A Tool for Evolutionary Threat Analysis of Smart Grids

Tommaso Zoppi(✉), Andrea Ceccarelli, and Marco Mori

University of Florence, Viale Morgagni 65, Florence, Italy
{tommaso.zoppi,andrea.ceccarelli,ma.mori}@unifi.it

Abstract. Cyber-security is becoming more and more relevant with the advent of large-scale systems made of independent and autonomous constituent systems that interoperate to achieve complex goals. Providing security in such cyber-physical systems means, among other features, identifying threats generated by novel detrimental behaviors. This paper presents a tool based on a methodology that is intended to support city evolution and energy planning with a focus on threats due to novel and existing interconnections among different components. More in detail, we report a tool demonstration which shows the application of a tool devised to (i) deal with security threats arising due to evolutions in a Smart City - intended as a complex cyber-physical system -, and (ii) consequently perform threat analysis.

Keywords: Threat analysis · Smart Grids · Evolution · IRENE

1 Threat Analysis for Planned Evolution

Most of the approaches supporting the achievement of safety and security requirements are based on threat analysis processes that are mostly intended for static scenarios, with limited inherent solutions to support the planned evolution of infrastructures and especially of Smart Cities. Tackling evolution of Smart Cities as main challenge, in [3, 4] a threat analysis methodology was identified that lists the occurring threats through an incremental approach. Concisely, the methodology is based on both observing changes in the Smart City topology and performing a threat analysis on detected changes. The focus is on: (i) new threats that can arise from the addition of new components (either buildings or connections), and (ii) threats that are no longer affecting the Smart City due to some topology changes. This methodology is intended to support city evolution and energy planning with a focus on threats due to interconnections among different components of the grid.

The methodology was developed within the project IRENE [7, 2], which is focused on collaborative city planning for resilient energy management. The project investigates the interplay and coordination of social, economic and technical components to improve robustness of the urban electricity network.

Starting from the methodology we presented in [3] *in the context of the IRENE project, this paper is intended to complement the live demonstration of the threat analysis tool that will be performed at the workshop IRENE @ SmartGift 2017.*

E.T. Lau et al. (Eds.): SmartGIFT 2017, LNICST 203, pp. 205–211, 2017.
DOI: 10.1007/978-3-319-61813-5_20

2 Background: Evolutionary Threat Analysis (ETA)

In order to provide safety and security requirements for Smart Grids, it is essential to analyze the interdependencies regulating the flow of information among constituent systems. Their interactions and interdependencies may generate cascading effects, which represent possible security threats and damages. To avoid such situations, the transient threat analysis should be supported by new approaches that are able to deal with cascading contingency chains revealing the effect of evolving the grid.

The evolutionary threat analysis described in [3, 4] adopts the guidelines defined from NIST in the SP 800-30 [1] regarding both the approach to follow and the main steps to perform and validate the risk assessment process. In particular, it follows an *asset-oriented* approach as defined in the NIST standard, by identifying threat events depending on critical assets of the grids, i.e., the internal behavior of a component (e.g., a hospital) and their possible interactions. It supports an incremental threat identification process that is carried out after the grid evolution. Starting from the identification of impacts or consequences of the addition/removal of assets, the approach identifies the threats and/or the vulnerabilities that can arise due to this scenario's evolution. Consequently, the mitigation strategies to apply/remove are identified according to the traceability of their threats. The methodology is not reported here for brevity; please refer to [3, 4] for more details.

3 Background: Tool Design, Implementation and Configurations

The methodology in [3, 4] has been implemented in a tool.

Design Choices. According to the purpose of the IRENE project, the tool is intended to support city planners when they have to plan – or to assess – an evolution of the existing grid that contributes to provide smart services in the near future. Anyway, evolving the grid leads to modifications that can introduce new threats or vulnerabilities (e.g., a substation that is fundamental for critical grid components) as well as architectural changes that need to be supported by the whole grid (e.g., energy rebalancing due to a failure in a connection or a generic component).

Moreover, considering threats happening in a scenario as a formal relation between two components (i.e., threats and scenarios) allows viewing the results of the threat analysis as a Formal Concept Analysis (FCA) structure [8]. This view established hierarchical relations between scenarios depending on threats.

Implementation Choices. We describe here our implementation choices.

- *Language*. We choose *Java* as tool platform since it is not OS-dependent and other tools in the IRENE [7] toolset were developed with the same language. This will help the future integration of the single tools to build a unique toolset.
- *Interface and I/O*. The tool has not a graphical interface since it should be used in cooperation with other tools that offer a graphical user interface. However, the tool can be considered as a standalone resource that has its inputs and outputs into text

files. This allows a simple integration with other tools that can read and write the input and output files to tune the preferences of the threat analysis tool according to their actual needs.

- *Performance*. The complexity of the threat analysis implemented in this tool is not so relevant to require deep performance analysis. However, during the CPU-intensive phase – while threats for each evolution step are listed – the tool executes the most expensive tasks in dedicated threads, to not lock the main thread responsible to collect the outcomes of the created threads. This will increase the performances of the tool in workstations where several (physical or virtual) CPUs are available. The management of such threads is left to the *Java* scheduler that runs a preemptive priority-based scheduling algorithm.

Complexity. The tool performs satisfactorily with the scenario's inputs, given that it is polynomial with respect to the inputs. Consequently, we expect to have an acceptable scalability with larger scale Smart Grids.

Code characteristics. The implemented code was checked to obtain quality metrics in order to give an overview on its complexity and on how it is written, that we partially report in Table 1 Moreover, we refactored the code to eliminate the code flaws identified by *FindBugs* [5]. *FindBugs* was tuned to identify the following bug categories: security flaws, bad practices, dodgy code, and multi-threading correctness.

Table 1. Quality metrics values for the ETA source code

Metric			Results		
Code	Name	Detail	Average	St. Dev	Max
WMC	Weighted methods per class	Type	18.571	14.783	52
NOM	Number of methods	Type	6.571	7.178	29
MLOC	Method lines of code	Method	8.229	9.164	39

4 Tool Demo: Define Inputs, Outputs, Execution

We exercise the tool on a simple evolving Smart City (Smart Grid) scenario. We consider as possible ways to evolve the grid i) a simple change of the topology (e.g., the addition of a wire), and ii) a set of evolutionary features that lead to changes in the grid functionalities (see [3, 4]).

Inputs. The tool requires the following inputs: (i) a list of threats, (ii) a list of threat categories, (iii) a list of mitigations, (iv) a list of grid components, and (v) a (set of) scenario including a grid topology. More in detail, the *list of threats* and *threat categories* define the threat model, or rather the threats that the user is investigating for the specific study (e.g., cyber-security, environmental, etc.). The *mitigations* are a set of strategies

that can be instantiated and implemented on a specific grid to mitigate or avoid the detrimental effects of the happening of the threats. This information is therefore used to analyze the *grid scenario* the user wants to analyze, which can be either a brand new one or an evolution of a previously analyzed grid scenario. In both cases, the grid scenario is defined as a grid topology that is composed by the components in the *components list*, complemented with assumptions about the city scenario under investigation (e.g., seismic zone, prone to terrorism, etc.).

Outputs. The output of the tool is provided both in terms of a list of identified threats and a Formal Concept Analysis (FCA) file. In particular, it consists in a set of files listing the threats identified in each input scenario along with the mitigations that can be applied accordingly. The relations between evolutions of the same scenario are highlighted through the usage of the FCA output, which helps defining a hierarchy for the involved grid scenarios. The hierarchy is based on the threats that can arise in each scenario: a grid scenario is an ancestor of another one if it is interested by a subset of threats that occur in the child scenario.

Dependencies. The tool is written in Java 8. However, a compatible version is available for Java 7. Moreover, one of the outputs of the tool is a *.cex* file containing the result of the FCA. This needs to be opened by dedicated tools such as *ConExp* [6], which allow visualizing the FCA through the support of the *Colibri-Java* API.

Running the Tool. The tool is an executable *.jar* that can be run via command line on *Windows*, *OSX* and *UNIX* systems invoking the *Java Virtual Machine* with "java –jar <pathname>/WP2_ThreatAnalysis.jar". The tool is compiled with the current standard version of *Java* (*Java* 8); therefore, it cannot be run on systems where *Java* is not installed or if *Java* 7 or previous versions are installed.

5 Tool Demo: Results from Execution

Here we report the results of an execution of the tool using the scenario in Fig. 1, and the threats, mitigations and evolutionary steps described in [3, 4]. First, inputs are read. In the example, the tool shows that 38 threats, 19 mitigations and 120 possible novel interdependencies were loaded.

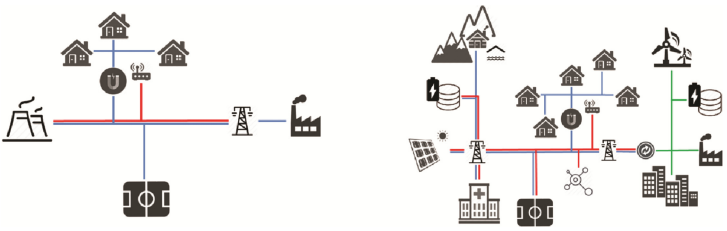

Fig. 1. Evolution steps from [3]: 0_InitialScenario (left) and 5_BuildingIndustrialDistrict (right)

Then, all the considered evolving scenarios are loaded and each of them is analyzed incrementally. In the example, 8 evolutions of the initial grid are considered: an initial definition of the grid ("*0_initialScenario*") and 8 temporal evolutions, leading the grid being exposed to 494 threats (414 structural threats and 80 threats due to novel interdependencies) in its final state.

In Table 2 we can observe a summary of the threat analysis executed on the scenarios mentioned above. In the first scenario all the components - and, consequently, the threats - are new, while other steps add or remove components from the previous scenario. The threats related to the scenarios changes accordingly: considering "*5_BuildingIndustrialDistrict*" we can observe that 2 components are added and 4 are removed, totalizing 384 structural threats and 81 threats due to novel interdependencies. We observed that 8% of this threats are new with respect to the previous "*4_InseringStorages*" scenario due to the addition of 2 components, while 17% of threats that affected the "*4_InseringStorages*" where eliminated due to the removal of 4 components.

Table 2. Threat analysis summary

Scenario	Components				Threat statistics (%)			
	Prev	Add	Del	Tot	Structural	Dependencies	Add	Del
0_InitialScenario	0	15	0	15	91	9	100	0
1_DiscoveringResources	15	4	0	19	88	12	26	0
2_GrowingNumberOfPeople	19	6	0	25	86	14	28	0
3_AddingKeyBuildings	25	4	2	27	85	15	14	6
4_InseringStorages	27	6	0	33	81	19	24	0
5_BuildingIndustrialDistrict	33	2	4	31	83	17	8	17
6_InsertionSCADA	31	2	2	31	83	17	8	9
7_InstallingMicroGrids	31	0	0	31	83	17	0	0
8_ImprovingDecarbonisation	31	2	0	33	84	16	6	0

Figure 2 shows the graphical interface to specify/import the input threats of each scenario and the results of the FCA. The latter consists in a graph where large nodes are evolution steps (i.e., scenarios), represented according to a hierarchical view. In particular, we can see how scenarios 6 and 7 are represented in the same point since they are threatened by the same items. The only change here is the substitution of the Basic Data Centre (BDC) component with a SCADA that is not threatened by any additional item. Lastly, we can observe that a hierarchy is established between two or more scenarios if the evolution step only adds threats (as for scenarios 0, 1, 2); in other cases we may have both to consider additional threats and to discard others thus changing the base threats set (as for scenarios "*4_InseringStorages*" and "*5_BuildingIndustrialDistrict*" previously discussed).

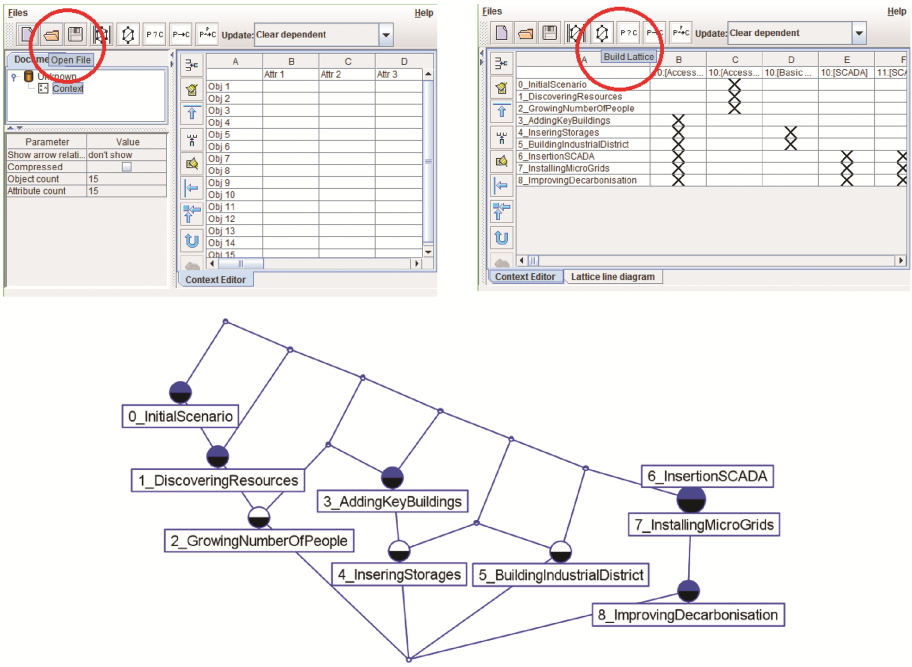

Fig. 2. Steps to build a graphical lattice with *ConExp* FCA tool: loading file, showing components, and depicting the hierarchical relationships among scenarios

6 Concluding Remarks

Tackling evolution of Smart Cities as main challenge, within the IRENE project we identified a threat analysis methodology that was implemented into a tool. *This paper described the tool above, complementing the live demonstration of the threat analysis tool that will be performed at the workshop IRENE @ SmartGift 2017.* Concisely, the novelty of the tool we presented relies in (i) identifying new threats that can arise from the addition of new components, (ii) build formal threats-scenarios and scenario-scenario relationships, and (iii) propose suitable high-level mitigations.

We lastly underline that the usefulness of the tool – which is assessed based on feedbacks of users as presented in [9] - are strictly related to the set of threats and mitigations that are chosen. For example, depending on the targeted scenario, a list of threats containing more threats due to physical attacks can be more representative than the IRENE list, which is mainly based on threats to cybersecurity.

Acknowledgments. This work has been partially supported by the projects JPI Urban Europe IRENE, FP7-ICT-2013-10-610535 AMADEOS and FP7-IRSES DEVASSES.

References

1. Grid, NIST Smart. Guide for Conducting Risk Assessments. Special Publication 800-30, September 2012
2. Jung, O., et al.: Towards a collaborative framework to improve urban grid resilience. In: 2016 IEEE International Energy Conference (ENERGYCON), Leuven, pp. 1–6 (2016)
3. IRENE, D2.1 – Threat Identification and Ranking (2015). http://ireneproject.eu/public-deliverables/
4. Mori, M., Ceccarelli, A., Zoppi, T., Bondavalli, A.: On the impact of emergent properties on SoS security. In: 2016 11th System of Systems Engineering Conference (SoSE), Kongsberg, pp. 1–6 (2016)
5. Ayewah, N., et al.: Using FindBugs on production software. In: Companion to the 22nd ACM SIGPLAN Conference on Object-Oriented Programming Systems and Applications Companion (2007)
6. ConExp. https://code.google.com/archive/p/colibri-java
7. IRENE project, Improving the Robustness of Urban Electricity Networks. http://ireneproject.eu/
8. Ganter, B., Wille, R., Wille, R.: Formal Concept Analysis. Springer, Heidelberg (1999)
9. Alexandr, V., et al.: Towards security requirements: iconicity as a feature of an informal modeling language. In: 3rd International Workshop on Requirements Engineering for Self-Adaptive and Cyber Physical Systems (RESACS) (2017, to appear)