Run-Time Risk Management in Adaptive ICT Systems

Mike Surridge, Bassem Nasser, Xiaoyu Chen, Ajay Chakravarthy, Panos Melas
IT Innovation Centre
Gamma House, Enterprise Road,
Southampton, UK
+44 23 8059 8866
{ms,bmn,wxc,ajc,pm}@it-innovation.soton.ac.uk

***Abstract*—We will present results of the SERSCIS project related to risk management and mitigation strategies in adaptive multi-stakeholder ICT systems. The SERSCIS approach involves using semantic threat models to support automated design-time threat identification and mitigation analysis. The focus of this paper is the use of these models at run-time for automated threat detection and diagnosis. This is based on a combination of semantic reasoning and Bayesian inference applied to run-time system monitoring data. The resulting dynamic risk management approach is compared to a conventional ISO 27000 type approach, and validation test results presented from an Airport Collaborative Decision Making (A-CDM) scenario involving data exchange between multiple airport service providers.**

Keywords-Risk Management; Adaptive Systems; Secure ICT; Machine Reasoning.

I. INTRODUCTION

Today, critical infrastructures in areas like transportation involve multiple stakeholders (e.g. operating companies, customers, etc.), who are increasingly seeking to optimize their operations by exchanging data over ICT networks. A typical example is Airport Collaborative Decision Making (A-CDM) [1], in which service companies at an airport share information. This allows them to plan ahead and turn around aircraft more quickly, and make predictions of when each outgoing flight will be ready to depart. These are used by air traffic controllers to efficiently allocate airspace including take-off and landing slots. This has led EUROCONTROL to mandate the introduction of A-CDM in Europe as part of a wider collaborative decision-making framework for air traffic control under the Single European Skies (SESAR) initiative [2]. Similar data-sharing methods are increasingly used in transport hubs: other examples include Port Area Communities [3].

This agile data-sharing approach makes it easier to switch service providers to ensure availability at the right time if aircraft deviate from the pre-planned schedule. However, it also creates new risks from disruption of the data exchange. This in turn amplifies old risks as service providers become more dependent on each other as they eliminate slack resources in pursuit of greater efficiency. It is also very difficult for public infrastructure operators to analyze these risks as required by regulators [4]. Standard methods for analyzing risks in IT systems such as ISO 27000 [5], [6] require an a priori expert assessment of risks and controls based on the known structure of the system as a whole. This approach works well if it can be applied at design time in a conventional system lifecycle. It is far less effective in a dynamically composed system where each stakeholder designs only their own portion of the system, decides just in time how to deploy their resources and which external services to use, and (in extremis) decides how to respond to internal or external faults.

The SERSCIS project [7] aimed to address these challenges by developing a methodology and tools to manage risks in dynamic multi-stakeholder service oriented systems. The SERSCIS approach is based on semantic system models and extensive use of machine reasoning to analyze risks at design time, and at run time. Interactions between stakeholders are described by service level agreements (SLAs), making it possible to monitor and analyze other stakeholder behavior as well as in-house resources. Control strategies can be introduced that exploit the possibility of dynamic system reconfiguration, and can be activated in response to specific threats. While the focus of SERSCIS was on risks associated with interconnected IT services, the approach to modeling and managing risks also encompasses physical networks and some physical processes, driven by the needs of A-CDM validation scenarios.

In the following section we discuss related work to risk modeling and management. In section 3, we present the SERSCIS approach to threat modeling using semantic modeling and reasoning. Section 4 details the threat modeling activities and the relation with existing threat knowledge bases. Section 5 describes the threat activity assessment during system runtime. Section 6 discusses the implemented prototype in the context of A-CDM installation at Vienna Airport where as section 7 concludes with future work.

II. RELATED WORK

Conventional risk management methodologies [5], [6] are based on an analysis of assets that allow a system to achieve its purpose, threats to these assets that may prevent the system functioning correctly, and control strategies to protect those assets from potential threats. Assets are defined as anything that has value to the organization and which therefore requires protection. The notion of primary and secondary assets can be found in the literature. Primary assets like business processes and information are supported by secondary assets such as hardware, software, networks, personnel, physical spaces and organizational structure. These supporting assets are subject to potential threats from malicious or accidental disruption. In order to protect the primary assets, the threats to the secondary assets should be controlled and mitigated. In a mature risk management strategy, controls are introduced based on their cost-effectiveness, leaving some residual risks that are too unlikely or have too little impact on system assets to be worth controlling. We were particularly interested in the variant specified for use in the SESAR project [8] as this is directly relevant to the air traffic control sector and A-CDM scenarios. However, all these approaches depend on the use of security experts to identify potential threats, assess how likely it is that they will arise, and advise on how far control

strategies can reduce the likelihood of them having a significant impact on system assets.

Much research has been conducted to devise methods to support such analyses. For example, MEHARI [9] and CRAMM [10] provide a knowledge base of vulnerabilities, attack and control scenarios, which were found to be very similar in a recent formal comparison [11]. Other approaches include OCTAVE [12], which has been refined [13] to support a more qualitative approach for non-specialists. Many of these tools and methodologies provide generic threats (e.g. theft of media, fire, tampering with software, exceeding limits of operation) while others provide catalogues of specific threats per asset type (e.g. file erasure, OS, application software). However, they all depend on human analysis and interpretation by experts in IT security and/or the system being analyzed.

This has motivated research into the possibility of capturing human expertise in a model which could then be applied by non-experts. For example, Secure Tropos [14] provides a diagrammatic approach to risk modeling, which has been extended [15] to provide a domain model covering assets, risks and risk treatment related concepts, and asset security criteria for confidentiality, integrity and availability. The CORAS project [16], [17] also used a graphical approach to identify, explain and document security threats and risk scenarios. A graphical notation was developed to perform five security analysis tasks: Context establishment, Risk identification, Risk estimation, Risk evaluation and Treatment identification. Diagrams are created during each of these steps (similar to UML models) under the guidance of a domain expert. These approaches have been used successfully, but they still require human interpretation, so they cannot handle situations where assets may be rapidly added, removed, reconfigured or recomposed at run-time. In [18] a formal approach is proposed for reasoning. The relations between the risk components are defined focusing on events temporal dependencies. However the quantitative analysis of these dependencies was not addressed.

Semantic risk modeling is another approach based on machine understandable representations which offers the possibility of automated risk analysis. Blanco et al [17] provide a useful review of such approaches. For example, the NRL Security Ontology [18] provides a way to describe the security properties of Web Services, which was later used in a Web Service vulnerabilities ontology [19]. The most mature example prior to the developments reported here is the Security Ontology from Secure Business Austria (SBA) [20], which is based on the German IT Grundschutz Manual [21]. It provides a way to model systems with common threats and control strategies. The SBA approach goes a long way towards the goal of capturing security expertise in a form that can be reused (with supporting tools) by non-experts. However, this ontology describes threats and vulnerabilities using OWL instances, which works well when modeling a statically deployed system, but not for a dynamically composed system whose concrete composition is not known at design time. Having said that, it can be used to construct Bayesian belief graphs [22] describing the probability of threats impacting the system in the presence of controls. Poolsappasit et al [31] use such graphs to calculate dynamically changing threat levels via backwards Bayesian inference, but in their approach the graphs are constructed manually and only the probabilistic weights are adjusted. Our approach combines the two approaches, constructing belief networks automatically for a dynamically composed system and using both semantic and Bayesian inference to deduce threat activity levels.

III. SERSCIS APPROACH

In SERSCIS, our aim was to create a unified approach for such systems spanning both design time and run-time risk analysis. To do this, we needed four things:

O1. an ontology capturing security expertise using generic classes of assets, threats and controls;

O2. a way to specialize this to describe system-specific classes of assets and threats in a design time abstract system model;

O3. a way to instantiate the resulting classes at run-time to create a concrete system model representing the current configuration of the running system;

O4. a way to analyze this model to identify vulnerabilities and diagnose threats.

The starting point was a core model based on the approach used by SBA [20]. The SERSCIS variant was deliberately made simpler, with a view to reducing the number of facts that have to be asserted in Objective 3 (mentioned as O3 from now on) to support useful inferences in O4 above. In SERSCIS, we also made extensive use of SWRL rules to model the configuration of controls needed to protect assets from each class of threat, and the way some threats could be triggered as secondary effects of a primary disruption to the system.

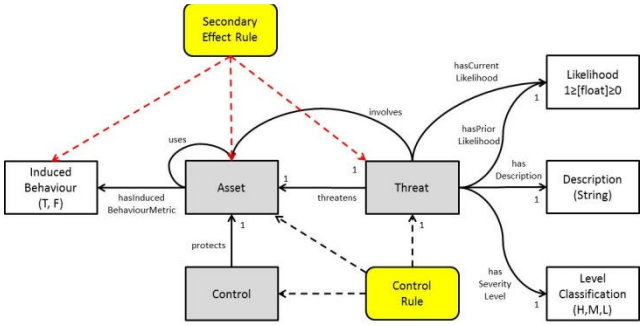

Figure 1. Core model classes and SWRL rules.

The core model classes were then specialized to provide a dependability model describing (still generic) types of assets, threats, controls and (mis-)behaviours found in adaptive ICT systems. This encoded expert knowledge, e.g. what types of threats are relevant in such a system, how do they arise in specific combinations of interconnected assets, and what controls are needed to block or mitigate them. For example, asset classes were defined for offered services, their clients (customers who pay for and control access to the services, and consumers who actually use them), and resources used to deliver the service. Resources are further divided into those specified by clients (CSResources), or selected by the service provider (PSResources). Further asset classes represent the physical infrastructure, including hosts (organizations of people, computers and other equipment in which services, resources and clients operate) and communication networks to which organizations may be connected). The resulting dependability model thus

provided O1 above. For full details of this semantic model, see [24].

For O2, a graphical system editor was developed allowing users to create system-specific sub-classes of the generic asset types from O1, and define the relationships between them. An automated process was then used to find system-specific asset combinations that may be subject to generic threat types from O1, and to auto-generate system-specific threat classes representing the various ways each generic threat could affect the specific system. These tools are also described in [24], while an application of the ontology to model A-CDM systems and scenarios is described in [24].

Here we are mainly concerned with the construction and analysis of the concrete run-time system model O3, and the use of this model to detect and diagnose attacks O4. For this we must consider the types of threats captured by the model, and how they are related to system behavior.

IV. THREAT MODELLING

The goal of threat modeling in SERSCIS is to identify the threats to different asset types from a stakeholder perspective at design time, and create a model allowing threats to be automatically assessed at runtime. Our threat models are semantic descriptions of the relationships between a class of threat and classes of involved assets. Rules are then defined describing what controls are needed protecting those assets to block or mitigate the threat. One of the main challenges in developing this model is to ensure that threat classes are defined in a consistent manner, with no unintended gaps or overlaps between threat classes.

Examination of ISO 27001 and 27005 showed that they only provide guidelines for threat identification, which cannot easily be mapped onto an abstract threat model in the absence of a concrete system. The next option was to base the SERSCIS threat ontology on knowledge bases underpinning some of the related tools. The CRAMM threat catalogue is not published, so attention focused on the MEHARI catalogue of threats per asset type. However, MEHARI focuses on low-level information assets such as individual data files and software configurations. It would be possible to construct a threat model from this, but it would be more suited to analyzing a single service rather than an ecosystem of data sharing systems like those found in airports.

After conducting the investigation of the above standards, the best starting point among established standards and tools was found to be IETF RFC 2828 [25]. This was well suited to the task because it provides an information security vocabulary, so it was designed to classify meaning. RFC 2828 defines a threat as a combination of a threat action (an event or situation that compromises a system) and its threat consequences (the nature of the resulting compromise). Factorizing threats along these lines reduces the number of distinct cases that need to be considered from N.M to N+M, after which a systematic approach can be used to find all relevant combinations of threat actions and consequences.

The first step was to decide on the set of consequences to be included in the model:

TABLE I. THREAT CONSEQUENCES

No	Consequence and meaning
1	The threat causes data to be passed to or read by the wrong party. (An asset will become indiscreet).
2	The threat causes corruption of data. (An asset will be become inaccurate).
3	The threat causes the system to run too slowly. (An asset will begin to underperform).
4	The threat causes the system to fail in its function. (An asset will become unreliable).
5	The threat steals access to system functionality. (An asset will become promiscuous).
6	The threat causes control of the system to be lost to an inappropriate party. (An asset will become untrustworthy)
7	The threat places more load on the system than it can handle. (An asset will become overloaded).
8	The threat makes it impossible for one or more users to interact with some or all of the system functions. (An asset will become unavailable).
9	The threat undermines other parties' trust that the system will work correctly on their behalf. (They will become dissatisfied with their interaction with the system).

Table 1 indicates how these consequences are related some of the classes of asset misbehavior defined in the SERSCIS asset model. Other types of behavior are related to possible threat actions, which are listed in Table 2:

TABLE II. THREAT ACTIONS

No	Action
1	Destruction of an asset.
2	An asset is accessed without proper restrictions.
3	Removal of an asset by an unauthorized party (theft).
4	Alteration of an asset (corruption).
5	An asset is misused or caused to fail by an authorized operator (insider attack).
6	Blocking messages to and from an asset.
7	Sending spurious messages to or from an asset (spoofing).
8	Altering messages to or from an asset.
9	Messages to and from an asset are read by someone who is not the source or the intended destination (snooping).
10	There is a bug in the software components of an asset.
11	The asset malfunctions for other internal reasons.
12	An asset malfunction becomes persistent (i.e. predictable).
13	An imposter pretends to be an asset (impersonation – the asset becomes unauthentic).
14	An imposter deceives an asset by pretending to be another asset (the other asset is unauthentic).
15	An asset's capacity is reduced (under-provisioning).
16	An asset is subjected to more use than expected (overuse).
17	An asset commits to do more than it can.
18	An asset is hacked over a network using a previously known security flaw (remote known exploit).
19	An asset is hacked over a network using a previously unknown security flaw (remote zero-day exploit).
20	Other action, including secondary effects.

To derive a comprehensive set of threats, one must take all meaningful combinations of action and consequence for each type of asset. In the SERSCIS threat model, threats are labeled using the threat action number, the consequence number, and a descriptive name usually referring to the type of asset targeted by the threat. In practice, one can ignore some combinations if they are not important for a particular class of applications, but the systematic approach ensures this is never done inadvertently. For A-CDM scenarios we were mainly interested in inappropriate data access or alteration, and reduced service performance or availability.

The threat action, consequence paradigm allows a systematic approach to threat identification; however it doesn't solve the issue of detecting threats during runtime. Detecting threats during runtime in SERSCIS is based on observations of asset (mis-)behaviors (e.g. Inaccessible, Inaccurate, Indiscreet, Over-committed, Overloaded, etc.). The behavior classes are defined as part of the SERSCIS model, representing the (generic) effects of threat actions from Table 2 on different asset types (see Table 3). They are listed in Table 3. Observations of these behaviors must be generated by analyzing system monitoring events. This is system dependent, but usually involves setting thresholds for parameters like throughput, response time or rates for various types of failures, and using these to decide whether the behavior is or is not present. Our approach to threat identification (see Section5) means one need not monitor for all behaviors.

TABLE III. ASSET BEHAVIOURS

Behavior	Applies to											
	Space	**Host**	**Network**	**Interface**	**ServiceGroup**	**Customer**	**Consumer**	**CSResource**	**PSResource**	**Third-Party**	**ResourceGroup**	**NetworkGroup**
Dissatisfied: the asset isn't happy with its interactions with the system.					Y	Y	Y	Y	Y		Y	
Inaccessible: the asset fails to recognise authorized users	Y	Y	Y					Y	Y		Y	Y
Inaccurate: information exchanged with the asset contains errors.		Y	Y		Y	Y	Y	Y	Y		Y	Y
Indiscreet: the asset allows inappropriate access to its data.		Y	Y		Y	Y	Y	Y	Y		Y	Y
Overcommitted: The asset is promising more than it can deliver.					Y							
Overloaded: the asset is interacting more than it should.		Y	Y	Y	Y	Y	Y		Y		Y	Y
Promiscuous: the asset does not properly restrict access.	Y	Y	Y		Y			Y	Y		Y	Y
Unaccountable: the asset denies responsibility for its actions.						Y						
Unauthentic: the asset may not represent or be what it claims.		Y	Y		Y	Y	Y	Y	Y		Y	Y
Unauthorized: the asset doesn't have the rights it needs.						Y	Y					
Unavailable: the asset can't be contacted.		Y	Y	Y	Y	Y	Y	Y	Y		Y	Y
Underperforming: interaction with the asset takes too long.		Y	Y		Y	Y		Y	Y		Y	Y
Unreliable: the asset fails for internal causes.		Y			Y	Y	Y	Y	Y		Y	

Threats can obviously cause asset (mis-)behavior through their threat consequences. Typically one must analyze the observed behavior to deduce which threat is responsible as discussed in Section 5. Some types of asset behavior can also be part of the threat action, i.e. their detection indicates that a threat action is in progress. The SERSCIS threat model captures this via secondary effect rules. For instance a DoS attack on a resource causes it to become unavailable, which causes a service using the resource to underperform.

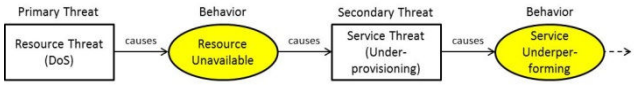

Figure 2. Secondary effect (threat behaviour) chain.

V. THREAT ACTIVITY ASSESMENT

The starting point for threat activity assessment was to assume we have a set of M observations $\mathbf{B}=\{B_i, i=1,\ldots,M\}$ of potential adverse behaviors in system assets. Each observation tells us whether a specific behavior is present or absent in a specific system asset. Note that this set only covers a subset of possible system behaviors. The rest are not observed, so we do not know whether they are present. The set **B** represents our evidence of threat activity in the system. At this point threats are considered independent, at the end of this section we explain how secondary effects capture chained threats.

We then suppose that the actual threat activity is represented by a vector of Boolean values $\mathbf{T}=\{T_j, j=1,\ldots,N\}$, covering all N potential threats in the system. Thus T_j=True if and only if threat j is active. Our problem is that we do not know which threat(s) are active, i.e. we do not know **T**. However, we do have an idea of how likely it is that each threat could be active, from our conventional design-time risk analysis. This is captured as a vector of prior probabilities $P(Tj|\varnothing)$, giving the expected probability that threat T_j is active given no evidence either way (i.e. the set of behavior observations is the empty set ∅). The prior probabilities reflect our initial ideas about how likely each threat is, i.e. they reflect our trust in the system's ability to counteract potential threats. But since we have some observations of the system, we can improve on this and find the current probability that each threat is active. This is simply $P(T_j|B)$, i.e. the current probability that threat T_j is active given the evidence of our observations **B**.

To calculate this, we can use Bayesian inference [26], which tells us that:

$$P(\mathbf{T}|\mathbf{B}) = P(\mathbf{T})\frac{P(\mathbf{B}\,|\,\mathbf{T})}{P(\mathbf{B})}$$

Here **T** is a specific set of true/false threat activity values. P(**T**) is the expected probability of that particular set of values. If threats are considered independent, this is just:

$$P(\mathbf{T}) = \prod_{j:T_j=\text{True}} P(T_j|\emptyset) \prod_{j:T_j=\text{False}} \{1 - P(T_j|\emptyset)\}$$

P(**B**|**T**) is the probability of getting our evidence (i.e. the M observations **B**) if the set of active threats is as specified by **T**. Finally, P(**B**) is the expected probability of getting those observations given no assumptions about which threats are active. If we sum over all 2^N possible values of **T**, the law of total probability gives this as:

$$\mathrm{P}(\mathbf{B}) = \sum_{\mathbf{T}} P(\boldsymbol{T}) \cdot P(\boldsymbol{B}|\boldsymbol{T})$$

If we want to know P(T_j|**B**), we have to sum P(**T**|**B**) over all combinations of active threats **T** in which T_j is active:

$$\mathrm{P}(T_j|\mathbf{B}) = \sum_{\mathbf{T}:\, T_j=\mathrm{True}} \mathrm{P}(\mathbf{T}|\mathbf{B})$$

$$\mathrm{P}(T_j|\mathbf{B}) = \sum_{\mathbf{T}:T_j=\mathrm{True}} \mathrm{P}(\mathbf{T})\cdot\mathrm{P}(\mathbf{B}|\mathbf{T}) \Big/ \sum_{\mathbf{T}} \mathrm{P}(\mathbf{T})\cdot\mathrm{P}(\mathbf{B}|\mathbf{T})$$

Everything here is easily calculated except for P(**B**|**T**) and the fact that we have to sum over all possible combinations of threat activity values **T**, i.e. we need to calculate 2^N terms, which rapidly becomes intractable. The usual way to deal with the exponential number of terms is to sample the possible threat activity configurations **T**. We will return to this shortly.

To calculate P(**B**|**T**), we assumed that each potentially observable behavior could be caused in one of two ways:

i) behavior i could be triggered by threat j, with probability P($T_j \rightarrow B_i$); or
ii) behavior i could arise spontaneously in the absence of any threat with probability P(B_i|$\mathbf{T_0}$).

where $\mathbf{T_0}$ represents the case of no active threats (i.e. all Tj are False). Spontaneous behavior was included in the model to allow for the possibility of false alarms, in which a behavior is detected due to monitoring errors or random variations in asset behavior that may look like misbehavior.

The contributions to adverse behavior causation P(Tj→Bi) and P(B_i|$\mathbf{T_0}$) were added to the core ontology from Fig. 1 by including further relationships, as shown in Fig. 3.

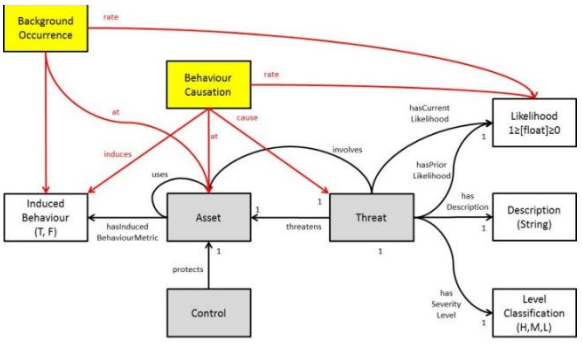

Figure 3. Core model behaviour causation extensions.

The results were relatively insensitive to the rates of threat triggering of behavior, provided the rates are close to 1.00 or 0.00 when they should be and not otherwise. However, this varies from system to system, so some tuning of P($T_j \rightarrow B_i$) is needed. For example, whether a threat causes underperformance depends on whether the compromised process is on the critical path, so this has to be set for system-specific threat classes. Some tuning of P(B_i|$\mathbf{T_0}$) was also needed to match the sensitivity and false alarm rates of the system-specific monitoring probes used. For this reason, the rates were not added directly to the semantic model, but stored in a separate human-readable spread sheet while this tuning process was carried out. We did not try to automate the tuning process using machine learning techniques similar to those used for Internet traffic [27], although this may be possible in future.

Now, if we make an observation to detect a given behavior, it should be present unless none of the possible causes triggered it. It is convenient to define the probability P(¬B_i) of not finding behavior i, which is given by:

$$\mathrm{P}(\neg B_i) = \{1 - \mathrm{P}(B_i|\mathbf{T_0})\} \prod_{j:T_j=\mathrm{True}} \{1 - \mathrm{P}(T_j \rightarrow B_i)\}$$

We can then calculate P(**B**|**T**) from:

$$\mathrm{P}(\mathbf{B}|\mathbf{T}) = \prod_{j:T_j=\mathrm{True}} \{1 - \mathrm{P}(\neg B_i)\} \prod_{j:T_j=\mathrm{False}} \mathrm{P}(\neg B_i)$$

These equations represent the Bayesian belief graph used in the Bayesian inference procedure. The software used the semantic model to determine which threat instances are related to which asset behaviors (i.e. to create the belief graph), and then looked up the corresponding probabilities taken from the associated spread sheet. Note that in [22], an ontology is used to compute a Bayesian belief graph which is then used in the forward direction to compute how likely it is that threats will be active and have consequences (including triggering secondary threats) given the available controls. We use the belief graph in the reverse direction, to deduce from the consequences which threats are active.

Finally, we come to the sampling strategy used to estimate the sum of terms in the numerator and denominator used to compute the posterior estimates of threat activity P(T_j|**B**). It is computationally infeasible to compute all 2^N terms given that a typical concrete system model may have thousands of threat instances (our A-CDM model peaked at 2,647 threat instances). One might use an Approximate Bayesian Computation (ABC) sampling strategy [28] in which terms are sampled based on an estimate of their contribution derived from previous samples. In our case it was clear that the chances of threats being active (i.e. causing adverse behavior) is low given that the systems of interest (such as airports) do employ extensive countermeasures. We therefore initially considered terms in which only zero, one or two threats were active at any time, and obtained reasonably good results with a simplified ontology [29], [30]. However, this approach wasn't effective if one threat was likely to cause another. This would make it likely that more than one threat will be active (a situation we did not sample), and undermine the assumption of independence used to calculate P(**T**).

In the final version, any threats classified by semantic reasoning as secondary (i.e. triggered by observed asset misbehavior, as a knock-on consequence of some other primary threat) are considered to be active by definition. We then sample the subspace of T by varying the status of all

remaining non-secondary (i.e. potential primary) threats. In effect, each sample then represents a set of correlated active threats, but different samples are independent of each other, since the differences between them come only from the independent primary threats. In effect, this means we use Bayesian inference only to find primary threats, as the secondary threats have already been identified by semantic reasoning and excluded from the Bayesian search:

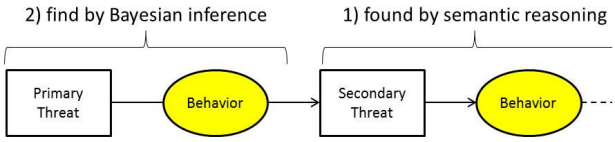

Figure 4. Sampling stragegy.

This approach was well suited to our purpose because:

- while the estimates for $P(T_j|\mathbf{B})$ are quite rough, comparison with prior probabilities $P(T_j|\varnothing)$ gives a clear signal whether observed behaviour supports the hypothesis that threat j is active;
- the secondary effect chain extends only as far as can be explained by observed behavior, so we do not need extra parameters modeling threat propagation probability;
- the run-time is kept to a minimum, which is a significant benefit given that Bayesian estimation is being used 'in the loop'.

The restriction to sampling at most one primary plus all secondary threats allowed the total run-time to be kept within 1 minute, while the number of threats increased from under 100 to well over 2000.

VI. EVALUATION

A. Evaluation Scenarios

The evaluation was carried out using a simulation of a typical A-CDM installation based on a subset of the Vienna Airport system. It was necessary to simulate the test system because A-CDM was not deployed at Vienna until near the end of the SERSCIS project, and also because it was not acceptable to inject faults into the real airport in order to test our threat detection and diagnosis approach. Details of the simulated airport are described in [24]. Here we focus on the attack scenarios simulated, and the results obtained from the semantically-driven Bayesian inference procedure.

The basic structure of the simulated airport is illustrated by the logical assets and relationships from Fig. 5.

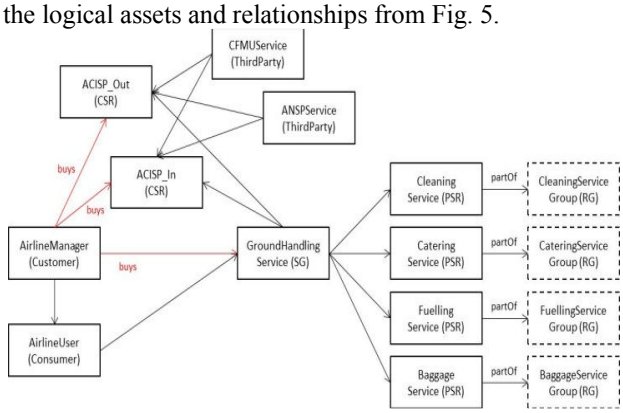

Figure 5. Logical assets in the airport system model.

For the sake of simplicity, Fig. 5 shows only logical services, without any of the physical assets involved (airport communication networks and service providers). The model represents the A-CDM network from the point of view of a ground handling service provider. The ground handler is responsible for coordinating services such as refueling, catering and baggage handling provided to the aircraft during its stopover, and updating the A-CDM data exchange service provided by the airport's ACISP. This service is also used by local air traffic control services, and by the Central Flow Management Unit (CFMU) operated by EUROCONTROL.

This set up was used to simulate six scenarios:

- unauthorized rescheduling of the refuelling service by a malicious activist impersonating a ground handler;
- jamming of radio communications between the ground handling service operator and the aircraft stand;
- a remote denial of service attack on a fuelling service over a connecting network;
- a worker strike at an airport service provider causing a slow-down in aircraft refuelling services;
- a malfunction in the ground handler's software for predicting aircraft turn-around schedules.
- a 'sunny day' in which there are no active threats and the airport operates normally with only random variations in turn around service provision;

All but the last scenario represents a different active threat, most of which also involve knock-on consequences. The aircraft arrival and target take-off times were taken from 124 flights passing through the real Vienna Airport on a day in July 2010.

B. Key Performance Indicators

To assess the impact of the threats on the simulated airport operations, we considered how they affect key performance indicators (KPI) defined by Austrocontrol (the national air traffic control service provider). These KPI included the number of aircraft taking off outside the slot tolerance windows, and the average error in predicted aircraft ready times, known as target off-block times (TOBT) issued when the aircraft is about to land.

TABLE IV. SIMULATED KPI

KPI	Scenario					
	1	2	3	4	5	6
Take-off outside slot	0%	0%	31%	0%	99%	0%
Average TOBT error	1m	57m	94m	11m	19m	0m

It is clear that all the simulated attacks had some impact on these KPI. Scenario 1 has the least effect, simply because rescheduling the fuelling service can usually send a spare crew when the ground handler complains that they are overdue. Only at peak times were there no spare crews available in the simulation, so only a few flights suffered any serious delays. In scenarios 1, 2 and 4 the ground handler was able to predict the consequences of the attack well enough to avoid aircraft taking off at the wrong time. In the remote denial of service and software malfunction scenarios this was not possible, so aircraft left the stand at the wrong time. (Normally air traffic control would prevent them taking off, but our simulation did not cover what

happened after the ground handler finished delivering the turn-around service).

While each attack produced a different effect on the KPI, apart from Scenario 1 this depended on when mitigating action was taken, so it isn't in general possible to deduce the cause of the problem from the KPI. However, using the semantically-driven Bayesian inference approach, it was possible to diagnose the root causes.

C. Threat Detection

In Scenario 1, the first adverse behavior detected is the fact that the fuelling service disagrees with the ground handler as to how often it is being updated. At this point, the SERSCIS tools reported four threats with higher than expected likelihood, i.e. $P(T_j|\mathbf{B}) > P(T_j|\varnothing)$ where $P(T_j|\varnothing)$ is typically around 1%:

- unauthorized access to data by impersonation of the ground handler (28.5%)
- unauthorized access to data due to faulty access control at the fuelling service (28.5%)
- unauthorized update to data due to faulty access control at the fuelling service (28.5%)
- unauthorized update of data by impersonation of the ground handler (14.1%)

Initially the fuelling service copes with the unauthorized rescheduling by substituting spare crews, leading the ground hander to assume the schedules are still accurate. The last two threats then drop off the list returned by the Bayesian estimator, leaving just the two possible reasons why the fuelling service is being indiscreet (which is still detectable). When the airport gets busy, the fuelling service can no longer honor the original schedule, and the ground handler detects that there must be a data corruption. At this point, the Bayesian activity estimator reports only two threats as possible causes:

- unauthorized update to data due to faulty access control at the fuelling service (65.5%)
- unauthorized update of data by impersonation of the ground handler (32.8%)

At this point a third threat is also identified as being active, but this is because it has been classified as a secondary effect: persistent inaccuracy in one of the available fuelling services. It is useful to note that at this point, if secondary effect classification were not used, we would have seen a much lower $P(T_j|\mathbf{B})$ for the above pair of possible root causes, and non-negligible increases in $P(T_j|\mathbf{B})$ relative to $P(T_j|\varnothing)$ for several other threats based on adverse behavior in the ground handler and in the data passed to the A-CDM service provider, etc. However, because these can all be caused by persistent inaccuracy in a fuelling service, they are automatically accounted for, and the Bayesian inference procedure does not get distracted from possible root causes. As far as the authors know, semantic and Bayesian inference have never before been combined in this way to enable root cause analysis using monitoring data from a set of interdependent system components.

Throughout this scenario, the machine reasoning tools were able to produce results within 1-2 minutes of the behavior changes being detected, despite the fact that the airport simulation and the simulated monitoring system plus semantic and Bayesian inference tools were all running in the same virtual machine on a single CPU.

Clearly, as with any data-driven methodology, the SERSCIS tools cannot diagnose a threat correctly until they have enough observations of asset misbehavior. In this case that was not until the effect of the attack on the fuelling service provider had become significant. However, even when only some misbehavior was evident, the inference method produced a shortlist that included the actual cause, plus with other causes whose investigation would have led the operator in the right direction (in this case, the obvious first step would be to call the fuelling service provider to find out what is happening). We did not simulate the effect of operator action to mitigate the threat, but in this case it may have been possible to fix the errant access control policy before the airport became busy. In the worst case some of the load could have been diverted to an alternate fuelling service provider.

The other scenarios show a similar pattern to Scenario 1. In Scenario 2, the SERSCIS tools identified the jamming attack on the ground handler's internal radio network as the primary active threat as soon as the first misbehavior was detected (the network became unavailable). In this case, a secondary effect is also identified, which accounts for the knock-on consequence that the ground handling service becomes unavailable. This ensures other possible causes of the service being unavailable (e.g. lack of available fuelling services) are not highlighted as possible active threats. In Scenario 2, the posterior probability of the jamming attack is only 15.7% due to the poor sampling strategy, but because it is the only threat for which $P(T_j|\mathbf{B})$ is boosted above $P(T_j|\varnothing)$, the signal is clear enough to an operator.

In Scenario 3, we get a similar effect: the correct attacks are identified as root causes even though $P(T_j|\mathbf{B})$ remains well below 50%. Two different variations were used for this scenario, one in which the remote attacker exploited a software bug to tie up the service itself, and one where the attacker used a packet flooding attack to deny service. Both cases produce the same set of secondary effects accounting for various knock-on effects (starting with the fuelling service being unavailable). Including these in all Bayesian samples allows correct identification of the primary cause in each case.

In Scenario 4, the fuelling service retains some availability but the loss of most of its workers to a wildcat strike makes it underperform. From the ground handler's point of view the cause of this is internal to the fuelling service provider, and the Bayesian inference simply highlights an underperforming fuelling service threat. After some time the effects lead the ground handler to underperform, and this is tracked by the SERSCIS tools by an increase in the chain of misbehavior and threats classified as secondary effects.

Finally in Scenario 5, the software malfunction leads the ground handling service to produce inaccurate TOBT predictions. The Bayesian inference procedure highlights four possible causes:

- ground handing service software malfunction (18.3%);
- impersonation of a back-up fuelling service to the ground handler (13.7%)
- inaccuracy in the back-up fuelling service (10.1%); and
- unauthorized update of ground handing service data (9%).

The first threat is the correct root cause diagnosis, and it is also the one found to be the most likely. The other three

appear because they too could account for the inaccurate predictions, and in this case we have no evidence ruling them out. The two threats involving a back-up fuelling service appear because in this scenario, there is no problem with the first-choice fuelling service and the back-up is never used. As a result, even though monitoring probes are in place to detect authentication failures and inaccuracy, they are never triggered so no observations can be included in the set **B** used by the Bayesian inference system. The last threat highlighted represents the possibility that input data is being corrupted by an unauthorized party, and this arises because our simulation did not include any reporting from customers (i.e. airlines) on the number of updates they had made. Consequently it was not possible to monitor whether any excess update requests were received and processed by the ground handling service.

This last scenario shows that in some situations one should make assumptions about at least some types of adverse behavior when no observations are available. Our approach has been to assume an observation remains valid until we get a new observation (i.e. we take the last received system monitoring event). This fails when a new component is introduced for which there are no prior events, or where the system cannot provide the necessary monitoring data. This is one area we plan to investigate in future.

Finally, Scenario 6 was tested just to check whether our system was likely to cause false alarms. In fact, no threats were ever highlighted in any test using the sunny day scenario, even though some flight delays were seen. This is due to the fact that some level of random misbehavior detection is built into the model through $P(B_i|\varnothing)$.

D. Comparision with SESAR

It is worth considering how the overall SERSCIS procedure differs from a conventional ISO 27000 compliant methodology. The comparison was made between our approach and SESAR for two reasons: SESAR is ISO 27001/27005 compliant and it is applied in the same context as ours i.e. Air Traffic research and development [8]. The comparison includes the design-time modeling processes summarized in Section 3 as well as the run-time elements described in Sections 4 and 5. The results are summarized in Table 5

TABLE V. COMPARISION WITH SESAR METHODOLOGY

Step	Original SESAR procedure	SERSCIS variation
Threat identification	Design-time analysis by a security expert.	Design-time automated threat class generation based on expertise encoded in a generic threat model.
Threat impact severity assessment	Original SESAR procedure adopted in SERSCIS. Design-time analysis based on a consideration of impact on primary and secondary assets.	
Threat prior likelihood	Design-time expert analysis of attacker motives/abilities in the presence of security controls.	Rough estimates are sufficient, may be based on historic attack frequency data if available.
Threat current likelihood	Not available: static design-time analysis only.	Dynamically inferred at run-time based on observed system behaviour.
System vulnerability detection.	Design-time expert analysis based on the planned security controls.	Run-time analysis based on actual security controls used by dynamically composed system assets.

The key differences are in threat identification stage and the current threat activity likelihood. In the conventional methodology, a security expert is involved in analyzing the specific system under consideration. In the SERSCIS approach, this is not necessary. Instead, the security expert creates a generic model that can be used by a system designer in an automated threat identification process. The conventional methodology then uses the design-time expert analysis as a basis for assessing run-time threat activity and system vulnerabilities. In SERSCIS these are updated at run-time using information from run-time monitoring of the system configuration and behavior. Dynamic changes can only be addressed in the conventional methodology by updating the original design-time analysis for each change. Since this involves humans it cannot be done 'in the loop' which is why SERSCIS makes such extensive use of machine reasoning so the process can be fully automated. The results of this run-time analysis are then presented to the user via a decision support GUI described in [30].

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a run-time dynamic approach to risk management for use in dynamic, multi-stakeholder ICT systems as part of the SERSCIS project. A semantic threat modelling approach was used to support design time threat identification and mitigation strategies. Semantically driven Bayesian inference enables the run-time analysis of threats and allows us to perform root cause analysis for threats. We have provided the results of our approach when applied to various scenarios simulating threats within an airport ICT system. We have also compared and contrasted the improvements of our approach over existing risk management procedures like SERSAR. In the future, we plan to extend the modelling work as a basis for relating socioeconomic trust in ICT systems as part of the OPTET project.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement no. 225336, SERSCIS and the EU funded OPTET project.

REFERENCES

[1] EUROCONTROL, "Airport CDM Implementation Manual, Version 3," 12 2008. [Online]. Available: www.euro-cdm.org/library/cdm_implementation_manual.pdf. [Accessed 10 2009].

[2] J"The Single European Sky ATM Research Programme (SESAR)," [Online]. Available: http://publish.eurocontrol.int/content/sesar-and-research. [Accessed October 2011].

[3] H. Moyano, "EU R&D Into Port Technologies," *Port Technology International,* vol. 42, pp. 20-22, 2009.

[4] European Commission, *Council Directive 2008/114/EC on the identification and designation of European critical infrastructures and the assessment of the need to improve their protection.,* 2008.

[5] *ISO/IEC 27001:2005. Information technology – Security Techniques – Information security management systems – Requirements,* International Organization for Standardization, 2005.

[6] *ISO/IEC 27005:2011. Information technology -- Security techniques -- Information security risk management,* International Organization for Standardization, 2011.

[7] "SERSCIS: Semantically Enhanced Resilient and Secure Critical Infrastructure Services," [Online]. Available: http://www.serscis.eu.

[8] J. Touzeau, E. Hamon, M. Krempel, B. Gölz, R. Madarasz and J. Alemany, "SESAR DEL16.02.01-D03: SESAR ATM Preliminary Security Risk Assessment Method.," 2011.

[9] Club de la Securite de l'Information Francais (CLUSIF), "MEHARI 2010 Risk Analysis and Treatment Guide," 08 2010. [Online]. Available: http://www.clusif.asso.fr/fr/production/ouvrages/pdf/MEHARI-2010-Risk-Analysis-and-Treatment-Guide.pdf. [Accessed 02 2013].

[10] Siemens Enterprise, "CRAMM v5.1 Information Security Toolkit," [Online].Available: http://www.cramm.com/files/datasheets/CRAMM%20%28Datasheet%29.pdf. [Accessed 02 2013].

[11] I. El Fray, "A Comparative Study of Risk Assessment Methods, MEHARI \& CRAMM with a New Formal Model of Risk Assessment (FoMRA) in Information Systems," *Computer Information Systems and Industrial Management,* pp. 428-442, 2012.

[12] C. J. Alberts and A. J. Dorofee, Managing information security risks: the OCTAVE approach, Addison-Wesley Professional, 2003.

[13] R. A. Caralli, J. F. Stevens, L. R. Young and W. R. Wilson, "Introducing octave allegro: Improving the information security risk assessment process," Carnegie-Mellon University, 2007.

[14] Tropos project, [Online]. Available: http://www.troposproject.org. [Accessed 2012].

[15] R. Matulevius, N. Mayer, H. Mouratidis, E. Dubois, P. Heymans and N. Genon, "Adapting secure tropos for security risk management in the early phases of information systems development," in *Proceedings of the 20th international conference on Advanced Information Systems Engineering (CAiSE '08)*, Montpellier, France, 2008.

[16] I. Hogganvik and K. Stølen, "A graphical approach to risk identification, motivated by empirical investigations.," in *Proceedings of the 9th international conference on Model Driven Engineering Languages and Systems (MoDELS'06)*, Genova, Italy, 2006.

[17] C. Blanco, J. Lasheras, E. Fernandez-Medina, R. Valencia-Garcia and A. Toval, "Basis for an integrated security ontology according to a systematic review of existing proposals," *Comput. Standards & Interfaces,* vol. 33, no. 4, pp. 372-388, 2011.

[18] A. Kim, J. Luo and K. M. , "Security ontology to facilitate web services description and discovery," *Journal on Data Semantics,* vol. 9, pp. 167-195, 2007.

[19] A. Vorobiev and N. Bekmamedova, "An ontology-driven approach applied to information security," *Journal of Research and Practice in Information Technology,* vol. 42, no. 1, pp. 61-76, 2010.

[20] S. Fenz and A. Ekelhart, "Formalizing information security knowledge," in *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security (ASIACCS'09)*, Sydney, Australia, 2009.

[21] German Federal Office for Security in Information Technology (BSI), "IT Grundschutz Manual," 2005.

[22] S. Fenz, "An ontology- and Bayesian-based approach for determining threat probabilities," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS '11)*, New York, USA, 2011.

[23] A. Chakravarthy, M. Surridge, X. Chen, M. Hall-May, B. Nasser and T. Leonard, "SERSCIS Deliverable D2.2: System Modelling Ontology and Tools: Final Implementation v1.5," 01 2013. [Online]. Available: http://www.serscis.eu/wp-content/uploads/2013/02/SERSCIS_D2.2-v1.5.pdf. [Accessed 01 2013].

[24] M. Surridge, A.Chakravarthy, M. Hall-May, B. Nasser and R. Nossal, "SERSCIS: Semantic Modelling of Dynamic, Multi-Stakeholder Systems," in *Proceedings of the SESAR Innovation Days (2012)*, Braunschweig, Germany, 2012.

[25] R. Shirey, "RFC 2828: Internet Security Glossary," May 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2828.txt. [Accessed August 2012].

[26] A. Gelman, J. B. Carlin, H. S. Stern and D. B. Rubin, Bayesian Data Analysis, Second Edition, Boca Raton, FL, USA: Chapman and Hall/CRC, 2003.

[27] T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys and Tutorials,* vol. 10, no. 4, pp. 56-76, 2008.

[28] T. Toni, D. Welch, N. Strelkowa, A. Ipsen and P. H. M. Stumpf, "Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems," *Journal of the Royal Society Interface,* vol. 6, pp. 187-202, 2009.

[29] M. Surridge, "Semantic Security Modeling for Run-Time Threat Analysis," in *The Cyber Security & Privacy EU Forum, 2012*, Berlin, Germany, 2012.

[30] D. Kostopoulos, G. Leventakis, V. Tsoulkas and N. Nikitakos, "An Intelligent Fault Monitoring and Risk Management Tool for Complex Critical Infrastructures: The SERSCIS Approach in Air-traffic Surface Control.," in *14th International Conference on Computer Modelling and Simulation (UKSim)*, Cambridge, UK, 2012.

[31] N. Poolsappasit, R. Dewri and I. Ray, "Dynamic Security Risk Management Using Bayesian Attack Graphs," *IEEE Transactions on Dependable and Secure Computing,* vol. 9, no. 1, pp. 61-74, 2012.