A TOOL TO FORMALISE AND TEST ATTACK SCENARIOS OF SOFTWARE INTENSIVE CRITICAL INFRASTRUCTURES

Claudio Balducelli & Giordano Vicoli
ENEA (Italian Agency for New Technologies, Energy and Environment)[1]

Keywords: intrusion detection, attack trees, attack patterns, critical infrastructures, attack modelling

Abstract

Modelling and simulation of attacks scenarios could help to discover the hidden vulnerability points inside software intensive critical infrastructures, like energy distribution networks, railway networks etc. that are controlled and supervised by SCADA (System Control And Data Acquisition) systems.

To define the possible attack scenarios is necessary to analyse the attackers characteristics and the resources and the opportunities that could be available for the attackers.

In this paper the formalism of "attack trees" is proposed as usefull modeling technique for attacks: more attack sequences are generated from a single tree configured and "edited" using ATP (Attack Tool Platform). ATP provides users with a set of instruments useful for building (editing) attack actions profiles, a tree of this profiles, tree instances (named attack scenarios) and then for running simulation sessions using these instances.

The same attack action can be associated to different action profiles having different levels of difficulty. In this way an attach tree could include different attack paths with different associated attack difficulty levels.

The utilization of attack trees realizes a formal representation of the attack sequences to be tested. In such way this formalisation represents also a formal strategy to elicit knowledge and information about potential vulnerabilities of the infrastructures.

ATP tool was used in SAFEGUARD[2] project to test attack scenarios against an electrical transmission network controlled by a SCADA system. The laboratory testing environment and the results are also described.

Introduction

This paper is a follow-up of a work already proposed in [1] where a sort of reference language to model and implement intrusions and faults scenarios was proposed. In that work a proposal was done to formalize the required paths of attacks or system faults through the definition of

[1] Via Anguillarese 301, 00060 Rome, Italy
[2] A multi-agent system to safeguard Large Complex Critical Infrastructures developed inside EU FP5 program

attack trees. The root of an attach tree represent an event that could significantly harm the infrastructure's mission.

Every path in the attack tree represents a unique type of attack (or a unique type of fault propagation) for the infrastructure.

Different types of nodes and links can be utilized during the design phase of an attack tree: AND node, OR node or XOR node. Such a model allows also the insertion of a certain degree of certainty inside the different paths.

Attack trees are in some way similar to fault trees that was extensively used in nuclear applications [2]. The main difficulties to apply fault trees depends on the difficulty to asses the failure rates of the different nodes of an information system and to evaluate the interdependencies between the different components failures [3].

The utilization of such attack trees realizes also a formal representation of the attacks. This formalism could help the attack analyst to classify attack knowledge and to realize a more efficient method to elicit attackers expertise and behaviour.

The attacks may be planned by unintentional or malicious attackers, they may be single persons, communities or organisations. Single intruders generally have limited resources to conduct the attack activity. Organisations, and especially terrorist organisations, may have a lot of resources available. They could be generally able to conduct "distributed attacks" composed by sets of efficient sequences of single attacks in different points of the layered infrastructures.

Studies and the analyses of the "intruders' behaviour", in relation to the intrusion typologies and the types of knowledge and preferences of the involved actors, are necessary to support the generation of realistic attack scenarios and with an higher level of occurrence.

To make more effective such analyses, in this work an Attacks Test Platform (ATP) tool is proposed, based on the previous defined attack trees, that can be used for modelling and simulating attack scenarios. This tool is a suite of applications provided to test the robustness of an information intensive critical infrastructure against malicious attacks. The ATP components are instruments with which users may configure a set of attack sequences against a SCADA system or a generic information intensive system. The tool was developed in the framework of "SAFEGUARD", an European research project having the objective to realise a "Multi-agent System to safeguard Large Complex Critical Infrastructures".

To develop an attack sequence knowledge and expertise of potential attackers are required. The formalism of the "attack trees" is used to generate more attack sequences from a single tree that is configured and "edited" using the ATP tool. The tool provides users with a set of instruments that they can use for building (editing) attack actions profiles, a tree of this profiles, tree instances (named attack scenarios) and then for running simulation sessions using these instances.

The same attack action can be associated to different action profiles having different levels of difficulty. In this way an attach tree could include different attack paths with different associated attack difficulty levels.

Attack Test Platform generality

The three principal functionalities of the Attack Test Platform (ATP), as described in the following, are:

- attack trees editing
- scenarios generation from a tree
- scenarios running

Attack tree editing

An attack tree is composed by two type of nodes: logical node and action node.

The logical is the node that represents a point of decision, it is a step that has the property to define several sub-set of the tree. In simpler words the logical step provides a combinatory logic node for having several solution of attacks at that point of the tree.

The action is the node that represents the effective attack, the wicked action on attacked system.

The initial logical node of the attack tree represents the final objectives of the attacks; The terminal leafs (action nodes) of the tree represent the actions that must be executed for reaching the objectives. We can have three types of logical nodes:

· <AND> node (all children nodes must be executed)
· <XOR> node (only one of the children nodes must be executed)
· <OR> node (any combination of the children nodes must be executed)

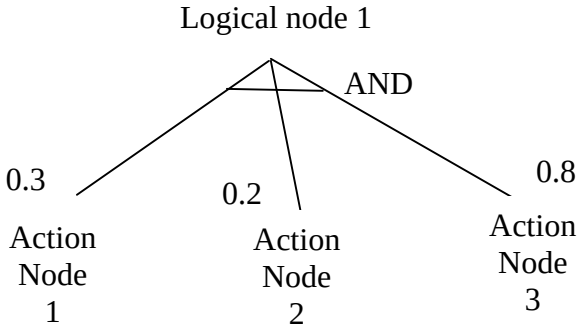

Fig 1 – graphical representation on a node

As visualised in fig 1 to the action nodes is possible to associate a certainty factor as a real value between 0.0 and 1.0. This number represents the probability (a score) that the action will have success; this probability normally depends about the degree of difficulty associated of the action. Here 0.0 is associated to infinite difficulty level and 1.0 to absence of difficulty. The visualised three generates only one scenario composed by all the three actions with a difficulty global factor of:

0.3 x 0.2 x 0.8 = 0,048 probability of success

In the figure is visualised the graphical interface, Attack Tree Editor (ATE) used to edit and visualise an attack tree.

The tree formalisation use “steps” to represent the nodes.

Here it is possible to visualize the Logical Steps (L) and the Action Steps (A). Every children steps are included inside a "children folder" that can be open and closed to see the children steps. Every steps has attributes like the "NAME", the "OPERATOR" for the logical steps or the "SCORE" for the action steps.

ATE graphical interface is the first program that must be used to begin the attack formalisation activity. ATE allows the user to open a TREE initial model for building a new TREE attack structure. It is also possible to open an old TREE model to add or modify parts of the tree.

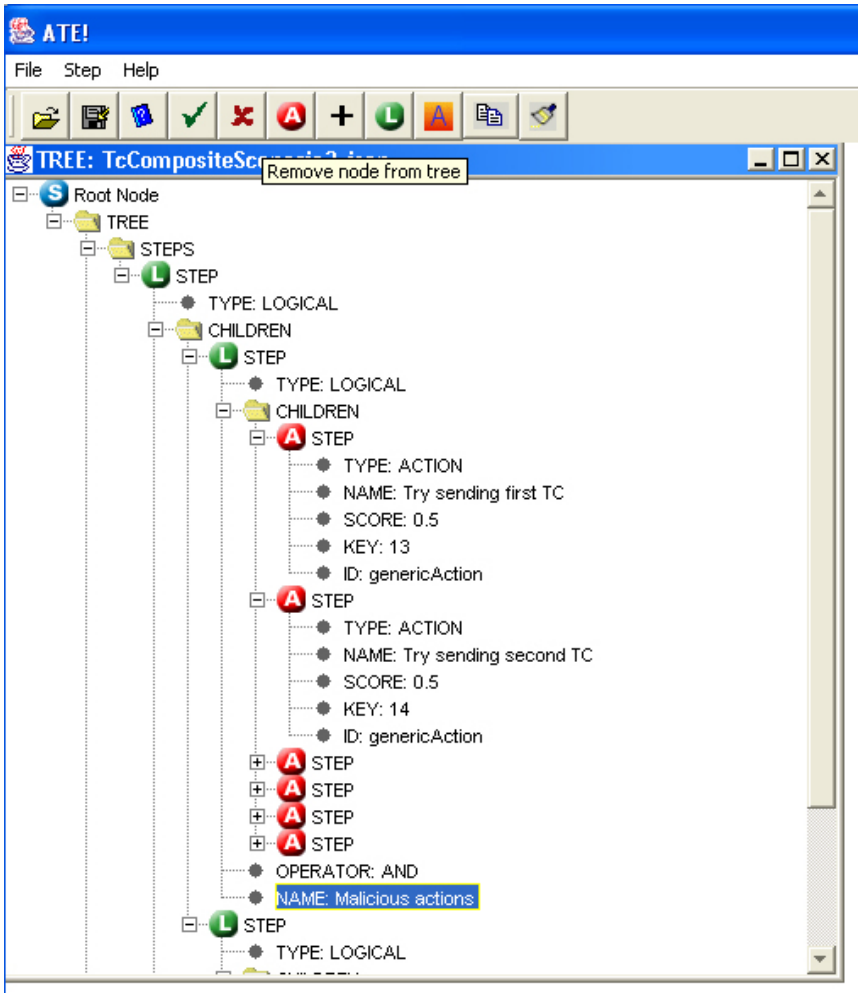

Fig 2 – ATE graphical interface

Actions that you may do on a existing TREE are:

- add (or remove) a Logical step or an Action step;
- add (or remove) an attribute (it's not a node, but a couple key-value that represents an attribute for the parent node).
- change the node structure;
- copy and paste a sub-tree from a tree toward another tree;
- save all;
- open and change an old tree.

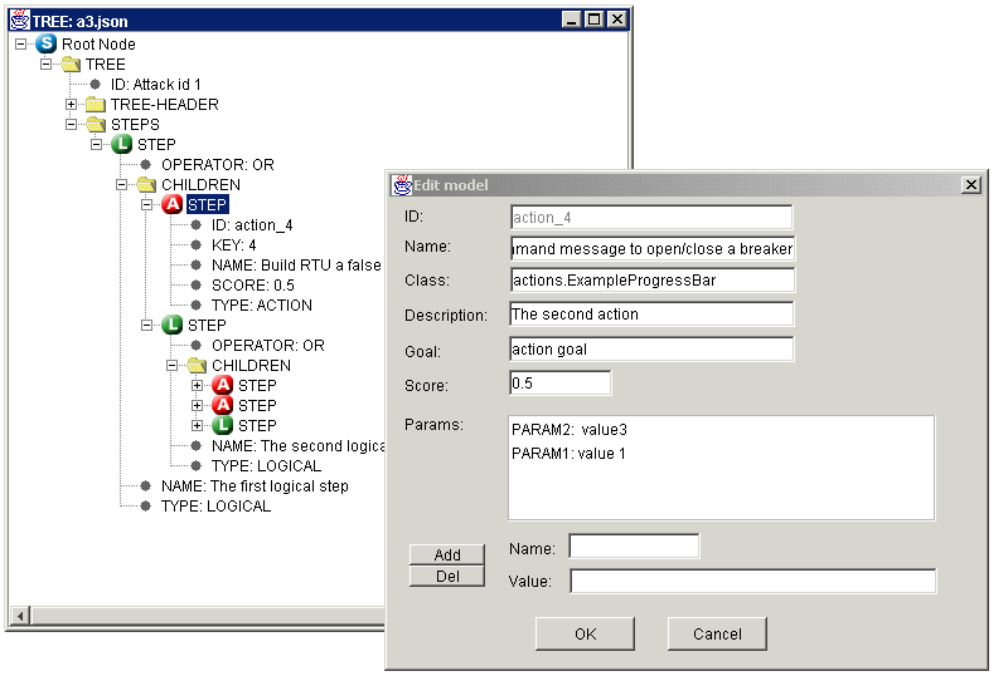

Fig 3 – the meta-descriptor dialog for the selected action on tree.

ATE allows also to edit a meta-description of the actions defined inside the tree, as it is visualised in fig 3. In the meta-description is possible associate a class for the action and also some action parameters. This information will be necessary to run the action inside a testing environment as it will be described in the following. Action class indicate the name of the program that will execute the action in the testing environment and the parameters are the relative program parameters like values for program running time, program running options etc.

Scenarios generation from a tree

ASE (Attack Scenario Editor) application let the operator to build a set of scenarios from a given tree. A single scenario represents a sequence of actions the test environment has to perform on the target network. From a given tree more scenarios are generated, and this mainly depends about the total number of <OR> or <XOR> operators that are defined inside the tree.

User can open a tree like the one showed in picture 2 and he can generate every possible scenario with an automatic mechanism. ASE explores the tree structure and for each logical node it observes the “OPERATION” attribute.

If a logical node has three branches and if the attribute value is “AND” the procedure exports every children node inside a single new scenario; if the attribute is “XOR” the procedure export only single children nodes inside three different new scenarios; if the attribute is “OR” the procedure export any combination of the children nodes inside seven different new scenarios.

Examples

As an example we have formalised an attack tree (see Table 1) containing the general procedure for an attack toward some user workstations of a local area network of a society.

TABLE 1 – Example of a simple attack tree against a workstation of a local area network

---

Goal: Stealing information, blocking services, modify data to/from a target machine
Precondition: The attacker is a user of a local area network

AND 1. (0.9) Identify the subnet where reside a PC with the information
2. Enter in the subnet
XOR 2.1 (0.9) The subnet is physically linked to the main network. Possibility to enter in any node (PC) of the local network
2.2 (0.2) The subnet is not physically linked to the main network. Possibility to enter only in a node (PC) belonging to the subnet
3. (0.95) Identify the IP address of the gateway machine
4. (0.95) Identify the IP address of the target machine
5. Build a message reading clone machine for the target machine
AND 5.1 (0.9) Get the MAC address for the gateway machine
5.2 (0.9) Get the MAC address for the target machine
6. Start malicious activity
XOR 6.1 (0.9) Start sniffing and stealing information
6.2 Blocking some services furnished by the server
XOR 6.2.1 (0.3) The attacker machine is connected to his own network plug-in
6.2.2 (0.6) The attacker machine is connected to a public network plug-in
6.3 (0.2) Modify user information exchanged with a server

Postcondition: The attacker must have time to acquire, analyse the information, and work on the attacker machine

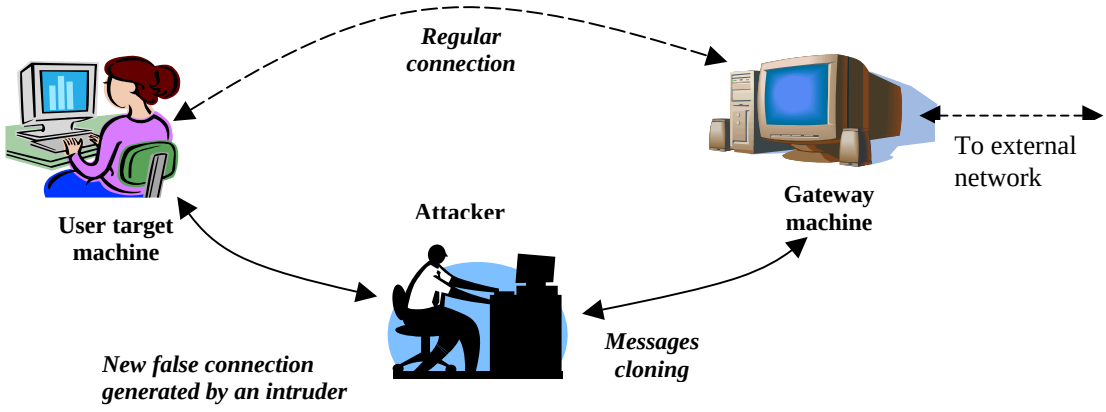

Fig 4 – Attacker intrusion inside a local area network

In the previous example fig. 4 shows how an attacker coud enter inside a subnet of local area network remotely or locally as defined by the logical step n. 2.

The attacker has to front different difficulties, that are dependent about the type of subnet where the target machine resides: if the subnet is physically connected to the main network the attacker may work from any other machine of the main network (step 2.1), otherwise he have to enter in some way to work on a workstation physically connected to the target subnet (step 2.2).

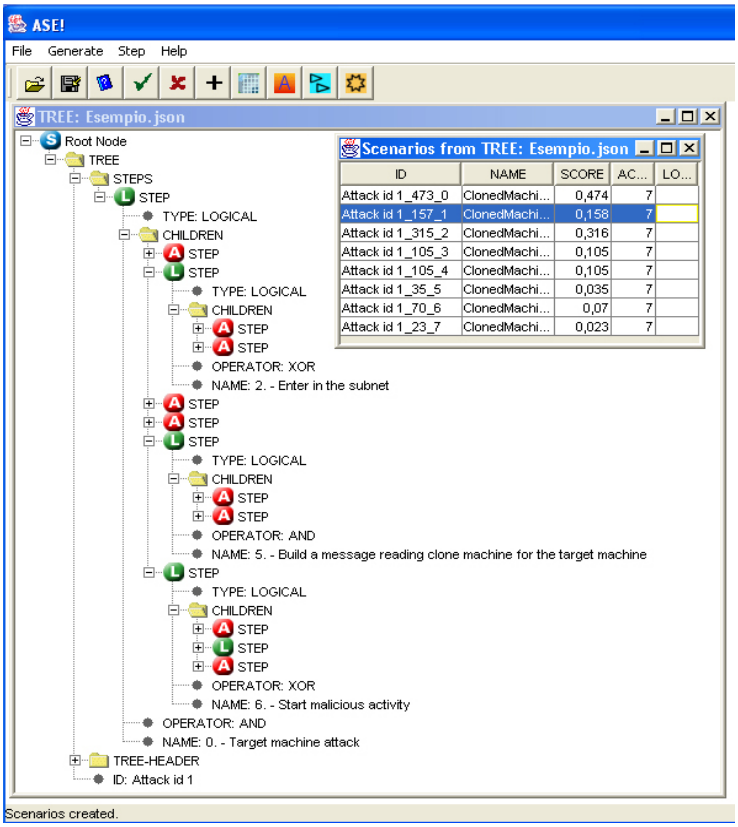

Fig.5. – ASE interface and scenario generation from a single tree

In both cases firstly he identifies the MACs (Media Access Control address) of the target machine and of the machine functioning as gateway as well (step 5.1, 5.2). Then he will set-up in the target machine the MAC address of his machine as a false address of the gateway machine. The same false setting is done for the gateway machine. In such way all the messages that the target machine will send/receive from the gateway will flow trough the intruder machine. The final objectives of the attacker may be of three type with different difficulties:

- Sniffing and stealing information from the target machine (Step 6.1). This is the most easy task and it is difficult for the network administrator to find out the intruder.
- Blocking some of the services offered to the target machine by the central server(step 6.2). This task is easy to do but is also easy find out the intruder especially if he work from his room and use his own plug-in for the machine.
- Modify the information exchanged with the server.

The last task is more difficult as it is necessary to know the content of the exchanged information (step 6.3).

In fig 5 is visualised ASE application interface where the previous tree is implemented.

Using “Generate” menu is possible to launch an automatic mechanism for creating from the tree every possible scenarios. The application, when this operation is completed, shows a table with scenarios generated. In this table is showed the scenario “SCORE” that represents the successful difficulty of the a scenario calculated composing the difficulties of every single action.

Table 2 show the structure of the seven generated scenarios corresponding to seven different paths inside the attack tree. Then user may open, modify and save all the generated scenarios.

Table 2 – Generated scenario table

Scenario	0	<1., 2.1, 3., 4., 5.1, 5.2, 6.1 > with 0,474 of difficulty
Scenario	1	<1., 2.1, 3., 4., 5.1, 5.2, 6.2.1> with 0.158 of difficulty
Scenario	2	<1., 2.1, 3., 4., 5.1, 5.2, 6.2.2> with 0,316 of difficulty
Scenario	3	<1., 2.1, 3., 4., 5.1, 5.2, 6.3 > with 0.105 of difficulty
Scenario	4	<1., 2.2, 3., 4., 5.1, 5.2, 6.1 > with 0.105 of difficulty
Scenario	5	<1., 2.2, 3., 4., 5.1, 5.2, 6.2.1> with 0.035 of difficulty
Scenario	6	<1., 2.2, 3., 4., 5.1, 5.2, 6.2.2> with 0.070 of difficulty
Scenario	7	<1., 2.2, 3., 4., 5.1, 5.2, 6.3 > with 0.023 of difficulty

Attack tree reusing

A specific attack tree can be reused inside another tree. For example the attack tree generated in the previous paragraph, that model an attacker working inside a local area network, can be reused inside another tree containing the working procedure of attackers working on machines installed on the external network, using an internet connection.

TABLE 2 – Example of a simple attack tree with attacker working on the external network

Goal: Stealing information, blocking services, modify data to/from a target machine
Precondition: The attacker works on an external network and use the "buffer-overflow" mechanism to obtain the super-user rights in the attacked system.

AND 1. Find the weakness of a personal Web site
XOR 1.1 (0.5) Find a known software bug
1.2 (0.2) Decide to make more attempt
2. Operate on the Web site interface
AND 2.1 (0.7) Find a function for which super-user rights are required
2.2 (0.2) Use the buffer overflow technique to inject malicious software
2.3 Enter and start working inside the local area network of the user

Postcondition: The attacker must have time to acquire, analyse the information, and work on the attacker machine

The previous table shows the attack tree relative to an attacker working from an external network. In this case the attacker must find the "weakness" of the software used to implement a personal Web site of a user or a society. He can decide to find a known software bug of a specific version of the software that allows to exploit the "buffer overflow" process; otherwise he can also to execute more attempts, but with minor probability of success. Buffer overflow is the process that an intruder can utilises to inject inside a victim system a software code able to make malicious activity, as for example to obtain to work with super-user rights and privileges. Despite many buffer overflow protection mechanisms were analyzed and implemented[4], buffer overflow remains a valid method to exploit the vulnerability of software systems.

In the attack tree of Tab. 2 the function of step 2.1, if executed, gives for a certain time the super-user rights to the external user. If, in the mean time, the attacker is able to inject a malicious software, he could acquire the super-user rights and the possibility to work as a privileged user in the sub-net where the workstation of the victim resides.

In this way the attacker has the possibility to work inside the local sub-net. Executing step 2.3, is equivalent to "reusing" the tree described before, that includes the attack procedure against a user belonging to a local area network.

ATP allows to substitute a logical step of a tree with an already defined tree. In such way the complete tree of fig. 6 is generated.

If we ask ASE to generate all possible scenarios we obtain a list of 16 scenarios, 9 more respect to the previous tree. The obtained difficulties to execute these scenarios are variable from 0.033 to 0.001. It means that for an external attacker the overall difficulty to conduct the attack is 10 times higher respect to an internal attacker: this appears as reasonable result.

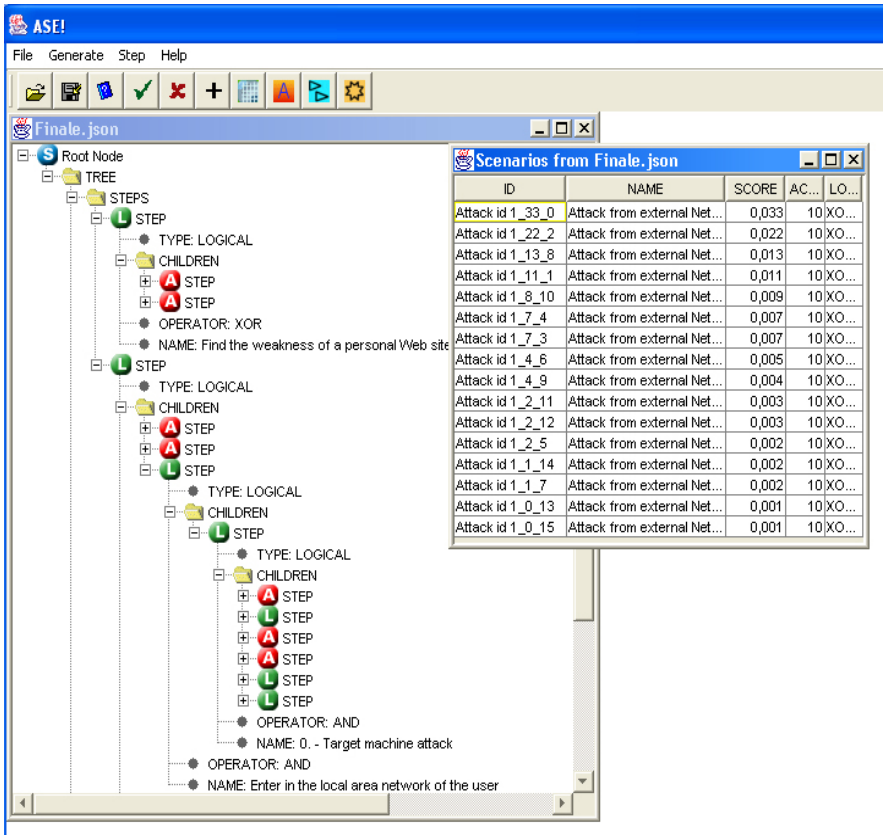

Fig. 6 – Attack tree relative to an external network attack

Attack scenarios running

Attack scenarios are the sequence of action steps that an attacker must execute to obtain the attack goal. ATP allows to "execute" the action steps of an attack scenarios in a simulated environment. The action steps of a scenario could be customized and made more specific, adding new features to actions previously defined in the tree that generated such scenario.
More in particular at every action must be associated to the "time" on which it will be executed. In such way a certain action could fire after or before other actions. The same action can be also repeated more times during a certain time length. This is the typical behaviour of an attacker that try more time to sniff or corrupt data inside transferred data packets buffers.
The actions can be also linked to the piece of malicious code realising the attack and a set of parameters, eventually required by the code, can be defined.

Experiments about attacks on SCADA systems

ATP was used to run simulated attacks scenarios against the SCADA software environment used as testing environment for SAFEGUARD agents[5]. The situation is illustrated in fig 7. An attacker is able to enter inside the Wide Area Network on which are connected the workstations of the Control Centre[3] (CC) of an electrical transmission network. These workstations are part of the Supervisory and Control System used by the electrical operators to monitor and control data coming from Remote Terminal Units[4] (RTU). Fig 7 shows how the attacker can sniff the messages exchanged between a CC and an RTU and how, with sufficient skill, he may also modify information stored inside tele-commands packets, producing in such way "false commands", aimed to generate unexpected remote operations, like an unexpected disconnection of an electrical line. Also in this case, as in the example described previously, the intruder could be "internal" (an employer/associate of the electrical company), or "external", if the Wide Area Network make available some services that require the internet connection; the availability of these services may be used as a weakness for attacks that make use of the buffer-overflow mechanism.

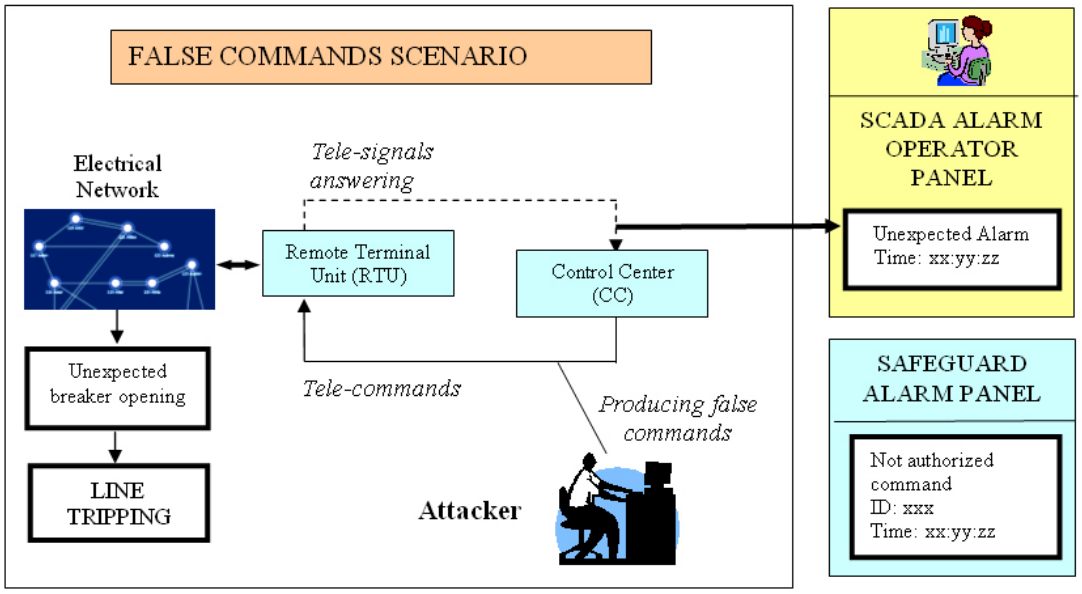

Fig 7. – Intrusion against a SCADA software environment

Fig 8 shows the ATP interface utilised to run the false command attack scenario. The complete attack sequence is composed by 7 action steps visible on the attack panel. The times on which every action is fired, is visible on the TIME column, where time increases from the bottom to the top. Certain actions relative to sending false tele-commands are repeated more times (100 times every 5 second in this case). Repeating more time these actions the attacker try to find the right command ID of a certain tele-command.

[3] A Control Center is the place in which the electrical network operators could monitor the status of the network, receive alarms and send remote commands from/to the peripheral electrical devices (breakers, transformer, generators etc)

[4] Remote Terminal Units are computerized devices acting as data monitoring front-ends of the Control Centers, normally located inside the electrical sub-stations and connected to Control Centers with a Wide Area Network.

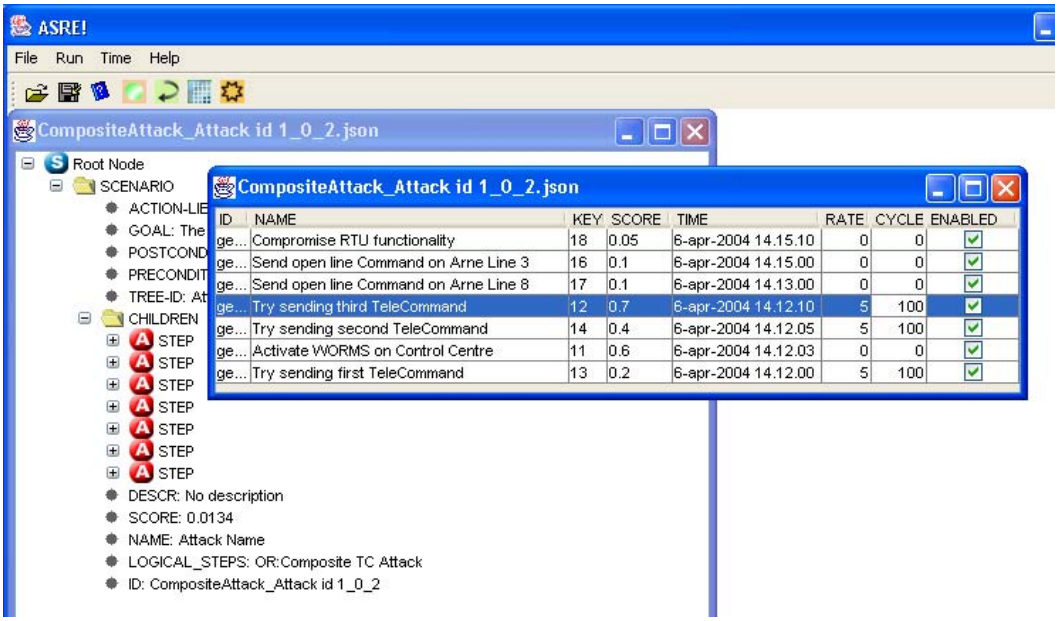

Fig 8 – The ATP interface to run false command attack scenarios

If a true command ID is found, it may be possible also to generate commands able to disconnect electrical lines, load shedding, change transformer tap positions ect.

Conclusions

Using ATP many different attack scenarios was configured and executed inside the Safeguard test bed. In particular the following scenarios was tested:

- False command scenario inside an electrical network
- Loss commands scenario inside and electrical network
- Perturbations inside SCADA communication packets
- Worms generation inside electrical Control Centres
- Data corruptions scenario inside electrical Data Base
- Malicious scan of Tele-communication packets
- Various types of buffer overflow attacks

More attack trees was configured including or a single type of the previous attacks or more than one, generating in such way a “composite” type of attacks.

Test was executed on the emulated SCADA environment to analyse the anomaly detection capability of the Safeguard agents.

All the most important functionalities of ATP are tested and the results suggest the introduction of some additional improvements. For example, it seem useful the introduction of a new type of actions like the “manual executed actions”. These actions will be used to wait the execution of manual action by the attacker. This allows to suspend temporary the attack sequence if it is necessary to wait for the execution of a manual activity (login into a server, browsing a data base, searching etc.) whose duration cannot be “a priori” determined.
Despite these new needed capabilities, the Attack Tool Platform is already now a useful software platform that support definition and archiving in a formal way knowledge and expertise about attacks and malicious activity.

References

[1] Balducelli C., "Modelling Attack Scenarios against Software Intensive Critical Infrastructure", in proceedings of 10th Annual Conference of The International Emergency Management Society, june 3-6, 2003, Sophia Antipolis, Provence, France

[2] Roberts, N. H., V. W. Vesely, D. F. Haasl, and F. F. Goldberg (1981). Fault Tree Handbook, Systems and Reliability Research, Office of Nuclear Regulatory Research. U.S. Nuclear Regulatory Commission. Washington, D.C. 20555.

[3] McQueen M., Boyer W, Flynn M., Alessi S., "Quantitative Risk Reduction Estimation Tool for Control Systems: Suggested Approach and Research Needs", in proceedings of The International Workshop of "Complex Systems & Infrastructure Protection", march 28-29, 2006, Rome, Italy

[4] Ruwase O., Lam M. S., "A Practical Dynamic Buffer Overflow Detector", http://suif.stanford.edu/papers/tunji04.pdf

[5] Balducelli, C., Lavalle L., Vicoli G., "Novelty Detection and Management to Safeguard Information Intensive Critical Infrastructure", in Proceedings of 11th Annual International Conference of The International Emergency Management Society, , may 18-21, 2004, Melbourne, Australia

Author Biography

Claudio Balducelli is a senior scientist working at ENEA as project manager since 1983 in the field of AI technologies applied to operator decision support systems for emergency industrial accidents. His interests include operator models, knowledge formalization, planning, computerized procedures, plant diagnosis, case based reasoning, learning and fuzzy algorithms. He co-ordinated also the prototypical implementation of various site applications like a cooperative training system for the Genoa Oil Port managers (MUSTER project) and an emergency operator support system for major Oil Deposits and Pipelines in Italy. He is team leader inside SAFEGUARD FP5 project (Safeguarding Critical Infrastructures), and IRRIIS (Integrated Risk Reduction of Information-based Infrastructure Systems) FP6 project.

Giordano Vicoli from 1988 to 1992 took part in research projects in the field of design and development of expert systems for diagnosis and control of industrial plants. From 1993 he has been working in the field of design and development of decision support systems and training applied in emergency management of high risk industrial plants. His interests was in the development of distributed application with J2EE and CORBA technologies. He works at the SAFEGUARD project work-packages developing the emulated SCADA test environment and furnishing the expertise necessary to formalize the attack scenarios. He is actually team leader in IRRIIS (Integrated Risk Reduction of Information-based Infrastructure Systems) FP6 project.