universidad de león

RIaSC

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, ENERGÍA Y TURISMO

incibe_
INSTITUTO NACIONAL DE CIBERSEGURIDAD

Multigraph Project: First Steps towards the Definition of a Multiple Attack Graph Model Simulator

Mattia Zago[§], Juan José Andreu Blázquez[†], Manuel Gil Pérez[†], Gregorio Martínez Pérez[†]
[§] *University of Verona,* [†] *University of Murcia*
mattia.zago@studenti.univr.it, {juanjose.andreu, mgilperez, gregorio}@um.es

***Abstract*—This paper presents the design and implementation of a software component acting as a simulator and aiming to help in the deployment of novel attack graph models. It is also intended to help comparing these novel approaches with already existing designs and implementations. It has also as an objective to determine those aspects of existing models that have not been completely defined or specified by their authors and thus may need some completion before being used in lab or real attack scenarios.**

I. Introduction and Motivation

In the literature, there exist several approaches to risk analysis through attack-graph based models as they represent an interesting data structure that allows modelling multiple ways of penetrating a network. However, selecting one or another approach to be consider in a potential lab or real scenario seems to be a complex task mostly because some of these models are just defined theoretically and simple experimentation has been provided on them. Additionally, it seems to be a complex task to compare two models to determine, under certain equivalent circumstances, which of the two is performing better.

With this aim in mind a software component defining the basic data structures and general interfaces of the detection and reaction parts is defined as part of the Multigraph project, a joint effort between the University of Verona and the University of Murcia, and whose early results are being described as part of this paper. The main target of the project is to build a simulator allowing security researchers and practitioners to implement different models and try to compare them, when possible, using similar assumptions. It is also intended to provide and describe a set of common interfaces that any potential researcher willing to design, implement and test a new model in the future could follow as a basic template to guide its design and deployment.

II. High-Level View of the Structure of the Simulator

The attack graph model simulator aims to provide a tool that can analyse a specific network by means of different models, offer the user (e.g., either a security specialist or a system administrator) a graphical visualization of the graph, and produce results that are consistent and can be easily compared to one another, independently of the model used for the analysis. To this end, the models are implemented following a common structure and interface, which allows the simulator to interact with them.

A. Core class implementing a model

The model implementation is defined by a set of classes including one core class that has to implement a common interface named *DecisionInterface*. This interface includes different methods that enable the simulator to do the real continuous interaction with the model and provide the results based on such interaction.

The core class is working in a multi-thread environment and all the additional classes required by any particular model to be implemented have to communicate with this core class.

Several core classes representing each one a different model already implemented in the simulator are provided to the user of the simulator when it is first run as depicted in Fig. 1.

B. Configuration

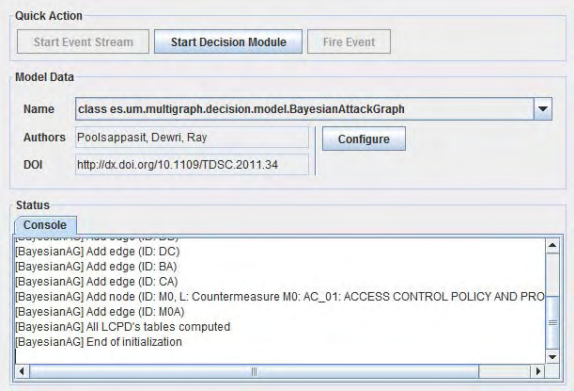

Fig. 1. Simulator main window

As each attack graph model is having a different set of parameters, another important aspect considered while designing the simulator was the definition of different template windows that any designer of a model could use to get the key values that need to be provided to make the model run under certain circumstances.

As an example in Fig. 2 some of the values required by the model [1] to run are depicted, together with some of the most relevant actions that can be considered to provide certain dynamism to the model execution while running it either step by step or as a whole.

C. Visualization

The simulator has been also designed to provide a graphical representation of the model and the graph being analysed during execution. This enables the user to see a representation

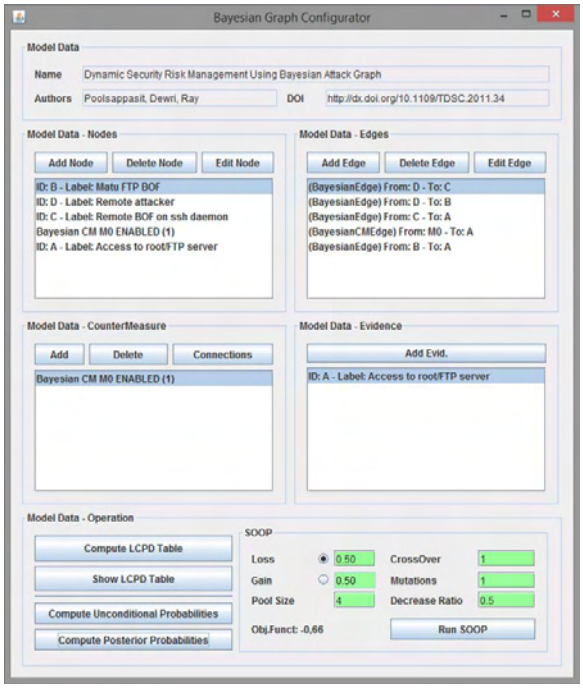

Fig. 2. Model configuration frame

of the model and interact with it. This is achieved through the use of a graph visualization class that receives as an input the nodes and edges of the model, and presents them on a screen. This visualization is using the library JGraphX [2] to display the graph.

An example of the representation of one sample graph for the model [1] is provided in Fig. 3.

In our simulator a given attack graph model can provide a

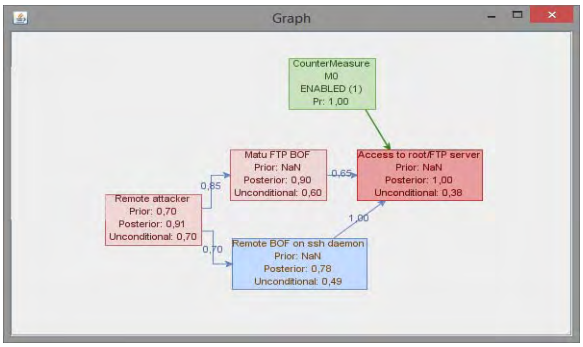

Fig. 3. Example of graph visualization for a given model

basic graphical interface to interact with it adding and deleting nodes and edges, changing the properties of these, etc. Whether or not this should be able to be done at any moment during runtime, or only at the beginning, depends on the specific features of the model being used.

In addition to that, the current version is able to represent the data at the lowest level and it represents the different classes of nodes through different colours and shapes. We are also intending to provide a multi-layered graph with abstraction and folding capabilities.

III. First Set of Lessons Learnt

The first issue that have emerged from this work is related with the innate difficulties while comparing different attack graph models. This is mostly due to the lack of a common structure or even a similar approach to the problem. There are several attempts to create a standard way to manage this problem, but each one has some issues at a certain point.

As a matter of example, and based on our design and implementation experience with different models existing in the simulator, there is a scale for the risk management (i.e., low, medium, high) but it is not mandatory or even recommended. The same happens inside the network analysis: how it should be defined the probability of a service being really compromised? Which should be the target: services, machines or both? And, what happens if the system is distributed or virtualized and then including different tenants?

In fact, as part of this effort, it has been also faced the problem that even taking two models with the same basic assumptions (e.g., same graph structure: services as nodes, exploits as edges, countermeasures as nodes and evidences as attributes), the outputs of these models can be different.

This is leading to the identification of a major issue when considering the comparison between different attack graph models, which is the lack of a standard normalized way to approach the problem, including specially the way to provide the input to the model and the way the model is describing its outputs.

IV. Conclusion

This paper describes a first attempt towards the creation of a simulator willing to help with the definition of new attack graph models. A first version of this simulator has been created and several models have been already implemented using it. Moreover, a first comparison between different existing proposals has been performed. As future work, we are planning to include new implementations of attack graph models and improve the current definition of the common interfaces, as well as providing general access to researchers to the simulator, so it can be used to design novel approaches.

Acknowledgements

This work has been partially funded with the support from the Spanish MICINN (project DHARMA, Dynamic Heterogeneous Threats Risk Management and Assessment, with code TIN2014-59023-C2-1-R) and the European Commission (FEDER/ERDF).

References

[1] 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, January/February 2012.

[2] JGraphX library, [Online] https://github.com/jgraph/jgraphx.