Metrics for Cyber Robustness

F.Baiardi¹, F.Tonelli¹ and A.Bertolini², M.Montecucco²
¹Department of Computer Science, University of Pisa
²Haruspex 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. The metrics strongly contribute to cyber awareness as they anticipate how a system can resist to some distinct set of attackers. Furthermore, they support a comparison of alternative versions of a system with respect to this ability to drive an investment to increase cyber robustness. 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 metric relates the probability an attacker is successful with the time it takes to implement its escalations. In turn this time depends on the alternative escalations it can implement, the number of attacks in these escalations, and the attack success probabilities.

Then, we define two financial metrics to evaluate the loss when an attacker reaches a goal and it owns the corresponding rights for a time interval. We relate each metric with the stress through an impact that pairs a loss with each time interval an attacker owns the access rights in a goal. This implies that the owner suffers a loss only when, and if, the attacker reaches a goal. The loss due to the attacks in an escalation is neglected with respect to the one the attacker produces when controlling some resources. Even in a destructive attack, the attacker needs some access rights to produce a loss.

\( AvLoss \) 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 acquires the access rights in a goal at \( t \) and it is the first
order derivative the stress due to the attacker.

*CyVar* is a financial metric that extends *Value-At-Risk*, *VAR*, to evaluate the loss more accurately. *VAR* is a risk measure for a security investment that focuses on the odds of losing money. It has three inputs: a time $t$, a confidence level $c$, and a loss amount $l$. Then, it returns the probability that the investor loses more than $l$ at $t$. $c$ is confidence level of this probability. *CyVar* returns the same information in a scenario where attackers escalate their privileges. We compute *CyVar* at $t$ by splitting $t$ into two intervals: the one it takes to reach a goal and the one it owns the corresponding rights. We compute the probability the attacker reaches a goal in an interval through the stress.

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 rank the countermeasure to deploy. The forecast of the attacker behaviours is also fundamental to compute the probabilities of interest. Obviously, distinct approaches and tools may be adopted.

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 the 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] is 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\textsuperscript{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 \cite{iec31010} suggests. Furthermore, Haruspex supports security-by-design because it can assess and manage ICT risk starting from the models of the system and of the attackers that it can build even before system deployment.

The Haruspex suite is an integrated set of tools assesses and manages ICT risk in a scenario where intelligent attackers aim to control some components of an ICT system by escalating its privileges through a sequence of attacks. 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 preferences and priorities. After building the models, the engine is the suite tool that discovers the escalations of each attacker by simulating its behaviour 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 \cite{gametheory}. In each iteration of the game, 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 escalations that include the new ones. The game ends either when the engine cannot implement an escalation or when the manager cannot stop some escalations. This tool is not analysed in the following because cannot redesign the whole system.

\subsection{Modelling a System and the Attackers}

We describe in more details the Haruspex models to simulate the target system and the attackers\textsuperscript{1}. Haruspex describes 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. The module vulnerabilities enable the attacks in the escalations. To describe social engineering attacks such as phishing, some components model the users or the system administrators.

Haruspex models an attack and its actions 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 the success probability. This probability depends on both the attacker and the action complexity.

The target system model describes the modules, their vulnerabilities, and the attacks they enable. Haruspex supports both public and suspected vulnerabilities. Attackers may discover suspected vulnerabilities when they are attacking the system. Haruspex pairs each suspected vulnerability with the probability it becomes public. This supports a what-if analysis of how new vulnerabilities affect the overall risk.

Haruspex assumes that no single attack grants all the rights in a goal and that the attacker needs to escalate its privileges 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 intelligent 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, $\lambda(\text{ag})$, a non-negative integer, models how the attacker selects the attacks in its escalations. The selection strategy defines how $ag$ ranks alternative escalations, while $\lambda(\text{ag})$ defines the length of the

\textsuperscript{1} An ancient Tuscany forecaster
sequences it ranks. \( \text{ag} \) randomly selects the attacks in an escalation if \( \lambda(\text{ag}) = 0 \). Otherwise, \( \text{ag} \) selects the sequence to implement by ranking all those with, at most, \( \lambda(\text{ag}) \) attacks. We use sequences rather than escalations because they may grant a proper subset of the rights in a goal. The strategy ranks any sequence \( \text{ag} \) can implement by exploiting the access rights it owns and it always returns a sequence that leads to a goal, if it exists. When the set of rights of \( \text{ag} \) is small, none of the sequences it can implement may lead to a goal. Here, \( \text{ag} \) selection strategy ranks the sequences according to the attributes their attacks. An assessment pairs each agent with a predefined selection strategy that may consider distinct attributes, as an example:

- \( \text{maxProb} \): considers the success probability of a sequence;
- \( \text{maxIncr} \): considers the number of rights the sequence grants;
- \( \text{maxEff} \): considers the ratio between success probability and execution time of a sequence.

There is no guarantee that any strategy returns a sequence leading to a goal.

An attacker runs a vulnerability scanning of some nodes to collect information on the vulnerabilities enabling the attacks in the sequences it can implement. Hence, larger values of \( \lambda(\text{ag}) \) increase the selection time due to the ranking of longer sequences. This shows that \( \lambda(\text{ag}) \) describes how \( \text{ag} \) solves the "collect or exploit" dilemma 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 \textit{continuity} defines the number of attacks an agent 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 uses the target system model and those of the agents in a scenario to implements an Haruspex experiment with multiple independent \textit{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 \textit{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 \textit{continuity}.

The \textit{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 \textit{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 \textit{engine} collects observations on the escalation of each agent, the components it has attacked and any goal it has reached. The observations build the sample that is the tool output. The confidence level of each statistic increases with the number of runs in the experiment because the \textit{engine} collects one observation in each run. The \textit{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. The \textit{engine} generalizes \textit{adversary simulation} as it executes the model of each attacker to forecast its behaviour against the target system.

### 3.3 Validation of the Suite

The accuracy of the suite predictions fully depends upon 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 exercise 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 score defending teams based on their performance.

One of the teams has used the 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 of the suite is the output of a vulnerability scanning of the nodes to defend because 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 can freely access any module of the target system. 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 more valuable to the defender. To model that agents fully exploits any information it acquires, we have assumed $\alpha_{ag}=3$ 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 they have computed the optimal list of vulnerabilities to patch in the time interval before the attacks of the red team. The team members manually apply the patches.

The target system is affected by more than one thousand vulnerabilities. The manager returns a list with about 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 approach that the suite adopts can minimize the overall complexity and focus the discovery on those escalations some attacker implements.

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 but the next sections use it to define financial metrics. Instead, the impact maps the time an attacker owns the rights in a goal 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 $St_{S \text{at,sg}}^{\text{S}}(t)$, the security stress of $S$ at $t$ due to $at$ that aims to reach any goal in $sg$. If $Pr\text{Succ}^{S}_{\text{at,sg}}(t)$
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, that each escalation is successful under the condition \( at \) selects it. The probability of selecting an escalation is not independent from the one of selecting a distinct one because both these probabilities 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 two probability that determines \( \PrSucc^S_{at,sg}(t) \) namely the one that a 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 \( t \) tolerates a larger number of attack failures because \( at \) can handle the failure of an attack with a larger number of times.

Suppose that the vulnerabilities of \( S \) enable \( at \) to implement two escalations to reach the only goal in \( sg \). These escalations include, respectively, three and four attacks. Each attack takes two units of time 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 may be lower than 1/8 any time there is a non-null probability that \( ag \) selects some sequences that do not lead to a goal. \( Str^S_{at,sg}(t) \) is at most 6/16 for any \( t \in [8..10) \) because in this interval \( at \) can implement any escalation, provided that all its attacks are successful, and the first one even if at most one of its attacks fails. The probabilities of the three events are, respectively, 1/8, 1/16 and 3/16. Again, \( at \) may select an attack sequence that differs from the two escalations.

To discuss in more details how some attributes of \( S \) and of \( at \) influence \( Str^S_{at,sg} \), we consider 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 trying to change the shape of \( S \). This force is ineffective till \( t_0 \) when the escalations of \( at \) begin to change shape of \( S \). As \( t \) increases, the selection strategy of \( at \) and its look-ahead become less and less critical because \( at \) can select and implement longer escalations. \( S \) definitely 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 execution time of attacks and the length of the shortest escalation to \( g \) in \( sg \). \( t_1 \) depends on the success probability of attacks in the sequences that \( at \) selects. This probability determines the time to implement a sequence because \( at \) may repeat an attack for a number of times before it succeeds. \( t_1 - t_0 \) 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 \( Str^S_{at,sg}(t) \) can evaluate cyber robustness in a more robust way 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 do not return accurate information on the boundaries of the interval where \( S \) can resist to \( at \).

\( Str^S_{at,sg}(t) = 1 - Str^S_{at,sg}(t) \) is the inverse of the stress and it 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:

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

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.

The stress is undefined if the attackers in $sa$ have distinct goals because they have distinct motivations and result in distinct impacts. However, we may define this 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 of attackers with the same goals only.

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

$Str^{S}_{at,ag}(n)$ is an alternative definition of security stress where $n$ is the largest number of attacks $at$ can execute to reach a goal in $sg$. Obviously, $n$ also includes failed attack executions. $Str^{S}_{at,ag}(n)$ 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 the time it has available. However, $Str^{S}_{at,ag}(n)$ neglects the work to collect information about $S$ to select the attack to implement. Obviously, we cannot deduce a closed form for $Str^{S}_{at,ag}(n)$ but we can approximate even this definition as 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 by $Str^{S}_{at,ag}(n)$ the adoption of this approximation.

### 4.2 Impact Function

$Str^{S}_{at,ag}(t)$ evaluates how long it takes $at$ to acquire the control of $S$. This information is critical when $at$ can produce an impact immediately after acquiring the control. A typical example may be a terrorist aiming to shutdown an ICS to destroy or sabotage a production plan or to create a large-scale pollution. In a commercial context, $at$ may aim to steal some IP, e.g. a component design or some source code, or to reduce the efficiency of a production plan.

A quantitative evaluation of the loss is fundamental to discover if the return of a security investment is larger than the potential loss.

The loss for the owner of $S$ due to an attacker is strongly related to the time that it owns the access rights in a goal before the discovery of its attacks. For this reason, we define the loss through an impact function $Imp^{S}_{at,ag}(t)$. $Imp^{S}_{at,ag}(t) = l$ implies the owner loss is $l$ if $at$ owns the rights in $g$ for $t$.

$Imp^{S}_{at,ag}(t)$ is monotone non decreasing in $t$ and $Imp^{S}_{at,ag}(0) = 0$ but its shape fully depends on both $S$ and $at$ motivations. To analyse these dependencies we suppose that $at$ aims to reach the rights in $g$ to read some information in $S$ and steal some IP. After reaching $g$, $at$ needs $t_{s}$ to steal the information. If $at$ starts the exfiltration as soon as it reaches $g$, then $Imp^{S}_{at,ag}(t)$ increases if $t$ belongs to $0...t_{s}$ and it is constant for larger values. The second order derivative of $Imp^{S}_{at,ag}$ 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 $0...t_{l}$ and
increases for $t$ in $t_1...t_4$ if the exfiltration of further information initially increases the loss but the contribution of further information decreases.

$t_4$ depends on the amount of information to exfiltrate as well as the risk tolerance. The latter dependency arises because the probability $S$ detects an exfiltration increases with the communication bandwidth it exploits. Hence, by using a larger bandwidth $at$ reduces the exfiltration time at the cost of increasing the probability that $S$ discovers the exfiltration. Impact functions with a positive, decreasing first order derivative also model an attack to destroy some data. Now $at$ aims to acquire the privilege of updating the data to overwrite them with some garbage. This requires a time $t_4$ 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 $Imp_{ag}^S(t)$ is constant and the second order one is zero if $at$ aims to sabotage the production. The same function applies even when $at$ aims to steal some IP provided that $S$ steadily produces new IP to steal. If $at$ aims to damage the plan, $Imp_{ag}^S(t)$ steadily increases till it reaches a threshold value $t_j$ and is constant for larger times. The second order derivative of $Imp_{ag}^S(t)$ increases in the interval $0...t_j$ if the cost of restoring the ICS increases with the time $at$ controls it.

In the following, we use $Imp_{ag}^S(t)$ instead of $Imp_{at}^S(t)$ to denote the approximation through the output of a Haruspex experiment where $ag$ models $at$. We also assume that $Imp_{ag}^S(t)$ is defined for any value of $t$ even if some mechanisms of $S$ may discover a privilege escalation or its results, such as an illegal file update. These mechanisms may 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 $Str_{at,sg}^S(t)$ decreases with the number of vulnerabilities of $S$. We have experimentally verified this assumption is not true because a lower number of vulnerabilities of $S$ may increase $Str_{at,sg}^S(t)$. 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 an $at$ has only a partial information on $S$ and that $Str_{at,sg}^S(t)$ is related to the sequences the attacker can select and to the probability of selecting each sequence. Hence, an attacker 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 the selection of other escalations $at$ believes are worse than the stopped ones. The (owner) problem is that these escalations are actually better than those 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 $Str_{at,sg}^S(t)$ takes into account a large number features of $S$ and of $at$ but it also considers how some features of $S$ are related 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

$AvLoss_{ag, sg}^S(t)$ is a metric to evaluate the average loss of the owner at $t$ due to the agents in $sa$ that aim to
reach the goals in \( sg \). This metric is defined in terms of \( Str_{ag,sg}^S(t) \) and \( Imp_{ag,g}(t' - t) \), it is monotone not decreasing in \( t \), and \( AvLoss_{ag,sg}(t) = 0 \) if \( Str_{ag,sg}(t) = 0 \).

### 6.1 AVLoss: One Agent with One Goal

At first, we consider the agent \( ag \) that aims to reach \( g \), if \( Str_{ag,sg}^S(t) \) is first order derivative of \( Str_{ag,sg}^S(t) \) then

\[
AvLoss_{ag,sg}^S(t) = \int_{t=0}^t Str_{ag,sg}^S(t') Imp_{ag,g}(t' - t') dt'.
\]

\( AvLoss_{ag,sg}^S(t) \) is the sum for \( t' \) in the interval \( 0...t \) of the loss if \( ag \) reaches \( g \) at \( t' \). The weight of each loss is the probability \( ag \) owns the rights in \( g \) for \( t' - t \). This probability is the one that \( ag \) reaches \( g \) at \( t' \), that is the first order derivative of \( Str_{ag,sg}^S(t) \) at \( t' \). Obviously, a finite sum replaces the integral if \( Str_{ag,sg}^S(t) \) changes in a discrete way in \( 0...t \) because a finite number of points contribute to the loss.

Suppose that \( Str_{ag,sg}^S(t) \) linearly increases for values of \( t \) in \( 100...200 \) while \( ag \) always reaches a goal in \( sg \) for larger times. Since \( Str_{ag,sg}^S(200) = 1 \), \( Str_{ag,sg}^S(t) = (t-100)/(100) \) for \( t \in 1...100 \). Instead, \( Imp_{ag,g}(t' - t) \) increases as \( t' \) for \( t \) in the range \( 0...100 \). Then, for \( t \geq 100 \), \( AvLoss_{ag,sg}^S(t) = \int_{t=100}^{t=200} 1/100 \int_{t'=t}^{t'=t'} dt' \). If, instead, \( Str_{ag,sg}^S(t) \) is a step function that increases of \( 1/100 \) at each integer in \( 100...200 \), \( AvLoss_{ag,sg}^S(t) \) is a sum with one value for each integer in \( 100...200 \) not larger than \( t \).

### 6.2 AVLoss: More General Scenarios

If \( ag \) aims to reaches any goal in \( sg = \{ g_1, ..., g_d \} \), we compute the losses \( AvLoss_{ag,sg}^S(t) \) for any \( g \in sg \) and return their weighted average. As usually, the weight of each loss is the probability \( ag \) reaches the corresponding goal. We approximate these probabilities as the percentage of runs \( ag \) reaches the corresponding goal in a Haruspex experiment where it can reach any goal in \( sg \).

The loss \( AvLoss_{ag,sg}(t) \) due to a set of agents \( sag = \{ a_1, ..., a_d \} \) where each has the goals in \( sg \) is the sum of \( AvLoss_{ag,sg}^S(t) \) for any \( ag_i \in sag \).

### 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 escalations 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

\( CyVar_{ag,g}(v,t) \) is the probability of a loss larger than \( v \) at a time \( t \) due to \( ag \) that aims to reach \( g \). To compute \( CyVar_{ag,g}(v,t) \), first of all we compute \( t(v) \) as the minimum of \( Sle \), the set with any time \( t_i \) where \( Imp_{ag,g}(t_i) \geq v \). \( Sle \) includes any time \( t_i \) bounded by \( t \) and that results in a loss that is at least \( v \). If \( Sle \) is empty, then \( CyVar_{ag,g}(v,t) = 0 \). Otherwise, \( t(v) \) exists and \( CyVar_{ag,g}(v,t) \) 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,

\[
Sle = \{ t_i \mid t_i \leq t \text{ and } Imp_{ag,g}(t_i) \geq v \}
\]

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

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

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

The definition of $\text{CyVar}^S_{ag,g}(v,t)$ does not assume that $ag$ reaches $g$ with a probability equal to $I$ provided that enough time is available.

As an example, suppose that $\text{Imp}_{ag,g}^S(t) = \alpha \cdot t^2$ and that we are interested in $\text{CyVar}^S_{ag,g}(\beta, \gamma)$ the probability of a loss larger than $\beta$ at $\gamma$. $ag$ can produce a loss $\beta$ if it owns the rights in $g$ for at least $\sqrt{\beta/\alpha}$ units of time. This loss has the same probability 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,

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

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

### 7.2 CyVar: One Agent with Alternative Goals

If $sg=\{g_1, ..., g_n\}$, we can define $\text{CyVar}^S_{ag,sg}(v,t)$ provided that we know $\text{Imp}_{ag,sg}^S(t)$ for each $g_i$ in $sg$.

A first approximation of $\text{CyVar}^S_{ag,sg}(v,t)$ computes $\text{CyVar}^S_{ag,g_i}(v,t)$ for each $g_i$ in $sg$ and considers the worst case. Here $\text{CyVar}^S_{ag,g_i}(v,t)$ is defined in terms of $\text{Str}_{ag,g_i}^S(t)$ because we execute a distinct experiment for each $g_i$ in $sg$ where a run ends only when, and if, $ag$ reaches $g_i$. $\text{CyVar}^S_{ag,sg}(v,t)$ is the percentage of runs where $ag$ is successful. Since the worst outcome for the owner is the largest loss probability, we have that

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

A more accurate approximation considers the contribution of each goal 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 $\text{CyVar}^S_{ag,sg}(v,t)$ by considering each impact function $\text{Imp}_{ag,sg}^S(t)$ where $g_i \in \{g_1, ..., g_n\}$. For $\text{Imp}_{ag,sg}^S(t)$, we consider the time $t$, that $ag$ should own the rights in $g_i$ to produce an impact $v$. Then, we compute the probability that $ag$ reaches $g_i$ in $(t - t_o)$ as the percentage of runs where this happens. $\text{CyVar}^S_{ag,sg}(v,t)$ is the sum of all the probabilities.

### 7.3 CyVar: Agents with Alternative Goals

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

We compute $\text{CyVar}^S_{sa,sg}(v,t)$ 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 decomposition $dv$, $pr(dv)$ is the product of the probabilities that the loss of each $ag_j$ is larger than $dv_j$. For each $sa$, this
probability is 1 if \( d_i = 0 \) and it is \( CyVar^S_{sa,sg}(dv_i, t) \) otherwise. If \( S_v \) includes any decomposition of \( v \), then:

\[
CyVar^S_{sa,sg}(v, t) = \sum_{sd \in S_v} pr(sd)
\]

Informally, 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 one element 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 \( \{ v, 100-v \} \) where \( v \) belongs to 0...100. Then,

\[
CyVar^S_{sa,sg}(100, 200) = \sum_{v \in 0..100} CyVar^S_{ag_1,sg}(v, 200) \cdot CyVar^S_{ag_2,sg}(100-v, 200)
\]

### 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 by applying the alternative metrics to quantify the return of an investment that deploys one of them.

#### 8.1 The Three Versions

Any version of the ICS of interest is an ICT network segmented into four types of subnets: Central, Power Context, Process, and Control.

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. The structuring of the ICS into subnets follows the *defence-in-depth* strategy.

\( S_2 \), the second ICS version, doubles the number of nodes by replicating each node without altering the number of connections between subnets.

\( S_3 \), see Fig. 8-2, includes 98 nodes as \( S_1 \) but it is a more accurate implementation of the *defence-in-depth* strategy because it splits the Central subnet into two subnets. Each subnet includes 24 nodes and there is a connection from one subnet to the Power Context subnets.

#### 8.2 Modelling Attackers and Their Impact

Any attacker aims to control the production plan to reduce the efficiency of power production. 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, $\text{Imp}^S_{\text{at},g}(t) = n \cdot \text{Low} \cdot t$ where $n$ is the number of PLCs that at controls for a time $t$. Initially, each attacker owns some rights on a node in the Central subnet.

Haruspex defines the scenario of interest by introducing four classes of agents: $T_1, ..., T_4$. All the agents have one goal and those in the same class have the same goal but adopt distinct selection strategies. Haruspex introduces these classes where agents only differ because of their selection strategies to handle the uncertainty due to the lack of information on the attacker preferences. An assessment analyses the losses due to distinct agents in the same class to discover the most dangerous attacker. Since agents in the same class cover uncertainty about attacker preferences, we decompose a financial loss into the contribution of agents that belong to distinct classes.

Agents in the $T_4$ class aims to control both the PLCs in the ICS. Hence, their goal $g_4$ includes access rights on both the PLCs. Instead, $T_2$ agents have no preference on which PLC to control. $g_3$ 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, $\text{Imp}^S_{\text{at},g}(t)=2 \cdot \text{Low} \cdot t$ if $ag$ belongs to the $T_4$ class, while $\text{Imp}^S_{\text{at},g}(t)=\text{Low} \cdot t$ 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 $T_1$ 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 lower 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, $S_4$ is the most fragile version because of the low number of attacks an agent needs to reach a goal. The number of nodes in $S_3$ confuses the agents and increases both the time to acquire information on the nodes and the one to reach a goal. Finally, $S_1$ 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, $\text{AvLoss}(t)$ is positive after eight hours and twenty minutes and it increases as $t^2/2$ after a bit more than 14 hours. Instead, in $S_5$, 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 $S_1$ this interval ends after 24 hours.

### 8.5 CyVar for the Three Versions

Let us assume that $\text{Low}=10$ 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 $V/(2 \cdot \text{Low})=100/20=5$ hours. This implies that $ag$ reaches $g_2$ in less than 19 hours. In $S_5$, the most dangerous $T_1$ agent always reaches $g_1$ in less than 19 hours. Hence, $\text{CyVar}_{{S_5}_{\text{at}},g_1}(100, 24)=1$ and the owner will suffer this loss in
24 hours anytime a \( T_1 \) agent attacks \( S_1 \). The situation strongly changes in \( S_2 \) where the probability that the loss of a \( T_1 \) agent is 100 belongs to the range [0.2 ... 0.3]. Lastly, \( CyVar_{ag}^{S_1}(100, 24)=0 \) because in \( S_3 \) a \( T_1 \) agent cannot reach \( g_i \) in 19 hours.

Hence, the owner can avoid any loss by changing the structure of the ICS from the one of \( S_1 \) to the one of \( S_3 \). This is cost effective if its cost is lower than 100.

The impact of \( ag \) in the \( T_3 \) 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. \( CyVar_{ag,g_2}^{S_1}(100, 24)=1 \). Instead, this never happens in both \( S_1 \) and \( S_2 \), i.e. \( CyVar_{ag,g_2}^{S_2}(100, 24) = CyVar_{ag,g_2}^{S_3}(100, 24)=0 \). When considering \( T_3 \) and \( T_4 \) agents, the investment to change the structure of the ICS from \( S_2 \) to \( S_1 \) 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 set \( sa \) with two agents that belong, respectively, to \( T_3 \) and \( T_4 \). The agents begin their attacks simultaneously, aim to control a distinct PLC and are independent, because they do exchange privileges or information when implementing an escalation. Hence, the agent have an impact is \( V \) if the sum of the times that they own the rights in, respectively, \( g_3 \) and \( g_4 \) is larger than 10 hours.

At first, we consider a decomposition where each agent reaches its goal in less than 19 hours and it owns these rights for, at least, 5 hours. This always happen in \( S_1 \), i.e. \( CyVar_{sa,g_3}^{S_1}(100, 24)=1 \). In \( S_3 \), 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 \( S_2 \).

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. We can deduce that \( CyVar_{sa,g_3}^{S_2}(100, 24)\approx 0.5 \) 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.

We compute the overall impact and the corresponding \( CyVar \) by considering 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. These two values determine the largest impact of the two agents.

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. \( CyVar \) uses the stress to compute the probability of a loss in an interval. \( AvLoss \) uses the stress to evaluate 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 exist to produce the same data.

We have applied the proposed metrics to evaluate three versions of an ICS in a scenario where attackers have distinct goals. The evaluation applies the metrics to 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.

**Figure 8-1: First Version of the ICS**
Metrics for Cyber Robustness

Figure 8-2: Third Version of the ICS

Figure 8-3: First Version: Stress Curve of the Most Dangerous Agents
Metrics for Cyber Robustness

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

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


of Computer Science and Information Technologies 2(6), 2627–2631 (2011)


