Producing Data to Assess and Manage ICT Risk by Simulating Threat Agents

F.Baiardi, F.Tonelli

Dipartimento di Informatica, Università di Pisa, Italy
`[baiardi,tonelli]@di.unipi.it`

Most definitions of risk agree it is an increasing function of the probability of some events according to historical data on the occurrence of theses events. Hence, no data are available to assess the risk due to the adoption of a new technology or of a version of a system that introduces a large number of changes. As an example, since the events of interest to assess a smart meter widely differ from those of a dumb one, historical data on dumb meters are useless to assess a smart one. Similar problems arise when designing an innovative system because no data may be available to assess alternative designs. The "no data"' problem prevents a proactive approach to risk because we can assess a system after its deployment and attacked instead than when designing it. Usually, an assessment solves the lack of data by estimating risk according to personal experience of the assessors. Even if they are competent and fair, the number of factors to consider is so large that the resulting evaluation may be subjective, disputable, hard to justify or properly communicated.

The Haruspex[1] methodology tackles the "no data" problem to assess an ICT system under attack by intelligent, goal oriented threat agents. These agents represent humans or programmed malware that aim to control some predefined resources. Haruspex models the target system and each attacking agent and then executes these models to simulate the joint evolution of the agents and of the system. By analyzing the evolution we discover which goals each agent reaches and how long this takes. We handle stochastic factors in the models by applying a Monte Carlo method that returns synthetic data it collects in multiple simulations. These data replace the historical one so that a risk assessment can be run before the deployment. Another improvement with respecto to current approaches it that it can adopt a *what-if* approach to discover how distinct agents or alternative system implementations affect the overall risk.

The Haruspex model of the target system describes the target system as a set of interconnected modules. Each module defines some operations that the other modules invoke provided that they own the corresponding privilege. Vulnerabilities are defects in some modules that enable some elementary attacks. The agents know some vulnerabilities and discover the other ones in the evolution. An attack consists of actions an agent executes to illegally acquire some privileges. To execute any attack, an agent needs some privileges on the operations of some modules. An attack succeed with a probability that depends upon both its structural properties and the agent that executes it.

[1] An ancient Tuscany forecaster

The model of an agent describes how the agent chains the elementary attacks to reach a goal anytime this takes more than one attack. Each attack in a chain escalates the agent privileges to execute further attacks that grant further privileges till the agent acquires all those in a goal. The selection of the chain to implement depends upon the agent goals, its legal privileges and the information on the system it has available. The modeling of this selection strongly influences the realism of an agent model and, hence, the accuracy and the precision of the simulation. In an AI perspective, the problem of building and ranking chains to a goal is strongly related to an intelligent agent behavior and it is more challenging than classical planning. First of all, each step in a chain requires a distinct time and it may fail. This increases the complexity of evaluating the time to implement a chain. Furthermore, the agent may discover some information to build a chain only after some preliminary attacks. As an example, an agent can access full information on a protected network only after successfully attacking some account on a network node. The chain an agent selects also depends upon the time it invests to collect and analyze information on the target system.

Haruspex introduces four attributes to model an agent: *look-ahead*, *ranking strategy*, *persistence* and *continuity*. The assessment can specify these parameters for each agent. The look-ahead is inspired to how a chess player selects its move by taking into account both the current situation and the one after playing some moves. An agent look-ahead is a non-negative integer that defines the complexity of a ranking strategy by determining the amount of time the agent spends to evaluates the future gains of its current choice. An agent with a zero look-ahead randomly selects the attack to implement. Otherwise, the agent selects the chain to implements by ranking all those with a length lower than or equal to its look-ahead. At first, the strategy only considers chains leading to a goal. If no chain reaches a goal because of a low look-ahead, the strategy considers all the chains the agent can implement. The ranking of chains considers the attributes of the attacks in a chain such as the success probability, the time to implement the attacks or the number of rights it grants to the agent. The persistence of an agent defines the number of times it repeats a failed attack before selecting an alternative chain. Lastly, the persistence defines the number of attacks of a chain the agent executes before invoking again the strategy. This defines the compromise between the selection overhead and the ability of chains enabled by newly discovered vulnerabilities.

The modular models of the system and of the agents enables Haruspex to compute a statistical sample starting from simpler and more easily measurable factors. In this way, the assessment only has to supply basic probabilities, such as the one that an attack is successful or of discovering a vulnerability. The assessment can evaluate these probabilities more easily than those related to a whole attack chain. Haruspex preserves the overall complexity of the assessment in the joint evolution of the models of target system and of the agents and in the adoption of a Monte Carlo method that repeats this evolution to collect a statistical sample. This solution increases the freedom of an assessment in selecting the agent parameters with respect to methodologies that have to constrain the parameters to apply a more formal method. In turn, the freedom increases the accuracy of the assessment because it can consider a large number of parameter combinations and evaluate how each combination contributes to the success probability of the corresponding agent and to the time to reach a goal. Since an assessment should discover the most dangerous agent for a system, namely the one that reaches its goals in the shortest time, the ability of freely selecting the agent parameters guarantees to cover all the agents of interest.

The Haruspex methodology has driven the design and the implementation of a set of tools to automate risk assessment and management. The suite includes tools to build the models of interest, to execute them and to analyze the output of a simulation to discover the most effective security investment. The tools interact through a database with the system and the agent models and the samples. The kernel of the suite includes the *builder*, the tool that returns the system and the agent models, the *engine* and the *manager*. The last two tools are those most affected by our assumption on intelligent agents that builds a plan and update it when the system changes.

An *engine* experiment consists of independent *runs* that simulate, for the same time interval, the evolution of some agents and of the system. At the beginning of a run each agent owns its legal privileges only. A run ends either when all the agents reach their goals or when reaching the end of the interval. At each time step the *engine* determines the newly discovered vulnerabilities and it selects the chain each idle agent implements according to the agent ranking strategy. An agent is busy for the selection time and for the one implement some attacks according to the agent persistence. The selection time includes both the ranking time and the one to collect information about the attacks in the ranked chains. The *engine* determines the outcome of each agent attack and, according to this outcome, it grants the corresponding privileges. In each run, the *engine* collects data on the evolution that it stores in a database to compute statistics of interest. The confidence level of these statistics increases with the number of runs in an experiment and the *engine* starts a new run till reaching the level of interest for the assessment.

The *manager* runs a sequence of experiments to select a cost effective set of countermeasures to improve the system robustness. A countermeasure is a change to the target system that breaks a chain by strongly reducing the success probability of an attack. The tool runs a first experiment and it selects the countermeasures to break the chains the agent implements to reach their goal. To minimizes both the number and the cost of countermeasures, the selection privileges countermeasures for attacks shared among chains. Then, *manager* updates the system model to take into account the changes due to countermeasures and it runs another experiment to discover any new chains the agents implement. Being intelligent, the agents may select distinct chains when countermeasures have broken the other ones. If some agents still reach their goals, the *manager* starts a new iteration that selects further countermeasures and runs another experiment. The iterations end after breaking all the chains.