    - I/O dependencies and authenticity/integrity
    - Integrity issues in served/flowing data
    - Exposing IoT sensor internals
    - Exposing IoT controller internals

Controls fall into the following broad categories:

- Organisational measures
    - Staff screening, training
- User intervention
- Physical security

  Controlling physical access to spaces

    - Physical locks & keys, chip & PIN, biometrics, ID checks,
- Device security

  Controlling direct access to devices; preventing alteration of software on devices.

    - Login password checks, chip & PIN, biometrics, anti-malware, secure host configuration, secure BIOS, remote wiping
- Data security

  Encryption of data flows or stored copies; replicated storage.

    - Encryption, keys, replication, data access control, DB access control
- Network security

  Network access control and routing restrictions

    - Radio subnet encryption, network AuthN via X.509, PSK, SIM, Bluetooth SSP, EAP-TLS, EAP-PSK, etc, blocked segments and interfaces, bandwidth management, DoS filter
- Service security

Access control and privilege restriction mechanisms

- TLS, AuthN, passwords, strong password, OTP, SMS, X.509, etc

- Client security

  Spam filtering, passwords

- Software security and vulnerability management

  Software testing, pen testing, patching, device certification

- Resource management

  Elastic hosting, process prioritisation

Considerable refactoring of the knowledgebase has been done to create v5 and add in what is needed to model additional aspects. In the CyberKit4SME project, the work on modelling data centres and Kubernetes systems was done.

In the following two sections, asset types that may be used in the system models are indicated through capitalisation, e.g. “Data Centre”, “Cluster”, “Pod” etc.

I.5.2.b. Data centre infrastructure

A “Data Centre” asset has been introduced to simplify the modelling of data centres. A Data Centre is a physical space that contains network resources (hosts and networks). Although it forms part of the physical layer, it therefore also participates in the network layer, and would normally be added to a system model with other network layer assets. The main relationships of a Data Centre are shown in Figure 15.

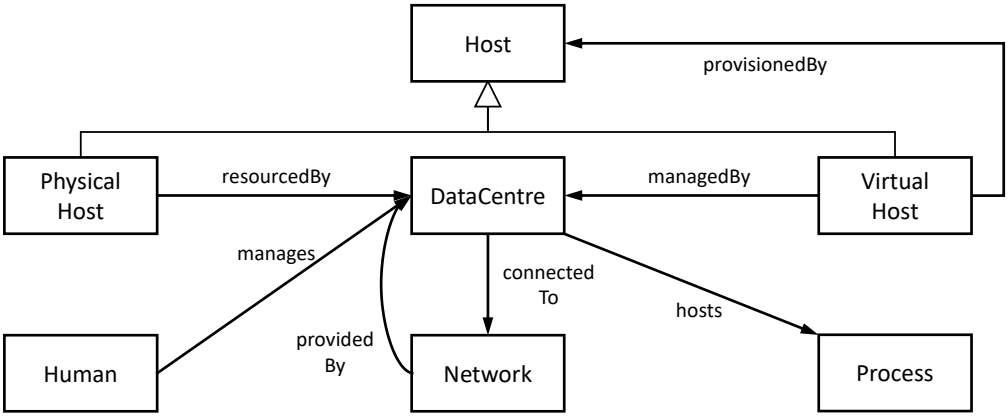

Figure 15. Data Centre Relationships in the Network Layer

There are two ways a Physical Host can be related to a Data Centre:

- locatedIn: signifies only that the Host is physically in the Data Centre. This is a physical layer assertion with no direct relevance to the network layer, which is why it does not appear in Figure 15;
- resourcedBy: signifies that the Host is physically in the Data Centre, and that it will be used to support virtualised host and network assets provided or managed by the Data Centre.

The domain model assumes that each Data Centre has at its disposal a physical Router providing a Wired LAN and a physical server Cluster connected via that LAN, both ‘resourced by’ the Data Centre. The domain model checks if they are already defined in the system model, and if not, it adds them. Best practice in SSM system modelling is to leave out the network infrastructure in a Data Centre. It is usually best to just add the Data Centre and let the domain

model fill in the details. Other elements can then be related to the Data Centre, leaving the domain model to link them with Data Centre infrastructure elements.

For example, the Data Centre infrastructure includes a Wired LAN but not a WiFi subnet. If the Data Centre does have a WiFi subnet to support mobile devices belonging to users working there, one can specify that this is provided by the Data Centre, rather than adding a separate WiFi router located in the Data Centre. The domain model inference rules will then link the WiFi subnet to the Data Centre router in a reasonable way, so the SSM user doesn't need to worry about ensuring connectivity via assets that will be added by inference. Similarly, if a Server (or any Fixed Host) is 'located in' but not 'managed by' the Data Centre, it is not necessary to think about how this Server could communicate via an external network such as the Internet. It is enough to specify that the Server is locatedIn the Data Centre, and the Data Centre is connected to the external network, and the domain model will add the necessary connections within the Data Centre.

I.5.2.c. Pod and container management

A Container is a stateless virtual machine, which means it can easily and automatically be deployed, stopped, or restarted. A Pod is really a compositional context for one or more Containers, in which the Containers can be related to each other and to persistent storage volumes. This means a Pod acts somewhat like a Host, providing an (internal) communication mechanism between Containers, routing communications between Containers and external subnets, and providing access to persistent storage. For this reason, the domain model represents a Pod as a type of Virtual Host.

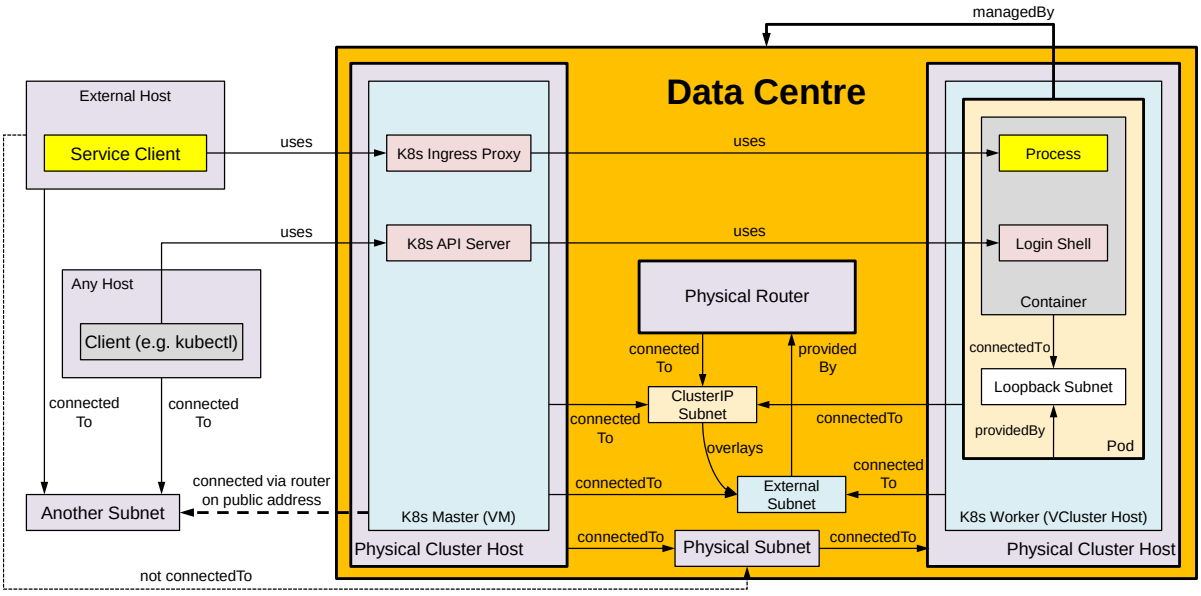

Figure 16. Inferred Management Infrastructure in a Data Centre. If the assets and relationships shown with thick lines are asserted then the rest can be inferred.

If any Pods are managed by a Data Centre, they are not (like other Virtual Hosts) provisioned directly by the Data Centre's physical Cluster but run on a virtualisation platform assumed to have a Kubernetes-like architecture. The elements of this are shown in Figure 16. It is not necessary to define these explicitly in the system model. The presence of at least one Pod managed by the Data Centre triggers domain model inference rules that create this infrastructure by finding what is missing and filling in the gaps.

The Kubernetes (K8s) documentation does not completely define everything, as some aspects are left to the K8s operator. In these areas, reasonable assumptions have been made to arrive at the structure shown in Figure 16. The rules that create this structure do so one element at a time, and check whether there is an asset able to fulfil the same role before adding the

element. This means it is possible to override the structure of Figure 16 by explicitly defining some elements in the system model. For example, some operators may have K8s Master and Worker nodes connected directly to a physical subnet, instead of provisioning a separate (virtual) external subnet. This can be specified by defining it explicitly (possibly including some of the underlying Data Centre infrastructure elements described previously). The domain model should then detect which elements are present and avoid creating them again, leaving a tailored variation of the K8s structure.

The domain model also uses inference rules to fill in some relationships between Processes, Pods, Containers and Data Centres. If a Pod 'contains' a Process, a Container is inserted running on the Pod and holding the Process, which may be managed by a Data Centre (if not it must be provisioned by some other Host). If a Process is hosted directly by a Data Centre, it is assumed to run in a Container on a Pod managed by the Data Centre, so that alone would trigger construction of the K8s platform elements from Figure 16.

Best practice for SSM model creation is the following:

- stateless Processes (those that don't access stored data locally) may be 'hosted' directly by the Data Centre, leaving the domain model to put them in Containers inside Pods running on the K8s cluster;
- other Processes can be 'contained' in Pods that are 'managed by' the Data Centre, leaving the domain model to insert Containers inside those Pods;
- Processes should be contained in the same Pod if they are tightly coupled (so the Pod must be included explicitly to get Processes in the same Pod even if the Processes are stateless).

Kubernetes will deploy additional pods according to the load and the configured scaling rules. In our model, A Pod is a ScalableHost that runs on a ClusterHost (though the Pod is not itself a ClusterHost).

- ScalableHost: means the asset represents a collection that can be automatically scaled as loads increase;
- ClusterHost: means the asset represents a collection containing more than one host instance, where the instances have identical properties but may be doing different things.

Two Processes are 'tightly coupled' if they must be scaled together, and not otherwise. Thus, a web service with a back-end database would normally use different Pods as it may be appropriate to scale the former separately from the latter. If the web service were divided into two separate processes that have a 1-to-1 relationship, then one might have one Pod containing both (even if the web service is stateless), but still with a separate Pod for the back-end database.

I.6. Interfaces

Human operators interact with the SSM using the web interface already described in Section I.3.

I.6.1. Interfaces with other tools

The SSM service has an extensive REST API using JSON format messages. The API is used by the web user interface and other clients and described using OpenAPI[31] so that client libraries can be automatically generated.

[31] OpenAPI: https://swagger.io/specification/

I.6.1.a. Asset mapping

All the integration scenarios we are looking at involve an interface between an SSM system model and some information coming from another tool that has some data about the actual information system (rather than a model of it). This means that in all cases the first challenge is to have a way to map from real-world information to the model (and vice versa).

Status: prototype stage

Each asset in the system model can have arbitrary key-value pairs added it (with key repetition permitted). This data is not used by the SSM for threat and risk analysis but assists with the asset mapping. As described in ProTego D4.2[32], for process assets in the system model we add the following additional metadata:

- “cluster” (optional): Kubernetes cluster identifier. This is required only when there are reports coming from more than one Kubernetes cluster;
- “hostname”: this can be a fully-qualified domain name, an IP address, or a Kubernetes-style hostname including the namespace (i.e. <service-name>.<namespace-name>);
- “process”: the software name. Taken from the CPE identifier (“vendor:product”);
- “version” (optional): software version. Taken from the CPE identifier (“version:update” or just “version” if “update” is not present). The “version” meta-data is only required when there are multiple versions of the same process.

The Common Platform Enumeration[33] (CPE) identifier is used by OpenVAS to identify the software process it has found (and which it is reporting on). The identifier takes the form:

- “cpe:/a:vendor:product:version:update”.

For example:

- “cpe:/a:microsoft:internet_explorer:8.0.6001:beta”.

The ProTego work was concerned with software vulnerabilities so needed to identify Process assets, but in CyberKit4SME we will also need to add metadata to other asset types, such as “cluster” and “hostname” for Host assets or username for Human assets.

Relevant metadata to assist in mapping should be:

- Process:
  - “process”: the software name. Taken from the CPE identifier (“vendor:product”);
  - “version” (optional): software version. Taken from the CPE identifier (“version:update” or just “version” if “update” is not present). The “version” meta-data is only required when there are multiple versions of the same process.
- Host:
  - “cluster” (optional): Kubernetes cluster identifier. This is required only when there are reports coming from more than one Kubernetes cluster;
  - “hostname”: this can be a fully-qualified domain name, an IP address, or a Kubernetes-type hostname including the namespace (i.e. <service-name>.<namespace-name>).
- Network:

[32] ProTego D4.2: https://protego-project.eu/deliverables/

[33] Common Platform Enumeration: https://cpe.mitre.org/

    - “tag”: although there is no universal network identifier, it is likely that the source of discovery data (such as Keenaï) will be able to assign a tag (label) to a network based on IP address range for instance.

- Human:
    - “username”: the username, as found in log files.

Additional arbitrary metadata may also be added of course, for instance ports that host is known to have open (either from operating system processes such as SSH or “hosted” processes such as a web server). This can be used for creating Keenaï filter rules (see Section [not in extract]) or just for human inspection and comprehension.

One important aspect of the SSM system models is that not every asset in the real world is represented separately in the model. For instance, a room containing 20 identical PCs in the same environment (e.g. connected to the same network) would be represented as a single Host asset in the model, as the same threats and controls would apply to all. If all such assets are represented individually then the model is harder to read, slower to run and the output will be unnecessarily verbose.

To address this, we can make use of the fact that metadata keys in the SSM may be repeated, so for instance a host can list all the IP addresses that it represents.

I.6.1.b. Interface to receive system asset and relationship information

Status: design stage

As described in Section I.5.1.a and in Section [not in extract], there are various sources of information that may be used to help build the initial SSM system model. The sources of information will need to feed data into a common interface. The SSM service API already has methods to add new assets or relationships (as used by the web user interface for instance) but for information arising from external tools a different interface is required to

- Check that the asset or relationship does not already exist;
- Map metadata information provided by the external tool to existing assets in the model;
- Merge new asset information with an existing asset if appropriate;
- Separate existing merged assets into multiple assets when necessary;
- Place new assets on the canvas in a sensible place;
- Potentially seek human assistance.

In the initial model building stage, the interface would be used to add information. However, we also want to be able to semi-automatically keep a system model up to date with changes in the real-world system and so the ability to disable, enable and delete asset should also be supported. This can be done using a REST API and standard HTTP verbs: POST (for add), PATCH (for the enable and disable edits), and DELETE. The GET method will also be supported to enable an external tool to query as to whether an asset or relation already exists in a system model.

All messages to this API must also be in the context of an existing model which is specified in the URL. For example, to add a new asset one would send a POST request with an HTTP body using the format of the following example to an endpoint such as:

`https://example.com/system-modeller/meta/models/{modelID}/assets`

```
{
    "label": "server1",
    "type": "Server",
    "metadata": [
```

```
        {
            "key": "hostname",
            "value": "server1.example.com"
        },
        {
            "key": "etc",
            "value": "etc"
        }
    ],
    "status": "enabled"
}
```

The response would include the SSM system model ID and label for the new host asset. The status would be assumed to be “enabled” but could be specified explicitly. Labels in system models must be unique, so this could be specified by the external tool or automatically assigned.

To update (enable/disable) or delete the asset, a PATCH request would be sent, either using the same “metadata”, the label or specifying the asset ID (if known).

To add a relationship between assets a message in the format of the following example would be sent in a POST request to a URL such as:

`https://example.com/system-modeller/meta/models/{modelID}/assets`

```
{
    "from": {
        "id": "4b4bdbb7",
    },
    "to": {
        "label": "IE",
        "type": "ApplicationProcess",
        "metadata": [
            {
                "key": "process",
                "value": "microsoft:internet_explorer"
            },
            {
                "key": "version",
                "value": "8.0.6001:beta"
            }
        ],
    },
    "type": "hosts"
}
```

An asset ID, label or metadata that uniquely identified an asset could be used to specify the two ends of a relationship. Such a method would first check for the existence of the two assets rereferred to and create or update them as necessary. The ID of the relationship would be returned. Deleting a relationship would be done using an HTTP DELETE request and a similar message body.

I.6.1.c. Interface to respond to queries about the system model

To support the Keenaï use case of identifying anomalous events (see Section [not in extract]) the SSM needs to be able to provide information about a system model to external tools in a way that they can use.

Status: production

The SSM already has an interface to provide the system-model information in a JSON document. The endpoint and format are as shown in the following example:

```
https://example.com/system-modeller/models/{modelID}/report
```


```
{
    "name": "CyberKit4SME example model",
    "domain": "http://it-
innovation.soton.ac.uk/ontologies/trustworthiness/domain-network",
    "assertedAssets": {
      "cebf302e": {
        "label": "My Process",
        "type": "ApplicationProcess",
        "controls": [],
        "misbehaviours": [],
        "trustworthinessAttributes": []
      },
      "e1fa3001": {
        "label": "My Server",
        "type": "Server",
        "controls": [],
        "misbehaviours": [],
        "trustworthinessAttributes": []
      }
    },
    "inferredAssets": {},
    "relations": {
      "b587fd56": {
        "from": "e1fa3001",
        "to": "cebf302e",
        "type": "hosts"
      }
    },
    "threats": {},
    "misbehaviours": {},
    "trustworthinessAttributes": {},
    "controls": {},
    "controlStrategies": {}
}
```


The report format does not currently include the additional metadata fields so would not be appropriate for an external tool which needed to map to the real-world system to interpret. The metadata fields will be added using the same JSON format as described in the previous section.

Status: design stage

The report format as it is was not designed for the CyberKit4SME use case: it just outputs almost all the system model data.

If necessary we can provide some reporting options to reduce the size of the response document, for instance by only returning the model topology, that is the asserted assets and their relationships (and not all the threats, misbehaviours, controls, etc).

Another option would be to just provide a subset of the asserted assets and relationships centred around a specific asset of interest. When investigating an incident, the system model can be queried to return all the assets linked to an asset of interest (and their relations) or all assets up to *n* hops away in order to provide context for an investigation.

I.6.1.d. Interface for operational risk analysis

In the H2020 ProTego and Fog Protect projects we have developed a technique for updating the system model based on vulnerability data so that an "operational model" can be created and analysed (see Section I.7 and ProTego D4.2[34] for more information).

Status: prototype

Connectors have been developed to receive vulnerability data from OpenVAS and OWASP ZAP[35] and use this data to update the system model. In CyberKit4SME we will consolidate the prototype integration with OpenVAS and develop a custom adaptor to receive additional vulnerability data from Keenaï.

An example of the format the SSM accepts for OpenVAS vulnerability data follows:


```
{
  "cvss_version": 2,
  "identifiers": {
    "ip_address": "192.168.143.211",
    "port": "80/tcp"
  },
  "vulnerabilities": [
    {
      "name": "Nginx Server Multiple Denial Of Service Vulnerabilities 01
 - Jan16",
      "family": "Denial of Service",
      "product": "cpe:/a:nginx:nginx:1.9.5",
      "cves": [
        {
          "CVE-2016-0742": []
        },
        {
          "CVE-2016-0746": []
        },
        {
          "CVE-2016-0747": [
            "CWE-399"
          ]
        }
      ],
      "cvss_base_vector": "AV:N/AC:L/Au:N/C:P/I:P/A:P",
      "qod": "30"
    },
...
```


[34] ProTego D4.2: https://protego-project.eu/deliverables/

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

The SSM interprets the “identifiers” to find the asset and uses the CVE[36], CWE[37] and CVSS[38] data to determine how to adjust the model configuration. The risk analysis is then re-run and some optimal security control recommendations are then found.

To ease integration, the highest level risks and the recommendations are placed on a Kafka message queue. An example of the risk report follows:


```
{
  "Risks": [
    {
      "ObjectToIdentify": {
        "ObjectName": "SomeAsset",
        "ObjectIdentifiers": []
      },
      "RiskDescription": "Alteration or corruption designed to mislead, s
uch that the affected asset is no longer what it claims to be.",
      "RiskImpact": "High",
      "RiskName": "LossOfAuthenticity",
      "RiskLikelihood": "High",
      "RiskLevel": "Very High"
    },
    …
  ],
  "OverallRiskLevel": "Very High",
  "RiskVector": {
    "VeryHigh": 1,
    "High": 5,
    "Medium": 0,
    "Low": 445,
    "VeryLow": 983
  }
}
```


A list of security control suggestions along with their effects is provided in a separate message. For example:


```
{
  "ExistingRisk": "Very High",
  "ExistingRiskVector": {
    "VeryHigh": 1,
    "High": 5,
    "Medium": 0,
    "Low": 445,
    "VeryLow": 983
  },
  "ObjectToIdentify": null,
  "Recommendations": [
    {
      "RecommendationId": 22,
```


[36] Common Vulnerability and Exposures: https://cve.mitre.org/

[37] Common Weakness Enumeration: https://cwe.mitre.org/

[38] Common Vulnerability Scoring System: https://nvd.nist.gov/vuln-metrics/cvss

```
      "ControlStrategy": "LS.L.VSPSH.1-
VSPSH_7fb13229_27bca80e_265cd9f2_dcd16e78-CSG-DisableGatewayHost-
Runtime",
      "Action": [
        {
          "Control": "CS-Disable-7fb13229",
          "ControlAsset": {
            "AssetLabel": "Router-OSRDataCentre",
            "AssetType": "Router",
            "ControlLabel": "Disable",
            "Uri": "er_c7a837b5"
          }
        }
      ],
      "ExpectedRisk": "High",
      "ExpectedRiskVector": {
        "VeryHigh": 0,
        "High": 2,
        "Medium": 1,
        "Low": 146,
        "VeryLow": 1285
      },
      "ExpectedMisbehaviours": [
        {
          "ObjectToIdentify": {
            "ObjectName": "SomeAsset",
            "ObjectIdentifiers": []
          },
          "RiskDescription": "Represents a state in which a data asset is
 somewhat out of date, or a process is delayed by outdated or (temporaril
y) unavailable inputs.",
          "RiskImpact": "Medium",
          "RiskName": "LossOfTimeliness",
          "RiskLikelihood": "Very High",
          "RiskLevel": "High"
        },
        …
      ],
      …
    },
    …
  ]
}
```


This interface and message formats will continue to be developed in other projects and will be adapted to CyberKit4SME as required.

I.7. Usage

As described briefly in CyberKit4SME D2.2 “Technical Specifications”, the SSM models information systems and automates much of the risk assessment procedure described in ISO 27005 (thereby supporting ISO 27001 compliance). Through automation, a risk assessment is made methodical and reproducible and a security analyst may do a better job in less time.

It is useful to think about three usage scenarios for the SSM, though they overlap to a large degree: security-by-design, audit and change management, and Operational Monitoring and Recommendation. The first two scenarios involve extensive human interaction with the software through the web user interface already shown in Section I.3.

I.7.1. Security-by-design

If you are designing a new information system, best practice (and indeed, regulation) says that security should be designed in to the system. The SSM literally provides a blank canvas for a design to be drawn upon along with all the potential system elements (including security controls) and their relationships. At any point, the threats may be analysed and a risk assessment may be executed, providing valuable information to the designer such as what security controls may be added where and their effect. In this way, a system may be designed secure before implementation.

I.7.2. Audit and change management

In this scenario you have an existing information system and want to assess the risks arising from the design of the system and/or those introduced or mitigated by changes (such as a new part which is being designed). This might be part of a formal ISO 27001 audit, or an internal assessment. It is a small variation on the *Security-by-design* scenario sketched above but is the more common scenario given the prevalence of existing systems.

To use the SSM the user interacts with the SSM software with some parts being manual and others computed by the SSM. The procedure is:

1. User: draws a model of the existing system, including relevant assets (networks, hosts, processes, data, people, places) and their relationships (such as which process uses what data).

   This requires an understanding of the physical/virtual infrastructure (network, hosts), the software and data used by a company and the environment (people, places).

2. User: identifies the primary assets for the business (generally data and processes) and indicate the impact on the business that failures in those assets (such as loss of confidentiality) would cause.

   This requires an understanding of the business.

3. SSM: finds the threats to the system automatically using the built-in knowledgebase and through its understanding of attack-paths and threat cascades.

   The concept of a “threat” is taken to mean an undesirable situation and the SSM can identify not only standard cyber-security threats but also situations leading to non-compliance with regulation (such as GDPR).

   This would normally be done (imperfectly) by a trained security analyst.

4. User: specifies what control measures are already in place (such as passwords, firewalls, etc).

   This requires an understanding of the information-security measures already in place.

5. SSM: computes the risk of every threat to the system automatically.

   This is very hard to do by hand and would be done by a security analyst. It involves the use of the specified impacts, the inter-connectedness of the assets (to see how failures in the secondary assets affect the primary assets) and an understanding of how the controls that are in place effect the likelihood of each threat.

6. User: adapts the system with additional controls suggested by the SSM or with other proposed changes and recomputes the risk (step 5) until the residual risk is acceptable for the business.

   This requires an understanding of what information-security measures it is possible and cost-effective to implement.

All together the ISO 27005 risk assessment procedure is a complex process, requiring knowledge about many aspects of a business. Such a procedure has at times been considered

too difficult for SMEs to manage, hence simpler descriptions of necessary security measures such as the UK's Cyber-Essentials[39] have been introduced by regulators.

The risk assessment of the system design carried out in this scenario (and the first one) is with a long time horizon: it looks at what is likely over a long period – the future risk – and proposes security controls to match. For example, the assumption is made that vulnerabilities will be found in any software process over time and so the likelihood of such exploits is set to "medium" by default in order to raise associated risks and encourage the implementation of security controls. One appropriate security control is to have a software patching policy for instance but of course a system should also be designed to resist zero-day attacks for which no patch yet exists. If a software process in the model is known to be hardened against attack (it may have been pen-tested for instance), or known to be especially vulnerable then the default likelihood associated with this can be changed appropriately.

I.7.3. Operational monitoring and recommendation

Work in the Fog Protect and ProTego projects have developed a prototype system for:

- updating the design-time model to be synchronised with the current state and create an operational (or "run-time") model;
- performing a risk assessment with a short time horizon, to assess the current risk;
- providing appropriate security control recommendations which would have immediate effect.

Updating a system design model to be an operational model primarily involves adjustments to the model's attributes to take into account the current set of known vulnerabilities. For instance, an OpenVAS scan of the operational system will reveal whether there are known vulnerabilities (or not) in the CVE database. Those software processes where a vulnerability is found are then configured in the model to increase the likelihood of a successful attack, and those where no vulnerability is found are configured to reflect that a successful attack is very unlikely (in the short term).

In such an operational modelling scenario, it is likely (we hope) that the standard controls such as having a software patching *policy* are already in place and described in the model. To address immediate risks, other controls may be proposed such as *applying* a software patch (i.e. following the policy and doing it *now*) or disabling a service or network path.

Figure 17 shows an operational system where the SSM is linked to a SIEM and to scanners such as OpenVAS (adapted from a Figure developed for the ProTego project). In the Figure:

1. The blue lines show the system model being built and a mapping between assets in the system model and those understood by the SIEM being configured.
2. The red lines show the Information System being monitored by the SIEM and OpenVAS and events flowing into the SIEM and the prototype "Run Time Microservice" which understands how to update the system model for different events.
3. With an updated model, the solid green lines show an updated risk level and suggesting mitigating controls flowing back down the stack, and dashed green lines show manual human-in-the-loop operations which the system administrator may take as a response.

[39] Cyber-essentials: https://www.ncsc.gov.uk/cyberessentials/overview

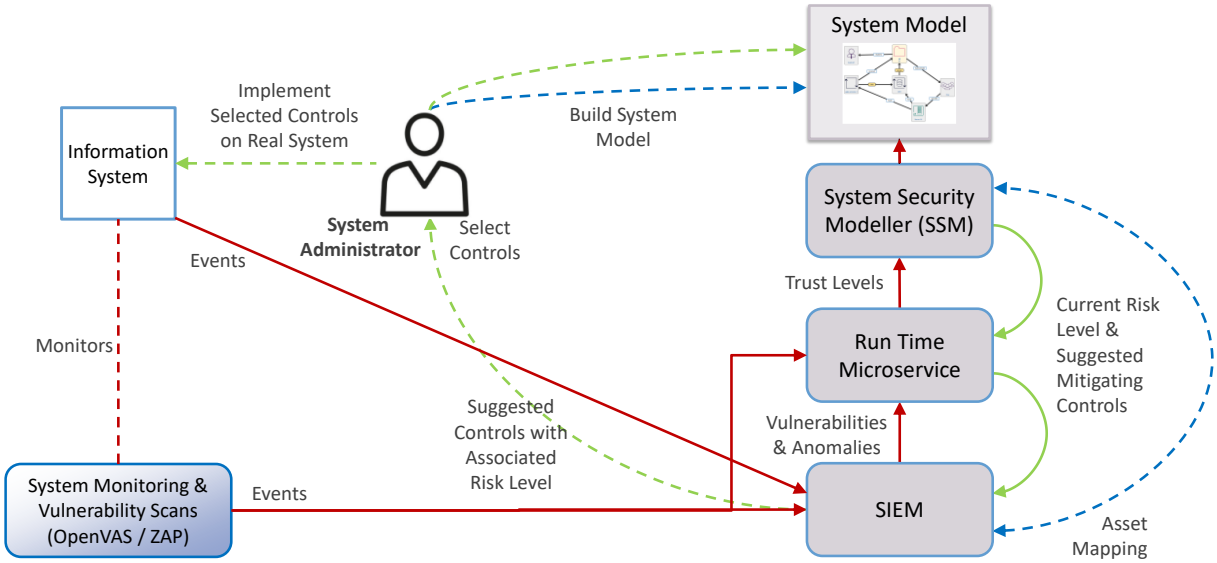

Figure 17. Integration of the SSM into an operational system.

I.7.4. Challenges for SMEs

The three scenarios bring challenges that will need addressing to enable non-experts to use the tools. They are summarised in Table 3.

Table 3. High-level challenges for SMEs to use the SSM in the three scenarios.

	Usage Scenario			
Challenge for SME	***Security-by-design***	***Audit and Change Management***	***Operational Monitoring and Recommendation***	***Addressed by***
Understanding how to represent the intended design in the SSM	X	X		Documentation, inference of missing parts (I.5.2)
Understanding the best place to add security controls	X	X		Documentation, attack paths (I.5.1.b), operational risk analysis techniques (I.6.1.d)
Understanding the identified attack paths in the system	X	X		Attack path analysis tools (I.5.1.b)
Extracting appropriate data (e.g. a report) from the SSM to inform the implementation	X			Better reporting
Knowing what assets make up the existing information system and how they are related		X		Model discovery (I.5.1.a)

Extracting an appropriate report describing the results of the risk analysis and what needs to change		X		Better reporting
Integration of the SSM with other tools (such as SIEM, OpenVAS)			X	Operational risk analysis (I.6.1.d)

I.8. Validation

The SSM, along with its built-in documentation, was deployed early-on in the CyberKit4SME testbed with an account created for each validation scenario. The system models developed for CyberKit4SME D3.1 “Cyber Security Baseline Analysis Report”, along with the associated in-development knowledgebase were provided as example starting points.

The validation partners have tried to use the SSM, have explored the baseline models and have developed some further models. Full details will be reported in CyberKit4SME D3.2 “Interim Validation Case Studies Report”.

Initial feedback is as follows (“+”, “=” and “-“ signs mean appreciation, general remark and doubts, respectively):

[+] SME people with decent technical background (IT administrators, IT managers, IT ingeneers, Software Developers, ...) appreciated the power and completeness of SSM;

[+] SME people with decent technical background (IT administrators, IT managers, IT ingeneers, Software Developers, ...) had no specific;

[=] SME people with no technical background had no significant problems to use items/elements to build up a diagram;

[-] SME people with no technical background hasn’t been confident on how to configure values in “impact”, “likelihood” and “risk” columns;

[-] Users with less or no technical background were not comfortable with the tool: they didn’t fully understand purpose and goals of SSM (“When I finish a model, what is supposed to do with that?”)

[-] many reported view and GUI could be improved (view is frequently centered in the upper left corner of the diagram canvas)

[+] many reported documentation is complete and comprehensible;

[+] SME people with less or no technical background had no significant problems to use items/elements to build up a diagram;

[-] SME people with less or no technical background appreciated warnings, alerts and dialog boxes showed by the tool but users considered these aids not so “actionable” because they were not able to identify what to do, based on these messages;

[+] users with technical background described alert, warnings and dialog boxes as useful and clear contribution to the general knowledge of SSM results.