**Fabrizio Baiardi,**
**Federico Tonelli**
*Università di Pisa - Dipartimento di Informatica*

# Metriche per la cyber robustezza

***Metrics for cyber robustness***

*Sommario: La cyber robustezza misura per quanto tempo un sistema ICT può resistere ad attaccanti intelligenti che elevano i loro privilegi per aumentare i loro privilegi fino al raggiungimento dei propri obiettivi. Questo articolo propone tre metriche per valutare questo attributo di un sistema. La metrica base è chiamata security stress e valuta la probabilità che un attaccante raggiunga un obiettivo in un intervallo di tempo fissato a priori. La relazione tra questa probabilità ed il tempo a disposizione dell'attaccante valuta la robustezza globale. Questa metrica è il punto di partenza per definirne altre due che valutano la robustezza cibernetica attraverso l'impatto finanziario generato dagli attacchi ad un sistema ICT.*

*Dopo aver definito le metriche, il lavoro discute come sia possibile approssimare il security stress utilizzando l'output della suite Haruspex. Gli strumenti della suite predicono come un sistema ICT verrà attaccato, prima che l'attacco avvenga effettivamente. Per calcolare queste previsioni, gli strumenti simulano l'interazione tra il sistema e gli attaccanti nello scenario di interesse. L'output della suite permette di calcolare le metriche e restituisce delle contromisure da adottare per produrre una versione del sistema più robusta. Infine, il lavoro esemplifica le metriche definite utilizzandole per confrontare tre versioni di un sistema di controllo industriale.*

*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 an interval of time. The relation between the interval length 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.*

## 1. Introduzione

La cyber robustezza è un attributo di un sistema ICT (Information and Communications Technology) che ne valuta la capacità di resistere ad attaccanti intelligenti e con obiettivi decisi a priori, cioè ad attacchi informatici realizzati da minacce, e.g. attaccanti, intelligenti e persistenti che vogliono controllare alcuni componenti del sistema. I diritti corrispondenti definiscono l'obiettivo degli attaccanti che essi stabiliscono prima di iniziare gli attacchi. Tali attaccanti vengono anche indicati con il termine di *advanced persistent threat (APT)*. Gli attaccanti considerati aumentano i loro privilegi, ad esempio i diritti di accesso, attraverso un'elevazione di privilegi, o *privilege escalation*. Una privilege escalation è una sequenza di attacchi che termina quando l'attaccante acquisisce l'insieme di diritti obiettivo e che gli permettono di controllare alcune risorse di sistema. Le contromisure per questi attaccanti devono interrompere le varie escalation che essi possono eseguire, mentre la sola contromisura possibile contro un attaccante che ha già raggiunto un obiettivo è la revoca di alcuni dei privilegi che esso ha acquisito.

Questo articolo presenta alcune metriche per quantificare la cyber robustezza. Le metriche contribuiscono fortemente a determinare il livello di consapevolezza del proprietario di un sistema ICT sulla sicurezza che il sistema è in grado di offrire. Infatti, esse predicono come un sistema possa resistere ai vari attaccanti. Inoltre, esse permettono di confrontare versioni alternative di un sistema per stabilire come ognuna possa resistere agli attacchi. Questo confronto permette di scegliere gli investimenti per aumentare la robustezza del sistema.

La prima metrica definita è il *security stress*, o più semplicemente *stress*, ad un tempo $t$. Lo stress del sistema a $t$ è la probabilità che un attaccante raggiunga il proprio obiettivo entro $t$. In altri termini, lo stress cui un sistema è soggetto al tempo $t$ aumenta con la probabilità che l'attaccante riesca ad eseguire, entro $t$, una escalation per raggiungere il proprio obiettivo. Lo stress al tempo $t$ è una valutazione sintetica di numerosi attributi di un sistema ICT quali, ad esempio, il numero delle escalation alternative che l'attaccante può implementare, il numero di attacchi in queste escalation e la probabilità di successo di ogni singolo attacco.

La prima metrica è la base della definizione di due metriche finanziarie che misurano la perdita causata da un attaccante che ha raggiunto il proprio obiettivo e che mantiene i diritti corrispondenti per un tempo $t$. Queste metriche sono collegate alla *stress* da una funzione che valuta l'impatto provocato da un attaccante quando questo possiede i diritti in uno dei suoi obiettivi e che mappa in una perdita ogni intervallo di tempo nel quale questo attaccante mantiene tali diritti nell'obiettivo. Le metriche definite implicano che il proprietario del sistema subirà una perdita solo quando, e se, l'attaccante raggiunge un obiettivo. Quindi, assumiamo implicitamente che le perdite causate dagli attacchi in un'escalation siano trascurabili rispetto a quelle che un attaccante può produrre dopo aver raggiunto un obiettivo. Per spiegare quest'assunzione, è opportuno ricordare che anche in un attacco

distruttivo, l'attaccante può produrre una perdita solo dopo aver ottenuto alcuni diritti di accesso.

La prima metrica finanziaria*,* che chiamiamo *AvLoss,* misura la perdita in un tempo *t* come la somma pesata delle perdite dovute ai singoli attaccanti. Il peso di ogni perdita è la probabilità che l'attaccante acquisisca i diritti di accesso in un obiettivo al tempo *t* ed è la derivata di primo ordine della *stress* causata dall'attaccante.

La secondo metrica finanziaria*,* che chiamiamo *CyVar*, estende la nozione di Valore-A-Rischio (VAR) al campo ICT al fine di valutare più accuratamente la perdita causata dagli attaccanti al tempo *t*. VAR è una misura di rischio di un investimento che si focalizza sulla probabilità di perdere denaro. Essa ha tre parametri: un tempo *t*, un livello di confidenza *c* ed una perdita *l* e restituisce la probabilità che un investitore perda più di *l* al tempo *t*. *c* è il livello di confidenza della probabilità restituita. *CyVar* restituisce la stessa informazione in uno scenario dove gli attaccanti elevano i propri privilegi attaccando un sistema ICT. Per calcolare *CyVar* al tempo *t,* partizioniamo l'intervallo *0..t* in due intervalli: quello che l'attaccante impiega per raggiungere l'obiettivo e quello in cui mantiene i diritti su quell'obiettivo. Il danno viene calcolato mappando il secondo intervallo mediante la funzione di impatto. La probabilità che l'attaccante causi questo danno è quella che raggiunga il suo obiettivo nel primo degli intervalli. Si calcola questa probabilità mediante la *stress*.

Un input fondamentale per tutte le metriche che, come quelle definite, vogliano misurare la capacità di un sistema di resistere agli attacchi è la probabilità che un attaccante possa raggiungere un obiettivo. Poiché non esistono delle espressioni di forma chiusa per calcolare, od approssimare, questa probabilità, proponiamo di approssimarla adottando una metodologia Monte Carlo e la suite Haruspex. La suite Haruspex supporta una valutazione e ed una gestione del rischio ICT che utilizza dei modelli formali del sistema ICT da analizzare e degli attaccanti. La suite costruisce un campione statistico per calcolare le statistiche per valutare e gestire il rischio ICT. Il campione viene costruito applicando una metodologia Monte Carlo che utilizza simulazioni multiple ed indipendenti di come gli attaccanti considerati elevano di propri privilegi quando attaccano il sistema ICT di interesse. Il campione permette di predire il comportamento degli attaccanti, di scoprire quali vulnerabilità saranno sfruttate e di classificare le contromisure da implementare. La predizione dei comportamenti di un attaccante è fondamentale anche per calcolare altre probabilità quali ad esempio quella che un componente sia attaccato. Ovviamente, l'approccio che utilizza la suite Haruspex non è l'unico possibile ed altri approcci e/o strumenti possono supportare la valutazione delle metriche proposte.

Descriviamo brevemente la struttura di questo lavoro. La sezione 2 esamina brevemente lo stato dell'arte sulle metriche di interesse. La sezione 3 presenta la metodologia Haruspex e gli strumenti della suite che restituiscono le informazioni che permettono di approssimare le metriche di interesse. La sezione 4 definisce il *security stress* e la sua approssimazione attraverso gli output della suite Haruspex. Vengono

anche discusse alcune definizioni alternative della funzione d'impatto. La sezione 5 analizza brevemente un paradosso sul ritorno dell'investimento in sicurezza. Ognuna delle due sezioni successive definisce una metrica finanziaria. La sezione 8 applica le metriche proposte a tre versioni alternative di un sistema di controllo industriale. Scopo dell'applicazione è quella di valutare le diverse versioni ed il ritorno di un investimento per sostituire una versione con un altra. L'ultimo capitolo presenta le prime conclusioni e discute i possibili sviluppi.

## 2. Related Works

I contenuti di questo lavoro estendono e generalizzano [1-2] che hanno definito, rispettivamente, il *security stress* e *CyVar*. La suite Haruspex generalizza la metodologia della simulazione degli avversari [3]. [4-8] descrivono gli strumenti della suite, la loro applicazione e come questi automatizzino le fasi di gestione e valutazione del rischio ICT. [9-12] discutono la simulazione dell'elevazione di privilegi. [13] esamina la modellazione di attacchi e delle difese nei sistemi critici. [14] analizza attacchi multi-passo, cioè escalation, e ricostruisce i passi di un attaccante utilizzando le tracce ed una ontologia degli attacchi predefinita. [15] presenta un approccio basato su modelli per simulare attacchi e raccogliere informazioni sulla capacità di recupero (*resiliency*).

[16] esamina metriche per la *resiliency* nei sistemi cibernetici. [17-19] definiscono alcune metriche per la robustezza senza però integrarle con le simulazioni di attacchi. La metrica in [20] si focalizza sulle vulnerabilità *zero-day* mentre [21] propone metriche per la difesa, ma analizza gli attacchi in modo isolato senza correlarli con l'elevazione di privilegi dell'attaccante. [22-26] esaminano metriche sulla sicurezza alternative a quelle qui proposte. [27] esamina alcune metriche di sicurezza per lo sviluppo del software. La metrica in [28] è simile al *security stress* poiché considera la quantità di lavoro necessaria per attaccare un sistema. Anche [29] considera il lavoro degli avversari. [30,31] analizzano le relazioni tra metriche ed investimenti sulla sicurezza. [32] analizza l'investimento ottimo per la sicurezza. [33] considera attacchi contro smart-grid. Infine, [34] discute la valutazione ed il miglioramento della *resiliency* nelle infrastrutture critiche.

## 3. La Suite Haruspex

Questa sezione presenta sia la metodologia Haruspex che la suite che ne supporta l'applicazione. Inoltre, discute la validazione della metodologia e della suite.

Haruspex[1] è una metodologia basata su modelli che adotta un metodo predittivo per calcolare le probabilità per valutare e gestire il rischio. Il metodo predittivo basato su modelli è uno dei tre metodi previsto dallo standard IEC 31010 [35]. Haruspex permette di valutare il rischio ICT in uno scenario dove alcuni attaccanti intelligenti, noti a priori, cercano di controllare alcuni componenti del sistema target della valutazione. Gli attaccanti elevano i propri privilegi componendo gli attacchi in sequenze. Haruspex predice il comportamento degli attaccanti definendo i modelli degli attaccanti e del sistema ed eseguendoli in modo interattivo. Questo approccio permette di valutare e gestire il rischio ICT anche durante il progetto di un sistema poiché utilizza solamente i modelli del sistema e degli attaccanti. Questi modelli si possono costruire ancor prima di implementare realmente il sistema.

La suite Haruspex è un insieme integrato di strumenti per supportare la metodologia. Il kernel della suite è composto da 4 strumenti: il *builder*, il *descriptor*, l'*engine* ed il *manager.* I primi due strumenti costruiscono, rispettivamente, il modello del sistema e quello di ogni attaccante nello scenario da analizzare. La costruzione dei modelli richiede la conoscenza di alcuni parametri semplici, come le vulnerabilità in ogni modulo del sistema, gli attacchi che esse abilitano e la probabilità di successo di ogni attacco. Queste informazioni sono disponibili in numerosi database pubblici e privati. Il modello di ogni attaccante è un agente i cui attributi ne descrivono le preferenze e le priorità.

Dopo aver costruito i modelli, l'*engine* è lo strumento della suite che scopre le escalation che ogni attaccante utilizza eseguendo i modelli delle entità in uno scenario e simulando il comportamento di ogni attaccante contro il sistema d'interesse. La simulazione preserva l'intera complessità dello scenario e le interazioni tra il modello del sistema da analizzare e quello degli attaccanti permettono di scoprire le escalation che questi ultimi eseguono per elevare i loro privilegi.

Il *manager* è il principale strumento per gestire il rischio che iterativamente seleziona alcune contromisure ed invoca l'*engine* per valutare come la loro adozione modifica il rischio generato dal sistema da analizzare. Ad ogni iterazione, il manager migliora la propria selezione e lancia una nuova simulazione per valutare il rischio generato dalla sua adozione. Questo comportamento è quello tipico di una forma di gioco estensiva [36]. In ogni iterazione del gioco, il *manager* è il giocatore che seleziona le contromisure più economiche per fermare un insieme di *escalation* scoperte in precedenza. L'adozione delle contromisure produce una nuova versione del sistema. Invece, l'*engine* è un giocatore che implementa le escalation dei vari attaccanti contro la nuova versione del sistema. Nell'iterazione successiva, l'insieme di escalation che il *manager* considera viene esteso con quelle eseguite nell'iterazione precedente. Il gioco termina quando l'*engine* non può più implementare un'escalation o quando il *manager* non può più fermare qualche escalation dell'*engine*. Questo strumento non sarà analizzato nel seguito poiché fuori dai temi trattati in questo articolo.

[1] *Un antico sacerdote etrusco che prediceva il futuro in modo scientifico a partire dalle viscere degli animali*

### 3.1 Modellare un Sistema e gli Attaccanti

Descriviamo nel dettaglio i modelli utilizzati da Haruspex per simulare il sistema d'interesse e gli attaccanti [36]. Haruspex modella il sistema come un insieme di moduli interconnessi ognuno dei quali definisce delle operazioni che gli utenti e gli altri moduli possono invocare solo se, e quando, possiedono i diritti di accesso corrispondenti. Le vulnerabilità dei moduli abilitano gli attacchi che vengono composti in una escalation. Per descrivere gli attacchi di *social engineering,* come il *phishing,* si possono introdurre ulteriori componenti che modellano gli utenti o gli amministratori del sistema.

Haruspex modella un attacco e le sue azioni attraverso alcuni attributi. Due di questi attributi descrivono, rispettivamente, i privilegi per eseguire le azioni e quelli che un attaccante acquisisce ogni volta che l'attacco considerato ha successo. Altri attributi descrivono il tempo per eseguire le azioni dell'attacco e la probabilità di successo. Questa probabilità dipende sia dall'attaccante che dalla complessità dell'azione. Gli attributi degli attacchi sono dedotti classificando le vulnerabilità che li abilitano. La classificazione utilizza la descrizione delle vulnerabilità in database pubblici.

Il modello del sistema d'interesse descrive i moduli, le loro vulnerabilità e gli attacchi che esse abilitano. Haruspex supporta vulnerabilità sia pubbliche che sospette. Gli attaccanti possono scoprire alcune delle vulnerabilità sospette quando stanno attaccando il sistema. Haruspex associa ad ogni vulnerabilità sospetta la probabilità che essa diventi pubblica ad un certo tempo.

L'introduzione delle vulnerabilità sospette permette di eseguire un'analisi *what-if* che scopra come tali vulnerabilità interagiscono con quelle pubbliche per modificare il rischio complessivo. Haruspex assume che nessun attacco singolo possa concedere tutti i diritti in un obiettivo e che quindi l'attaccante debba estendere i propri privilegi attraverso una sequenza di attacchi. Ogni attacco permette di ottenere alcuni diritti di accesso che abilitano l'esecuzione di altri attacchi successivi nell'escalation fino quando l'attaccante non acquisisce tutti i diritti in un obiettivo.

Essendo intelligenti, gli attaccanti selezionano l'escalation da eseguire in base alle proprie preferenze e priorità. Haruspex modella un attaccante attraverso un agente intelligente *ag*. Gli attributi di *ag* descrivono i diritti iniziali dell'attaccante, i suoi obiettivi, e le informazioni che esso ha a disposizione sul sistema attaccato. Un'accurata modellazione di come un attaccante seleziona gli attacchi nella propria escalation influenza fortemente l'accuratezza dell'intera simulazione e delle predizioni calcolate. Gli attributi di *ag* che determinano questa selezione comprendono la strategia di selezione ed il *look-ahead*, $\lambda(ag)$, un intero non negativo. La strategia di selezione definisce le priorità di *ag* ed ordina le diverse escalation che può eseguire. Invece, $\lambda(ag)$ definisce la lunghezza delle sequenze di attacco che *ag* considera quando sceglie quella da eseguire. Nel caso in cui $\lambda(ag)$

sia uguale a *0*, allora *ag* seleziona casualmente un attacco nell'escalation. Altrimenti, *ag* seleziona l'escalation da implementare classificando tutte quelle che comprendono, al più, $\lambda(ag)$ *attacchi*.

Nel seguito, ci riferiremo a queste escalation come sequenze poiché, in generale, esse permettono di ottenere un sottoinsieme proprio dei diritti in un obiettivo. La strategia ordina le sequenze di attacco che *ag* può implementare utilizzando i diritti di accesso che ha a propria disposizione e quindi restituisce la prima delle sequenze. La strategia restituisce sempre una sequenza che porta ad un obiettivo, se questa esiste. Quando l'insieme dei diritti di *ag* è piccolo, *ag* può implementare solo sequenze che non permettono di raggiungere un obiettivo. In questo caso, la strategia di *ag* che ordina le sequenze in base ad alcuni degli attributi degli attacchi in ogni sequenza. Strategie diverse considerano sottoinsiemi diversi. Ogni valutazione del rischio associa ad ogni agente una delle strategie di selezione predefinita. Alcune strategie sono:

- *maxProb:* considera la probabilità di successo degli attacchi;
- *maxIncr:* considera il numero di diritti che la sequenza concede;
- *maxEff:* considera il rapporto tra probabilità di successo e tempo di esecuzione.

Tuttavia, non solo non c'è garanzia che la strategia restituisca una sequenza che permetta di raggiungere un obiettivo, ma qualcuno degli attacchi della sequenza restituita può essere inutile per raggiungere un obiettivo. Un attacco è inutile se l'agente può raggiungere un obiettivo senza utilizzare i diritti che esso concede.

Per scoprire le vulnerabilità negli attacchi delle varie sequenze che può scegliere, un attaccante esegue una scansione di alcuni nodi. Tali scansioni restituiscono le informazioni sulle vulnerabilità che abilitano attacchi nelle sequenze alternative. Poiché ogni scansione richiede un tempo non nullo, valori più grandi di $\lambda(ag)$ aumentano il tempo della selezione poiché *ag* deve costruire e selezionare sequenze di attacco più lunghe. Questo dimostra come $\lambda(ag)$ permetta di risolvere il dilemma *"colleziona o sfrutta" che un attaccante fronteggia quando deve decidere se raccogliere ulteriori informazioni o sfruttare quelle di cui già dispone per selezionare un attacco. Abbiamo sperimentalmente verificato che valori di look-ahead più grandi di 3 forzano la valutazione di sequenze lunghe senza aumentare l'accuratezza della scelta.*

*La continuità, un'altra proprietà, definisce il numero di attacchi che un agente implementa prima di invocare nuovamente la propria strategia di selezione. Agenti con un basso valore di continuità sfruttano* al meglio le vulnerabilità sospette appena scoperte al costo di un maggior numero di selezioni.

### 3.2 L'Engine

Questo strumento utilizza il modello del sistema e quelli degli attaccanti in uno scenario per implementare un esperimento Haruspex.

Un esperimento contiene più *run* tra di loro indipendenti. Ogni *run* simula il comportamento degli agenti e del sistema in uno stesso intervallo di tempo. All'inizio di un *run*, tutti gli agenti hanno a disposizione solamente i diritti legali dell'attaccante che modellano. Un *run* termina o quando tutti gli agenti raggiungono uno dei loro obiettivi o quando termina l'intervallo di tempo. Ad ogni passo, l'*engine* determina le vulnerabilità sospette che ogni agente scopre. A questo punto, viene considerato ogni agente inattivo ed invocata la sua strategia di selezione. L'agente è occupato per il tempo necessario a raccogliere le informazioni sulle vulnerabilità, selezionare una sequenza ed, infine, implementare un numero di attacchi della sequenza selezionata uguale alla sua *continuità*. La sequenza restituita ed i vari attributi dell'agente determinano il tempo che esso impiega ad eseguire gli attacchi nella sequenza selezionata. Se la strategia non restituisce una sequenza, allora l'agente rimane inattivo e sarà riconsiderato solo dopo la scoperta di almeno una vulnerabilità sospetta.

L'*engine* determina l'esito di un attacco in base agli attributi dell'attacco stesso e, se ha successo, concede all'agente i diritti corrispondenti. Un agente ripete un attacco fallito per un numero di volte uguale alla sua *resistenza*, un ulteriore attributo dell'agente. L'agente considera la sequenza successiva nella propria classifica quando il numero di fallimenti dello stesso attacco raggiunge la propria *resistenza*.

Alla fine di un *run,* l'*engine* raccoglie le osservazioni sulle escalation di ogni agente, sui componenti che gli agenti hanno attaccato e su ogni obiettivo che gli agenti hanno raggiunto. Queste osservazioni costruiscono il campione statistico che è l'output dello strumento. Il livello di confidenza di ogni statistica che si può calcolare aumenta in base al numero di *run* dell'esperimento che ha generato il campione poiché l'*engine* colleziona un'osservazione per ogni *run*. L'utente può richiedere che l'*engine* lanci un nuovo *run* fino a quando una statistica predefinita non raggiunge il livello di confidenza stabilito a priori. Tali statistiche possono considerare, ad esempio, i moduli che un agente attacca od il tempo che l'agente impiega per raggiungere un obiettivo.

L'*engine di Haruspex* generalizza l'approccio basato sulla simulazione di avversari poiché esegue il modello di ogni attaccante per predire il suo comportamento nei confronti del sistema di cui interessa valutare la robustezza.

### 3.3 Validazione della Suite

L'utilità e l'accuratezza della suite dipendono completamente dall'abilità dei suoi strumenti di mimare in maniera accurata come gli attaccanti selezionino ed implementino le proprie escalation. Validare gli strumenti in base ad attaccanti reali è raramente possibile poiché i dati sugli attacchi reali non sono facilmente reperibili. Per questa ragione, abbiamo validato la suite in alcuni esercizi di cyber difesa che hanno messo in competizione alcuni team di difensori di varie nazioni con un

team attaccante, il red team. Lo scenario creato nell'esercizio considera una nazione fittizia, la cui industria è bersagliata da un crescente numero di attacchi informatici. L'esercizio si svolge in più giorni ed assume che nel primo giorno siano eseguiti con un insieme di attacchi informatici di basso profilo contro un sistema che i team nazionali devono difendere. L'intensità degli attacchi aumenta fino ad arrivare, nel secondo giorno ad attacchi di spionaggio e sabotaggio. Nel tentativo di rendere l'esercizio più realistico possibile, i difensori devono eseguire attività supplementari come, ad esempio, delle sfide di analisi forensi e legali. L'esercizio è creato come un gioco competitivo, dove i team difensori sono valutati in base alle loro performance.

Uno dei team nazionali ha utilizzato la suite *Haruspex* per analizzare il sistema da difendere, predire il comportamento del red team e, in base a questa predizione, selezionare le vulnerabilità da patchare. Patchare una vulnerabilità prima che il red team inizi i suoi attacchi è l'unica contromisura a disposizione dei difensori. Poiché un team ha a disposizione poco tempo per applicare le patch, il costo di una patch è dato dal tempo per applicarla. In questo caso, il *manager* seleziona l'insieme di patch da applicare minimizzando il tempo necessario per applicarle.

L'unico input che la suite ha avuto a disposizione è stato l'output dello scanning per scoprire le vulnerabilità dei nodi da difendere. Per mancanza di tempo non si sono potuti applicare strumenti per analizzare il codice sorgente dei moduli del sistema. L'adozione di Haruspex ha dovuto affrontare alcuni problemi, poiché l'esercizio assume che alcuni nodi siano sconosciuti, cioè che il team difensore non può analizzarli e non può quindi ottenere informazioni sui moduli software che essi eseguono. Questi nodi possono anche contenere malware in grado di attaccare altri nodi. Questo contraddice gli assiomi di Haruspex che assumono che la valutazione del rischio di un sistema possa accedere liberamente ad ogni informazione disponibile sul sistema stesso. Abbiamo risolto la contraddizione assumendo il caso pessimo in cui il red team controlla questi nodi e che li può usare per lanciare ulteriori attacchi. Le simulazioni hanno considerato agenti che mirano a controllare le parti più critiche del sistema da difendere. Per modellare agenti che sfruttano interamente ogni informazione che acquisiscono, abbiamo assunto $\lambda(ag)=3$ per ogni *ag*. L'incertezza sulla strategia di selezione è stata gestita adottando l'approccio standard di Haruspex per gestire la mancanza d'informazioni su alcuni attributi. Quest'approccio introduce diversi agenti per ogni possibile valore di un attributo, simula i loro attacchi e quindi considera il caso pessimo, cioè l'agente che produce il maggior impatto nel più breve tempo. Durante l'esercizio, gli strumenti hanno considerato tutte le escalation degli agenti ed hanno calcolato la lista ottima di vulnerabilità da patchare nell'intervallo di tempo che precede gli attacchi del red team. I membri del team difensore hanno applicato manualmente le patch.

Il sistema bersaglio era afflitto da più di 1000 vulnerabilità. Il manager ha restituito una lista con circa il *2%* di queste vulnerabilità. Questo conferma l'abilita dell'*engine* di predire il comportamento degli

attaccanti e di scoprire le vulnerabilità critiche, quelle che permettono agli attaccanti di raggiungere i loro obiettivi, anche in presenza di un gran numero di vulnerabilità non critiche. Molti approcci alternativi ad Haruspex si focalizzano sulla scoperta di tutte le possibili escalation prima di selezionare quelle critiche. Ciò forza a gestire l'enorme complessità di scoprire ogni escalation. L'approccio Monte Carlo, adottato dalla suite, minimizza l'intera complessità perché si focalizza sulla scoperta delle sole escalation che gli attaccanti implementano.

Il team che ha applicato la suite Haruspex ha ottenuto eccellenti risultati per quanto riguarda la difesa della propria rete.

## 4. Security Stress ed Impatto

Questa sezione definisce il *security stress*, o più semplicemente *stress*, e l'impatto. La *stress* valuta la cyber robustezza in termini della probabilità che alcuni attaccanti raggiungano un obiettivo in un certo intervallo di tempo. Non solo questa è una metrica autonoma, ma può anche essere usata per definire altre metriche. In particolare, nella prossima sezione la utilizzeremo per definire metriche finanziarie. Invece, l'impatto è una funzione che mappa il tempo in cui un attaccante mantiene i diritti in un obiettivo che ha raggiunto nella corrispettiva perdita per il proprietario del sistema. Nel seguito, denotiamo un attaccante come *at*, l'obiettivo di *at* come *sg* e un obiettivo in *sg* come *g*.

### 4.1 Security Stress

Prima di tutto consideriamo un solo attaccante, vincolo che elimineremo nel seguito. Se $Str^{S}_{at,sg}(t)$ è il *security stress* di un sistema *S* al tempo *t* causato da *at* che mira a raggiungere un qualunque obiettivo in *sg* e se $PrSucc^{S}_{at,sg}(t)$ è la probabilità che *at* abbia successo ed implementi un'escalation per raggiungere un obiettivo in *sg* entro un tempo *t*. Allora,

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

$PrSucc^{S}_{at,sg}(t)$ è la somma di tutte le probabilità di successo delle singole escalation che *at* ha a disposizione per raggiungere un obiettivo in *sg* moltiplicata per la probabilità che *at* la scelga. La probabilità di selezionare le diverse escalation non sono indipendenti poiché esse dipendono dalle priorità e dalle preferenze di *at*. La complessità di calcolare i due fattori che influenzano $PrSucc^{S}_{at,sg}(t)$, cioè la probabilità che una sequenza di attacchi abbia successo e quella che *ag* selezioni ogni sequenza alternativa è la ragione che, come detto in precedenza, impediscono di dedurre un'espressione di forma chiusa per $PrSucc^{S}_{at,sg}(t)$. È da notare che tra le sequenze che *at* può scegliere vi sono anche quelle che non portano ad un obiettivo. Per il momento, ci focalizziamo sulle proprietà del *security stress* e discuteremo nel seguito come calcolarlo. Poiché è una distribuzione di probabilità, $Str^{S}_{at,sg}(t)$ è monotona non decrescente in *t* e $Str^{S}_{at,sg}(0)=0$. $Str^{S}_{at,sg}(t)$ aumenta con *t*

per due ragioni. Prima di tutto, all'aumentare di *t*, *at* può implementare escalation più lunghe, cioè può raggiungere un obiettivo anche mediante sequenze di attacchi più lunghe. In generale, questo aumenta nettamente la *stress*. La seconda ragione è che un valore più grande di *t* permette di tollerare un numero maggiore di attacchi falliti poiché *at* può ripetere un attacco per un numero più grande di volte. Per esemplificare queste relazioni, consideriamo un sistema banale dove *at* può raggiungere un solo obiettivo in *sg* attraverso due escalation con, rispettivamente, tre e quattro attacchi. Ogni attacco impiega due unità di tempo e la propria probabilità di successo è *0.5*. $Str^{S}_{at,sg}(t)$ vale zero se *t* è più piccolo di *6*, mentre $Str^{S}_{at,sg}(t)$ è, al più, *1/8* per ogni $t \in [6..8)$ poiché *at* può implementare una delle due escalation, ammesso che nessun attacco fallisca. La *stress* potrebbe essere più bassa di *1/8* se non è nulla la probabilità che *ag* possa selezionare anche delle sequenze che non portano ad un obiettivo. $Str^{S}_{at,sg}(t)$ è al più *6/16* per ogni $t \in [8..10)$ poiché in questo intervallo *at* può implementare una qualunque delle due escalation, ammesso che tutti gli attacchi abbiano successo o, nel primo caso, che al massimo un attacco fallisca. Le probabilità dei tre eventi sono, rispettivamente, *1/8*, *1/16* e *3/16*. Di nuovo, non ci sono garanzie che *ag* selezioni una delle due escalation.

Per discutere più in dettaglio come alcuni attributi di *S* e di *at* influenzino $Str^{S}_{at,sg}$ consideriamo due tempi:

- $t_0$ è il più breve tempo dove $Str^{S}_{at,sg}(t) > 0$;
- $t_1$ è il più breve tempo dove $Str^{S}_{at,sg}(t) \approx 1$.

$t_0$ è il tempo per implementare la più breve escalation per un obiettivo in *sg* mentre *at* ha sempre successo per tempi più grandi di $t_1$. Non richiediamo che $Str^{S}_{at,sg}(t) = 1$ poiché $Str^{S}_{at,sg}(t)$ potrebbe raggiungere *1* solo asintoticamente. Nell'esempio precedente, $t_0 = 6$, mentre $Str^{S}_{at,sg}(t)$ raggiunge *1* asintoticamente poiché tutte le sequenze includono attacchi con una probabilità di successo nettamente minore di *1*. Assumiamo che $t_0$ e $t_1$ esistano e consideriamo *at* come una forza che tenta di cambiare la forma di *S*. Questa forza è inefficace fino a $t_0$, quando l'escalation di *at* inizia a cambiare la forma di *S*. All'aumentare di *t*, la strategia di selezione di *at* ed il suo *look-ahead* diventano via via meno critici poiché *at* può selezionare ed implementare escalation più lunghe e che possono comprendere anche attacchi inutili. *S* si rompe definitivamente dopo $t_1$ poiché *at* seleziona ed implementa un escalation che porta sempre a *g*. $t_1 - t_0$ valuta quanto a lungo *S* riesce a resistere ad *at*, almeno parzialmente, prima di rompersi.

$t_0$ dipende sia dal tempo di esecuzione degli attacchi sia dalla lunghezza dell'escalation più breve per un obiettivo *g* in *sg*. $t_1$ dipende dalla probabilità di successo degli attacchi nelle escalation che portano a *g*. A sua volta, questa probabilità determina il numero medio di ripetizioni di un attacco prima che abbia successo. $t_1 - t_0$ dipende sia dalla deviazione standard della lunghezza dell'escalation sia dalla probabilità di successo dei loro attacchi. Queste dipendenze da attributi di *S* dimostrano che $Str^{S}_{at,sg}(t)$ può valutare la cyber robustezza in un modo più accurato di metriche che considerano solamente valori medi,

come ad esempio il tempo medio o il numero medio di attacchi in una escalation.

In effetti, queste metriche non possono restituire informazioni sull'intervallo nel quale *S* può resistere ad *at*.

$Sur^{S}_{at,sg}(t) = 1- Str^{S}_{at,sg}(t)$ è l'inversa dello *stress* ed è la funzione di sopravvivenza [37] che determina la probabilità che *S* sopravviva agli attacchi di *at* per raggiungere un obiettivo in *sg*.

Possiamo calcolare la *stress* causata da un insieme di attaccanti *sa* sotto l'assunzione che essi abbiano gli stessi obiettivi in *sg*. Per ogni *t*, la *stress* causata da *sa* è la più grande *stress* dei suoi attaccanti:

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

Se questa *stress* è sempre causata dallo stesso attaccante $at_m$ in *sa*, allora ci riferiremo ad $at_m$ come l'attaccante più pericoloso perché raggiunge sempre un obiettivo prima degli altri.

La *stress* non è definita se gli attaccanti in *sa* hanno obiettivi distinti poiché essi causano un impatto diverso. Comunque, per questo caso, possiamo definire la *stress* come la media pesata di quello dei singoli attaccanti in *sa*. Il peso di un attaccante valuta la sua contribuzione all'intera *stress*. Nel seguito consideriamo solo la *stress* di attaccanti con lo stesso obiettivo.

L'adozione di un metodo Monte Carlo permette di superare i problemi posti dalla mancanza di un'espressione in forma chiusa per $Str^{S}_{at,sg}(t)$ poiché il metodo permette di approssimarlo con la percentuale di *run* in un esperimento Haruspex nel quale l'agente *ag* che modella *at* raggiunge un obiettivo in *sg* prima di *t*. Denotiamo quest'approssimazione sostituendo $Str^{S}_{at,sg}(t)$ con $Str^{S}_{ag,sg}(t)$. L'esperimento simula *ag* per almeno un tempo *t* e raggiunge il livello di confidenza prefissato sul tempo necessario ad *ag* per raggiungere un obiettivo in *sg*. Questo livello è anche quello dell'approssimazione di $Str^{S}_{at,sg}(t)$ attraverso $Str^{S}_{ag,sg}(t)$. Considerazioni simili si applicano al calcolo della *stress* di un insieme di attaccanti. In questo caso ci riferiremo all'agente più pericoloso e non all'attaccante più pericoloso.

$Str^{S}_{at,sg}(n)$ è una definizione alternativa del *security stress* nella quale *n* è il più grande numero di attacchi eseguiti durante un'escalation di *at*. Ovviamente *n* include anche gli attacchi falliti. $Str^{S}_{at,sg}(n)$ collega lo *stress* con il numero di attacchi invece che del tempo per implementarli. Questa definizione evidenzia la quantità di lavoro di *at* anziché sul tempo che *at* ha a propria disposizione.

Comunque, $Str^{S}_{at,sg}(n)$ non considera il lavoro per raccogliere le informazioni su *S* necessarie ad *at* per selezionare l'attacco da implementare. Ovviamente non possiamo dedurre una forma chiusa nemmeno per $Str^{S}_{at,sg}(n)$ ma possiamo approssimare anche questa definizione con la percentuale di *run* in un esperimento nel quale l'agente *ag,* che modella *at*, raggiunge un obiettivo in *sg* eseguendo, al più, *n* attacchi. Indichiamo con $Str^{S}_{ag,g}(n)$ l'utilizzo di questa approssimazione.

### 4.2 Funzione d'impatto

L'informazione che $Str^{S}_{at,g}(n)$ restituisce, ovvero il tempo che *at* impiega per acquisire il controllo di *S* è critica quando *at* può produrre un impatto, e.g. una perdita per il proprietario di *S*, immediatamente dopo aver acquisito tale controllo. Un tipico esempio potrebbe essere un terrorista che cerca di disattivare un'infrastruttura critica per distruggere o sabotare un impianto di produzione industriale o per creare un inquinamento su larga scala. In un contesto commerciale, *at* potrebbe mirare al furto di alcune informazioni, ad esempio il progetto di un componente o alcuni codici sorgenti, o alla riduzione dell'efficienza di un impianto di produzione.

Una valutazione quantitativa impatto è fondamentale per scoprire se il ritorno dell'investimento in sicurezza è della proporzionale alla perdita potenziale.

L'impatto causato da un attaccante è strettamente collegato al tempo in cui questo mantiene i diritti di accesso in un obiettivo che ha raggiunto prima che venga scoperto il suo attacco. Per questa ragione, definiamo la perdita attraverso una funzione d'impatto $Imp^{S}_{at,g}(t)$. $Imp^{S}_{at,g}(t) = l$ implica che la perdita del proprietario di *S* è *l* se *at* mantiene i diritti in *g* per *t*.

$Imp^{S}_{at,g}(t)$ è monotona non decrescente in *t* ed $Imp^{S}_{at,g}(0)=0$ ma la sua forma dipende completamente da *S* e dalle priorità di *at*. Per analizzare queste dipendenze, supponiamo che *at* miri ad acquisire i diritti in *g* per esfiltrare alcune informazioni in *S* e rubare così alcuni dati importanti. Dopo aver raggiunto *g*, *at* ha bisogno di $t_s$ per trasferire le informazioni. Se *at* inizia l'esfiltrazione non appena ha raggiunto *g*, allora $Imp^{S}_{at,g}(t)$ aumenta se *t* è compreso in $0..t_s$ ed è costante per valori più grandi. La derivata di secondo ordine di $Imp^{S}_{at,g}(t)$ è strettamente negativa in $0..t_s$ ogni volta che l'esfiltrazione di ulteriori informazioni contribuzione contribuisce in modo decrescente sull'intera perdita. La stessa derivata aumenta per *t* in $0…t_{s1}$ e decresce per *t* in $t_{s1}…t_s$ se l'esfiltrazione di ulteriori informazioni aumenta inizialmente la perdita ma il contributo di ulteriori informazioni diminuisce.

$t_s$ dipende dall'ammontare di informazioni da esfiltrare ed anche dalla tolleranza al rischio di *at*. Questa tolleranza è influente poiché la probabilità che *S* scopra un'esfiltrazione d'informazioni aumenta con la percentuale della banda di comunicazione di S che l'esfiltrazione sfrutta. Quindi, utilizzando una maggiore percentuale di banda, *at* riduce il tempo di esfiltrazione al costo di un aumento della probabilità che essa sia scoperta.

Le funzioni d'impatto con una derivata di primo ordine positiva e decrescente permettono di modellare anche un attacco che distrugge alcuni dati. In questo caso, *at* deve acquisire i privilegi per aggiornare i dati per sovrascriverli con valori non significativi. La scrittura richiede un tempo $t_s$ che aumenta se si vuole minimizzare la probabilità che essa sia scoperta. Supponiamo adesso che *S* sia un sistema di controllo industriale e che *at* miri a sabotare la produzione o a danneggiare l'impianto di produzione. Stuxnet è un esempio ben conosciuto del

secondo caso [38,39]. La derivata di primo ordine di $Imp^{S}_{at,g}(t)$ è costante e la derivata di secondo ordine è zero se *at* mira a sabotare la produzione. La stessa funzione applica permette di modellare anche il caso in cui *at* cerca a rubare alcune informazioni, a patto che *S* produca continuamente nuove informazioni da rubare. Se *at* mira a danneggiare l'impianto allora $Imp^{S}_{at,g}(t)$ aumenta costantemente fino a raggiungere un valore di soglia $t_d$ ed è costante per tempi più grandi. La derivata del secondo ordine di $Imp^{S}_{at,g}(t)$ aumenta nell'intervallo $0…t_d$ se il costo di ripristino del sistema di controllo è proporzionale al tempo in cui viene controllato da *at*.

Nel seguito usiamo $Imp^{S}_{ag,g}(t)$ invece di $Imp^{S}_{at,g}(t)$ per denotare l'approssimazione attraverso l'output di un esperimento nel quale *ag* modella *at*. Assumiamo anche che $Imp^{S}_{ag,g}(t)$ sia definita per ogni valore di *t* anche se alcuni meccanismi di *S* potrebbero scoprire un'elevazione di privilegi od il suo risultato, come ad esempio l'aggiornamento illegale di un file. Questi meccanismi possono introdurre un limite superiore su *t* che riduce l'intera perdita.

## 5. Stress e ritorno dell'investimento in sicurezza

Discutiamo brevemente l'assunzione, comunemente introdotta per valutare ogni investimento in sicurezza, che garantisce il ritorno positivo di ogni rimozione di una qualunque vulnerabilità in *S*. Possiamo riformulare quest'assunzione dicendo che $Str^{S}_{at,sg}(t)$ decresce al diminuire del numero di vulnerabilità di *S*. Abbiamo verificato sperimentalmente che quest'assunzione non è vera poiché la riduzione del numero di vulnerabilità di *S* può aumentare $Str^{S}_{at,sg}(t)$. Quindi, il patching di una vulnerabilità o l'implementazione di una contromisura può, in realtà, aumentare le probabilità di successo di un attaccante.

Per spiegare perché un numero più basso di vulnerabilità possa aumentare la probabilità di successo di un attaccante, un supponiamo che *at* abbia solo informazioni parziali su *S* e ricordiamo che $Str^{S}_{at,sg}(t)$ dipende dalle sequenze che l'attaccante può eseguire ed alla probabilità di scegliere ed eseguire ogni sequenza. Quindi, la mancanza d'informazioni può far sì che l'attaccante selezioni alcune escalation con una bassa probabilità di successo o con attacchi inutili. Il patching di alcune vulnerabilità di *S* potrebbe fermare alcune di queste escalation e forzare *at* a scegliere altre escalation che *at* credeva peggiori rispetto a quelle bloccate. Il problema è che queste escalation sono in realtà migliori di quelle precedentemente selezionate anche se *at* non ne era a conoscenza.

Per inquadrare il problema si consideri un sistema che comprende un *honeypot* per rallentare l'attaccante. Se si eliminano le vulnerabilità dell'*honeypot*, la robustezza del sistema diminuisce. La mancanza d'informazioni a disposizione dell'attaccante trasforma parte del sistema in un *honeypot* anche se il proprietario non lo ha previsto.

L'aumentare del rischio al diminuire del numero di vulnerabilità è solo un'ulteriore istanza del paradosso di Braess [40]. Il lavoro di Braess dimostra come un aumento del numero di percorsi a disposizione può

aumentare la congestione del traffico. In sicurezza informatica, si ha il fenomeno duale: un numero minore di cammini a disposizione dell'attaccante riduce il tempo per raggiungere un obiettivo poiché aumenta la probabilità che l'attaccante scelga la migliore escalation.

L'aver permesso l'applicazione del paradosso di Braess anche alla sicurezza informatica conferma che $Str^{S}_{at,sg}(t)$ può considerare sia un grande numero di caratteristiche di *S* e di *at* separatamente, come anche le relazioni tra tali caratteristiche.

L'unica soluzione che può evitare il paradosso di Braess è quella che valuta la cyber robustezza di una nuova versione di un sistema prima del suo *deployment* o, meglio, durante il suo progetto. Questo è possibile purché le metriche utilizzate permettano di predire il comportamento degli attaccanti nei confronti della nuova versione.

## 6. Perdita attesa in un intervallo

$AvLoss^{S}_{ag,sg}(t)$ è una metrica che valuta la perdita media del proprietario del sistema *S* al tempo *t* causata dagli agenti in *sa* che mirano a raggiungere gli obiettivi in *sg*. Questa metrica è definita in termini di $Str^{S}_{at,sg}(t)$ e $Imp^{S}_{ag,g,i}(t\text{-}t')$, è monotona non decrescente in *t* e $AvLoss^{S}_{sa,sg}(t)=0$ se $Str^{S}_{sa,sg}(t)=0$.

### 6.1 AvLoss: Un agente con un obiettivo

Consideriamo inizialmente un unico agente *ag* che mira a raggiungere un obiettivo g, se $Str'^{S}_{ag,g}(t)$ è la derivata di primo ordine di $Str^{S}_{ag,g}(t)$ allora

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

$AvLoss^{S}_{at,g}(t)$ è la somma per *t'* nell'intervallo *0..t* della perdita generata dal fatto che *ag* raggiunge *g* in *t'*. Ogni perdita è pesata con la probabilità che *ag* mantenga i diritti nell'obiettivo per *t-t'*. Questa probabilità è quella che *ag* raggiunga *g* a *t'*, che è la derivata di primo ordine di $Str^{S}_{ag,g}(t)$ a *t'*.

Ovviamente, una somma finita rimpiazza l'integrale se $Str^{S}_{ag,sg}(t)$ cambia in maniera discreta in *0..t* poiché la perdita è generata da un numero finito nell'intervallo *0..t*.

Supponiamo ad esempio che $Str^{S}_{ag,\ g}(t)$ aumenti linearmente per valori di *t* in *100…200* mentre *ag* raggiunge sempre un obiettivo in *sg* per periodi più grandi. Poiché $Str^{S}_{ag,g}(200)=1$ allora $Str^{S}_{ag,g}(t)=(t\text{-}100)/(100)$ per $t \in 1…100$. Invece, $Imp^{S}_{ag,g,i}(t\text{-}t')$ cresce come $t^2$ per *t* nell'intervallo *0…100*. Quindi, $t \geq 100$, $AvLoss^{S}_{at,g}(t) = \int^{t}_{100} 1/100\ (t'^2)\ dt'$. Se, invece, $Str^{S}_{ag,g}(t)$ è una funzione a gradini che aumenta di *1/100* ad ogni intero in *100…200*, $AvLoss^{S}_{at,g}(t)$ è una somma con un valore per ogni intero in *100…200* non più grande di *t*.

### 6.2 AvLoss: Scenario più generale

Se *ag* mira a raggiungere un qualunque obiettivo in $sg = \{g_1, ..., g_n\}$, la perdita si può calcolare come media pesata delle perdite $AvLoss^S_{at,g}(t)$ per ogni $g \in sg$. Al solito, il peso della perdita dovuta al raggiungimento di $g_i$ è la probabilità che *ag* raggiunga $g_i$. Le probabilità di raggiungere i vari obiettivi possono essere approssimate con la percentuale di *run* in cui *ag* raggiunga il corrispettivo obiettivo in un esperimento in cui può raggiungere un qualunque obiettivo in *sg*.

La perdita $AvLoss^S_{sag,sg}(t)$ causata da un insieme di agenti $sag = \{a_1, ..., a_f\}$, dove ogni agente ha alcuni obiettivi in *sg* è la somma di $AvLoss^S_{agi,sg}(t)$ per ogni $ag_i \in sag$.

## 7 Valore a Rischio per ICT

$CyVar^S_{ag,g}(v,t)$ è una metrica finanziaria più accurata di *AvLoss* ed estende il *Valore A Rischio* (VAR) al rischio ICT causato da attaccanti con obiettivi predefiniti. Dopo aver definito *CyVar* per un singolo agente con un obiettivo, considereremo il caso di più obiettivi ed infine quello di più agenti con gli stessi e/o diversi obiettivi. Assumiamo che tutti gli agenti inizino la loro escalation simultaneamente e che si richieda per *CyVar* lo stesso livello di confidenza degli esperimenti che hanno generato il campione per approssimare la *stress*.

### 7.1 CyVar: Un agente con un obiettivo

$CyVar^S_{ag,g}(v,t)$ è la probabilità di una perdita più grande di *v* ad un tempo *t* causata da un agente *ag* che mira a raggiungere *g*. Per calcolare $CyVar^S_{ag,g}(v,t)$ calcoliamo inizialmente *t(v)* come il minimo di *Sle*, l'insieme con ogni tempo $t_h$ tale per cui $Imp^S_{ag,g}(t_h) \geq v$. *Sle* include un qualunque tempo $t_h$ limitato da *t* e che provoca una perdita di almeno *v*. Se *Sle* è vuoto, allora $CyVar^S_{ag,g}(v,t)=0$. Altrimenti, *t(v)* esiste e $CyVar^S_{ag,g}(v,t)$ è la probabilità che *ag* mantenga i diritti in *g* per almeno *t(v)*. Questa è la stessa probabilità che *ag* raggiunga *g* in, al più, *t - t(v)*. Quindi,

$Sle = \{ t_h \mid t_h \leq t \text{ e } Imp^S_{ag,g}(t_h) \geq v \}$
$t(v) = \text{se } Sle \neq \phi \text{ allora } \min(Sle) \text{ altrimenti } 0$
e

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

La definizione di $CyVar^S_{ag,g}(v,t)$ non richiede che *ag* raggiunga *g* con una probabilità uguale ad 1, ammesso che ci sia abbastanza tempo disponibile.

Informalmente, per calcolare $CyVar^S_{ag,g}(v,t)$ invertiamo $Imp^S_{ag,g}(t)$ per scoprire $t_o$, il tempo in cui *ag* deve mantenere i diritti in *g* per produrre una perdita *v*. Utilizzando $t_o$, calcoliamo il tempo *t(v)* che *ag* ha a propria disposizione per raggiungere *g*. Ovviamente, ad ogni aumento di $t_o$ corrisponde una diminuzione di *t(v)* e della probabilità che *ag* raggiunga *g* in *t(v)* e quindi anche di $CyVar^S_{ag,g}(v,t)$.

Ad esempio, supponiamo che $Imp^S_{ag,g}(t) = \alpha \cdot t^2$ e che siamo interessati a $CyVar^S_{ag,g}(\beta,\gamma)$, la probabilità di una perdita più grande di $\beta$ al tempo $\gamma$. *ag* può produrre una perdita $\beta$ se mantiene i diritti in *g* per almeno $\sqrt{\{\beta/\alpha\}}$ unità di tempo. Quindi la probabilità di una perdita più grande di $\beta$ è la stessa che *ag* raggiunga *g* in, al più, $t=\gamma-\sqrt{\{\beta/\alpha\}}$. Ogni volta che *t* è positivo, questa probabilità è uguale allo *stress* al tempo $\gamma - \sqrt{\{\beta/\alpha\}}$. Quindi,

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

Il calcolo di $CyVar^S_{ag,g}(v,t)$ assume che *ag* inizi i suoi attacchi al tempo *0*. Se questo accade con probabilità *att(ag)* allora una perdita più grande di *v* ha una probabilità $att(ag) \cdot CyVar^S_{ag,g}(v,t)$. Questa formulazione tiene anche conto del caso in cui non ci siano attacchi. Possiamo anche definire $CyVar^S_{ag,g}(v,t)$ nel caso in cui sia definita una distribuzione della probabilità che *ag* inizi i suoi attacchi a *t*.

### 7.2 CyVar: Un agente con obiettivi alternativi

Se $sg=\{g_1, ..., g_n\}$, possiamo definire $CyVar^S_{ag,sg}(v,t)$ purché conosciamo $Imp^S_{ag,gi}(t)$ per ogni $g_i$ in *sg*. Una prima approssimazione di $CyVar^S_{ag,sg}(v,t)$ calcola $CyVar^S_{ag,gi}(v,t)$ per ogni $g_i$ in *sg* e considera il caso peggiore. In questo caso, $CyVar^S_{ag,gi}(v,t)$ è definito in termini di $Str^S_{ag,gi}(t)$ poiché eseguiamo possiamo eseguire un esperimento distinto per ogni $g_i$ in *sg* in cui un *run* termina solo quando, e se, *ag* raggiunge $g_i$. $CyVar^S_{ag,gi}(v,t)$ è la percentuale di *run* nei quali *ag* ha successo. Poiché il caso peggiore per il proprietario del sistema è la probabilità più elevata di subire la perdita, abbiamo che

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

Un'approssimazione più accurata considera il contributo di ogni obiettivo alla perdita complessiva. Calcoliamo quest'approssimazione attraverso un esperimento in cui ogni *run* termina quando *ag* raggiunge un qualunque obiettivo in $sg=\{g_1, ..., g_n\}$.

Quindi approssimiamo $CyVar^S_{ag,sg}(v,t)$ considerando ogni funzione d'impatto $Imp^S_{ag,gx}(t)$ dove $g_x \in \{g_1, ..., g_n\}$. Per $Imp^S_{ag,gx}(t)$, consideriamo il tempo $t_x$ per quale *ag* dovrebbe mantenere i diritti in $g_x$ per produrre un impatto *v*.

A questo punto possiamo approssimare la probabilità che *ag* raggiunga $g_x$ in $(t - t_x)$ con la percentuale di *run* nel quale questo accade. $CyVar^S_{ag,sg}(v,t)$ è la somma di tutte le probabilità.

### 7.3 CyVar: Agenti con obiettivi alternativi

Definiamo $CyVar^S_{sa,sg}(v,t)$ dove $sa=\{ag_1, ..., ag_k\}$ è un insieme di agenti che condividono gli obiettivi in $sg=\{sg_1, ..., sg_k\}$ sotto l'assunzione che gli agenti non interagiscano o non cooperino tra loro.

Calcoliamo $CyVar^S_{sa,sg}(v,t)$ considerando le decomposizioni alternative di $v$ in una tupla $dv$ con $k$ valori non negativi $\{dv_1, ..., dv_k\}$ dove $\sum_{j=1,k} dv_j=v$. $pr(dv)$ è la probabilità che un qualunque agente in $sa$ causi una perdita almeno uguale a quella corrispondente in $dv$. A causa dell'indipendenza tra gli agenti, per ogni decomposizione di $dv$, $pr(dv)$ è il prodotto delle probabilità che la perdita di ogni $ag_j$ sia più grande di $dv_j$. Per ogni $sa_j$, questa probabilità vale 1 se $d_j = 0$ altrimenti vale $CyVar^S_{saj,sgj}(dv_j,t)$. Se $Sv$ è l'insieme che comprende una qualunque decomposizione di $v$, allora:

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

Informalmente, quest'approssimazione calcola la probabilità che gli agenti in $sa$ causino una perdita più grande di $v$ in tre passi. Il primo decompone $v$ in un insieme di tuple, ognuna con un elemento per ogni agente in $sa$. Per ogni tupla, il secondo passo calcola la probabilità che ogni agente causi almeno la corrispondente perdita. A questo punto può essere necessaria un'ulteriore decomposizione se $sg$ comprende più di un obiettivo. Il terzo ed ultimo passo somma tutte le probabilità delle varia tuple.

Ad esempio, se $sa =\{ag_1, ag_2\}$ calcoliamo $CyVar^S_{sa,sg}(100, 200)$ decomponendo $100$ in $101$ insiemi con la struttura $\{v, 100\text{-}v\}$ dove $v$ appartiene a $0…100$. Quindi,

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

## 8. Un esempio

Questa sezione applica le metriche proposte a tre versioni di un sistema di controllo industriale che supervisiona e controlla un impianto di generazione di corrente. La prima versione è il sistema attualmente in uso. Il proprietario vuole valutare un investimento in sicurezza al fine di selezionare ed implementare una delle altre due versioni alternative. Analizziamo queste versioni applicando le metriche discusse in precedenza per quantificare in modo accurato il ritorno dell'investimento necessario per adottare la versione considerata.

### 8.1 Le tre versioni

Ognuna delle tre versioni del sistema di controllo industriale considerato nel seguito è una rete ICT segmentata in quattro tipi di sottoreti: *Central*; *PowerContext*; *Process*; *Control*.

Gli utenti dell'*intranet* eseguono dei processi per la generazione di corrente attraverso i nodi in una sottorete *Central*. Gli operatori

dell'impianto interagiscono con i server SCADA attraverso i nodi di una sottorete *PowerContext*. I server SCADA ed i sistemi nella sottorete *Process* controllano la generazione della corrente. Infine, il sistema guida l'impianto attraverso alcuni componenti logici programmabili (PLC) in una sottorete *Control*.

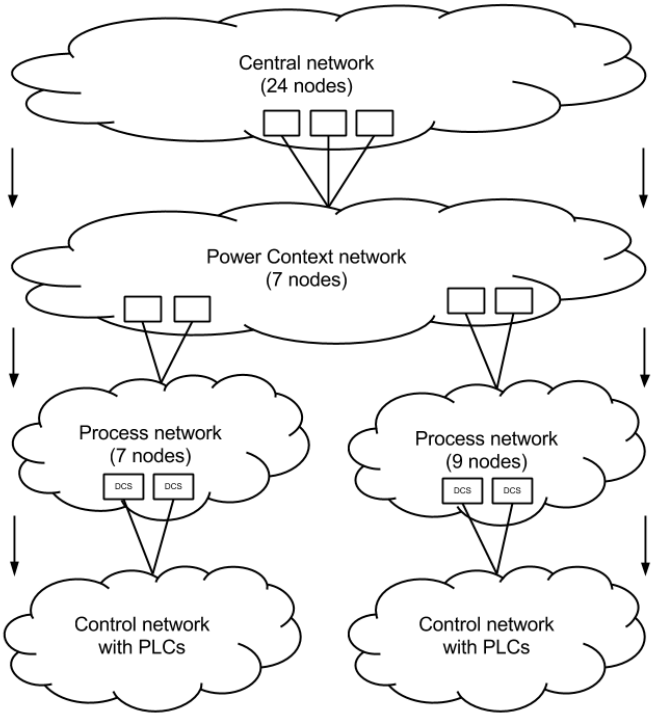



*Figura 8- 1. Prima Versione del Sistema di Controllo Industriale.*

*La* Fig. 8-1 mostra $S_1$, la versione del sistema attualmente in uso [41]. Essa include 49 nodi segmentati in 6 sottoreti. La struttura del sistema è definita in modo da soddisfare il principio della *defence-in-depth*. La sottorete *Central* include 24 nodi, la sottorete *PowerContext* include 7 nodi. Le sottoreti *Process* 1 e 2 includono, rispettivamente, 9 e 7 nodi. Un diverso PLC collega ogni sottorete *Process* ad una sottorete *Control*. Tre nodi connettono la sottorete *Central* con la sottorete *PowerContext*. Due coppie di nodi nella sottorete *PowerContext* sono connessi a quelli nella sottorete *Process*. Infine, c'è una connessione fra due nodi in ogni sottorete *Process* e la corrispondente sottorete *Control*.

$S_2$, la seconda versione del sistema, raddoppia il numero dei nodi di $S_1$ replicando ogni nodo di ogni sottorete senza alterare il numero di connessioni tra le sottoreti.

$S_3$, si veda la Fig. 8-2, è formata da 98 nodi, come $S_2$, ma soddisfa in modo più rigoroso l'approccio della *defence-in-depth* poiché partiziona la sottorete *Central* in due sottoreti. Ogni sottorete include 24 nodi ed esiste una connessione fra una sottorete e la sottorete *PowerContext*.

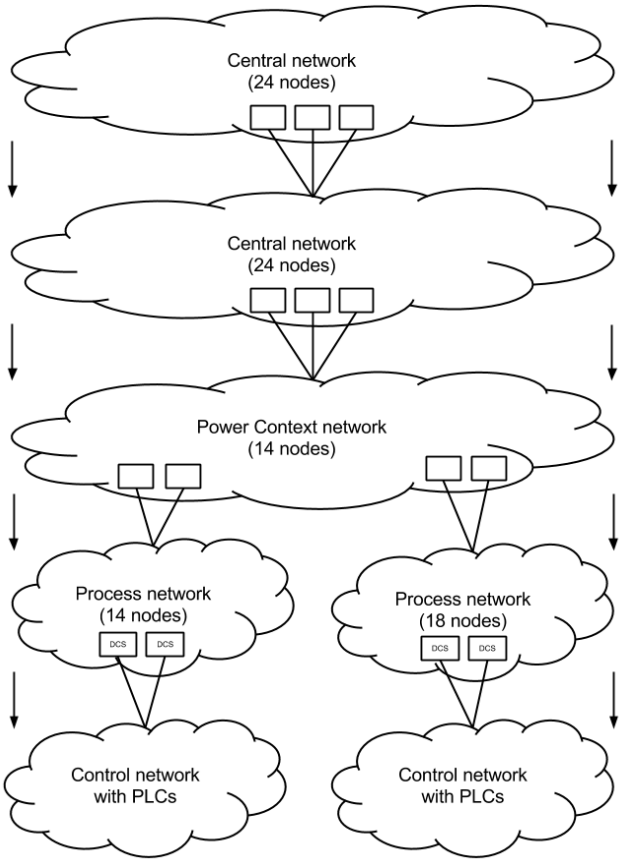



*Figura 8- 2. Terza Versione del Sistema di Controllo Industriale.*

### 8.2 Modellare gli attaccanti ed il loro impatto

Un qualunque attaccante cerca di controllare l'impianto per ridurre l'efficienza della produzione di corrente. Sotto queste assunzioni, se un attaccante controlla un PLC per un tempo *t*, la perdita per il proprietario sarà $Low \cdot t$. Inoltre, la perdita aumenta con il numero di PLC che l'attaccante controlla. Quindi, $Imp^{S}_{at,g}(t) = n \cdot Low \cdot t$ dove *n* è il numero di PLC che *at* controlla per un tempo *t*. Inizialmente, ogni attaccante alcuni possiede unicamente alcuni diritti su un nodo nella sottorete *Central*.

Uno scenario Haruspex comprende quattro classi di agenti: $T_1$, ..., $T_4$. Tutti gli agenti nella stessa classe hanno lo stesso obiettivo ma adottano strategie di selezione diverse. Haruspex introduce queste classi, i cui agenti differiscono solamente nella strategia di selezione, per gestire l'incertezza sulle preferenze dell'attaccante. Nel seguito, una perdita finanziaria verrà decomposta nelle varie contribuzioni di agenti che appartengono a classi distinte. Questo vincolo tiene conto del fatto che i vari agenti di una stessa classe sono stati introdotti solo per scoprire la peggiore combinazione di attributi di un singolo agente.

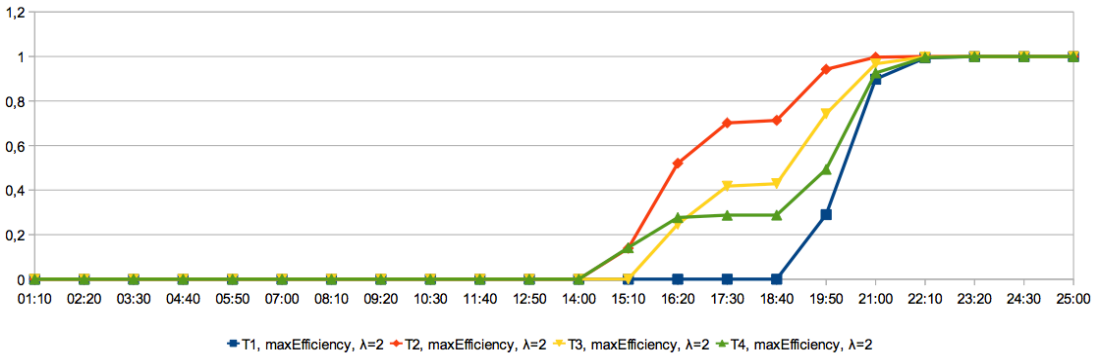
Gli agenti in $T_1$ cercano di controllare entrambi i PLC nel sistema. Quindi, il loro obiettivo $g_1$ include i diritti di accesso su entrambi i PLC. Invece, gli agenti della classe $T_1$ non hanno preferenze su quale PLC controllare. $g_3$ è l'obiettivo degli agenti nella classe $T_3$ mentre $g_4$ è

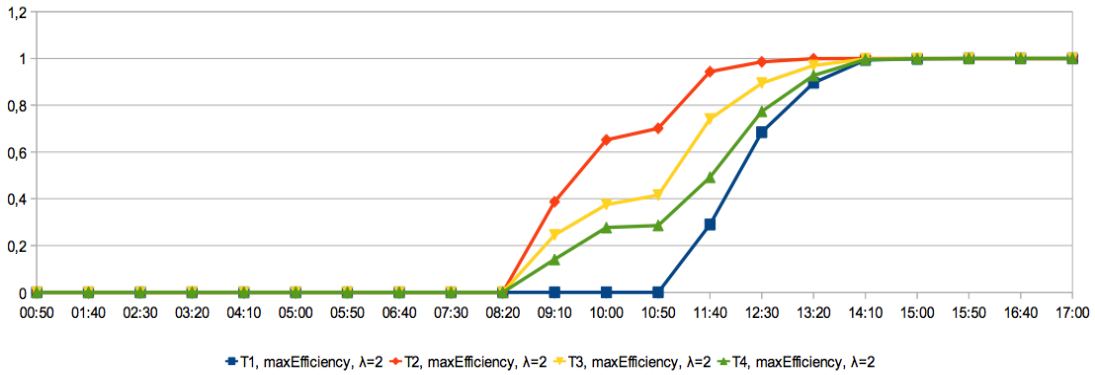
quello degli agenti nella classe $T_4$. Ognuno degli ultimi due obiettivi comprende solo i diritti di accesso su un singolo PLC. Come conseguenza, $Imp^{S}_{at,g}(t)=2 \cdot Low \cdot t$ se $ag$ appartiene alla classe $T_1$, mentre $Imp^{S}_{ag,g}(t)=Low \cdot t$ se $ag$ appartiene ad una qualunque altra classe.

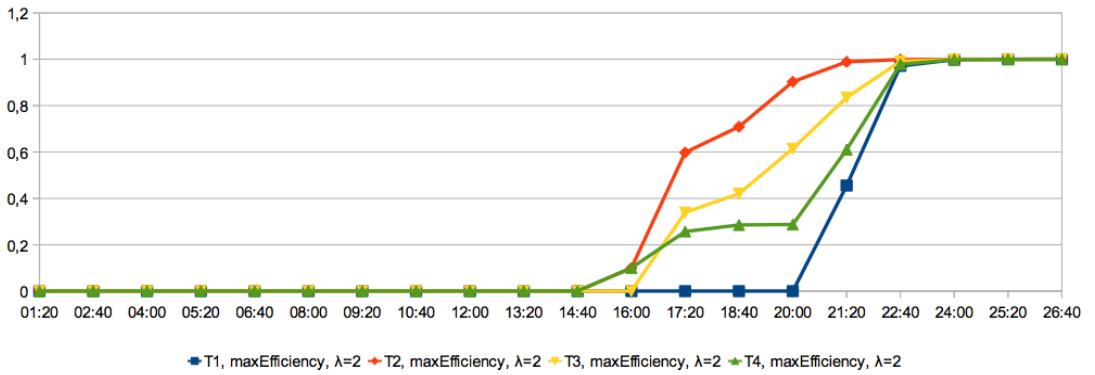
### 8.3 Stress di ogni versione

Nel seguito, utilizziamo le ore come unità di tempo. Le Fig. 8-3, 8-4 ed 8-5 mostrano le curve di *stress* degli agenti più pericolosi per ogni classe per le diverse versioni del sistema di controllo industriale. Il livello di confidenza di queste curve è del *95%*.

*Figura 8- 3 Prima Versione: Curve di Stress degli agenti più pericolosi.*
*Figura 8- 4 Seconda Versione: Curve di Stress degli agenti più pericolosi.*
*Figura 8- 5 Terza Versione: Curve di Stress degli agenti più pericolosi*



Le curve evidenziano che in $S_1$ l'agente più pericoloso della classe $T_2$ raggiunge $g_2$ in circa *12* ore. Un qualunque agente di ogni altra classe raggiunge l'obiettivo in circa *14* ore, cioè circa due ore dopo. L'agente più pericoloso per $S_2$ è un agente della classe $T_2$ che raggiunge $g_2$ in circa *21* ore. Gli altri agenti impiegano un'ora in più. Entrambi gli agenti delle classi $T_3$ e $T_4$ impiegano più tempo di un agente della classe $T_2$ che può scegliere liberamente quale PLC controllare.

Ogni agente impiega più tempo per raggiungere il proprio obiettivo in $S_3$ che in $S_2$. Gli agenti rimanenti raggiungono il loro obiettivo più di due ore dopo.

Come previsto, $S_1$ è la versione più fragile a causa del basso numero di attacchi che un agente deve implementare per raggiungere un obiettivo. Il numero di nodi in $S_2$ confonde gli agenti ed aumenta sia il tempo per acquisire informazioni sui tali nodi che su quello per raggiungere l'obiettivo. Infine, $S_3$ è la versione più cyber robusta poiché il numero dei suoi nodi e delle sue sottoreti aumenta sia il numero di attacchi che l'agente deve eseguire sia il tempo per raggiungere l'obiettivo. Ciò riduce la *stress* di un attaccante.

### 8.4 AvLoss di ogni versione

Per ogni versione consideriamo la perdita media dovuta all'agente più pericoloso. Per semplificare l'analisi utilizziamo un'interpolazione lineare della funzione di *stress*. Per tutti i sistemi, la perdita media aumenta come $t^2$. In un primo intervallo, il coefficiente è la metà della pendenza della retta che interpola la funzione di *stress*. Quindi, il coefficiente è ½. La differenza critica è nella posizione relativa del primo intervallo. $AvLoss^{S1}{}_{at,g}(t)$ è positiva dopo *8* ore e *20* minuti ed aumenta come $t^2/2$ dopo poco più di *14* ore. Invece, in $S_2$, l'intervallo corrispondente inizia dopo *14* ore e termina dopo poco più di *22* ore. Quindi, in questa versione la perdita inizia quando nell'altra c'è il picco massimo. Infine, in $S_3$, questo intervallo termina dopo *24* ore.

### 8.5 CyVar di ogni versione

Assumiamo che *Low=10* e che interessi valutare e gestire il rischio causato dagli agenti nelle classi $T_1$ ,...,$T_4$ se *V=100* e *t=24* ore. L'impatto di *ag* nella classe $T_1$ è *V* se esso mantiene i diritti in $g_1$ per $V/(2 \cdot Low)=100/20=5$ ore. Questo implica che *ag* deve raggiungere $g_1$ in meno di *19* ore. In $S_1$, l'agente più pericoloso della classe $T_1$ raggiunge sempre $g_1$ in meno di *19* ore. Quindi, $CyVar^{S1}{}_{ag,g1}(100, 24)=1$ e il proprietario del sistema soffrirà tale perdita in *24* ore ogni volta che un agente della classe $T_1$ attaccherà $S_1$. La situazione cambia radicalmente in $S_2$, dove un agente della classe $T_1$ causa una perdita pari a *100* con una probabilità che appartiene all'intervallo *[0.2…0.3]*. Infine, $CyVar^{S3}{}_{ag,s}(100, 24)=0$ poiché in $S_3$ un agente della classe $T_1$ non può raggiungere $g_1$ in *19* ore.

Quindi, il proprietario del sistema può evitare ogni perdita cambiando la struttura del sistema di controllo industriale da quella di $S_1$ a quella di

$S_3$. Questa modifica è conveniente da un punto di vista economico se il suo costo è inferiore a *100*.

L'impatto di *ag*, appartenente alla classe $T_2$, è *V* se esso mantiene i diritti in $g_2$ per *10* ore. Quindi, *ag* dovrebbe raggiungere $g_2$ in meno di *14* ore. Questo accade sempre in $S_1$, cioè $CyVar^{S1}_{ag,g}(100, 24)=1$. Invece, questo non accade mai né in $S_2$ né in $S_3$, cioè $CyVar^{S2}_{ag,g2}(100, 24) = CyVar^{S3}_{ag,g2}(100, 24) = 0$. Quando si considerano gli agenti delle classi $T_3$ e $T_4$, l'investimento per cambiare la struttura del sistema da $S_2$ a $S_3$ non ha ritorno perché questi agenti non raggiungono il loro obiettivo in un intervallo di tempo interessante quando attaccano $S_2$. Invece, il ritorno dell'investimento per cambiare la struttura del sistema di controllo da $S_1$ a $S_2$ è positivo poiché il cambiamento riduce realmente l'impatto generato dagli agenti.

Consideriamo adesso un insieme *sa* con i due agenti che appartengono, rispettivamente, alle classi $T_3$ e $T_4$. Gli agenti iniziano ad attaccare simultaneamente, cercano di controllare un PLC diverso e sono indipendenti poiché non si scambiano diritti od informazioni quando implementano una escalation. Quindi, essi generano un impatto uguale a *V* se la somma dei tempi, nei quali mantengono i diritti rispettivamente in $g_3$ e $g_4$, è più grande di *10* ore.

Consideriamo inizialmente una decomposizione dove ogni agente raggiunge il proprio obiettivo in meno di *19* ore e mantiene questi diritti per almeno *5* ore. Questo accade sempre in $S_1$, cioè $CyVar^{S1}_{sa,sg}(100, 24)=1$. In $S_2$, un agente raggiunge $g_3$ in meno di *19* ore con una probabilità uguale a *0.6* mentre la probabilità che l'altro agente raggiunga $g_4$ nello stesso tempo è uguale a *0.5*. Quindi, la probabilità congiunta dei due eventi è uguale a *0.3*. Queste probabilità cambiano solo leggermente in $S_3$.

In una scomposizione alternativa, un agente mantiene i diritti per *6* ore e l'altro per *4* ore. Quindi, il primo dovrebbe raggiungere il proprio obiettivo in meno di *18* ore mentre il secondo ha, al più, *20* ore disponibili. In $S_2$, la probabilità dei due eventi sono rispettivamente, *0.9* e *0.3* e la probabilità congiunta è *0.27*. Possiamo dedurre che $CyVar^{S2}_{sa,sg}(100, 24)>0.5$ poiché abbiamo considerato solo due delle possibili scomposizioni.

In $S_3$, la probabilità di due eventi sono, rispettivamente, *0.3* e *0.6* e quella congiunta è meno di *0.2*.

Calcoliamo l'impatto complessivo e il corrispondente *CyVar* considerando quanto ogni agente contribuisce e le corrispettive probabilità. Quest'analisi per $S_2$ mostra che l'agente nella classe $T_3$ può mantenere i diritti nel proprio obiettivo per, al più, *10* ore dell'intervallo di *24* ore. Invece, l'altro agente può mantenere i diritti nel proprio obiettivo per poco meno di *9* ore. Questi due valori determinano il più grande impatto che i due agenti possono causare.

Determiniamo l'impatto minimo considerando che entrambi gli agenti mantengano i diritti nei corrispettivi obiettivi per almeno *2* ore. Quindi, l'impatto degli agenti in *sa* varia da *100(10+9)* a *100(2+2)*. Il corrispondente impatto per $S_3$ è simile e questo conferma ulteriormente il basso ritorno d'investimento del cambiamento di $S_2$ in $S_3$.

## 9. Conclusione

La cyber robustezza misura la capacità di un sistema ICT di resistere ad attaccanti intelligenti che elevano i propri privilegi attraverso sequenze di attacco. Abbiamo proposto metriche alternative per valutare quest'attributo del sistema. Il *security stress* considera la probabilità che attaccanti raggiungano i loro obiettivi in un intervallo di tempo. *CyVar* utilizza la *stress* per calcolare la probabilità di una determinata perdita in un intervallo. *AvLoss* utilizza la *stress* per valutare la perdita media in un intervallo. Possiamo valutare le metriche proposte in questo lavoro attraverso i campioni che la suite Haruspex produce simulando il comportamento degli attaccanti. Ovviamente, strumenti ed approcci alternativi possono produrre gli stessi dati.

Abbiamo applicato le metriche proposte per valutare tre versioni di un sistema di controllo industriale in uno scenario dove alcuni attaccanti avevano obiettivi distinti. La valutazione applica le metriche per misurare il ritorno di un investimento per cambiare la prima versione con un'altra.

Gli sviluppi futuri di questo lavoro considerano la definizione di metriche dipendenti dalle contromisure da implementare per fermare un attaccante. Queste metriche misureranno il lavoro per impedire che gli attaccanti raggiungano i loro obiettivi e non quello degli attaccanti per elevare i propri privilegi.

### *Bibliografia*

[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: 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 Nether- lands (2013)

[9] Kotenko, I., Konovalov, A., Shorov, A.: Agent-based modeling and simulation of botnets and botnet defense. In: Conference 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 Conference (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). doi:10.1109/MILCOM.2012.6415565

[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, 2003. 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:Anetworksecurity 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). doi:10.1109/MILCOM.2010.5680489

[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)