Metrics for Cyber Robustness

F.Baiardi[1], F.Tonelli[1] and A.Bertolini[2], M.Montecucco[2]
[1]Department of Computer Science, University of Pisa
[2]Haruspex S.R.L.
ITALY
[1]{baiardi,tonelli}@di.unipi.it
[2]{alessandro.bertolini,marcello.montecucco}@haruspex.it

ABSTRACT

Cyber robustness measures how long an ICT system can resist to attackers that compose attacks to escalate their privileges till reaching their goals. This paper proposes three metrics to evaluate this ability. The basic one is the security stress that considers the probability that an attacker reaches a predefined goal in a time interval. The relation between the interval size and the probability evaluates the overall robustness. This metric is the starting point to define two metrics that evaluate cyber robustness through the financial impact. We approximate the security stress using the output of the Haruspex suite. The suite tools forecast how a system is attacked by simulating the interaction between the system and some attackers. The output of the suite supports both the computation of the metrics and the design of more robust versions. Lastly, we apply the metrics to compare three versions of an industrial control system.

***Keywords**: Robustness; Targeted Attack; Privilege Escalation; Impact*

1.0 INTRODUCTION

Cyber robustness is the ability of an information and communication technology (ICT) system of resisting to intelligent and goal oriented attackers, i.e. advanced persistent threats. Attackers increase their privileges, e.g. access rights, through a privilege escalation. An escalation is a sequence of attacks that ends when the attacker acquires a predefined set of rights, its goal, and controls some system resources. The only countermeasure against an attacker that has reached a goal is to revoke some of the privileges it has acquired.

This paper defines some metrics to quantify cyber robustness. These metrics strongly contribute to cyber awareness as they anticipate how a system can resist to its attackers. Furthermore, they can drive a security investment by supporting a comparison of alternative versions of a system with respect to this ability.

The first metric is the *security stress* or simply *stress* at a time *t*. This metric is equal to the probability that an attacker reaches its goal at time *t*. This relates the success probability of an attacker and the time it has available to implement its escalations. In turn, this time depends on the alternative escalations an attacker can implement, the number of attacks in these escalations, and the attack success probabilities.

Then, we introduce an *impact* function to map into a loss the time interval an attacker owns some access rights. We use the *stress* and the *impact* to define two financial metrics, *AvLoss* and *CyVar*. Both metrics evaluate the loss at a time *t* due to an attacker that control some system components and neglect the one due to the attacks in an escalation. These metrics splits *t* into two times, the one the attackers take to reach a goal and the one they own the corresponding rights. We compute the probability to reach a goal in an interval through the *stress*.

AvLoss measures the expected loss at a time *t* as the weighted sum of the losses due to distinct attackers. The weight of each loss is the probability that the attacker reaches a goal at *t*. This probability is the first order derivative of the *stress* due to the attacker.

CyVar evaluates the loss more accurately by extending *Value-At-Risk* or *VAR*, a risk measure for a security investment. *VAR* focuses on the odds of losing money and it has three inputs: a time t, a confidence level c, and a loss amount l and it returns the probability the investor loses more than l at t. c is confidence level of this probability. *CyVar* returns the same information for attackers that escalate their privileges.

A fundamental input for all the metrics is the probability an attacker reaches a goal in a time interval. Since there is not a closed form expression to compute or even approximate this probability, we adopt a Monte Carlo method and the Haruspex suite. The Haruspex suite supports a model based ICT risk assessment and management. It builds a statistical sample to compute the statistics to assess and manage ICT risk by applying a Monte Carlo method with multiple simulations of how the attackers escalate their privileges when attacking an ICT system. The sample enables the assessment to forecast the behaviour of the attackers, to discover which vulnerabilities they exploit, and to compute the probabilities of interest. Obviously, distinct approaches and tools may be adopted to apply the proposed metrics.

We structure this paper as follows. Sect.2 briefly reviews related works. Sect.3 outlines the Haruspex methodology and the tools that return information to compute the metrics of interest. Sect.4 defines the *security stress* and its approximation through the outputs of Haruspex tools. Then, it discusses alternative definitions of the impact function. Sect.5 briefly analyses a paradox on the return of a security investment. Both the following sections define a financial metric. Then, Sect.8 applies these metrics to three alternative versions of an industrial control system and it evaluates both the versions and the cost effectiveness of the investment to change one version into another. Lastly, we draw some conclusions.

2.0 RELATED WORKS

This work extends and generalizes [1,2] that defined, respectively, the security stress and *CyVar*. The Haruspex suite generalizes adversary simulation [3]. [4-8] describe the suite tools, their application, and how they automate ICT risk assessment and management. [9-12] discusses the simulation of privilege escalations. [13] reviews attack and defence modeling for critical systems. [14] analyzes multi-step attacks, i.e. escalations, and reconstructs the attacker steps using its traces and a predefined attack ontology. [15] presents a model based approach to simulate attacks to collect information on resiliency.

[16] reviews resilience metrics for cyber systems. [17-19] define some robustness metrics without integrating them with the simulation of the attacks. The metric in [20] focuses on zero-day vulnerabilities while [21] proposes metrics for cyber defence but it consider attacks in isolation without relating them with attacker escalations. [22-26] review alternative security metrics. [27] reviews security metrics for software development. [28] defines a metric similar to the *security stress* as it considers the amount of work to attack a system. Also [29] considers the adversary work. [30,31] investigate the relation between metrics and security investments. [32] analyzes the optimal security investment. [33] considers attacks against smartgrids. [34] discusses how to evaluate and improve resiliency of critical infrastructures.

3.0 THE HARUSPEX SUITE

This section outlines both the Haruspex methodology and the suite. Then, it introduces the tools to support the defined metrics.

Haruspex[1] is a model based methodology that adopts a predictive method to compute the probabilities to assess and manage risk. This is one of the three methods the IEC 31010 standard [35] suggests. Furthermore,

[1] An ancient Tuscany forecaster

Haruspex supports *security-by-design* because it can build even before a system deployment the models of the system and of the attackers it uses to assess and manage ICT risk.

The Haruspex suite is an integrated set of tools to apply the Haruspex methodology to a scenario where intelligent attackers select and implement a sequence of attacks to escalate their privileges and control some modules of an ICT system. The suite kernel consists of four tools: the *builder*, the *descriptor*, the *engine* and the *manager*. The first two tools build the model of the system and the one of each attacker in the scenario using simple and easily measurable parameters, such as the vulnerabilities in each system module, the attacks they enable, and the success probability of each attack. The model of each attacker is an agent with some attributes that describe its goals, the access rights it aims to acquire and its preferences. After building the models, an assessment discovers the escalations of each attacker by applying the *engine,* the suite tool that simulate the behaviour of the attackers against the target system. The simulation preserves the overall scenario complexity and the interactions between the target system model and those of the attackers mimic how the latter escalate their privileges. The *manager* is a risk management tool that iteratively selects some countermeasures and invokes the *engine* to evaluate how they affect the overall risk. Then, it improves the selection and starts a new iteration. This resembles an *extensive form* game [36]. In each game iteration, the manager is a player that selects the cheapest countermeasures against a set of escalations to produce a new system version, while the *engine* is a player that implements the agent escalations against this version. In the next iteration, the *manager* considers a set of escalations that includes also those the *engine* has implemented. The game ends as soon as the *engine* cannot implement an escalation or the *manager* cannot stop some escalations. This tool is not analysed in the following because cannot redesign the whole system.

3.1 Modelling a System and the Attackers

We describe in more details the Haruspex models to simulate the target system and the attackers [36]. Haruspex models the target system as a set of interconnected modules. Each module defines some operations to be invoked by the users and the other modules if and when they own the corresponding access rights. To describe social engineering attacks such as phishing, some modules model the users or the system administrators. The target system model describes each module, its vulnerabilities, and the attacks they enable. The input to build the system models include a complete list of all its vulnerabilities. The assessment can also introduce some suspected vulnerabilities and pair each of these vulnerability with the probability it becomes public when the system is under attack. This supports a *what-if* analysis of how suspected vulnerabilities affect the overall risk.

The module vulnerabilities enable some attacks. An attack consist of some actions that Haruspex models through some attributes. Two attributes describe, respectively, the privileges to execute the actions and those an attacker acquires anytime the attack is successful. Other attributes include the time to execute the actions and their success probability. This probability depends on both the attacker and the action complexity.

Haruspex assumes that no single attack grants all the rights in a goal and that the attacker can reach any of its goals through a sequence of attacks. Each attack grants some access rights that enable the execution of the following attacks in the escalation till the attacker acquires all the rights in a goal.

Being intelligent, attackers select their escalations according to their preferences. Haruspex models an attacker through an agent *ag*. The attributes of *ag* describe the attacker legal privileges, its goals, and the information on the target system it has available. Two attributes, the *selection strategy* of *ag* and the *look-ahead*, $\lambda(ag)$, a non-negative integer, models how the attacker selects the attacks in its escalations. The selection strategy defines how the attacker ranks alternative attack sequences, while $\lambda(ag)$ defines the length of the sequences it ranks. *ag* randomly selects the attacks in an escalation if $\lambda(ag)=0$. Otherwise, *ag* selects the sequence to implement by ranking all those with, at most, $\lambda(ag)$ attacks. We use sequences rather than escalations because they may grant a proper subset of the rights in a goal only. The strategy ranks any sequence *ag* can implement by exploiting its current access rights and it always returns a sequence that leads to a goal, if it exists. When its current set of rights is small, *ag* cannot implement a sequence that leads to a goal. Here, the strategy ranks the sequences according to the attributes of their attacks. An assessment pairs each agent with a predefined selection strategy that may consider distinct attributes, as an example:

- *maxProb*: considers the success probability of a sequence;
- *maxIncr*: considers the number of rights the sequence grants;
- *maxEff*: considers the ratio between success probability and execution time of a sequence.

There is no guarantee that any of these strategies always returns a sequence leading to a goal.

An attacker runs a vulnerability scanning of some nodes to discover the vulnerabilities enabling the attacks in the sequences it ranks. Hence, larger values of $\lambda(ag)$ increase the selection time due to the scanning of a larger number of nodes. This shows that $\lambda(ag)$ describes how *ag* solves the "collect or exploit" dilemma an attacker faces when deciding whether to collect further information or exploit the available one to select an attack. We have experimentally verified that look-ahead values larger than *3* force the evaluation of longer sequences without improving the ranking.

The *continuity* defines the number of attacks in a sequence *ag* implements before invoking again its strategy. A low continuity exploits at best newly discovered vulnerabilities at the cost of a larger selection overhead.

3.2 The Engine

This tool generalizes *adversary simulation* as it uses the models of the target system and of the agents in a scenario to forecast the behaviour of each attacker against the target system. The tool adopts a Monte Carlo method by implementing an Haruspex experiment with multiple independent *runs*. Each run simulates the behaviours of the agents for the same time interval. When a run begins, each agent only owns the legal privileges of the attacker it models. A run ends either when all the agents reach one of their goals or at the end of the interval.

At each time step, the *engine* determines the suspected vulnerabilities the agents discover. Then, it considers each idle agent and it invokes its strategy. The agent is busy for the time to select a sequence plus the one to implement a number of attacks equal to its continuity. The *engine* determines the success of an attack according to the attack attributes and, if it is successful, the agent acquires the corresponding privileges. An agent repeats a failed attack for a number of times equal to its *persistence*, a further attribute. The agent considers the next sequence in the ranking when the number of failures of the same attack reaches its persistence. An agent is idle anytime it cannot select an attack because it lacks some privileges and it may leave this state only after the discovery of a suspected vulnerability.

At the end of a run, the *engine* collects observations on each agent escalation, the modules it has attacked and any goal it has reached. The observations collected in all the runs build the sample that is the tool output and that the assessment uses to compute the statistics of interest. The confidence level of each statistic increases with the number of runs in the experiment because the *engine* collects one observation in each run. The *engine* starts a new run till some predefined statistic reaches the required confidence level. The statistics may consider, among others, the modules an agent attacks or the time it takes to reach a goal.

3.3 Validation of the Suite

The accuracy of the suite predictions fully depends on the suite ability to mimic in an accurate way the attacker behaviours. Validating the tools with respect to real attackers is rarely possible because data on real attacks is seldom available. For this reason, we have validated the suite in some real-time network exercises where some defending teams were put against one attacking team, the red team. The exercise scenario considers a fictional country which industry fell under increasing cyber attacks. Day 1 started with low-level hacktivist campaigns and led to espionage and sabotage attacks against the networks of the defenders by the end of day 2. In addition to technical defence, the exercise includes tasks such as legal assignments and forensics challenge, to make it as lifelike as possible. The exercise is built up as a competitive game that scores defending teams based on their performance.

One of the teams has used the Haruspex suite to analyse the system to defend and to select the vulnerabilities to patch. Patching vulnerabilities before the red team begins its attacks is the only feasible countermeasure. Since the time to deploy the patches is low, the cost of a patching is the time to apply it and the *manager* minimizes the time to apply the patches it returns.

The only input we could feed to the suite is a vulnerability scanning of the nodes to defend. Lack of time has prevented an analysis of the module source codes. The scanning has posed a further problem because the exercise rules prevents any defender team from scanning some nodes that can even store malware to attack other nodes. This contradicts the basic Haruspex axioms that an assessment has a complete information on any target system module. We have solved this contradiction by assuming that the red team fully controls these nodes and use them to launch some attacks.

We have considered agents that aim to control the system resources more valuable to the defender. To model that agents fully exploits any information they acquire, $\lambda(ag)=3$ for any *ag*. To handle uncertainty about the red team selection strategy, we have applied the Haruspex standard approach to manage lack of information on some attributes. This approach introduces distinct agents for each possible attribute value and then considers the worst case. In the exercise, the tools have considered all the escalations of these agents and have computed the optimal list of vulnerabilities to patch in the time interval before the attacks of the red team. The team members manually applied the patches.

The target system is affected by more than one thousand vulnerabilities. The manager returns a list with less than 2% of these vulnerabilities. This confirms the *engine* ability of forecasting the attacker behaviours and of discovering the critical vulnerabilities the attackers exploit even in the presence of a huge number of less important vulnerabilities. Most alternative approaches focus on the discovery of all the escalations before selecting the critical ones and have to face the huge complexity of discovering any escalation. The Monte Carlo method minimizes the overall complexity by discovering the most frequent escalations.

The team that has applied the Haruspex suite has scored excellent results as far as concerns network defence.

4.0 SECURITY STRESS AND IMPACT

This section defines the *security stress*, or *stress* and the *impact*. The *stress* evaluates the cyber robustness a system *S* in term of the probability that some attackers reach a goal in a time interval. It is an autonomous metric and the next sections use it to define financial metrics too. Instead, the *impact* maps the time an attacker owns some access rights into the corresponding owner loss. In the following, we denote an attacker by *at*, by *sg* the goals of *at*, and by *g* one goal in *sg*.

4.1 Security Stress

We define $Str^{S}_{at,sg}(t)$, the *security stress* of *S* at *t* due to *at* that aims to reach any goal in *sg* . If $PrSucc^{S}_{at,sg}(t)$ is the probability that *at* selects and implements an escalation that reaches a goal in *sg* within *t* then,

$$Str^{S}_{at,sg}(t)=PrSucc^{S}_{at,sg}(t)$$

$PrSucc^{S}_{at,sg}(t)$ is the sum, for all the possible escalations, of the probability each escalation is successful at *t* under the condition *at* selects it. The probabilities of selecting two distinct sequences are not independent because they both depend on the priorities and preferences of *at*. We cannot deduce a closed form expression for $PrSucc^{S}_{at,sg}(t\ )$ because of the complexity of computing the probabilities that determine $PrSucc^{S}_{at,sg}(t)$ namely the one that each sequence of attacks is successful and the one that *ag* selects each sequence. For the moment being, we focus on the *security stress* properties and discuss its computation in the following. Being a probability distribution, $Str^{S}_{at,sg}(t)$ is monotone, non-decreasing in *t* and $Str^{S}_{at,sg}(0)=0$. $Str^{S}_{at,sg}(t)$ increases with *t* for two reasons. First of all, *at* can implement longer escalations, i.e. it can reach a goal through a longer sequence of attacks. In general, this sharply increases the *stress*. The second reason is that a larger value of *t* can tolerate a larger number of attack failures.

Consider an attacker *at* that does not need to run a vulnerability scanning as it already knows the vulnerabilities of *S*. *at* can reach the only goal in *sg* by implementing two escalations that include, respectively, three and four attacks. Each attack takes two time units and its success probability is *0.5*. $Str^{S}_{at,sg}(t)$ is zero if *t* is smaller than *6*, while $Str^{S}_{at,sg}(t)$ is, at most, *1/8* for any $t\in[6..8)$ because *at* can implement one of the two escalations provided that all the attacks are successful. The *stress* is lower than *1/8* any time *ag* may select some sequences that do not lead to its goal. $Str^{S}_{at,sg}(t)$ is at most *6/16* for any $t\in[8..10)$ because in *0..t* *at* can implement any of the two escalations, provided that at most one attack in the first one fails. This occurs with a probability equal to *3/8*. Several reasons can reduce the *stress*. As an example, *at* may select a sequence that does not lead to a goal or need some time to scan the node of *S*.

We discuss in more details how some attributes of *S* and of *at* influence $Str^{S}_{at,g}$ by considering two times:

- t_0 is the shortest time where $Str^{S}_{at,sg}(t)>0$,
- t_1 is the shortest time where $Str^{S}_{at,sg}(t) \approx 1$.

t_0 is the time to implement the shortest escalation to a goal in *sg,* while *at* is always successful for times larger than t_1. We require that $Str^{S}_{at,sg}(t) \approx 1$ because $Str^{S}_{at,sg}(t)$ may reach *1* only asymptotically. In the previous example, $t_0 = 6$, while $Str^{S}_{at,sg}(t)$ reaches *1* asymptotically because all the sequences include attacks with a success probability strictly lower than *1*. Assume both times exist and consider *at* as a force acting on the shape of *S*. This force is ineffective till t_0 when the escalations of *at* begin to change the shape of *S*. As *t* increases, the selection strategy of *at* and its attributes become less and less critical because the time *at* has available increases. *S* definitively cracks after t_1 because *at* always selects and implements an escalation that leads to *g*. $t_1 - t_0$ evaluates how long *S* resists to *at*, at least partially, before cracking.

t_0 depends on both the attack execution times and the length of the shortest escalation to *g* in *sg*. t_1 depends on the success probability of attacks in *at* sequences. This probability determines the time to implement a sequence as it constrains the number of repetitions of a failed attack. $t_1 - t_0$ depends on both the standard deviation of the lengths of the escalations to *g* and the success probabilities of their attacks. These dependencies show that $Str^{S}_{at,sg}(t)$ measures cyber robustness more accurately than metrics that only consider average values, such as the average time or the average number of attacks in an escalation to *g*. In fact, these metrics cannot measure the interval of time *S* can withstand the attacks of *at*.

The inverse of the *stress* $Sur^{S}_{at,sg}(t) = 1 - Str^{S}_{at,sg}(t)$ is a survival function [37] that plots the probability that *S* survives to the attacks of *at* to reach a goal in *sg*.

We can compute the *stress* of a set of attacker *sa* provided that they have the same goals in *sg*. At each time, the stress due to *sa* is the largest stress of its attackers:

$$Str^{S}_{sa,sg}(t) = max\{at \in sa,\ Str^{S}_{at,sg}(t)\}$$

If the *stress* is always due to the same attacker at_m in *sa*, then we denote at_m as the most dangerous attacker that always reaches a goal before the other ones.

In general, if the attackers in *sa* have distinct goals, they also have distinct motivations and result in distinct impacts. We may define the resulting *stress* as the weighted average of those of the attackers in *sa*. The weight of an attacker evaluates its contribution to the overall stress. In the following, we consider the stress due to attackers with the same goals only.

A Monte Carlo method can overcome the lack of a closed form expression and approximate $Str^S_{at,sg}(t)$ as the percentage of runs in an Haruspex experiment where the agent *ag* that models *at* reaches a goal in *sg* before *t*. We denote this approximation by replacing $Str^S_{at,sg}(t)$ with $Str^S_{ag,sg}(t)$. The experiment simulates *ag* for at least *t* and it reaches the confidence level of interest on the time *ag* takes to reach a goal in *g*. This level is also the one of the approximation through $Str^S_{ag,g}(t)$. Similar considerations apply to the *stress* of a set of attackers where we refer to the most dangerous agent and not to the most dangerous attacker.

$Str^S_{at,sg}(n)$ is an alternative definition of *stress* where *n* is the largest number of attacks *at* can execute to reach a goal in *sg*. Obviously, *n* also counts failed attacks. $Str^S_{at,g}(n)$ relates the stress to the number of attacks instead than to the time to implement them. This focuses on the work of *at* instead than on available time. However, $Str^S_{at,g}(n)$ neglects the collection of information about *S* to select an attack sequence. Obviously, we cannot deduce a closed form for $Str^S_{at,g}(n)$ and we approximate it through the percentage of runs in a Haruspex experiment where the agent *ag* that models *at* reaches *g* by executing, at most, *n* attacks. We denote this approximation by $Str^S_{at,g}(n)$.

4.2 Impact Function

$Str^S_{at,g}(t)$ evaluates how long it takes *at* to acquire the control of *S*. This time is critical to evaluate the owner loss because *at* produces a loss only when it controls some modules. A typical example is a terrorist aiming to shutdown an ICS to sabotage a production plan or to create a large-scale pollution. In a commercial context, *at* may aim to steal some intellectual property or to reduce the efficiency of a production plan. The loss for the owner of *S* is related not only to $Str^S_{at,g}(t)$ but also to the time *at* owns the access rights in *g* before *S* discovers its attacks. For this reason, we also introduce an impact function $Imp^S_{at,sr}(t)$. $Imp^S_{at,sr}(t) = l$ implies the owner loss is *l* if *at* owns the rights in *sr* for *t*

$Imp^S_{at,sr}(t)$ is monotone non decreasing in *t* and $Imp^S_{at,sr}(0) = 0$ but its shape fully depends on both *S* and *at* motivations. To analyse these dependencies we suppose that *at* aims to read some information in *S* to steal some IP. Let us suppose the time to exfiltrate the information is t_s. $Imp^S_{at,sr}(t)$ increases if *t* belongs to $0...t_s$ and it is constant for larger values. The second order derivative of $Imp^S_{at,sr}$ is strictly negative in $0...t_s$ anytime the exfiltration of further information has a decreasing contribution to the overall loss. The same derivative increases for *t* in $0...t_{s1}$ and decreases for *t* in $t_{s1}...t_s$ if the exfiltration of further information initially increases the loss but the contribution of further information decreases. t_s depends on the amount of information to steal as well as on *at* risk tolerance. In fact, the probability *S* detects an exfiltration increases with the communication bandwidth it exploits. Hence, *at* can reduce the exfiltration time by using a larger bandwidth at the cost of increasing the probability of being discovered.

Impact functions with a positive, decreasing first order derivative also model the loss due to data loss. Now *at* needs the privilege of updating the data to overwrite them with some garbage. This requires a time t_s that increases if the overwrite has to be stealthy.

Suppose now that *S* is an industrial control system, ICS, and that *at* aims to sabotage the production or to damage the production plan. Stuxnet is a well known example of the latter [38,39]. The first order derivative of $Imp^S_{at,sr}(t)$ is constant and the second order one is zero if *at* aims to sabotage the production. We adopt the same function even when *at* aims to steal some information provided that *S* steadily produces new information to steal. If *at* aims to damage the plan, $Imp^{S}_{at,sr}(t)$ steadily increases till it reaches a threshold value t_d and is constant for larger times. The second order derivative of $Imp^{S}_{at,sr}(t)$ increases in the interval $0\ldots t_d$ if the cost of restoring the ICS increases with the time *at* controls it.

In the following, we use $Imp^{S}_{ag,sr}(t)$ instead of $Imp^{S}_{at,sr}(t)$ to denote the approximation through the output of a Haruspex experiment where *ag* models *at*. Furthermore, *sr* is always a set of goal *sg* of some agents. We also assume that $Imp^{S}_{ag,sg}(t)$ is defined for any value of *t* even if some mechanisms of S can discover an escalation or its results, i.e. an illegal file update. This introduce an upper bound on *t* that reduces the overall loss.

5.0 STRESS AND RETURN OF A SECURITY INVESTMENT

We discuss now the widely used assumption that any security investment to remove any vulnerability of *S* always has a non negative return. We can rephrase this assumption by saying that $Str^{S}_{at,sg}(t)$ decreases when the number of vulnerabilities of *S* decreases. We have experimentally verified this assumption is not true because a reduction in the number of vulnerabilities of *S* may increase $Str^{S}_{at,sg}(t)$. Hence, the patching of a vulnerability or the deployment of a countermeasure may actually increase an attacker success probability.

To explain why a lower number of vulnerabilities may increase an attacker success probability, consider that *at* has only a partial information on *S* and that $Str^{S}_{at,sg}(t)$ is related to the sequence *at* can select and to the probability it selects each sequence. Hence, *at* may select some escalations with a low success probability or a sequence longer than an escalation because some of its attacks are useless. The patching of some vulnerabilities of *S* may stop some of these escalations and force *at* to select further escalations it believes are worse than the stopped ones. The (owner) problem is that these escalation are actually better than those *at* previously selected even if *at* is not aware of it. This is actually another instance of the Braess's paradox [40] that shows that a larger number of paths may increase traffic congestion. In ICT security, a lower number of paths reduce the time to reach a goal as it increases the probability of selecting the best escalations.

This confirms that not only $Str^{S}_{at,sg}(t)$ takes into account a large number features of *S* and of *at* but it can also relate some features of *S* to those of *at*.

The only solution to avoid the Braess's paradox in ICT is to evaluate the cyber robustness of a new version before its deployment. This requires to predict the behaviour of the attackers against the new version.

6.0 EXPECTED LOSS IN AN INTERVAL

Each metric of the owner loss at a time *t* we consider splits the time interval $0..t$ into the time the attacker takes to reach a goal and the one it owns the rights it has acquired.

The metric $AvLoss^{S}_{ag,\,sg}(t)$ evaluates the owner average loss at *t* due to the agents in *sa* that aim to reach the goals in *sg*. This metric is monotone not decreasing in *t*, and $AvLoss^{S}_{ag,\,sg}(0)=0$.

6.1 AVLoss: One Agent with One Goal

If *ag* is an agent that aims to reach *g* and $Str'^{S}_{ag,\,sg}(t)$ is first order derivative of $Str^{S}_{ag,\,sg}(t)$ then

$$AvLoss^{S}_{ag,g}(t) = \int_{t' \in 0..t} Str'^{S}_{ag,\,sg}(t')\, Imp^{S}_{ag,g,i}(t-t')dt'.$$

$AvLoss^{S}_{at,g}(t)$ is the sum for *t'* in the interval $0\ldots t$ of the loss that *ag* produces if it reaches *g* at *t'*. The weight of each loss is the probability *ag* owns the rights in *g* for *t-t'*. This has the same probability that *ag* reaches *g* at *t'*. In turn, this is the first order derivative of $Str^{S}_{ag,\,sg}(t)$ at *t'*. Obviously, a finite sum replaces the integral if

$Str^S_{ag,\ sg}(t)$ changes in a discrete way in $0\ldots t$ because now the loss is due to a finite number of contributions. Suppose, as an example, that $Str^S_{ag,\ sg}(t)$ linearly increases for values of t in $100\ldots200$ while ag always reaches a goal in sg for larger times. Since $Str^S_{ag,sg}(200)=1$, $Str^S_{ag,sg}(t) = (t\text{-}100)/(100)$ for $t \in 1\ldots100$. Instead, $Imp^S_{ag,g,i}(t\text{-}t')$ increases as t^2 for t in the range $0\ldots100$. Then, for $t \geq 100$, $AvLoss^S_{ag,g}(t) = \int^t_{100} 1/100\ (t'^2)\ dt'$. If, instead, $Str^S_{ag,sg}(t)$ is a step function that increases of $1/100$ at each integer in $100\ldots200$, $AvLoss^S_{ag,g}(t)$ is a sum with one value for each integer in $100\ldots200$ not larger than t.

6.2 AVLoss: More General Scenarios

If ag aims to reaches any goal in $sg = \{g_1, \ldots, g_n\}$, $AvLoss^S_{ag,sg}(t)$ is the sum of the losses due to the distinct goals in sg. Hence.

$$AvLoss^S_{ag,sg}(t) = \Sigma_{\ g \in sg}\ AvLoss^S_{ag,g}(t)$$

This assumes that ag models an attacker that stops its attack as soon as it reaches any goal in sg. Under this assumption, goals in sg are mutually exclusive because at can reach at most one goal in sg.

If $sag = \{a_1, \ldots, a_f\}$ and each agent in sag has the goals in sg, then

$$AvLoss^S_{sag,sg}(t) = \Sigma_{\ agi \in sag}\ AvLoss^S_{agi,sg}(t)$$

i.e. the average loss due to a set of agents is the sum of the average loss due to each agent. We assume all the agents begins their attacks simultaneously.

7.0 VALUE AT RISK FOR ICT

CyVar is a financial metrics more accurate than *AvLoss* that extends the Value-At-Risk statistic to the ICT risk due to attackers with predefined goals. At first, we define *CyVar* for one agent with one goal, then we cover a set of goals and, lastly, a set of agents with the same or distinct goals. We assume all the agents begin their attacks simultaneously and that *CyVar* has the same confidence level of the Haruspex experiment(s) to generate the sample(s) to approximate the *stress*.

7.1 CyVar: One Agent with One Goal

$CyVar^S_{ag,g}(v,t)$ is the probability of a loss larger than v at a time t due to ag that aims to reach g. To compute $CyVar^S_{ag,g}(v,t)$, first of all we compute $t(v)$ as the minimum of Sle, the set with any time t_h where $Imp^S_{ag,g}(t_h) \geq v$. Sle includes any time t_h bounded by t and that results in a loss that is at least v. If Sle is empty, then $CyVar^S_{ag,g}(v,t)=0$. Otherwise, $t(v)$ exists and $CyVar^S_{ag,g}(v,t)$ is the probability that ag owns the rights in g for at least $t(v)$. This is the same probability that ag reaches g in, at most, t - t_i. Hence,

$$Sle = \{\ t_h \mid t_h \leq t \text{ and } Imp^S_{ag,g}(t_h) \geq v\ \}$$

$$t(v) = \text{if } Sle \neq \phi \text{ then } min(Sle) \text{ else } 0$$

$$CyVar^S_{ag,g}(v,t) = \begin{cases} Str^S_{at,g}(t\text{-}t(v)) \text{ if } t(v) \neq 0 \\ 0 \text{ if } t(v) = 0 \end{cases}$$

Informally, to compute $CyVar^S_{ag,g}(v,t)$ we invert $Imp^S_{ag,g}(t)$ to discover t_h, the time ag has to own the rights in g to produce a loss v. Using t_h, we compute the time $t(v)$ ag has available to reach g. Obviously, any increase in t_h simultaneously reduces any of $t(v)$, the probability that *ag* reaches *g* in $t(v)$, and $CyVar^S_{ag,g}(v,t)$.

The definition of $CyVar^S_{ag,g}(v,t)$ does not assume that *ag* always reaches *g* provided that it has available enough time.

As an example, suppose that $Imp^S_{ag,g}(t)= \alpha \cdot t^2$ and that we are interested in $CyVar^S_{ag,g}(\beta,\gamma)$ the probability of a loss larger than β at γ. *ag* can produce a loss β if it owns the rights in *g* for at least $\sqrt{\beta/\alpha}$ units of time. This probability of this loss is the same one that *ag* reaches *g* in, at most, $t=\gamma-\sqrt{\beta/\alpha}$. Anytime *t* is positive, this probability is the *stress* at $\gamma -\sqrt{\beta/\alpha}$. Hence,

$$CyVar^S_{ag,g}(\beta, \gamma)=Str^S_{at,g}(\gamma - \sqrt{\beta/\alpha})$$

This holds provided that *ag* starts its attacks at *0*. If this occurs with a probability $att(ag)$ then the probability of a loss larger than *v* is $att(ag) \cdot CyVar^S_{ag,g}(v,t)$. This takes into account even the case of no attacks. We can also define $CyVar^S_{ag,g}(v,t)$ if there is a probability distribution *ag* start its attacks at *t*.

7.2 CyVar: One Agent with Alternative Goals

If $sg=\{g_1, ..., g_n\}$, the definition of $CyVar^S_{ag,sg}(v,t)$ requires we know $Imp^S_{ag,gi}(t)$ for each g_i in *sg*.

A first approximation of $CyVar^S_{ag,sg}(v,t)$ considers the worst outcome,i.e. the probability of the largest loss

$$CyVar^S_{ag,sg}(v,t) = max\{CyVar^S_{ag,gi}(v,t),\ g_i \in sg\}.$$

We approximate $Str^S_{ag,gi}(t)$ as the percentage of runs where *ag* is successful in an experiment a run ends only when, and if, *ag* reaches g_i..

A more accurate approximation considers the contribution of each goal in *sg* to the loss. We compute this approximation through an experiment where each run ends when *ag* reaches any goal in $sg=\{g_1, ..., g_n\}$. Then, we approximate $CyVar^S_{ag,sg}(v,t)$ by considering each impact function $Imp^S_{ag,gx}(t)$ where $g_x \in \{g_1, ..., g_n\}$. For $Imp^S_{ag,gx}(t)$, we consider the time t_x that *ag* should own the rights in g_x to produce an impact *v* and approximate the probability that *ag* reaches g_x in $(t - t_x)$ through the percentage of runs where this happens. $CyVar^S_{ag,sg}(v,t)$ is the sum of all the probabilities.

7.3 CyVar: Agents with Alternative Goals

We define $CyVar^S_{sa,sg}(v,t)$ where $sa=\{ag_1, ..., ag_k\}$ is a set of agents sharing the goals in $sg=\{sg_1, ..., sg_k\}$ under the assumption that agents in *sa* do not interact or cooperate so that they are pair wise independent.

We compute $CyVar^S_{sa,sg}(v,t)$ by considering the alternative decompositions of *v* into a tuple *dv* with *k* non negative values $\{dv_1, ..., dv_k\}$ where $\sum_{j=1,k} dv_j=v$. $pr(dv)$ is the probability that any agent in *sa* results in a loss not smaller than the corresponding one in *dv*. Because of agent independence, for each *dv*, $pr(dv)$ is the product of the probabilities each ag_j result in a loss larger than dv_j. For each ag_j, this probability is *1* if $d_j = 0$ and it is $CyVar^S_{agj,sg}(dv_j,t)$ otherwise. If *Sv* includes any decomposition of *v*, then:

$$CyVar^S_{sa,sg}(v,t)=\sum_{sd \in Sv} pr(sd)$$

This shows that this approximation computes the probability that agents in *sa* result in a loss larger than *v* in three steps. The first one decomposes *v* into a set of tuples, each with a distinct value for each agent in *sa*. For each tuple, the second step computes the probability that each agent results in the corresponding loss. A further decomposition may occur if *sg* includes more than one goal. The third and last step sums all the probabilities.

As an example, if $sa = =\{ag_1,\ ag_2\}$ we compute $CyVar^S_{sa,sg}(100, 200)$ by decomposing 100 into 101 sets with the structure $\{v, 100\text{-}v\}$ where $v \in 0...100$. Then,

$$CyVar^S_{sa,sg}(100, 200) = \sum_{(v \in 0..100)} CyVar^S_{ag1,sg}(v, 200) \cdot CyVar^S_{ag2,sg}(100\text{-}v, 200)$$

8.0 AN EXAMPLE

This section applies the proposed metrics to three versions of an ICS that supervises and controls power generation. The first version is a real system actually in use. The owner wants to evaluate a security investment to select and deploy one of the two other versions. We analyse these versions through the alternative metrics to quantify the return of an investment that deploys one of them.

8.1 The Three Versions

Any ICS version is an ICT network segmented into four types of subnets: Central, Power Context, Process, and Control. The structuring of each version into subnets follows a *defence-in-depth* strategy.

Users of the intranet run the business processes of power generation through the nodes in a Central subnet. The plant operators interact with the SCADA servers through the nodes in a Power Context subnet. The SCADA servers and the systems in a Process network control power generation. Finally, the ICS drives the plant through some programmable logical components, PLCs, in a Control subnet.

S_1, the version of the ICS actually in use, [41] includes 49 nodes segmented into six subnets, see Fig. 8-1. The Central subnet includes 24 nodes, the Power Context includes 7 nodes. Then, Process subnet 1 and 2 include, respectively, 9 and 7 nodes. There is a connection from each Process subnet to a Control subnet with one PLC. Three nodes connect the Central subnet to the Power Context one. Two pairs of nodes in the Power Context network are connected to those in one Process subnet. Lastly, there is a connection from two nodes in each Process subnet to the corresponding Control subnet. S_2, the second ICS version, doubles the number of nodes by replicating each node without altering the number of connections between subnets. Lastly, S_3 includes 98 nodes as S_2 but it is a more accurate implementation of the *defence-in-depth* strategy because it splits the Central subnet into two subnets, see Fig. 8-2. Each subnet includes 24 nodes and there is connection from one subnet to the Power Context subnets.

8.2 Modelling Attackers and Their Impact

Any attacker aims to control the generation plan to reduce its efficiency. Under these assumptions, if it controls a PLC for a time t, the loss is $Low \cdot t$. Furthermore, the loss increases with the number of PLCs the attacker controls. Hence, $Imp^S_{at,g}(t) = n \cdot Low \cdot t$ where n is the number of PLCs that at controls for a time t. Initially, each attacker only owns some rights on a node in the Central subnet.

We introduce four classes of agents, $T_1, ..., T_4$, to describe a scenario. All the agents have one goal and those in the same class have the same goal but adopt distinct selection strategies. Haruspex introduces classes with agents that only differ because of their selection strategies to handle the uncertainty due to the lack of information on the attacker preferences. The assessment can analyse the losses due to agents in the same class to discover the most dangerous one. Since agents in the same class cover uncertainty about attacker preferences, when analyzing the loss due to some agents we always assume they belong to distinct classes.

T_1 agents aim to control both the PLCs in the ICS. Hence, their goal g_1 includes access rights on both PLCs.

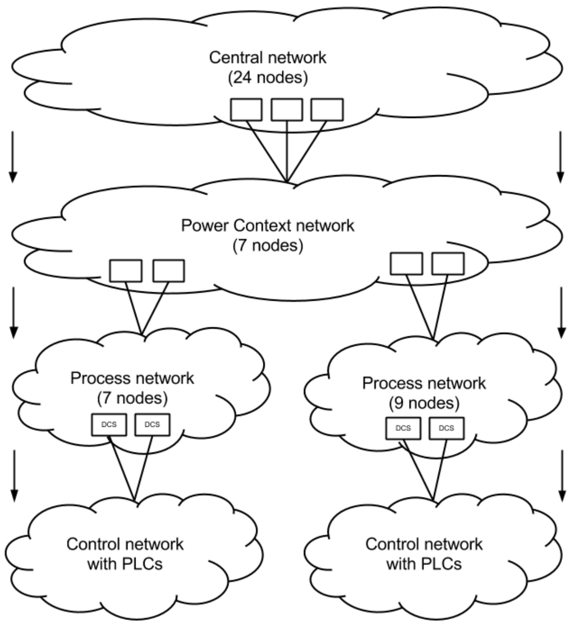

Figure 8-1: First Version of the ICS

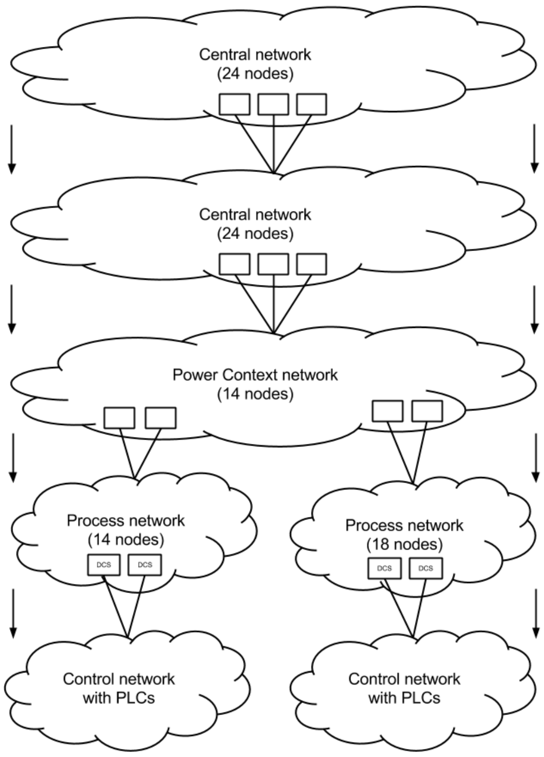

Figure 8-2: Third Version of the ICS

Instead, T_2 agents have no preference on the PLC to control and they stop their attacks as soon they control either PLC. g_3 is the goal of agents in T_3 while g_4 is the one of agents in T_4. Each of these goals includes access rights on one, predefined PLC. As a consequence, $Imp^S_{at,g}(t)=2 \cdot Low \cdot t$ if *ag* belongs to the T_1 class, while $Imp^S_{ag,g}(t)=Low \cdot t$ if *ag* belongs to any other class.

8.3 Stress of each Version

In the following, we use hours as time units. Fig. 8-3, Fig. 8-4, and Fig. 8-5 show the stress curves of the most dangerous agent in each class for each ICS version. The confidence level of these curves is 95%.

According to the figures, in S_1 the most dangerous T_2 agent reaches g_2 in about twelve hours. Any agent in another class reaches its goal in about fourteen hours, i.e. about two hours later. The most dangerous agent for S_2 is a T_2 agent that reaches g_2 in about 21 hours. Other agents take one more hour. Both T_3 and T_4 agents take longer than a T_2 agent that can freely choose which PLC to control.

In S_3, each agent takes longer to reach its goal than in S_2. The difference is low because a T_2 agent reaches g_2 only *20* minutes later than in S_2. The remaining agents reach their goal after more than two hours.

As expected, S_1 is the most fragile version because of the low number of attacks an agent needs to reach a goal. The number of nodes in S_2 confuses the agents and increases both the time to acquire information on the nodes and the one to reach a goal. Finally, S_3 is the most cyber robust version because its number of nodes and that of subnets increase both the number of attacks agents have to execute and the time to reach a goal. This results in the lowest *stress*.

8.4 AVLoss for the Three Versions

For each version, we consider the average loss due to the most dangerous agent. To simplify the analysis we use a linear interpolation of the stress function. For all the system the average loss increases with t^2. In a first interval, the coefficient is half the slope of the line interpolating the stress function. Then, the coefficient is *1/2*. The critical difference is in the position of the first interval. In the first version, *AvLoss(t)* is positive after eight hours and twenty minutes and it increases as $t^2/2$ after a bit more than 14 hours. Instead, in S_2, the corresponding interval begins after *14* hours and it ends after a bit more than *22*. Hence, in this version the loss begins when in the other it is peaking. Lastly, in S_3 this interval ends after *24* hours.

8.5 CyVar for the Three Versions

Let us assume that *Low=10* and that we aim to assess and manage the risk due to agents in $T_1, ..., T_4$ if $V = 100$ and $t = 24$ hours. The impact of *ag* in the T_1 class is *V* if it owns the rights in g_1 for $V/(2 \cdot Low) = 100/20 = 5$ hours. Hence, *ag* should reach g_1 in less than *19* hours. In S_1, the most dangerous T_1 agent always reaches g_1 in less than 19 hours. Hence, $CyVar^{S1}_{ag,g1}(100, 24)=1$ and the owner suffers this loss in *24* hours anytime a T_1 agent targets S_1. The situation strongly changes in S_2 where the probability that a T_1 agent results in a loss equal to *100* belongs to the range *[0.2 ... 0.3]*. Lastly, $CyVar^{S3}_{ag,s}(100, 24)=0$ because in S_3 a T_1 agent cannot reach g_1 in *19* hours. Hence, the owner can avoid any loss by deploying S_3. This deployment is cost effective if its cost is lower than *100*.

The impact of *ag* in the T_2 class is *V* if it owns the rights in g_2 for *10* hours. Hence, *ag* should reach g_2 in less than *14* hours. This always happens in S_1, i.e. $CyVar^{S1}_{ag,g}(100, 24)=1$. Instead, this never happens in both S_2 and S_3, i.e. $CyVar^{S2}_{ag,g2}(100, 24) = CyVar^{S3}_{ag,g2}(100, 24)=0$. When considering T_3 and T_4 agents, the investment to change the structure of the ICS from S_2 to S_3 has no return because these agents cannot achieve their goals in the interval of interest when attacking S_2. Instead, the return of the investment to change the ICS structure from S_1 to S_2 is positive because it actually reduces the agent impacts.

Consider now a T_3 agent and a T_4 one that begin their attacks simultaneously. These agents aim to control a distinct PLC and are independent, because they do exchange privileges or information. Hence, their impact is *V* if the sum of the times they own the rights in, respectively, g_3 and g_4 is larger than 10 hours.

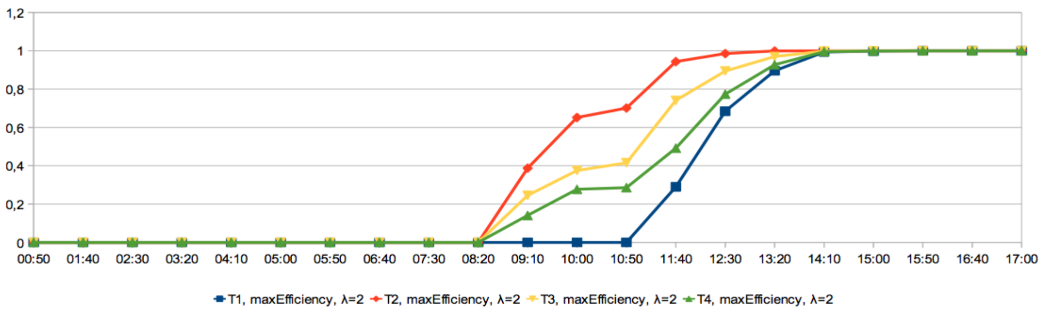

Figure 8-3: First Version: Stress Curve of the Most Dangerous Agents

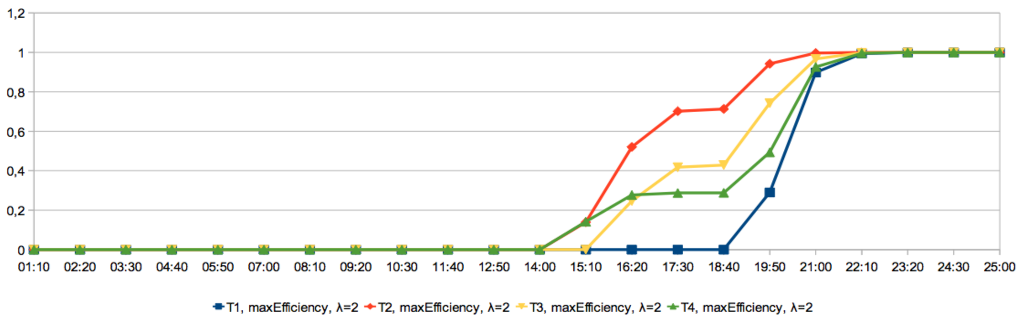

Figure 8-4: Second Version: Stress Curve of the Most Dangerous Agents

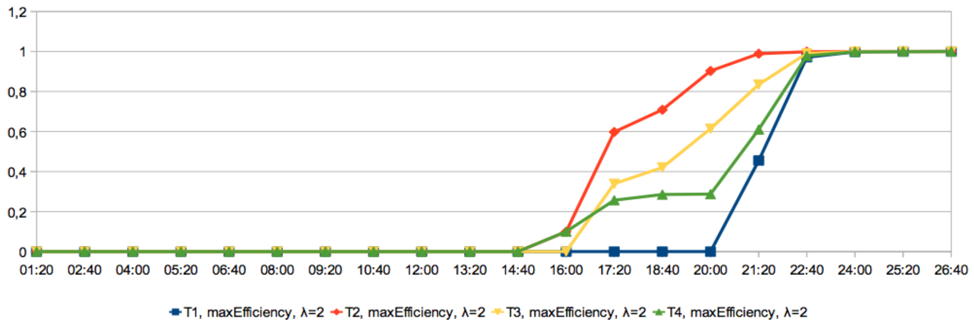

Figure 8-5: Third Version: Stress Curve of the Most Dangerous Agents

At first, we consider a decomposition where each agent owns these rights for, at least, *5* hours. This implies each agent should reach its goal in at most *19* hours. This always happens in S_1, i.e. $CyVar^{S1}_{sa,sg}(100, 24)=1$. In S_2, there is a *0.6* probability that one agent reaches g_3 in less than *19* hours while there is a *0.5* probability the other reaches g_4 in the same time. Hence, the joint probability is *0.3*. These probabilities change only slightly in S_3.

In an alternative decomposition, one agent owns the rights for *6* hours and the other for *4* hours. Hence, the first should reach its goal in less than *18* hours while the second one has, at most, *20* hours available. In S_2, the probabilities of the two events are, respectively, *0.9* and *0.3* and the joint one is *0.27*. This shows that $CyVar^{S2}_{sa,sg}(100, 24)>0.5$ because we have considered just two of the possible decompositions.

In S_3, the probabilities of the two events are, respectively, *0.3* and *0.6* and the joint one is less than *0.2*.

To compute the largest loss due to the two agents and the corresponding *CyVar*, we consider the agent contributions and the corresponding probabilities. This analysis for S_2 shows that the T_3 agent can own the rights in its goal for, at most, *10* hours out of a *24* hours interval. Instead, the other agent can own the rights in its goal for a bit less than *9* hours.

We determine the lowest impact by considering that both agents own the rights in the corresponding goal for at least *2* hours. Hence, the impact of agents in *sa* ranges from *100(10+9)* to *100(2+2)*. The corresponding impact for S_3 is similar and this further confirms the low return of the investment to change S_2 into S_3.

9.0 CONCLUSION

Cyber robustness measures the ability of an ICT system of resisting to attackers that escalate their privileges through a sequence of attacks. We have proposed alternative metrics for this ability. The *security stress* considers the probability attackers reach their goals in a time interval. *CyVar* and *AvLoss* use the *stress* to compute, respectively, the probability of a loss in an interval and the average loss in an interval. We can evaluate the metrics we have proposed through the sample that the Haruspex suite returns by simulating the attackers. Obviously, distinct tools and alternative approaches may be adopted to compute the sample or to approximate the metrics.

We have applied the proposed metrics to three versions of an ICS in a scenario where attackers have distinct goals. The metrics measure the return of an investment to change the first version into one of the other ones.

Future developments of this work concern the definition of metrics depending on the countermeasures to deploy to stop an attacker. These metrics will measure the work to prevent the attackers from reaching their goals rather than the work of the attackers to escalate their privileges.

[1] Baiardi, F., Tonelli, F., Bertolini, A., Bertolotti, R., Guidi, L.: Security stress: Evaluating ICT Robustness through a Monte Carlo Method. In: Ninth Conf. on Critical Infrastructures Sec., Lymassol, Cyprus (2014)

[2] Baiardi, F., Tonelli, F., Bertolini, A.: CyVar: Extending Var-At-Risk to ICT. Springer, Berlin, Heidelberg (2015)

[3] Hamilton, S.N., Hamilton, W.L.: In: Jajodia, S., Samarati, P., Cimato, S. (eds.) Adversary Modeling and Simulation in Cyber Warfare, pp. 461–475. Springer, Boston, MA (2008)

[4] Baiardi, F., Coro, F., Tonelli, F., Sgandurra, D.: Automating the Assessment of ICT Risk. Journ. of Information Sec. and Applications 19(3), 182–193 (2014)

[5] Baiardi, F., Sgandurra, D.: Assessing ICT Risk through a Monte Carlo Method. Environment Systems and Decisions, 1–14 (2013)

[6] Baiardi, F., Corò, F., Tonelli, F., Guidi, L.: Gvscan: Scanning Networks for Global Vulnerabilities. In: First Int. Workshop on Emerging Cyberthreats and Countermeasures, Regensburg, Germany (2013)

[7] Baiardi, F., Corò,F. ,Tonelli,F. ,Sgandurra,D.: A Scenario Method to Automatically Assess ICT Risk. In: Proc. Of Parallel and Dist. Processing 2014, Turin, Italy (2014)

[8] Baiardi, F., Corò, F., Tonelli, F., Guidi, L.: Qsec: Supporting Security Decisions on an IT Infrastructure. In: Eighth Conf. on Critical Infrastructures Sec., Amsterdam, The Netherlands (2013)

[9] Kotenko, I., Konovalov, A., Shorov, A.: Agent-based Modeling and Simulation of Botnets and Botnet Defense. In: Conf. on Cyber Conflict. CCD COE Publications. Tallinn, Estonia, pp. 21–44 (2010)

[10] Barreto, A.B., H., H., E., Y.: Developing a Complex Simulation Environment for Evaluating Cyber Attacks. In: The Interservice/Industry Training, Simulation and Education Conf. (I/ITSEC) (2012)

[11] Sarraute, C., Richarte, G., Lucángeli Obes, J.: An algorithm to find optimal attack paths in nondeterministic scenarios. In: 4th ACM Workshop on Security and AI. AISec '11, pp. 71–80. ACM, New York, NY, USA (2011)

[12] Futoransky, A., Miranda, F., Orlicki, J., Sarraute, C.: Simulating cyber-attacks for fun and profit. In: Proc. of the 2nd Int. Conf. on Simulation Tools and Techniques. Simutools '09, pp. 4–149 (2009)

[13] Ten, C.-W., Manimaran, G., Liu, C.-C.: Cybersecurity for critical infrastructures: attack and defense modeling. IEEE Trans. on Systems, Man and Cybernetics 40(4), 853–865 (2010)

[14] S.Rubinshtein, Puzis, R.: Modeling and reconstruction of multi-stage attacks. In: 2016 IEEE International Conference on Software Science, Technology and Engineering (SW- STE), pp. 135–137 (2016). doi:10.1109/SWSTE.2016.27

[15] Hassell, S., Beraud, P., Cruz, A., Ganga, G., Martin, S., Toennies, J., Vazquez, P., Wright, G., Gomez, D., Pietryka, F., Srivastava, N., Hester, T., Hyde, D., Mastropietro, B.: Evaluating network cyber resiliency methods using cyber threat, vulnerability and defense modeling and simulation. In: MILCOM 2012 - 2012 IEEE Military Communications Conference, pp. 1–6 (2012).

[16] Linkov,I.,Eisenberg,D.A.,Plourde,K.,Seager,T.P.,Allen,J.,Kott,A.:Resiliencemetrics for cyber systems. Environment Systems and Decisions 33(4), 471–476 (2013)

[17] Vaughn Jr., R.B., Henning, R., Siraj, A.: Information assurance measures and metrics - state of practice and proposed taxonomy. In: System Sciences, Proc. of the 36th Annual Hawaii Int. Conf., p. 10 (2003)

[18] Schudel, G., Wood, B.: Adversary work factor as a metric for information assurance. In: Workshop on New Security Paradigms. NSPW '00, pp. 23–30. ACM, New York, NY, USA (2000)

[19] Langweg, H.: Framework for malware resistance metrics. In: 2nd ACM Workshop on Quality of Protection, pp. 39–44. ACM, New York, NY, USA (2006)

[20] Wang, L., Jajodia,S., Singhal,A., Cheng,P., Noel,S.: k-zerodaysafety: A network security metric for measuring the risk of unknown vulnerabilities. IEEE Trans. Dependable Sec. Comput. 11(1), 30–44 (2014)

[21] Sandoval, J.E., Hassell, S.P.: Measurement, identification and calculation of cyber defense metrics. In: MILITARY COMMUNICATIONS CONFERENCE 2010, pp. 2174–2179 (2010).

[22] Jaquith, A.: Security Metrics: Replacing Fear, Uncertainty, and Doubt

[23] Payne, S.C.: A guide to security metrics. SANS Institute (2006)

[24] Swanson, M.: Security metrics guide for information technology systems. Technical report, NIST, US Department of Commerce (2003)

[25] Sarraute, C.: On exploit quality metrics – and how to use them for automated pentesting. In: Proc. of

8.8 Computer Security Conf. (2011)

[26] Jansen, W.: Directions in security metrics research. National institute of standards and technology. Computer Security Division (2009)

[27] Jain, S., Ingle, M.: Review of security metrics in software development process. International Journal of Computer Science and Information Technologies 2(6), 2627–2631 (2011)

[28] Pamula, J., Jajodia, S., Ammann, P., Swarup, V.: A weakest-adversary security metric for network configuration security analysis. In: 2Nd ACM Workshop on Quality of Protection, pp. 31–38. ACM, New York, NY, USA (2006)

[29] LeMay, E., Unkenholz, W., Parks, D., Muehrcke, C., Keefe, K., Sanders, W.: Adversary- driven state-based system security evaluation. In: 6th Int. Workshop on Security Measurements and Metrics, pp. 5–159. ACM, New York, NY, USA (2010)

[30] Böhme, R.: Security metrics and security investment models. In: Echizen, I., Kunihiro, N., Sasaki, R. (eds.) Advances in Information and Computer Security. Lecture Notes in Computer Science, vol. 6434, pp. 10–24. Springer, (2010)

[31] Böhme,R.:Security metrics and security investment models. In: 5th Int.Conf. on Advances in Inf. and Computer Security. IWSEC'10, pp. 10–24. Springer, Berlin, Heidelberg (2010)

[32] Gordon, L.A., Loeb, M.P.: The economics of information security investment. ACM Trans. Inf. Syst. Secur. 5(4), 438–457 (2002)

[33] Kundur, D., Feng, X., Liu, S., Zourntos, T., Butler-Purry, K.L.: Towards a framework for cyber attack impact analysis of the electric smart grid. In: First IEEE Int. Conf. on Smart Grid Communications, pp. 244–249 (2010). IEEE

[34] Alderson, D.L., Brown, G.G., Carlyle, W.M.: Operational models of infrastructure resilience. Risk Analysis 35(4), 562–586 (2015). doi:10.1111/risa.12333

[35] ISO, I.: Iec 31010: 2009. Risk management-Risk assessment techniques

[36] Rios Insua, D., Rios, J., Banks, D.: Adversarial risk analysis. Journal of the American Statistical Association 104(486), 841–854 (2009)

[37] La Corte, A., Scatà, M.: Failure analysis and threats statistic to assess risk and security strategy in a communication system. In: ICSNC 2011, The Sixth International Conference on Systems and Networks Communications, pp. 149–154 (2011)

[38] Byres, E., Ginter, A., Lingell, J.: How Stuxnet Spread - A Study of Infection Paths in Best Practice Systems. White Paper. Tofino Report, Abterra Technologies ScadaHacker.com (2011)

[39] Langner, R.: Stuxnet: Dissecting a cyberwarfare weapon. Security & Privacy, IEEE 9(3), 49–51 (2011)

[40] Braess, D., Nagurney, A., Wakolbinger, T.: On a paradox of traffic planning. Transportation Science 39(4), 446–450 (2005)

[41] Nai Fovino, I., Masera, M., Guidi, L., Carpi, G.: An experimental platform for assessing scada vulnerabilities and countermeasures in power plants (2010)