Appunti Tesi

1 Collegamenti tra SSG e CRA per IACS

CRA for IACS = cybersecurity risk assessment for industrial automation control systems
SUC = system under consideration
Considerando un Stackelberg Security Game (SSG) a somma zero, è possibile derivare input e output del gioco:

- **Input**:
  - insieme di asset condivisi (visibili sia dal difensore che dall'attaccante);
  - insieme delle risorse limitate per il difensore, che corrispondono ad una restrizione sulle proprie possibilità di difesa dei target, ovvero al numero massimo di assets che il difensore riesce a proteggere contemporaneamente;
  - insieme dei valori di payoff, che definiscono i guadagni/perdite sia per il difensore che per l'attaccante in caso di un attacco fallimentare o meno ad un determinato target, e tenendo in considerazione anche se un determinato asset è protetto o meno (il payoff di un determinato target definisce quanto esso è rilevante per il relativo agente razionale);
  - strategia mista iniziale per il difensore, che definisce la probabilità di ogni target di essere protetto dal giocatore.
- **Output**: l'output del SSG è rappresentato da una condizione di equilibrio, detto Stackelberg Equilibrium (SE), rappresentata da una coppia di strategie (una per il difensore e una per l'attaccante) tale per cui: il difensore e l'attaccante applicano la propria strategia ottimale. Generalmente si considera lo Strong Stackelberg Equilibrium (SSE), nel quale si aggiunge la condizione per cui in caso di più strategie ottimali per l'attaccante, quest'ultimo applica la strategia ottimale che massimizzi il payoff del difensore.

Un SSG in sé si potrebbe considerare come costituito dalle funzioni di payoff/utilità, ovvero dalle funzioni che "combinano" le varie componenti del gioco per arrivare ad ottenere un SE.
Negli input del SSG possiamo considerare, o meno, la strategia iniziale del difensore; se non la si considera, allora la possiamo determinare randomicamente. Inoltre, non si considera negli input di un SSG la strategia dell'attaccante, in quanto agendo per secondo può essere contraddistinto da un tipo di strategia arbitraria a seconda dei casi che si vogliono considerare (es. strategia greedy). Nel modello classico di un SSG, l'attaccante può selezionare un unico bersaglio, quindi le risorse dell'attaccante sono implicitamente limitate. In caso di modelli più complessi, è possibile ipotizzare un attaccante con risorse multiple (contemporaneamente può attaccare più assets), ma comunque limitate. Un'assunzione da fare in un SSG è che il payoff di una generica azione dipende solo dall'asset attaccato, e dal fatto che questo sia difeso o meno dal difensore. Inoltre, il difensore

avrà come obiettivo quello di minimizzare il danno massimo che può subire, sapendo che l'attaccante risponderà in modo ottimale. È attraverso i valori di payoff che vengono rappresentati costi e benefici delle strategie di ognuno delle 2 tipologie di agenti. L'ipotesi del gioco a somma zero rende necessari solo i payoff di uno dei 2 agenti, in quanto quelli dell'altro sarebbero ricavabili di conseguenza, andando così a ridurre gli input del gioco.

Da qui si possono ricercare delle **similitudini** tra la struttura di un SSG e il processo di CRA per IACS.

Nella fase **ZCR 5.1** vengono identificate le minacce per il sistema e queste ultime in un SSG possono essere comparate alle varie tipologie di attaccanti che possono essere interessati nel bersagliamento degli asset condivisi. Il tipo di un attaccante in un modello di gioco, infatti, può essere considerato un input implicito a seconda del quale possono essere definiti i vari comportamenti di attacco.

Inizialmente era stato pensato di identificare l'impatto, ovvero l'elemento identificato nella sezione ZCR 5.3 del processo di CRA, con il valore di payoff del difensore. Questo ragionamento era stato fatto in quanto il concetto di "impatto" in un SSG potrebbe essere tradotto in "quanto valore ha quel determinato asset per il difensore", e questa immagine viene proprio rappresentata nel valore di payoff, legato al target, del difensore. Però, in seguito al dialogo con un'esperta di Resiltech, si è arrivati alla conclusione che questa associazione sarebbe andata in contrasto con la letteratura relativa all'impatto nel processo di CRA.

La fase **ZCR 5.4** del processo di CRA per IACS ha come obiettivo quello di identificare la "likelihood" di un attacco ad ogni asset del SUC. In un SSG, tale fattore, rappresentato dalla metrica di Attack Feasibility Rating (**AFR**), corrisponde alla probabilità che l'attaccante decida di intraprendere un atto offensivo nei confronti di uno specifico target. Quindi la likelihood associata ad ogni asset andrebbe a corrispondere ad un elemento della strategia mista dell'attaccante.

L'obiettivo principale della fase **ZCR 5.5** è quello di determinare il **"unmitigated cybersecurity risk"** per ogni minaccia, e quindi assegnare un "risk score" ad ogni asset tramite la combinazione dell'AFR e l'impatto associati ad un target. Poiché l'impatto in sé non è associabile con il payoff del difensore in un SSG, allora è possibile associare quest'ultimo elemento all'unmitigated risk. Inoltre, una volta arrivati a uno SSE, la strategia ottimale del difensore corrisponde ad un vettore in cui ogni elemento costituisce la probabilità che il difensore decida di proteggere un determinato target. Ognuno degli elementi di questo vettore può essere considerato come il SL-T per ogni asset considerato (fase ZCR 5.6)

Un elemento che non è presente nel modello classico degli SSGs, ma che bensì viene considerato nel CRA per IACS, è la presenza di **path tra asset** che l'attaccante può sfruttare per il raggiungimento del proprio obiettivo finale. Tra i target interdipendenti si suppone che, per una minaccia, gli asset più facilmente raggiungibili siano quelli che si trovano all'inizio di un path; questi però rappresentano anche quelli di minor valore per l'attaccante. Contrariamente a ciò, gli asset che si trovano verso la fine del percorso di attacco sono più difficili da raggiungere ma costituiscono anche il vero obiettivo dell'attaccante. Viste queste considerazioni, è possibile una rappresentazione dell'interdipendenza tra gli asset, che in un SSG sono effettivamente indipendenti tra loro, tramite l'assegnazione di specifici valori di payoff all'attaccante. In questo senso, si avreb-

bero valori di payoff alti per l'attaccante in corrispondenza di asset significativi per quest'ultimo, e che quindi si troverebbero in coda al path che l'attaccante percorrerebbe in un corrispettivo sistema; mentre invece si assegnerebbero valori di payoff più contenuti a target che invece non sono considerati dall'attaccante come possibile obiettivo finale, e che quindi si troverebbero in cima a degli ipotetici path di attacco. A tale proposito, sarebbe possibile pensare a una numerazione degli asset, tramite indice, che rappresenterebbe la disposizione dei target in un path del sistema e strutturare i valori di payoff del difensore in relazione a tali indici.

Per quanto riguarda i valori di payoff dei 2 agenti di un SSG, dovranno essere fatte delle assunzioni, considerando che il SSG preso in considerazione è non a somma zero. In caso di attacco riuscito (obiettivo non difeso)

- Attaccante riceve un payoff positivo (successo dell'attacco):

$$U_A^u(t_i) > 0$$

  Tipicamente proporzionale al valore dell'obiettivo (es: danno causato, guadagno ottenuto).

- Difensore subisce una perdita (fallimento difensivo):

$$U_D^u(t_i) < 0$$

  Rappresenta il costo/danno subito per l'attacco riuscito.

In caso di attacco fallito (obiettivo difeso)

- Attaccante riceve un payoff negativo o nullo:

$$U_A^c(t_i) <= 0$$

  Può rappresentare perdita di risorse, rischio di cattura, o semplicemente "nessun guadagno".

- Difensore ottiene un payoff positivo o nullo:

$$U_D^c(t_i) >= 0$$

  Rappresenta il successo della protezione. Può essere zero (se difesa non produce valore diretto) o positivo (prevenzione di danni).

2 Path negli SSG

Questo lavoro, presentato in *Decision Support Systems 148 (2021) 113599* [1], propone un **sistema di supporto decisionale per la cyber-sicurezza** [2, 3]. L'obiettivo principale è assistere le organizzazioni nella selezione di un **portafoglio ottimale di controlli di sicurezza** per contrastare attacchi a fasi multiple (multi-stage attacks) [2, 3]. Gli autori sono Yunxiao Zhang e Pasquale Malacaria [1].

Il sistema è strutturato in diversi componenti interconnessi [2-4]:

- Un'**ottimizzazione preventiva**: Serve per selezionare un portafoglio difensivo iniziale prima che gli attacchi si verifichino. Mira a minimizzare il rischio di sicurezza potenziale [2, 3, 5, 6].

- Un **meccanismo di apprendimento**: Utilizza approcci consolidati come gli Hidden Markov Models (HMMs) [4], basati su lavori precedenti [5, 7-9], per rilevare e stimare possibili attacchi in corso [2-4, 6]. L'inferenza basata su HMM restituisce un vettore di credenza (belief vector) sullo stato dell'attaccante [4, 10].

- Un'**ottimizzazione online**: Questa componente seleziona un portafoglio ottimale per contrastare gli attacchi in corso rilevati dal meccanismo di apprendimento [2, 3]. Interviene quando viene rilevato un attacco e il difensore ha informazioni incomplete sullo stato esatto dell'attaccante [8, 10].

Per modellare attacchi a fasi multiple realistici, il sistema si avvale di un **grafo di attacco probabilistico** [7, 11].

- I nodi nel grafo rappresentano gli **"stati di privilegio"** dell'attaccante [7, 11, 12].

- Gli archi (edges) rappresentano le **vulnerabilità** che l'attaccante può sfruttare per passare tra gli stati [7, 11].

- Ogni arco è associato alla **probabilità di successo dello sfruttamento**, influenzata da una probabilità di base e dall'efficacia dei controlli applicati [13, 14].

- Il **rischio di sicurezza** è definito come la massima probabilità di successo che un attaccante raggiunga l'obiettivo percorrendo uno qualsiasi dei cammini possibili nel grafo [7, 12, 15].

L'ottimizzazione preventiva, relativa all'investimento iniziale in sicurezza, è formulata come un **gioco di Stackelberg standard** [5, 6]. In questo contesto, il difensore agisce come leader, scegliendo un portafoglio di sicurezza preventiva per minimizzare il rischio potenziale, mentre l'attaccante è il follower che mira a massimizzare il rischio selezionando il percorso più critico [5, 6]. Il problema viene risolto efficientemente convertendo il problema originale non lineare in un problema di programmazione lineare (LP) grazie alle proprietà delle matrici totalmente unimodulari e alla dualità forte [1, 16, 17].

L'ottimizzazione online, che gestisce la difesa contro gli attacchi in corso con informazioni incomplete sullo stato dell'attaccante (rappresentato da un belief vector) [8, 10], è formulata come un **gioco di Stackelberg Bayesiano** [8, 10, 18, 19]. L'obiettivo è minimizzare il rischio di sicurezza atteso, calcolato come l'aspettativa sui rischi per ciascun tipo di attaccante (la cui probabilità è data dal belief vector) [10, 20].

Un contributo tecnico chiave del lavoro è un **nuovo algoritmo efficiente per risolvere le ottimizzazioni online (giochi di Stackelberg Bayesiani) su grafi di attacco probabilistici** [10, 21]. I metodi classici come la trasformazione di Harsanyi [8, 18, 19, 22] sono inefficaci per grandi spazi di attacco a causa della potenziale matrice di payoff esponenzialmente grande [23, 24]. Anche approcci più recenti come DOBSS [4, 18, 25], HBGS [6, 18], e HUNTER [10, 18] non scalano sufficientemente bene per grafi di attacco di grandi dimensioni nel dominio della cyber-sicurezza [18, 26].

Il nuovo approccio sfrutta le proprietà delle **matrici totalmente unimodulari** e della **dualità forte** per convertire l'ottimizzazione online in un problema di **Programmazione Conica a**

Numeri Interi Misti (MICP) con coni esponenziali [2, 18, 21, 27]. Questo MICP può essere risolto efficientemente utilizzando risolutori esistenti (come MOSEK versione 9.2) [21, 28].

Gli autori dimostrano che il loro approccio per i giochi di Stackelberg Bayesiani su grafi di attacco è **molto efficiente**, richiedendo un numero significativamente inferiore di variabili di ottimizzazione rispetto a DOBSS e alla trasformazione di Harsanyi [25, 29, 30]. Le valutazioni numeriche mostrano che l'ottimizzazione online è efficiente anche per scenari realistici con un gran numero di nodi e tipi di attaccante [26, 28, 31], e fornisce **miglioramenti significativi nella mitigazione degli attacchi in corso** rispetto agli approcci precedenti [2, 32, 33].

Il documento include un **caso di studio** basato su uno scenario di rete universitaria per illustrare l'applicazione pratica del sistema [34, 35].

In sintesi, il lavoro offre un sistema completo per la cyber-difesa, con un particolare focus su un metodo efficiente basato su MICP per risolvere il complesso problema dell'ottimizzazione delle contromisure contro attacchi in corso in presenza di informazioni incomplete sullo stato dell'attaccante [36].

3 Terminazione di un SSG

Uno *Stackelberg Security Game (SSG)* è un gioco sequenziale tra due agenti: un **difensore (leader)** e un **attaccante (follower)**. È utilizzato per modellare scenari di sicurezza, come la protezione di infrastrutture critiche, aeroporti o reti informatiche.

Il gioco termina nel momento in cui l'attaccante osserva la strategia del difensore e decide il suo attacco. Tuttavia, il significato di "terminare" può variare a seconda del contesto. Di seguito sono elencati i principali modi in cui può concludersi uno SSG.

1. Risoluzione computazionale del gioco
   Il gioco termina *matematicamente* quando si trova una strategia ottimale per il difensore, assumendo che l'attaccante risponda razionalmente. Ciò avviene tramite:

   - Programmazione lineare o algoritmi di ottimizzazione per calcolare la strategia mista del difensore.
   - Utilizzo di algoritmi specifici come DOBSS (Decomposed Optimal Bayesian Stackelberg Solver) in presenza di attaccanti con tipi multipli.

   Una volta trovata questa strategia, il gioco è considerato "risolto".

2. Esecuzione pratica del gioco
   In un'applicazione reale (ad esempio in un aeroporto):

   - Il difensore fissa una strategia di pattugliamento (ad esempio, pattugliare certe aree con determinate probabilità).
   - L'attaccante osserva e agisce una sola volta (gioco a singolo turno).
   - Il gioco termina quando l'attacco avviene e si osserva l'esito.

3. **Fine simulazione o iterazione**
   In simulazioni o ambienti dinamici:

   - Il gioco può terminare dopo un numero predefinito di turni.

- Oppure quando si raggiunge una *convergenza* delle strategie (nessun miglioramento significativo nei payoff).

Convergenza algoritmica (nei casi dinamici o simulati)
In ambienti iterativi o RL (come se usi Gymnasium) l'equilibrio si raggiunge quando le strategie si stabilizzano. Quindi si possono usare criteri di convergenza tipo:

- Cambiamento medio nei payoff minore di una certa soglia
- Strategie stabili per N iterazioni
- Gradiente della funzione obiettivo vicino a zero

CONSIDERAZIONI DA VALUTARE IN SEGUITO ALL'USO DI UN GRAFO DI ATTACCO PROBABILISTICO

4. Condizioni di arresto in ambienti dinamici
   In versioni estese, come *repeated SSGs* o *learning-based SSGs*, il gioco può terminare:

   - Quando si raggiunge un certo livello di sicurezza desiderata.
   - Quando l'attaccante cambia comportamento in modo non modellabile.
   - Per limiti di tempo computazionale o risorse di simulazione.

Uno Stackelberg Security Game si considera terminato quando:

- Viene calcolata una strategia ottima del difensore (in modelli teorici).
- Si compie l'attacco (in scenari reali).
- **Si raggiunge una condizione di arresto definita (in simulazioni).**

L'output di un SSG è una condizione di equilibrio detta SSE, quindi potremmo dichiarare il gioco come finito nel momento in cui questa condizione viene individuata. Tra gli strumenti pratici per verificare il raggiungimento del SSE si può usare Gymnasium? Altrimenti si può usare framework specializzati, come ad esempio, Gambit che calcola e verifica equilibri?

4 Esempio SSG

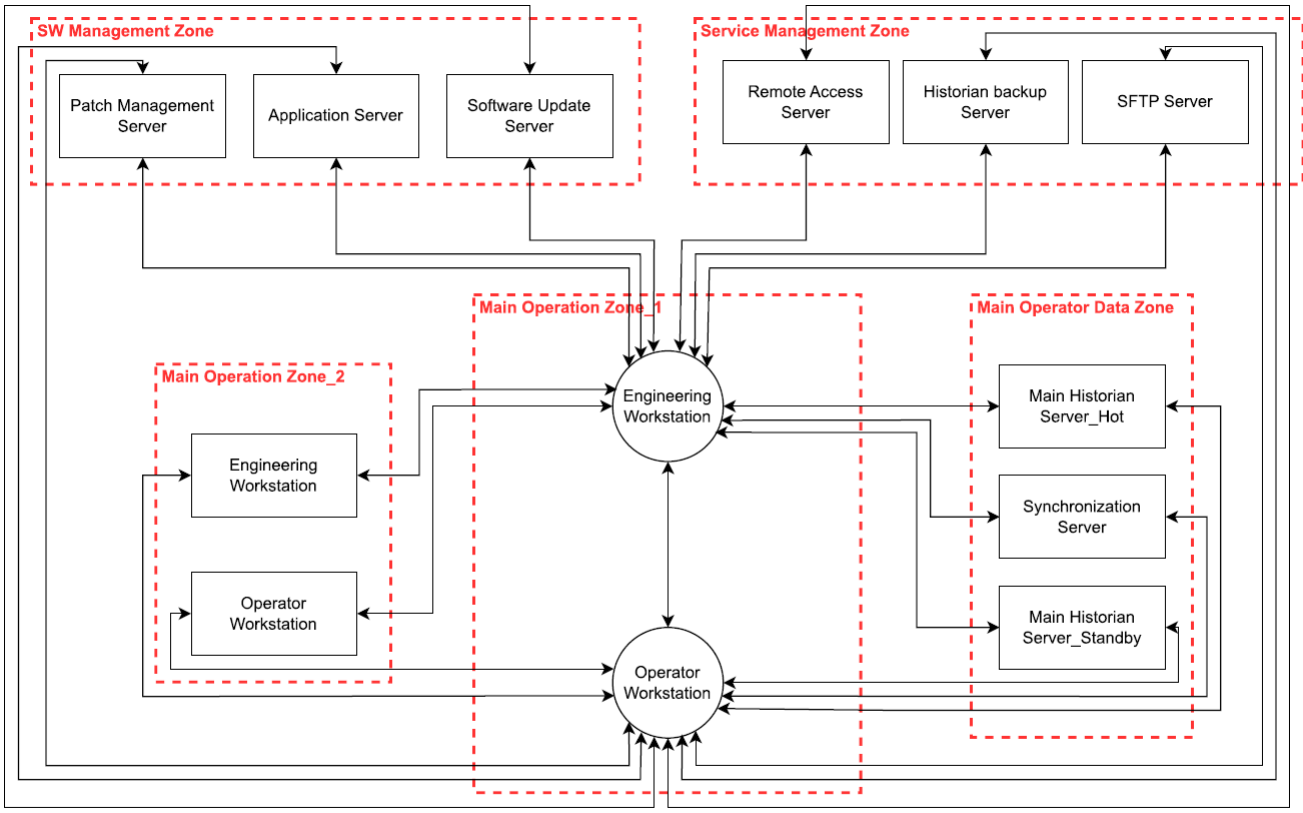
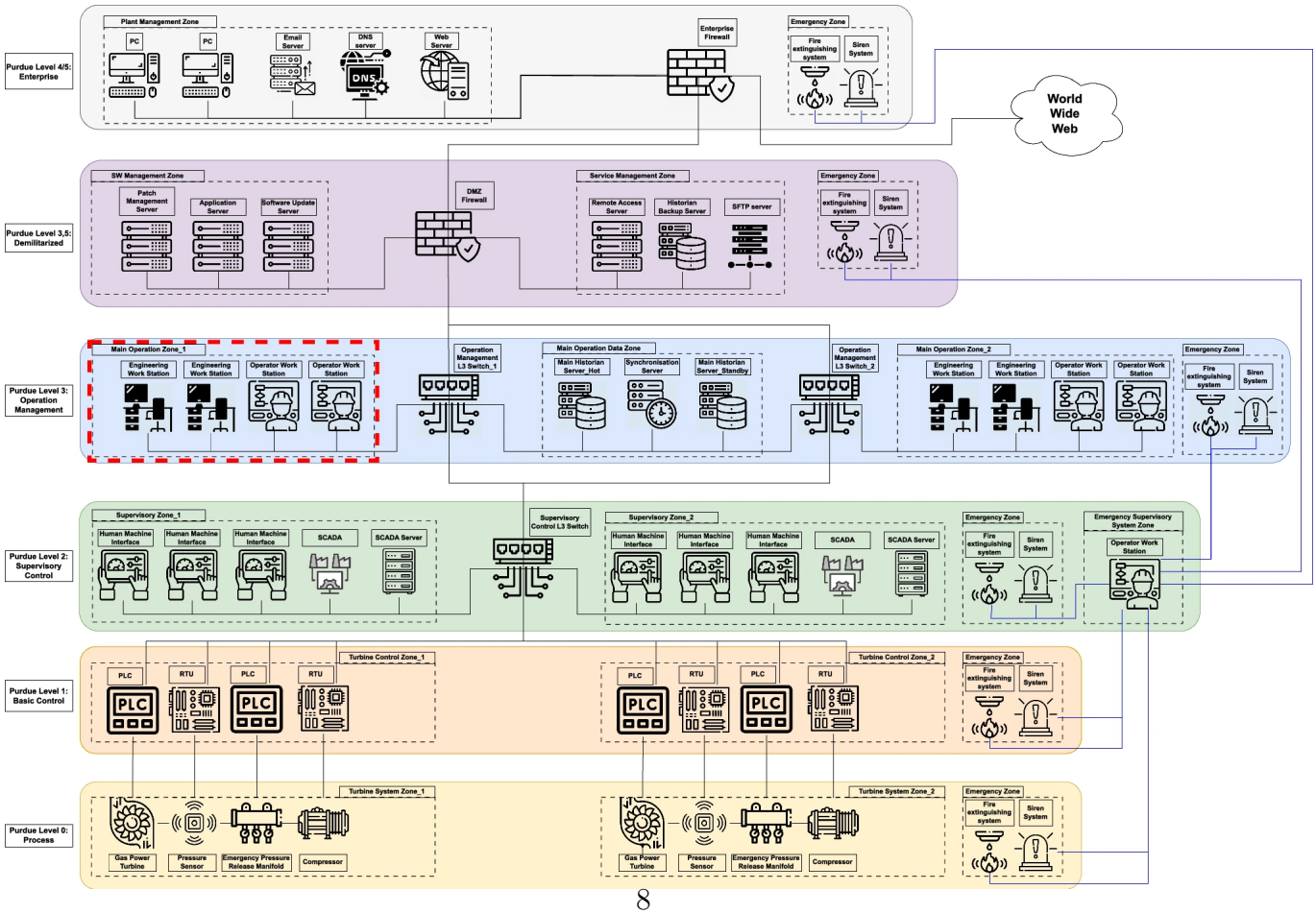


Fig. 3 DFD of the main operation zone 1



SUC per esempio per creazione attack graph (Purdue level 3 e 3.5 del file Brancat et al)

- Patch Management Server (PMS): sistema che automatizza il processo di distribuzione degli aggiornamenti software (patch) ai dispositivi in una rete, inclusi server e workstation.
- Application Server (AS): server che fornisce l'infrastruttura e le funzionalità logiche di supporto, sviluppo ed esecuzione di applicazioni nonché altri componenti server in un contesto distribuito.
- Software Update Server (SUS): server centrale che distribuisce aggiornamenti software (patch, nuove versioni, fix di sicurezza) ai dispositivi di una rete, evitando che ogni dispositivo scarichi gli aggiornamenti direttamente da Internet.
- Remote Access Server (RAS): è un server che consente agli utenti di connettersi da remoto a una rete privata o aziendale tramite una connessione sicura, come una VPN, un desktop remoto o un protocollo di accesso remoto (es. RDP, SSH).
- SFTP Server (SFTPS): un server che utilizza il protocollo SFTP per il trasferimento sicuro di file su una rete.
- DMZ Firewall (F): separa la rete interna di un'organizzazione da una zona demilitarizzata (DMZ), che ospita servizi accessibili da Internet. Questo crea un ulteriore livello di sicurezza, impedendo che eventuali compromissioni nella DMZ si propaghino alla rete interna.
- Synchronization Server (SS): è un sistema o componente di rete che ha il compito di mantenere dati o stati coerenti tra più dispositivi, server o applicazioni. In altre parole, assicura che le informazioni siano uguali e aggiornate su tutti i nodi coinvolti.
- Main Historian Server (MHS): è un tipo speciale di server utilizzato per raccogliere, archiviare, gestire e analizzare dati di processo nel tempo (dati storici), tipicamente in contesti industriali, di automazione o impianti SCADA (Supervisory Control and Data Acquisition).
- Operation Management L3 Switch (S3):
- Operator Workstation (OWS): è una postazione di lavoro (computer) usata dagli operatori di impianto per monitorare e controllare in tempo reale i processi industriali o di automazione
- Engineering Workstation (EWS): è un computer utilizzato in ambienti industriali o di automazione che serve per configurare, gestire e manutenere i sistemi di controllo.

Archi grafo $\rightarrow$ AFR (rappresentata in un intervallo da 0 a 1)
STRIDE model:

- Spoofing: Pretending to be something or someone other than yourself
- Tampering: Modifying something on disk, network, memory, or elsewhere
- Repudiation: Claiming that you didn't do something or were not responsible; can be honest or false (non considerata nell'esempio)
- Information disclosure: Someone obtaining information they are not authorized to access
- Denial of service: Exhausting resources needed to provide service

- Elevation of privilege: Allowing someone to do something they are not authorized to do

Analisi senza contromisure iniziale.

Si chiede all'owner quali contromisure del Fundamental Requirement sono già state implementate (guarda std 62443-3-3 annex B) per ogni minaccia STRIDE. (giustifica il perché delle probabilità assegnate) Non si esclude totalmente la minaccia, ma a seconda delle contromisure implementate si diminuisce la probabilità che quella minaccia avvenga.

Mappatura delle probabilità sugli archi da qualitativo a quantitativo (si considerano tutte le minacce del modello STRIDE, ad esclusione della Repudiation e per ogni minaccia per la quale è già presente una contromisura, la probabilità si abbassa alla fascia inferiore):

- high (H): $[0.8, 1]$ (1 minaccia risolta)
- medium-high (MH): $[0.6, 0.8)$ (2 minacce risolte)
- medium (M): $[0.4, 0.6)$ (3 minacce risolte)
- medium-low (ML):$[0.2, 0.4)$ (4 minacce risolte)
- low: $[0, 0.2)$ (5 minacce risolte)

Archi e probabilità associate (accanto ad ogni arco viene messa la lettera della minaccia per la quale, nel nodo iniziale, sono state implementate le contromisure):

- EWS → OWS (S, E): 0.69
- OWS → EWS (D, T): 0.77
- EWS → S3 (S): 0.92
- OWS → S3 (S, T, I): 0.57
- S3 → MHS (T): 0.93
- S3 → SS (I, D): 0.71
- S3 → F (E): 0.89
- MHS → SS (T, D, E): 0.40
- SS → MHS (S, I): 0.74
- F → PMS (D): 0.90
- F → AS (T): 0.82
- F → SUS (T, I, D, E): 0.38
- F → SFTPS (E): 0.86
- F → RAS (T, D, E): 0.44
- PMS → AS (S, I, E): 0.46
- AS → PMS (S): 0.80

- AS → SUS (D, E): 0.72
- SUS → AS (S, T, I, D): 0.26
- SFTPS → RAS (T, I, D, E): 0.30
- RAS → SFTPS (S, T, I, E): 0.24

Ad ogni nodo viene inoltre associato un impatto (il valore sarà rilevante nel momento in cui il nodo diventerà l'obiettivo di un determinato attaccante; l'impatto assume un valore da 0 a 10, dove 10 indica un danno molto ampio in caso di raggiungimento):

- EWS: 2
- OWS: 1
- S3: 3
- MHS: 4
- SS: 4
- F: 5
- PMS: 9
- AS: 10
- SUS: 7
- SFTPS: 6
- RAS: 8

Dato che abbiamo l'impatto e l'AFR, non rimane che mappare tali valori per ottenere il rischio ("risk") associato a una determinata. Considerando un path dell'attack graph, poiché ad ogni nodo si associa un determinato impatto e ad ogni arco si associa l'AFR. Quindi si calcola il rischio, usando la tabella del paper del lollo, considerando l'impatto del nodo da cui esce l'arco e la probabilità associata all'arco stesso. Quindi si ottiene un rischio per ogni arco del path (il rischio in questo caso corrisponde a "il rischio legato ad ogni asset da cui l'attaccante passa per arrivare al suo obiettivo") e per determinare il rischio associato al raggiungimento del nodo finale del path stesso, si prende il maggiore dei rischi trovati.

5 Idea Finale

Trascrizione PPT

La valutazione del rischio in cybersecurity è essenziale per proteggere proattivamente i sistemi industriali di automazione e controllo (IACS) dai rischi informatici. La teoria dei giochi offre un approccio strutturato per modellare le interazioni tra difensori e attaccanti. Gli **Stackelberg Security Games** (SSG) sono particolarmente adatti per ottimizzare l'allocazione delle risorse difensive.

Obiettivo: Introdurre gli SSG, il loro modello, i pro e contro, con un esempio semplice.

Obiettivo degli Stackelberg Security Games

Scopo: Ottimizzare l'allocazione delle risorse del difensore per minimizzare i rischi informatici contro un avversario razionale.

Il modello SSG prevede una dinamica leader-follower:

- Il **difensore** (leader) si impegna per primo in una strategia di sicurezza.
- L'**attaccante** (follower) risponde in modo ottimale, assumendo di conoscere la strategia del difensore.

Ciò è coerente con la necessità della cybersecurity di proteggere risorse critiche con risorse limitate (budget limitato per le contromisure).

Obiettivo: Identificare quali contromisure implementare e dove, per minimizzare i rischi informatici, rispettando i vincoli di budget.

Modello SSG: Input

- **Grafo di attacco:**
  - **Nodi:** stadi dell'attacco (es. shell remota come utente Apache).
  - **Archi:** vulnerabilità da sfruttare per passare al nodo successivo (es. CVE-2022-22720 – Apache HTTP Server 2.4.52).
  - Ogni arco ha un numero associato che rappresenta la probabilità che la vulnerabilità venga sfruttata (versione precedente da usare per un confronto nelle considerazioni della tesi: numero associato = rischio non mitigato può essere ottenuto da EPSS * CVSS).
- **Budget del difensore:** somma di denaro da investire nella difesa (es. 10k).
- **Contromisure:** requisiti di sicurezza con relativi costi ed efficacia sugli archi (es. segmentazione di rete, 1k, riduzione del rischio del 40% su CVE-2022-22720).

Modello SSG: Iterazioni

1. **Mossa del difensore:** sceglie per primo una strategia selezionando le contromisure da applicare e dove, rispettando i vincoli di budget.
2. **Mossa dell'attaccante:** osserva la strategia del difensore e sceglie il percorso nel grafo che gli offre il miglior ritorno.

Il ritorno dell'attaccante su un percorso corrisponde al rischio del difensore su quel percorso (gioco a somma zero).

Formula del ritorno dell'attaccante per ogni iterazione:

$$\max_{\forall \sigma} \left(\sum_{\forall e \in \sigma} R(e) \right)$$

dove:

- σ: percorsi nel grafo

- e: archi
- $R(e)$: rischio dell'arco

Il difensore aggiusta la propria strategia per ridurre il rischio dell'iterazione precedente. Le iterazioni continuano fino al raggiungimento dell'equilibrio.

Modello SSG: Output

L'equilibrio è raggiunto quando qualsiasi strategia difensiva diversa da quella corrente darebbe all'attaccante un ritorno maggiore rispetto all'iterazione precedente.

In altre parole, il gioco continua finché il difensore non trova una strategia tale che il ritorno per l'attaccante sia minore o uguale a quello dell'iterazione precedente.

Strategia del difensore: la migliore allocazione delle risorse che minimizza i rischi lungo i percorsi, rispettando i vincoli di budget.

Strategia dell'attaccante: il percorso che alla fine gli offre il miglior ritorno.

Se il ritorno finale dell'attaccante non è considerato un rischio tollerabile (è necessario fissare una soglia), il difensore deve aumentare il budget da investire nella difesa.

Considerazioni Post-Riunione (uso di CVSS ed EPSS obsoleto)

CVE → Il Common Vulnerabilities and Exposures, o CVE, è un dizionario di vulnerabilità e falle di sicurezza note pubblicamente mantenuto dalla MITRE Corporation.

CVSS → Il Common Vulnerability Scoring System (CVSS) è una norma tecnica aperta per valutare la gravità delle vulnerabilità di sicurezza di un sistema informatico. CVSS assegna un punteggio di gravità alle vulnerabilità, consentendo a chi si occupa di rispondere all'emergenza di stabilire la priorità di risposte e risorse in base al livello di minaccia. I punteggi vengono calcolati con una formula che dipende da diverse metriche che approssimano la facilità e l'impatto di un exploit. Il punteggio è espresso in una scala da 0 a 10, dove 10 indica il livello di vulnerabilità più grave.

EPSS → L'Exploit Prediction Scoring System (EPSS) è uno strumento basato sui dati per stimare la probabilità che una vulnerabilità software venga sfruttata in natura. Sebbene altri standard di settore siano stati utili per catturare le caratteristiche innate di una vulnerabilità e fornire misure di gravità, la loro capacità di valutare la minaccia è limitata. L'EPSS colma questa lacuna poiché utilizza le informazioni attuali sulle minacce provenienti da CVE e dati di exploit reali. Il modello EPSS produce un punteggio di probabilità compreso tra 0 e 1 (0 e 100%). Maggiore è il punteggio, maggiore è la probabilità che una vulnerabilità venga sfruttata.

Ciò su cui ci si concentra, in relazione al CRA per IACS, è la fase di sviluppo di un "Preventive Security Plan". A tale proposito si sviluppa il grafo di attacco probabilistico.

valore arco attack graph (rischio di attacco) = CVSS score (impatto) * EPSS score (probabilità dell'attacco) → payoff attaccante (opposto a quello del difensore) → da mappare a livello qualitativo

Rischio raggiungimento obiettivo finale da parte dell'attaccante = valore arco attack graph maggiore nel path intrapreso dall'attaccante stesso

Nel grafo di attacco probabilistico c'è la possibilità di avere più nodi iniziali così come più nodi finali, ovvero possono essere presenti più punti di accesso al sistema, così come più obiettivi per vari attaccanti. I primi all'interno del grafo possono essere rappresentati come nodi con soli archi uscenti, mentre i secondi come nodi con soli archi entranti. Quest'ultimo fatto non vuole dire che da quell'asset non ci siano più collegamenti ad altre componenti del sistema, ma che tutti gli altri possibili collegamenti sono "effimeri" per l'attaccante. Il fatto che i nodi finali siano contraddistinti dai soli archi entranti è dovuto al fatto che, nel momento in cui un attaccante li raggiungesse, esso terminerebbe la propria penetrazione all'interno del sistema. Tale condizione, però, varia a seconda del profilo dell'attaccante stesso.

First → sito per CVSS e EPSS

Rimane da fare un'analisi a risorse infinite

Nuove Considerazioni Post-Riunione (19/06/2025)

Considerare di fare 2 esempi, uno con il SUC del paper di Brancati et al e un altro basandosi sul sistema di esempio illustrato nel paper di ADVISE.

STRIDE GPT → Attack Tree, Per Attack graph usare NetworkX

Nuove considerazioni sui pesi degli archi del grafo: peso arco = probabilità che la vulnerabilità rappresentata dall'arco venga sfruttata da un attaccante (AFR).
Le vecchie considerazioni (peso arco = CVSS * EPSS) possono essere usate per un confronto nelle considerazioni della tesi.
Nell'attack graph il nodo obiettivo dell'attaccante avrà un attributo che ne indichi l'impatto.

Analisi delle tipologie di attaccanti sfruttando il modello STRIDE e la tabella della TAL (La TAL aiuta rapidamente i risk manager a identificare con precisione e a comprendere l'importanza degli agenti di minaccia rilevanti. La libreria è composta da 22 archetipi standardizzati definiti utilizzando otto attributi comuni; gli archetipi rappresentano agenti di minaccia esterni e interni, che vanno dalle spie industriali ai dipendenti non formati).

Per ogni arco del grafo, considerando una tipologia di attaccante, si segue un determinato processo di livello qualitativo:

- Si parte dalla probabilità di attraversamento dell'arco (= sfruttamento della vulnerabilità) più alta
- Si chiede all'owner del sistema se per quella vulnerabilità sono già state applicate delle contromisure di quelle disponibili
- A seconda delle contromisure già implementate si va a diminuire di conseguenza la probabilità di sfruttamento della vulnerabilità.

Le probabilità possono essere mappate a piacere dal livello qualitativo a quello quantitativo.

Le tipologie di attaccanti dipendono dal nodo dal quale l'arco parte, e quindi da quale struttura all'interno del sistema informatico è rappresentata dal nodo stesso.

Così facendo, si può evitare di usare per gli archi delle vulnerabilità troppo specifiche.

6 Struttura Tesi

- Introduzione (scrivi subito + revisione a fine lavoro): prefazione (2 pagine) (introduzione del problema (le 5 W))
- Capitolo 1: state of art, citare i paper analizzati senza copiare (analisi critica per quello che si trova nello state of art)
  (SSG → descrizione generale + focus sulle applicazioni CRA → introduzione generale paper lollo)
- Capitolo 2 (incluso nel capitolo 1): Incentrato sugli SSG, tutto quello che so (cos'è il problema e come viene implementato)
- Capitolo 3: applicazione degli SSG nel processo di CRA per un sistema cyberfisico, descrizione del caso di studio
- Capitolo 4 (capitolo modelling): Discussione dei risultati ottenuti + realizzazione modellino (descrizione dei dettagli implementativi del modello preso in considerazione)
- Capitolo 5: conclusioni (cosa ho fatto vedere nelle tesi + possibili sviluppi futuri)

Per ogni capitolo, all'inizio mettere qualche riga di introduzione al capitolo stesso. Da vedere se mettere i capitoli 3 e 4 insieme.

Il matching tra SSG e CRA nella parte conclusiva e dei lavori finali o da fare capitolo a parte, altrimenti cito entrambi gli approcci e poi nel capitolo dello sviluppo del modello si specifica il caso usato.

Quando si descrive lo scenario, giustificare lo scenario utilizzato piuttosto che giustificare ogni singolo valore usato nel modello.

Nella fase di modelling si può fare un'analisi di sensitività, ovvero far variare dei parametri di input per vedere le possibili debolezze e fare un'analisi degli attaccanti.