POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea

SimSCADA, un security serious game per gli ambienti SCADA

Relatori
prof. Antonio Lioy
ing. Andrea Atzeni

Candidato
Andrea Marchetti

Anno accademico 2018-2019

Sommario

Indice

Capitolo 1

Introduzione

1.1 La Sicurezza Informatica

Nel corso degli ultimi anni l'utilizzo di sistemi informatici ha avuto un utilizzo sempre più incrementale, sia in ambito industriale che per quanto riguarda i vari aspetti della vita quotidiana. L'Internet of Things è una realtà che si sta pian piano affermando in ogni aspetto della nostra vita, delineando quindi una sempre più grande e complessa rete di sistemi informatici interconnessi tra di loro. Parallelamente a tutto ciò si è reso necessario la definizione e l'implementazione di tecniche e protocolli per garantire la sicurezza informatica. Con il termine "sicurezza informatica" andiamo a identificare l'insieme dei prodotti, dei servizi, delle regole organizzative e dei comportamenti individuali che servono a proteggere i sistemi informatici a fronte di attacchi, danneggiamenti e/o accessi indesiderati, di natura accidentale o meno. Il fine è quello di riuscire a mantenere la confidenzialità, l'integrità e la disponibilità dei dati presenti in un sistema informatico [37].

L'aumentare dell'importanza della tecnologia nella nostra vita è testimoniato anche da altri dati: nell'annuale report di mercato di Cybersecurity Ventures [32], viene evidenziato come questo mercato continui ad essere in espansione, andando ad assumere attualmente un valore di circa 120 miliardi di dollari. Molte aziende continuano ad investire nei loro reparti di ricerca e sviluppo contro le minacce informatiche, facendo registrare un generale aumento della quantità di soldi esborsa. Questo dato va chiaramente preso in considerazione anche rispetto ai danni economici che possono scaturire a seguito di cyberattacchi, i quali possono generare cifre molto più alte di quelle investite. è ovvio quindi come allo stato attuale si ricorra sempre di più alla tecnologia, cercando di proteggersi nella maniera migliore possibile.

Purtroppo però non molto spesso si riesce in questo intento. Nell'ultimo anno si sono registrati numeri che stanno a testimoniare come occorra continuare ad investire risorse in questo ramo. Sono avvenute più di 53000 incidenti e oltre 2000 violazioni di dati confermate. Le vittime principali sono state le piccole imprese, molto spesso attaccate per motivazioni economiche, ma non mancano anche quelle di tipo strategico e di spionaggio. Rispetto al 2016, inoltre, si è registrato un lieve aumento tra gli organizzatori degli attacchi di figure operanti che lavoravano all'interno delle stesse aziende attaccate, a testimoniare ulteriormente come le minacce possano avere origine da qualsiasi parte, non necessariamente solo dall'esterno dell'ambiente di lavoro. La maggior parte degli attacchi è stata perpetrata attraverso l'utilizzo di ransomware, dei malware che rendono impossibile l'accesso ai dati alle vittime, a meno che non si effettui un pagamento per permettere lo sblocco delle risorse in questione. Sono stati numerosi anche gli attacchi attraverso le cosiddette tecniche di "ingegneria sociale": più di 1400 incidenti con quasi 400 casi di dati persi confermati [49].

1.2 Sistemi SCADA

La diffusione delle tecnologie IT è andata ad influenzare anche gli ambienti industriali definiti "isolati". Essi hanno basato la maggior parte dei loro protocolli di sicurezza proprio sul fatto che fossero degli ambienti con stretti e controllati rapporti con l'esterno. Questo ormai è un paradigma che si sta via via estinguendo, lasciando spazio ad ambienti industriali pienamente influenzati e coinvolti nell'Internet of Things.

Questo aspetto ha riguardato anche i sistemi SCADA (Supervisory Control and Data Acquisition), ossia quegli impianti di supervisione e controllo utilizzati soprattutto in ambiti industriali (tipicamente di generazione e distribuzione di energia elettrica e acquedotti), ma anche in aeroporti e ferrovie. Questi sistemi sono tipicamente composti da un gran numero di sensori collegati tra loro attraverso un sistema di comunicazione. Il nodo finale è costituito da uno o più computer supervisori ed hanno il compito di analizzare i dati raccolti dai sensori. Questi nodi inoltre rappresentano i punti di utilizzo, tramite cui è possibile interfacciarsi con l'impianto SCADA, ma possono non essere gli unici presenti nel sistema.Negli ultimi anni, infatti, si sono cominciate a sviluppare applicazioni per dispositivi handheld, ossia smartphone e tablet, per effettuare operazioni di controllo attraverso di essi. Questi diventano particolarmente utili quando si ha a che fare con sistemi SCADA con una vasta estensione geografica.

Sebbene tutti questi aspetti positivi appena analizzati, in realtà ci sono più insidie di quanto si pensi. I protocolli di sicurezza, così come quelli di comunicazione, utilizzati attualmente in ambiente SCADA, non tenevano per nulla conto delle minacce dovute al mondo esterno, dato che erano concepiti per ambienti isolati e affidabili, e sono risultati inefficaci per fronteggiare minacce IT. Inoltre le tecnologie introdotte hanno portato con loro anche le varie vulnerabilità di cui già soffrivano, andando quindi ad aumentare i pericoli a cui un sistema SCADA può essere sottoposto. A seguito anche di alcuni attacchi avvenuti a discapito di questi sistemi (uno su tutti il caso risalente al 2010 di Stuxnet, virus informatico con cui i governi di USA e Israele infettarono i sistemi di controllo di centrali nucleari iraniane [51] e che recentemente ha colpito anche delle centrali russe [42]), si è reso necessario sviluppare nuovi protocolli, in modo da garantire la sicurezza degli impianti SCADA e più in generale quella dei sistemi di controllo industriali.

1.3 Obiettivi del lavoro

I protocolli di sicurezza, e più in generale tutti gli strumenti necessari per avere un ambiente informatico sicuro, devono essere utilizzati nella maniera corretta, altrimenti si rischia di ottenere un effetto opposto a quello per cui sono stati destinati. è necessario quindi, oltre ai vari investimenti economici per la ricerca e lo sviluppo, effettuare un'adeguata formazione delle persone, in modo da educarli a come comportarsi in caso di situazioni a rischio.

Questo discorso è applicabile sia in ambito privato che negli ambienti industriali. In questi ultimi casi riuscire a garantire il corretto funzionamento dei sistemi di sicurezza significa riuscire a garantire l'incolumità di molta gente, sia fisica (pensiamo ai sistemi di controllo di acquedotti, impianti nucleari, ecc.) che quella dei dati a loro associati (se invece pensiamo a sistemi bancari, catastali, ecc.). Purtroppo l'evoluzione tecnologica è serrata e rimanere indietro costituisce un grosso vantaggio per chi commette cybercimini.

Lo scopo della tesi è quello di analizzare una possibile metodologia di formazione, ossia quella dei Serious Games, ed applicarla al campo della sicurezza informatica. In particolare gli argomenti di sicurezza che si affronteranno saranno strettamente legati all'ambiente dei sistemi SCADA. L'elaborato si divide in due parti principali: nella prima parte si effettua un'analisi teorica sul modello di apprendimento basato sull'utilizzo dei Serious Games insieme ad un'analisi di soluzioni già esistenti, così da vedere come questo metodo possa essere applicato alla materia della sicurezza informatica,. Vengono presentati inoltre i sistemi SCADA, la loro architettura, i loro protocolli di comunicazione, le criticità che li affligge ed i protocolli di sicurezza esistenti per difenderli. Nella seconda parte ci si concentra nella realizzazione di un Serious Game per simulare un sistema SCADA sotto attacco, costringendo l'utente a valutare attentamente la situazione e prendere le decisioni più adatte per contrastare l'emergenza.

Capitolo 2

Serious Games

In questo capitolo si analizza il concetto di Serious Game. Nella sezione 2.1 viene data inizialmente una definizione di Serious Game, analizzando i vari aspetti che li caratterizzano (2.1.1). Successivamente sono riportate un'analisi sulla classificazione delle varie tipologie di Serious Games (2.1.2) e una sui vantaggi che comporta il loro utilizzo in ambito formativo (2.1.3). Nella sezione 2.2 infine sono presi in considerazione alcuni Serious Games, descrivendo le loro funzionalità, gli obiettivi prefissati al momento della loro realizzazione dagli sviluppatori e come essi sono riusciti a raggiungerli.

2.1 Concetti generali

2.1.1 Definizione di Serious Games

Il concetto di Serious Game è stato a lungo analizzato e discusso nel corso del tempo. Uno dei primi cenni a questo argomento è presente addirittura in alcuni scritti di Platone in cui il filosofo descriveva l'importanza dei giochi per la formazione dei bambini, dato che potevano fornire uno stimolo fondamentale ai fini dell'apprendimento e della formazione [13]. Vi sono poi molti esempi storici di applicazioni di metodi educativi/didattici ai giochi. Chaturgana, identificato come il precursore degli scacchi, è uno dei primi giochi di cui si ha traccia in cui si deve applicare una tattica militare al fine di avere la meglio sull'avversario [34]. Il concept originario del Monopoli, Landlord's Game, creato nel 1902, invece, era stato studiato per mostrare i reali pericoli del capitalismo nella società di allora [34].

Il primo a parlare esplicitamente di "Serious Game" fu, negli anni '70, Clark Abt nel suo libro "Serious Games", in cui li definisce ([2]) come giochi che "hanno un intento educativo esplicito e attentamente ponderato e non sono concepiti per essere giocati primariamente con fini di divertimento". Questa descrizione è stata ripresa e adattata nel corso degli anni da numerosi altri studiosi, con il fine di permettere una maggior diffusione e comprensione del termine. Tra le varie definizioni si riporta quella data nel 2005 da David Michael e Sande Chen ("giochi che non hanno l'intrattenimento, lo svago o il divertimento come loro scopo primario") in [31], insieme a quella del 2006 di Kevin Corti [8] secondo cui l'apprendimento tramite Serious Game "è tutta una questione di bilanciare il potere dei videogiochi per accattivarsi ed attrarre l'utente finale con uno scopo specifico, come ad esempio sviluppare nuove conoscenze e abilità".

Nel 2007, invece, lo studioso e Game designer Ian Bogost propone in [4] il termine "Persuasive Games" come sostituto evolutosi a partire da quello precedente, ritenuto troppo restrittivo. In questa categoria, Bogost intende inserire tutti quei giochi che fanno uso della "retorica procedurale" (definita, sempre in [4], come la "pratica di usare i processi in maniera persuasiva") come strumento d'apprendimento dell'utente finale. Alla luce di quanto espresso da Bogost, è possibile racchiudere in questa categoria una gran varietà di giochi. Tra questi ne abbiamo alcuni il cui scopo principale non è quello di essere uno strumento di insegnamento, ma hanno trovato comunque un utilizzo in ambito educativo (giochi a cui è stato applicato un"purpose-shifting"). Ad esempio

la serie videoludica di "Assassin's Creed", in cui il giocatore viene immerso in diversi contesti storici, ricreati in maniera talmente accurata da essere utilizzati come strumento didattico nelle scuole [17, 48].

Onde evitare confusioni, ai fini di questo lavoro, sono definiti Serious Games tutti i prodotti in cui si unisce un fine "serio" alla struttura di un gioco. A questi sono affiancati quei giochi nati con fini di intrattenimento ma che, vuoi per la creazione di modifiche (chiamate "mod") ad hoc, vuoi per questioni di "purpose-shifting", si ritrovano ad essere utilizzati con scopi didattici/formativi. Insieme definiscono il "Serious Gaming", ossia qualsiasi gioco utilizzato con fini "seri" che possono o meno essere stati concepiti al momento della loro creazione (cf. fig. **??**).

Figura 2.1. Grafico che riassume la correlazione tra giochi, Serious Game e Serious Gaming

2.1.2 Classificazione

I Serious Games appaiono una categoria abbastanza ampia e ricca di elementi. Si è reso necessario, quindi, un sistema di classificazione, in modo da poter identificare al meglio i vari giochi appartenenti alla categoria. Fin dal 2002, sono stati fatti diversi tentativi per introdurre altrettanti sistemi di identificazione. Ognuno di essi è stato creato con il preciso scopo di migliorare i difetti del sistema precedente, ma purtroppo nonostante i vari tentativi nessuno di essi si è imposto in maniera generale sugli altri.

In ordine cronologico, le prime metodologie di classificazione sono basati su un singolo criterio di valutazione e possono essere suddivisi in due categorie:

Market-based in cui il principio di distinzione è dato dal mercato (ossia le persone che ci giocheranno) che utilizza il gioco in questione;

Purpose-based dove la classificazione avviene a seconda dell'intento e dello scopo per cui ogni gioco è disegnato e creato.

Le limitazioni di questi metodi sono numerose, in primis il fatto stessi che si basano solo su un solo criterio di distinzione. Inoltre per la classificazione "Market-based" il problema principale è che avviene solo tenendo conto dell'uso a cui è destinato il gioco e non del loro effettivo contenuto. Questo inconveniente è stato superato con il metodo "Purpose-based", afflitto però da un altro tipo di difetto: vi sono categorie troppo eterogenee che ne limitano l'affidabilità.

Questi metodi, però, hanno avuto il merito di aver ispirato gli studiosi Sawyer e Smith, tanto da creare nel 2008 un loro sistema di classificazione, basato su criteri multipli: il "Serious Game Taxonomy" ([40]). Con questo sistema si categorizzano i giochi secondo due principi:

Mercato: Organizzazioni governative e non governative, Difesa, Salute, Marketing e Comunicazione, Educazione, Settore aziendale, Settore Industriale;

Scopo: Giochi per la Salute, Giochi per la Pubblicità, Giochi per l'Allenamento, Giochi per l'Educazione, Giochi per la Scienza e Ricerca, Produzione, Giochi come Lavoro.

Come si può notare, questo sistema è una fusione di quelli precedenti e fa uso di entrambi i criteri di classificazione. Inoltre nella categoria "scopo" avviene una sotto-classificazione aggiuntiva, andando ad aggiungere una maggior complessità. Tutto ciò, comunque, permette una miglior comprensione dei Serious Games tramite una categorizzazione più precisa. Ciononostante, anche questo sistema ha dei punti deboli: il criterio "scopo" spesso risulta non sufficientemente accurato per alcune tipologie di gioco e sono presenti delle categorie di "mercato" e "scopo" che risultano coincidenti tra di loro.

Nonostante i difetti del loro sistema, resta comunque il fatto che l'idea lanciata da Sawyer e Smith di utilizzare più elementi di classificazione risulti abbastanza convincente, a tal punto da essere ripresa nel 2011 da parte di Damien Djaouti, Julian Alvarez e Jean-Pierre Jessel in [16] [cap. 6, pp. 118-136]. Questi 3 studiosi riprendono i 2 principi di classificazione di Sawyer e Smith, a cui però ne aggiungono un terzo, il "gameplay". Questo è un aspetto che caratterizza essenzialmente qualsiasi gioco, ma di cui bisogna tener conto anche in questo ambito, dato che non si può scindere completamente un Serious Game dalla sua dimensione ludica. Con il modello da loro teorizzato, i 3 studiosi tentano, quindi, di superare i precedenti sistemi il cui limite principale consiste nel focalizzarsi solo sulla parte "Serious", tralasciando quella del "Game".

In i tre aspetti distintivi del modello (fig.2.2) sono:

Gameplay: che riguarda al tipo di gameplay utilizzato nel gioco. In particolare, le informazioni fornite da questo criterio riguardano la struttura del Serious Game (come è giocato);

Scopo: che riguarda allo scopo per cui è stato designato il gioco. In particolare, gli eventuali scopi, oltre all'intrattenimento, a cui è destinato il Serious Game (perché è giocato);

Ambito: che riguarda alle applicazioni a cui è rivolto il gioco. In particolare,il tipo di mercato in cui è inserito e il pubblico che ne fa utilizzo (da chi è giocato).

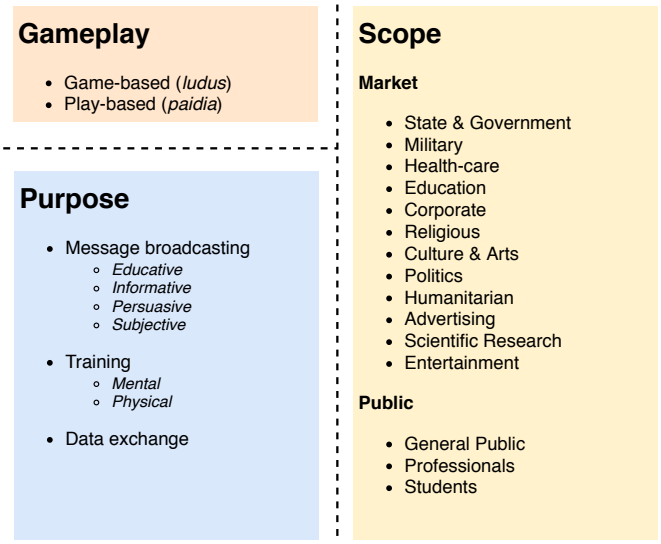

Figura 2.2. Modello G/P/S per la classificazione dei Serious Games; sono riportate anche le sottocategorie per ogni carattere distintivo

Per dimostrare l'efficacia del modello, il "G/P/S Model" è stato utilizzato per la creazione di un database di Serious Game [30] in cui sono stati raccolti più di 550 Serious Games. Inoltre,

essendo di natura collaborativa, se un prodotto risulta mancante all'interno della lista di giochi, è possibile aggiungerlo, classificandolo secondo lo stesso sistema.

Resta comunque anche esso un modello con dei limiti. Dato che è stato ideato per dare una visione generale dei Serious Games, non è possibile fornire informazione dettagliate su una specifica area d'interesse. Ad esempio, se si cerca un Serious Game associato al mercato dell'educazione, non possiamo differenziare tra quelli che trattano di matematica e quelli che trattano di storia. La differenziazione può avvenire solo secondo i criteri specificati dal modello. Ciò non toglie che può essere uno strumento utile per il Serious Gaming, dato che permette di identificare sia i vari Serious Games, che normali prodotti d'intrattenimento a cui è stato applicato un "purpose-shifting".

2.1.3 Vantaggi derivanti dal loro utilizzo

In questa sezione si esporranno i possibili vantaggi derivanti dall'utilizzo dei Serious Games. Gli aspetti benefici sono stati studiati (e tutt'ora continuano ad esserlo) ormai da numerosi scienziati nel corso degli anni. È ormai confermato il fatto che utilizzare videogiochi possa aiutare ad aumentare le capacità di problem-solving, affinare il pensiero strategico e migliorare l'attenzione, la vista e la coordinazione, a seconda dell'area di cervello attivata dal gioco in questione [46].

Inoltre in campo medico si fa sempre più ricorso ai videogiochi sia per questioni legate alla riabilitazione fisica [29, 6] , così come per combattere problemi psicologici: fobie, paure, disturbo da stress post-traumatico (PTSD) [14, 39] o disturbo da deficit di attenzione e iperattività (ADHD) [18, 27]. Questo grazie anche al progresso tecnologico che ha permesso una sempre più elevata immersione del paziente nel mondo virtuale.

Ovviamente, così come sono stati condotti studi sugli aspetti positivi, ne sono stati fatti altrettanti per analizzare i possibili effetti negativi derivanti da un uso improprio dei videogiochi. È stato documentato infatti che un utilizzo prolungato in periodi brevi può portare il giocatore ad assumere comportamenti aggressivi [3] o, addirittura, in alcuni casi, ad un distaccamento dalla realtà, causando l'insurrezione di patologie mentali (stress, ansia, depressione) e fisiche (obesità). Questi sintomi, nel 2018, sono stati ufficialmente identificati all'interno dell'undicesima revisione della Classificazione Internazionale delle Malattie (ICD-11) dal World Health Organization (WHO) come "gaming disorder", malattia appartenente alla famiglia dei disturbi mentali e comportamentali [50]. C'è da sottolineare, comunque, come quest'ultimo fatto sia stato ampiamente criticato e ritenuto prematuro, data anche la limitatezza di dati a supporto della teoria [1].

Comunque sia questi ultimi aspetti negativi sono più legati all'utilizzo di videogiochi in generale che a quello dei Serious Games. Inoltre,ai vantaggi sopracitati, bisogna aggiungere anche i benefici derivanti dalla loro natura pedagogica. I principali vantaggi sono:

Coinvolgimento Come è stato più volte sottolineato, i Serious Game sono innanzitutto dei videogiochi ed è indubbio che molte persone si divertano nel giocarli, con la conseguenza di spendere molto tempo in questa attività. Ciò è dovuto principalmente al coinvolgimento, generato dal divertimento, che un videogioco ben strutturato riesce a creare nel giocatore. È possibile sfruttare questo aspetto per mantenere alto il livello di concentrazione di uno studente, evitando distrazioni dovute a poco interesse e noia.

Sebbene la sua importanza, il divertimento non è l'unico modo per coinvolgere il giocatore. Si possono sfruttare anche altri aspetti psicologici per aumentare il coinvolgimento. Ad esempio, la curiosità, che può spingere un utente ad immergersi di più nel mondo virtuale solo per esplorarlo e scoprire tutti gli aspetti creati dagli sviluppatori.

Ma il metodo più efficace per garantire un buon coinvolgimento da parte di chi gioca è quello di creare continue sfide. Proponendo degli ostacoli da superare, degli obiettivi da raggiungere, è possibile mantenere alto il livello di concentrazione dell'utente, che proverà un senso di soddisfazione ogniqualvolta riuscirà a completare una sfida, gettandosi nella risoluzione di quella successiva. Questo aspetto è stato soggetto di studio da parte di numerosi studiosi. Tra questi, Mihaly Cziksentmihalyi in [10] e in [9], esplora ed espone i diversi stati mentali che una persona può attraversare nel momento in cui si trova coinvolto

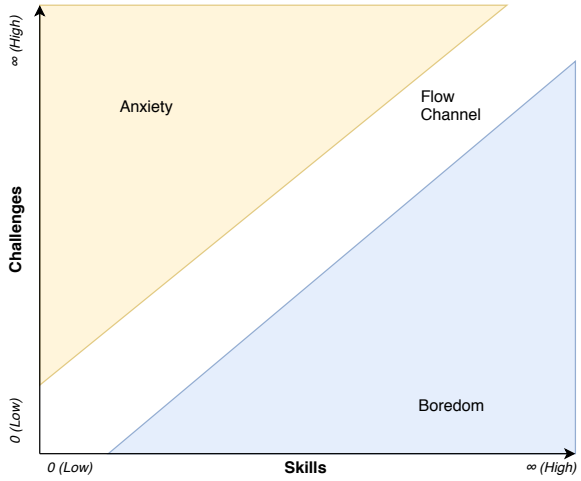

Figura 2.3. Grafico raffigurante la Teoria del Flusso di Cziksentmihalyi

in un'attività che richieda delle abilità e includa delle sfide. Una rappresentazione grafica è riportata in figura 2.3.

Uno sviluppatore di un Serious Game deve essere capace di mantenere il giocatore all'interno del cosiddetto "Canale di Flusso", tenendo alta la motivazione. Questa infatti rappresenta l'elemento chiave per cui l'apprendimento possa avvenire. Per fare ciò, oltre a proporre delle sfide, il Game designer deve anche contemplare un sistema di feedback che renda l'utente consapevole dei propri progressi e del raggiungimento degli obiettivi, coinvolgendolo attivamente nel processo di apprendimento. Il gioco, inoltre, riduce l'ansia associata al compimento degli errori, visti non più come un fallimento che impedisce i progressi, ma come una fase necessariamente da attraversare per imparare. Naturalmente, al fine di ottenere i risultati migliori, il contesto del gioco in cui si cerca di raggiungere questo grado immersione e coinvolgimento deve essere il più vicino possibile ad un contesto del mondo reale.

Apprendimento situato e contesto Nel 1991 Jean Lave e Etienne Wenger propongono in [28] una rivisitazione del concetto di apprendimento: i due studiosi affermano che questo non si sviluppa solo se indotto dall'insegnamento, ma anzi è un processo che si ottiene in determinati contesti, rapportandosi con altre persone (andando, quindi, a formare una comunità di "practioners") e in relazione al coinvolgimento in attività specifiche. In questo senso, l'apprendimento è il risultato di ciò che viene definito dagli autori come "legitimate peripheral partecipation", ossia un processo attraverso il quale i neo-membri di una comunità di "practioners", svolgendo inizialmente compiti semplici e poco rischiosi, acquisiscono gradualmente tutte le competenze necessarie per diventare esperti, finendo per essere coinvolti in ogni attività e pratica della comunità.

Sulla base di questa teoria, negli anni a seguire, alcuni studiosi hanno cercato di trovare un'applicazione anche in campo informatico. In particolare, nel 2000, Jan Herrington in [22] espone una lista di 9 features per identificare un modello di apprendimento situato, definendo poi un framework per il design degli ambienti d'insegnamento. Questi principi sono:

1. Fornire contesti che rispecchino il modo in cui le conoscenze verranno usate nella realtà;
2. Fornire attività realistiche;
3. Permettere l'accesso alle performance di esperti;
4. Fornire ruoli e prospettive multiple;
5. Supportare la costruzione di conoscenza collaborativa;
6. Promuovere la riflessione;
7. Rendere esplicita la conoscenza tacita;

8. Fornire l'aiuto e il sostegno degli insegnanti nei momenti critici;
9. Fornire giudizi durante le attività di insegnamento.

Ancora, nel 2004, David Williamson Shaffer in [41] sottolinea la potenza che può generare il mondo virtuale dei videogiochi e come questo renda possibile lo sviluppo dell'apprendimento situato, permettendo allo studente il confronto con molteplici contesti in cui imparare concetti anche complessi, senza però perdere di vista la loro relazione con i problemi del mondo reale.

Il contesto di un Serious Game, inoltre, gioca un ulteriore ruolo di supporto nell'insegnamento dei concetti di teoria. In accordo con diverse teorie pedagogiche, insegnare non significa fornire solamente nuove informazioni. È necessario integrarle con il bagaglio conoscitivo dello studente, in modo che siano assimilate nella maniera più corretta. Inserendole nel giusto contesto, grazie anche all'immersione che si viene a generare, è possibile per il giocatore creare i legami con quanto già appreso, necessari per facilitare la memorizzazione dei nuovi concetti.

Responsabilizzazione Molto spesso nei Serious Games, il giocatore viene catapultato in un contesto virtuale dove ricopre posizioni importanti dalle responsabilità elevate. Questo comporta che dovrà prendere delle decisioni le cui conseguenze avranno un impatto nell'ambiente che lo circonda. Data la possibilità di ricevere dei feedback visivi in tempi immediati, un giocatore può stabilire la corretta relazione causa-effetto delle sue decisioni, capire quali di esse fossero quelle corrette e rendere il processo di apprendimento più semplice e concreto.

Necessità di creare un modella della realtà Apprendere nozioni nel mondo reale può essere difficile anche a causa dei molteplici stimoli che una persona può ricevere, la maggior parte dei quali può anche risultare inutile. Nella creazione di un Serious Game (e più in generale di un qualsiasi videogioco), lo sviluppatore, per questioni legate al tempo e ai costi, deve adoperare delle operazioni di semplificazione della realtà, scegliendo cosa rappresentare fedelmente e cosa invece approssimare o ignorare. Questo comporta un vantaggio in termini di apprendimento, dato che il giocatore/studente si troverà ad interagire solo con gli elementi strettamente necessari per la sua formazione, evitando distrazioni causate dagli elementi superflui.

2.2 Case study

Come già affermato, nell'ultimo periodo, il boom dei Serious Games come strumento didattico si è diffuso anche in ambito informatico. A testimonianza è possibile confrontare la tabella 1 in [21] in cui è riportato una schematizzazione dei Serious Game per la cyber-sicurezza sviluppati negli ultimi anni. Nella stessa tabella, oltre a una breve tassonomia dei vari giochi, sono riportati anche i risultati ottenuti con il loro utilizzo. Come si può notare non tutti hanno registrato risultati, dovuto anche alla necessità di avere ulteriori valutazioni, ma in generale è possibile affermare che il loro utilizzo abbia dato dei benefici. Tra quelli più positivi, ne sono stati scelti tre Serious Games da analizzare e di cui riportare le principali caratteristiche, anche in relazione con i concetti espressi fin ad ora.

2.2.1 CyberCIEGE

CyberCIEGE è un videogioco sviluppato nel 2006 da parte della Naval Postgraduate School per insegnare concetti di sicurezza delle reti. Il suo sviluppo è stato sovvenzionato dalla US Navy ed è attualmente utilizzato dagli stessi sviluppatori in corsi didattici, così come da numerose agenzie ed istituzioni scolastiche come strumento di addestramento.

Il gioco è stato creato per permettere agli studenti di confrontarsi con delle decisioni a tema di sicurezza informatica, inserendoli in un ambiente sicuro e che incoraggia alla sperimentazione e alla riflessione. L'obiettivo è quello di mostrare i concetti astratti e le limitazioni dei vari meccanismi di difesa. Vi sono oltre 20 scenari che cercano di trattare il più possibile i vari

concetti. Lo studente, nel gioco, ricopre il ruolo di amministratore di sistema di una qualche azienda e in ognuno degli scenari vi sono degli obiettivi da completare, il cui completamento potrà permettere l'avanzamento al livello successivo. Per far ciò dovrà prendere delle decisioni che possono potenzialmente intaccare la sicurezza dell'ambiente in cui lavora. Nei vari livelli, per completare i propri obiettivi, sarà necessario acquistare server aggiuntivi, creare permessi di rete specifici per permettere l'accesso alle risorse di sistema agli utenti indicati dal gioco, occuparsi della sicurezza fisica dei sistemi, educare le nuove risorse e così via. È presente, inoltre, un sistema di ricompense che premia il giocatore nel momento del raggiungimento degli obiettivi (altrimenti, in caso contrario, verrà punito tramite lo stesso). I principali elementi del gioco sono:

Asset L'insieme delle informazioni che il giocatore dovrà proteggere dagli attacchi. Possono essere sia fisiche che virtuali e ognuna di esse richiede un trattamento diverso, a seconda che se ne voglia mantenere la segretezza, l'integrità o altre proprietà. Hanno un valore economico specificato che oltre a quantificare il danno che l'azienda subirebbe in caso di perdita della risorsa, caratterizza anche la frequenza degli attacchi subiti (più la risorsa sarà costosa, più frequentemente sarà attaccata).

Gli asset potrebbero dover essere condivisi all'interno dell'azienda, il che obbliga il giocatore a definire ed implementare una rete adatta affinché ogni utente possa accedervi senza correre alcun rischio, tenendo anche conto della loro dislocazione sul territorio. Ciò comporta, se necessario, anche l'eventuale setup di una VPN o di altre misure di protezione nel caso in cui l'accesso all'asset avvenga tramite Internet.

Utenti L'insieme degli impiegati virtuali presenti in ogni scenario. Come detto, al fine di completare gli scenari, è necessario che il giocatore fornisca loro le risorse di cui hanno bisogno, mettendoli nelle migliori condizioni di lavoro. A seconda del verificarsi o meno di queste condizioni, gli utenti possono assumere degli atteggiamenti diversi, a volte anche nocivi per la stessa azienda. Questo, unito anche al loro grado di abilità nell'utilizzo delle le risorse, permette di distinguere quattro categorie di utenza diversa:

- **Utenti normali** eseguono correttamente il loro lavoro utilizzando gli asset messi a loro disposizione. Possono apprendere nuove cose e migliorare la loro abilità. Ciò non toglie che in caso di procedure particolari da eseguire, all'insorgere di un problema, tendono ad irritarsi, cercando degli escamotage potenzialmente pericolosi. Proprio per questo non è possibile garantire loro la gestione di dati particolarmente sensibili;
- **Utenti affidabili** hanno un'abilità più elevata rispetto agli utenti normali. Ciò comporta che possono trattare dati sensibili, ricoprendo ruoli di alta responsabilità, e sono molto più tolleranti se messi di fronte a procedure complesse e restrittive da eseguire. Possono frequentare corsi speciali di aggiornamento;
- **Utenti arrabbiati** sono utenti normali le cui richieste sono state ignorate (o non soddisfatte) dal giocatore. Cercheranno di recar danno all'azienda, cercando però di non eseguire azioni troppo rischiose che comportino un loro licenziamento nel caso in cui vengano scoperti;
- **Utenti incompetenti** non hanno l'autorizzazione al trattamento di dati sensibili e generalmente non cercheranno di violare le politiche discrezionali dell'azienda. Tuttavia la loro incompetenza e l'ignorare i propri limiti li può portare a fraintendere la discrezionalità di suddette politiche,causando danni ad informazioni a cui non dovrebbero nemmeno aver accesso.

In poche parole, quindi, il giocatore è tenuto a gestire si i rischi esterni che potenziali rischi interni alla stessa azienda, dovuti a comportamenti dannosi degli utenti arrabbiati e/o inesperti.

Attaccanti Sono coloro che cercheranno di perpetrare gli attacchi, sia fisici che virtuali, ai danni dell'azienda. Sono suddivisi in due categorie principali:

- **Vandali** elementi spinti principalmente dalla noia e/o dalla pura curiosità tecnica. Non hanno motivazioni particolari che li invogli a spendere grosse risorse nell'attacco,

ma hanno le competenze necessarie per danneggiare esplicitamente gli asset. Alcuni possono essere spinti in realtà solo da intenti sperimentali e non si rendono conto della pericolosità delle loro azioni. Ciò non toglie che la loro presenza nei sistemi aziendali comporti l'intraprendere di azioni riparatorie da parte del giocatore;

- **Attaccanti professionisti** sono spinti da motivazioni prettamente economiche e, pertanto, hanno la propensione ad investire sia le risorse che il tempo nell'attacco. Cercano di effettuare attacchi clandestini la cui rintracciabilità risulti difficile da parte del giocatore.Tenteranno, inoltre, di infiltrarsi sia all'interno dell'azienda (facendo assumere dei loro agenti o corrompendo gli utenti già presenti) che all'interno delle compagnie partner fornitrici di software, in modo da poter installare Trojan Horse o trap door nei sistemi aziendali, così da garantirsi l'accesso agli asset voluti.

Zone Ogni scenario è suddiviso in una o più zone, corrispondenti, ossia, alle suddivisioni fisiche dell'ambiente di lavoro. È possibile definire per ogni zona una politica di accesso, più o meno restrittiva, per ogni tipo di utente, definendo anche cosa si può portare al suo interno e cosa no. Per aiutare a rispettare queste regole, si possono utilizzare meccanismi di protezione aggiuntivi, quali porte magnetiche (il cui meccanismo di sblocco può essere basato su sistemi biometrici) o guardie di sicurezza.

Gli utenti si possono muovere tra le zone rispettando le regole definite. Tuttavia occorre considerare che la rete aziendale si si estende attraverso zone multiple. Ciò significa che, in caso di attacco andato a segno, è possibile per gli attaccanti ottenere l'accesso virtuale anche a zone proibite. Occorre quindi proteggersi ed evitare il verificarsi di questa situazione, utilizzando le risorse messe a disposizione dal gioco, come ad esempio dispositivi per la cifratura della rete o l'instaurazione di una VPN.

Il gioco inoltre mette a disposizione del giocatore 2 ulteriori strumenti, aumentando la profondità del gameplay. Il primo è lo Scenario Development Tool (SDT). In poche parole è possibile produrre dei file di testo, scritti in Scenario Definition Language (SDL), per definire e descrivere degli scenari aggiuntivi, in modo da coprire quegli argomenti che non sono trattati in quelli predefiniti. Il linguaggio SDL descrive gli scenari in termini di utenti, asset, obiettivi degli utenti, attacchi e motivi per cui avvengono e impostazioni di sicurezza iniziale. Una volta definito ciò, il motore di gioco crea la topologia di rete, le impostazioni di sicurezza e, a partire dalla definizioni degli attacchi, stabilisce se essi andranno a buon fine o meno. gestirà, inoltre, in maniera automatica l'economia del gioco, lasciando quindi lo sviluppatore libero da questo compito. Ovviamente è necessario specificare un'ambientazione e gli obiettivi per completare lo scenario, in modo da permettere al giocatore di procedere attraverso una sequenza di sfide coerente e ben organizzata.

Il secondo strumento è una modalità di gioco alternativa: mentre normalmente ci si ritrova nei panni di chi deve difendere e prevenire gli attacchi alla rete gestita, nella modalità alternativa è possibile ricoprire il ruolo dell'attaccante, dando quindi la possibilità di vedere la rete creata precedentemente da una nuova prospettiva. Il nuovo ruolo ricoperto permette di dimostrare l'effettiva protezione garantita dai metodi difensivi implementati ai vari tipi di attacco, così come l'efficacia di quest'ultimi. Anche in questo caso è richiesta al giocatore una buona capacità decisionale per scegliere le tattiche di attacco più adatte, i cui esiti saranno subito mostrati, permettendo di valutare la tenuta difensiva della rete.

La valutazione del giocatore avviene su due livelli: il primo,più superficiale e immediato, è collegata all'andamento dell'economia del gioco. Prendere delle decisioni sbagliate, infatti, comincerà a far perdere fondi alla propria azienda, portandola ben presto sull'orlo della bancarotta (e di conseguenza alla fine del gioco). Al contrario, le decisioni corrette aiuteranno a generare introiti e permetteranno l'avanzamento ai livelli successivi. Oltre a questo, alla fine di ogni scenario viene mostrato al giocatore un log, riassumente tutte le decisioni prese durante il gioco, opportunamente analizzati dal sistema stesso, in modo da mettere in risalto le motivazioni dietro ad una determinata decisione. Questo può anche essere usato dagli insegnati, per registrare i progressi di uno studente e gli errori nei vari scenari. I log sono disponibili anche nel momento in cui si giochi uno scenario creato tramite lo SDT. Lo sviluppatore può configurare opportunamente dei trigger in modo da generare le parti di log nel momento in cui si verifichi una determinata situazione.

Nel corso degli anni, CyberCIEGE ha ottenuto ottimi risultati come strumento didattico, come dimostrano anche i risultati degli studi condotti sul suo utilizzo [23, 35, 26]. La sua efficacia è dovuta a vari fattori: innanzitutto il giocatore/studente ricopre, all'interno del gioco, un ruolo dall'elevata responsabilità. Ciò spinge ad affinare la propria abilità di individuare le giuste scelte da prendere nei vari contesti. Altri aspetti positivi sono sia l'immediatezza con cui sono presentati gli argomenti, mettendoli all'interno di un contesto ben delineato e soprattutto facilmente riconducibile alla realtà, sia la prontezza con cui vengono fornite le valutazioni. Inoltre la possibilità di creare scenari personalizzati, così come quello di cambiare il ruolo, permettendo quindi di affrontare gli argomenti da una nuova prospettiva, danno una profondità aggiuntiva al gameplay, offrendo ulteriori spunti di apprendimento.

Ovviamente non è esente da difetti: il gioco richiede che il giocatore abbia una discreta conoscenza degli argomenti trattati, pena la difficoltà di giocare anche negli scenari più semplici. Inoltre dal punto di vista grafico si potrebbe aumentare la chiarezza di alcuni menu ed interfacce, rese alcune volte in maniera confusionaria, tramite l'utilizzo di informazioni aggiuntive e/o messaggi di suggerimenti. Ciononostante, la praticità che lo caratterizza e l'immersione che si viene a generare dal suo utilizzo lo rendono uno strumento didattico molto valido e dall'alta efficacia.

2.2.2 Anti Phishing Phil

Anti Phishing Phil è un videogioco sviluppato nel 2007 da un team del CyLab Usable Privacy and Security Laboratory, facente parte della Carnegie Mellon University. Il gioco è stato supportato dalla US National Science Foundation per portare avanti l'iniziativa, da loro promossa, per aumentare la consapevolezza dei pericoli informatici [11]. Nel 2008 è stato acquistato dalla Wombat Security Technologies ed è tuttora da loro venduto, insieme ad altri prodotti, come strumento per il training. Nel 2009, il dipartimento di stato USA lo ha utilizzato per la formazione di oltre 55000 dipendenti [7].

La teoria presente all'interno gioco verte su un solo argomento, di cui però molto spesso si sottovaluta l'importanza: il phishing e le tecniche ad esso legate. Lo scopo è quello di educare la gente a riconoscere le trappole informatiche ed evitare di farsi ingannare, in modo da garantire la sicurezza dei dati sensibili, che siano quelli personali o aziendali. Il gioco ha un gameplay molto semplice ed intuitivo: al suo interno ci si ritrova a vestire i panni di un pesce chiamato Phil, imbattutosi in un gruppo di esche. A ogni esca è associato un URL, di cui occorre determinare la legittimità. In caso di indirizzo associato ad un sito phishing, lo si deve scartare in modo da non venire pescati e perdere le proprie vite. In caso contrario è possibile mangiare l'esca, guadagnando punti.

Il gioco è strutturato in quattro stage differenti, in ordine crescente per difficoltà, che vertono ognuno su diversi tipi di URL ingannevoli. Durante il gameplay, sono forniti al giocatore dei suggerimenti e delle indicazioni per mezzo di un altro personaggio (il padre di Phil). Per aumentare il grado di sfida, inoltre, sono inseriti all'interno di ogni scenario dei pesci nemici. Se Phil dovesse entrare in contatto con uno di loro, verrebbe mangiato e perderebbe una vita.

L'approccio allo sviluppo del videogioco si è diviso in 2 fasi, come riportato in [43]: innanzitutto gli sviluppatori hanno individuato quali argomenti erano interessati a far assimilare al giocatore. In particolare, la scelta è ricaduta su tre aspetti:

1. come identificare gli URL associati a siti di phishing;
2. dove trovare degli indizi per poter identificare se un sito web è affidabile o meno;
3. come utilizzare i motori di ricerca per individuare i siti web validi.

Una volta definiti questi concetti, per far si che gli utenti riuscissero a recepirli in maniera corretta, gli sviluppatori hanno attinto dalle scienze pedagogiche, applicando i principi di:

- **Riflessione:** ossia il processo tramite cui lo studente si focalizza mentalmente su ciò che sta apprendendo. Secondo degli studi, riportati in [43], l'apprendimento tramite Serious

Games aumenta se, durante il gameplay, sono presenti dei momenti in cui i giocatori hanno la possibilità di riflettere sulle nuove conoscenze apprese. Nel caso di Anti-Phishing Phil, il principio è applicato con la visualizzazione, alla fine di ogni livello, di un report con indicati i siti analizzati durante lo scenario, fornendo per ognuno di loro la corretta descrizione della loro natura;

- **Agenti story-based:** gli agenti sono elementi che aiutano e supportano lo studente durante il processo di apprendimento. Secondo questo principio, utilizzare degli agenti contestualizzati all'interno della storia del gioco migliora l'apprendimento. Le persone infatti tendono ad assimilare più informazioni se presentate in maniera organizzata e con dei rapporti causa-effetto, esattamente come accade quando si racconta una storia.In questo caso, Phil (il protagonista della storia e alter-ego del giocatore) rappresenta l'agente in questione;
- **Conoscenza concettuale-Conoscenza procedurale:** questi due tipi di conoscenze si influenzano a vicenda in maniera continua e danno vita ad un processo iterativo che dura per tutto il processo di apprendimento. All'interno del gioco questo viene reso tramite l'utilizzo di consigli procedurali specifici (es. "URL con numeri all'inizio sono solitamente truffe"), affiancati da conoscenze concettuali (es. spiegazioni sulle parti che formano un URL e quali sono le più importanti).

L'apprendimento del giocatore, basato sui principi appena esposti, all'atto pratico si sviluppa grazie all'ausilio di messaggi di training. Questi sono stati inseriti in quattro fasi/posti diversi all'interno del gioco:

- **Feedback durante il gameplay:** messaggi che vengono mostrati nel momento in cui si effettua una scelta (mangiare l'esca o meno) e che offrono un riscontro visivo immediato alla decisione presa;
- **Suggerimenti da parte del padre di Phil:** è possibile richiedere l'aiuto del padre di Phil durante il gameplay. Questi rispondere dando dei consigli su cosa osservare per effettuare la giusta scelta. Occasionalmente potrà mostrare i risultati di una ricerca sul web associata all'URL in questione. Con questi tipi di messaggi si forniscono le conoscenze concettuali relative all'utilizzo di un motore di ricerca e all'identificazione degli indirizzi affidabili;
- **Report di fine livello:** alla fine di ogni livello si mostra un sunto dei risultati conseguiti durante la partita. A fianco delle scelte errate sono riportate spiegazioni sul perché siano sbagliate e consigli su come identificare gli URL correttamente;
- **Tutorial tra livelli:** tra un livello e l'altro sono stati inseriti mini-tutorial per spiegare le restanti nozioni non coperte all'interno dei messaggi precedenti, soprattutto riguardo l'utilizzo dei motori di ricerca per identificare un sito inaffidabile.

All'interno di [43] è riportato anche una statistica per dimostrare l'efficacia del videogioco: sono stati condotti degli esperimenti su più di 1000 candidati in un arco temporale che va da settembre 2006 fino a febbraio 2007. I dati raccolti mostrano come l'utilizzo di Anti-Phishing Phil sia utile per migliorare l'abilità di riconoscere un sito di phishing da uno reale, soprattutto se paragonato ad altri metodi. I motivi di questi risultati, secondo le parole degli sviluppatori, sono da ricercare sia nel contenuto dei messaggi di training sia nella modalità in cui è presentato questo contenuto. È indubbio infatti che, nonostante non utilizzi un contesto realistico capace di generare un'elevata immersione nel giocatore, Anti-Phishing Phil ha il pregio di presentare gli argomenti in maniera semplice e diretta, guidando il giocatore nell'apprendimento e rendendo i concetti facilmente assimilabili da chiunque.

2.2.3 [d0x3d!]

[d0x3d!] è un Serious Game che differisce in maniera radicale dagli altri due casi appena studiati: è infatti un gioco di carte, di cui solo ultimamente è stata creata una versione per PC, disponibile sulla piattaforma digitale di Steam [44]. Sviluppato a partire dal 2012 da quattro studenti,

Zachary Peterson, Mark Gondree, Kate Lockwood e Joe Welch, è stato parzialmente supportato dal National Science Foundation, con l'assegnazione di due premi, e ha dato il via al progetto di ricerca TableTop Security, con l'obiettivo di investigare sull'utilizzo di giochi sia in classe che al di fuori per l'educazione alla sicurezza informatica [12, 47].

Come riportato in [20], l'obiettivo per cui è stato creato [d0x3d!] è quello di permettere una sempre più ampia diffusione delle discipline legate alla sicurezza informatica, cercando in particolar di modo di coinvolgere gli studenti più giovani, così da farli avvicinare fin da subito a questa disciplina, largamente ignorata nei piani di studio delle scuole primarie e secondarie. Inoltre, durante le fasi di design, al fine di aumentare le probabilità di successo, hanno puntato a rendere il gioco facilmente accessibile e giocabile (non è richiesta, infatti, alcuna conoscenza della materia da parte dei giocatori) e soprattutto lo hanno rilasciato sotto una licenza open-source, rendendolo di fatto completamente modificabile da parte di chiunque, in modo da rendere l'esperienza di apprendimento ancora più personale e coinvolgente.

Le meccaniche del gioco si basano su quelle di "Forbidden Islan", gioco da tavolo creato da Matt Leacock. I giocatori impersonano le figure di alcuni hacker e si trovano all'interno di una rete informatica (formata dalle carte) in cui devono trovare e collezionare quattro diversi asset. All'inizio del proprio è possibile intraprendere alcune azioni: compromettere un nodo, spostarsi su un nodo precedentemente compromesso e scambiare le carte con gli altri giocatori. Dopo di ciò si riceveranno delle informazioni (sotto forma di ulteriori carte) riguardanti la rete in cui ci si trova e le sue vulnerabilità. È possibile impadronirsi di un asset scartando quattro carte che lo raffigurano mentre si occupa il corrispondente punto di cattura. Alla fine del turno di tutti i giocatori, la rete si aggiusta, modificando la propria connotazione. Questo sta a simboleggiare l'intervento di un amministratore di sistema che cerca di impedire l'attacco alle proprie risorse.

Dovendo essere un gioco da utilizzare all'interno di classi di studenti, [d0x3d!] adotta un approccio di tipo collaborativo, piuttosto che competitivo: questo per aumentare la coesione tra gli studenti e rispecchiare la cooperazione necessaria per lo svolgimento di progetti e lavori di gruppo. La terminologia utilizzata al suo interno è molto semplice, ma comunque efficace. Questo per cercare di non spaventare gli studenti, creando in loro confusione ed inutili incomprensioni, e al contempo stimolare la loro curiosità.

La rete, come detto, è rappresentata tramite le carte, ognuna delle quali identifica un particolare nodo. Sebbene sia suggerita inizialmente una topologia a cui far riferimento, è possibile creare e esplorare le configurazioni più disparate. I nodi raffigurano le componenti più classiche che formano una rete aziendale, quali server DNS, serve mail, firewall, gateway di VPN. Quest'ultimi richiedono delle azioni aggiuntive da parte del giocatore, per essere violati, proprio come le loro controparti reali. Anche l'iconografia usata si muove in questa direzione, cercando di rappresentare in maniera semplice la vasta diversità dei client presenti all'interno di una rete aziendale (PC desktop, laptop, smartphone, tablet, ...).

I giocatori, in [d0x3d!], assumono possono assumere i ruoli di diversi tipi di hacker, ognuno specializzato in un tecnica di attacca: malware, social engineering, cryptoanalisi, botnet ed altri ancora. È possibile, durante il proprio turno, pescare una carta "zero-day exploit", con si può compromettere immediatamente ed occupare un nodo della rete: questa abilità, però, può essere utilizzata una sola volta durante il gioco. Sebbene possa sembrare irreale che un "zero-day exploit" permetta di attaccare qualsiasi tipo di obiettivo, senza alcuna distinzione, è stata fatta questa scelta di rappresentazione per rinforzare l'idea che e minacce sono ovunque e è possibile effettuare un attacco in maniera abbastanza semplice. Ovviamente l'utilizzo singolo permesso dal gioco richiede la pianificazione di una strategia ben congegnata, tipicamente associato agli "zero-days exploit".

La rappresentazione dei difensori, invece, è lasciata alla dinamica di adattamento della rete. Questo permette di illustrare le sfide relative ad un approccio alla sicurezza di tipo "penetrate-and-patch". Inoltre i giocatori possono meglio capire che questa tecnica non previene futuri attacchi, dato che dopo un cambiamento della rete è possibile comunque trovare nuove vulnerabilità per effettuare un nuovo tentativo di manomissione. La frequenza con cui avvengono le modifiche alla rete sono scandite da un contatore, simboleggiante la quantità di attenzioni che sta ricevendo l'infrastruttura dai suoi amministratori. Se questa diventa troppo alta, c'è la possibilità che la rete venga dismessa, causando la sconfitta di tutti i giocatori.

Anche in questo caso, sono state trovate delle limitazioni al gioco: [d0x3d!], in accordo con quanto affermato dagli stessi creatori, offre una rappresentazione della realtà a tratti troppo semplicistica. Le categorie degli asset rappresentati sono state scelte in maniera arbitraria, la rappresentazione fornita dei dati si focalizza solo su alcune proprietà di sicurezza (ignorando completamente altre), la rete è anch'essa rappresentata in maniera approssimata e viene fornita un solo approccio difensivo che è quello del "penetrate-and-patch", ignorando altre tecniche come i trusted system, meccanismi di difesa attivi o adattivi. Ciononostante, [d0x3d!] risulta un gioco molto adatto per approcciare l'argomento della sicurezza informatica, soprattutto per chi ha conoscenze limitate al riguardo. Sebbene la rappresentazione della realtà offerta è a tratti molto semplicistica, riesce a coinvolgere i giocatori, offrendo loro molti spunti di riflessione, offrendo la possibilità di iniziare un dialogo tra ciò che è stato osservato nel gioco e gli attacchi che avvengo nel mondo reale.

Capitolo 3

Sistemi SCADA

Con questo capitolo si cercherà di fornire un'analisi generale sui sistemi SCADA. Innanzitutto, nella sezione 3.1 verrà data la definizione generale di cosa sia un sistema SCADA e per quali scopi è principalmente utilizzato. Nella sezione 3.2 verrà fornita una descrizione dell'architettura generica di un sistema di questo tipo, analizzando le componenti presenti in un sistema SCADA tipo e come si affronta la loro implementazione. Infine, nella sezione 3.3, si confronterà gli SCADA con una tipologia di sistemi di controllo simili, i DCS. Verrà fornita anche una panoramica sull'evoluzione tecnologica che ha inevitabilmente influenzato anche il campo dei sistemi di controllo e come questa abbia portato alla definizione del concetto di "servizi SCADA".

3.1 Cosa è un sistema SCADA

La definizione di sistema SCADA è contenuta all'interno dello stesso acronimo che viene utilizzato per identificare un tipo particolare di sistemi di controllo industriale, o ICS (Industrial Control System). SCADA è l'abbreviazione di "Supervisory Control and Data Acquisition", ovverosia un sistema il cui scopo è di supervisione, controllo e acquisizione dei dati (funzionale ai fini dello svolgimento delle altre due operazioni). Gli scenari di utilizzo sono i più vari: dai classici sistemi di distribuzione e/o produzione energetica (impianti nucleari), ai sistemi di controllo fluidi (impianti di gestione della rete idrica/fognaria), ai sistemi di controllo del traffico, ferroviario e/o automobilistico, fino a sistemi geograficamente più contenuti, ma che necessitano comunque di controlli costanti (impianti di produzione industriale, stazioni di servizio, ecc.).

Dalla definizione data, però, non si riesce ad estrapolare cosa differenzi un sistema SCADA da un generico impianto di controllo distribuito (Distributed Control System, da qui in avanti DCS). Esistono infatti numero sistemi di controllo la cui classificazione è basata su diversi parametri, quali complessità dei processi controllati, distribuzione geografica più o meno ampia, distribuzione dell'intelligenza di controllo, tempo di reazione disponibile al verificarsi di un evento prodotto dal sistema stesso, modalità di interazione tra umani e macchinari e molti altri fattori ancora. Per capire veramente queste differenze occorre innanzitutto analizzare nel dettaglio come vengono svolti i tre compiti fondamentali (acquisizione dati, supervisione e controllo) e quali elementi, in generale, sono presenti nell'architettura di un sistema SCADA.

3.1.1 Supervisione (Supervisory)

La supervisione è la funzione principale a cui ogni sistema SCADA deve asserire (è possibile affermare che, senza supervisione, un impianto di controllo non può essere classificato come sistema SCADA). Tramite questa funzione è possibile monitorare lo stato in cui si trova il processo controllato e quale sarà la sua evoluzione. Per tale scopo, sono implementate tutte quelle funzionalità che permettono di visualizzare le informazioni che descrivono lo stato attuale del processo, così come sono utilizzate delle strutture dati al cui interno sono raccolte le cosiddette informazioni storiche, che descrivono tutti i possibili stati assumibili dal processo durante la sua evoluzione. Tutto

ciò è particolarmente utile nel momento in cui occorre identificare un eventuale stato anomalo del processo.

3.1.2 Controllo (Control)

Il controllo è la funzione che permette al sistema SCADA di prendere delle decisioni, in relazione allo stato attuale del processo controllato e alle sue future evoluzioni. Ovviamente questo compito può essere svolto in modalità differenti, ma la cui definizione dipende principalmente dal tipo di processo controllato, a seconda del quale è necessario creare un'architettura sia hardware che software specifica per il compito. Una concetto molto importante è che in un sistema SCADA il controllo del processo è concentrato per la maggior parte all'interno dell'unità di elaborazione (controllo centralizzato). In questo senso, l'elaboratore si serve del sistema di acquisizione dati per ottenere le informazioni ottenute dalla funzione di supervisione. Una volta elaborati queste dati grezzi (che non sono nient'altro che una rappresentazione dello stato del processo) se è necessario, cambia il valore dei parametri di stato che definiscono il processo di controllato, sfruttando sempre il sistema di acquisizione, ma nel senso opposto.

3.1.3 Acquisizione Dati (Data Acquisition)

L'acquisizione dei dati, sebbene sia di supporto alle precedenti funzioni, è quella che ricopre all'interno di un sistema SCADA una posizione fondamentale. È importante sottolineare che acquisizione dati, in questo contesto, sta ad identificare uno scambio bidirezionale tra l'unità di controllo e quella di supervisione. Vi sono casi in cui l'acquisizione dati è il compito principale di un sistema SCADA, poiché il controllo e la supervisione possono essere realizzate in maniera più superficiale o anche a posteriori. Esempi di questi tipo sono forniti sono i sistemi di telerilevamento, in cui si raccolgono dati che verranno, eventualmente, con modalità non sempre costanti.

In generale, però, l'acquisizione ha un ruolo funzionale allo svolgimento degli altri due compiti di un sistema SCADA. Senza scambio dei dati, infatti, l'unità di elaborazione non può avere le informazioni sullo stato attuale del processo, così come il sistema di supervisione non riceverà i valori dei parametri di controllo con cui gestire l'evoluzione dello stato del processo.

Lo scambio deve avvenire nella modalità più semplice possibile, senza alcuna manipolazione e/o elaborazione di sorta durante il percorso. Il processo di controllo avrà luogo all'interno del sistema di elaborazione. Ovviamente esistono situazioni in cui, le elevate dimensioni geografiche del processo controllato non permettono l'utilizzo di una sistema di controllo centralizzato, ma richiedono l'utilizzo di un sistema ad "intelligenza distribuita". In questo caso, però, non siamo più di fronte ad un sistema SCADA.

3.1.4 Analisi del processo da controllare

L'architettura di un sistema SCADA generalmente è sempre la stessa ed è costituita da delle componenti che si possono definire "standard". Al momento della definizione delle caratteristiche funzionali di un sistema SCADA, però, non si può ignorare il fattore rappresentato dal processo controllato. A seconda del tipo di analisi che si vuole condurre su di esso, infatti, si vanno a definire i vari parametri progettuali dell'impianto SCADA, sia quelli di tipo tecnologico che di tipo organizzativo. In questo senso, è utile classificare il processo da controllare in base a delle caratteristiche che aiuteranno poi a definire le funzioni fondamentali del sistema di controllo.

Tempo di reazione Con questo parametro si stabilisce con quanto ritardo è in grado di reagire il sistema di controllo ai cambiamenti di stato del processo durante la sua evoluzione. Solitamente, nei sistemi SCADA, si parla di reazioni "real-time", ossia che abbiano un ritardo trascurabile, anche se può capitare che questa caratteristica vada in contrasto con altri aspetti legati al sistema di controllo. Il primo tra questi è il limite dovuto alla tecnologia con cui si realizza il sistema di acquisizione dati. Un esempio di quanto detto è dato dagli impianti di controllo con grandi dimensioni geografiche dove, a causa di ciò, il trasporto delle informazioni avrà un tempo che non sarà mai non trascurabile ed andrà ad introdurre un ritardo nelle reazioni dell'intero sistema SCADA. Un'altra criticità può essere data dalla complessità dell'unità di elaborazione, spesso necessaria per soddisfare i vincoli di affidabilità e disponibilità, necessari per una corretta operatività dell'intero sistema SCADA. Per cercare di conciliare tutti i vari requisiti sono adottate diverse soluzioni, a seconda del risultato più conveniente per il caso in cui opererà il sistema SCADA:

- ridurre i tempi di reazione solo per una certa parte degli eventi generati da un processo;
- ridurre la complessità delle funzioni implementate nell'unità di elaborazione, e più in generale la complessità dell'intero sistema SCADA;
- utilizzare soluzioni tecnologiche ad hoc per il contesto in cui il sistema SCADA opererà.

Affidabilità L'affidabilità (intesa come "reliability") è un parametro altrettanto importante quando si parla di sistemi SCADA, di cui non si può fare a meno. Essendo utilizzati una grande quantità di componenti, ognuno dei quali con il proprio grado di affidabilità, al momento dell'implementazione del sistema occorre tenere conto di ognuno di questi valori. Ciò serve a stabilire le contromisure necessarie per evitare che, nel momento in cui si verifichi un malfunzionamento, questo influenzi l'intero impianto SCADA. Questo discorso vale sia per componenti di terze parti, di cui è sempre consigliabile verificare l'affidabilità dichiarata dal produttore, sia per parti auto-prodotte, per la cui realizzazione ci si dovrebbe affidare a strumenti adeguati cercando di ottenere un grado di affidabilità soddisfacente per l'utilizzo a cui è destinato.

Una considerazione da fare, quando si parla dell'affidabilità, è che, in alcuni casi, cercare di ottenere un grado elevato di questa proprietà può risultare troppo oneroso e con risultati non sempre all'altezza della cifra investita. Un eventuale guasto, infatti, potrebbe anche non pregiudicare il corretto funzionamento di un sistema di controllo, se questo avviene entro particolari condizioni. Un esempio è dato dagli impianti di rilevamento ambientale, in cui in caso di perdita di dati, si può parlare di evento trascurabile, se l'intervallo di tempo interessato dal disservizio è breve rispetto al tempo totale in cui il sistema ha lavorato.

Disponibilità Con questo parametro si indica il tempo totale in cui il sistema di controllo ha garantito il corretto funzionamento del processo controllato. Ovviamente, come nel caso dell'affidabilità, anche la disponibilità si può riferire all'intero sistema come alle singoli parti che lo compongono. In generale il soddisfacimento dei vincoli imposti dipende dal tipo di processo controllato: si possono avere situazioni con vincolo molto stringenti, così come processi per cui la disponibilità è una caratteristica che passa in secondo piano (ma comunque si avranno sempre dei vincoli da rispettare al momento dell'implementazione del sistema SCADA)

Interazione uomo-macchina Data la presenza di sistemi di controllo e supervisione, è necessario implementare anche un sistema con cui si fornisce ad un operatore la possibilità di interfacciarsi con l'impianto SCADA. Questi sono spesso indicate con la sigla HMI (Human Machine Interface), la cui complessità dipende, come per tutti gli altri parametri, dal processo controllato dal sistema SCADA. Per impianti in cui sono utilizzate procedure completamente automatizzate, basterà un'interfaccia che sia semplicemente in grado di permettere l'osservazione dei vari stati evolutivi del processo. D'altra parte, nei casi in cui, oltre all'osservazione, è richiesto all'operatore di effettuare anche operazioni di controllo, è necessario un'interfaccia che implementi tutte le funzionalità richieste per svolgere il compito assegnato, senza però andare ad inficiare sulla facilità di utilizzo, che deve rimanere in ogni caso adeguatamente alta.

In passato, vi erano casi particolari in cui era impossibile realizzare un qualsiasi tipo di interfaccia, per via delle numerose informazioni da riportare a schermo. Oggi, grazie all'utilizzo di monitor dalle grandi dimensioni o, se necessario, di "video wall',' questi casi sono sempre meno frequenti.

Dimensione geografica Anche in questo caso, la dimensione del sistema è strettamente vincolata dalla dimensione geografica in cui ha luogo lo svolgimento processo. Se, ad esempio, parliamo di un processo con un'area geografica limitata, come può essere una linea di produzione industriale o un sistema di depurazione, il sistema SCADA può essere collocato interamente in un unico edifico, solitamente lo stesso che ospita il processo da controllare. Se invece prendiamo in considerazione il controllo di un sistema di trasporto energetico, la cui dimensione è chiaramente vasta, è necessario che il sistema SCADA rispecchi la distribuzione geografica del processo stesso, in modo da rendere lo svolgimento del lavoro più semplice. Ovviamente, in questo caso, aumenta la complessità architetturale dell'impianto di controllo, in quanto si rende necessario affrontare le varie criticità che sono introdotte dalla presenza delle notevoli dimensioni in cui tutta l'operazione si svolge. Una su tutti è l'affidabilità dei sistemi di comunicazione, che deve risultare sempre molto alta, poiché un'eventuale corruzione delle informazioni trasportate potrebbe pregiudicare completamente le operazioni di controllo. Altra criticità può essere rappresentata dalla presenza di più postazioni di controllo, soluzione utilizzata in particolari casi di processi dalle elevate dimensioni geografiche. In questo caso occorre realizzare un adeguato sistema di interfacce uomo-macchina che permette l'accesso contemporaneo ai dati, senza incorrere in errori dovuti all'inconsistenza delle informazioni salvate nei database.

3.2 Architettura di un sistema SCADA

In generale, l'architettura di un sistema SCADA è composta da tre sottosistemi adibiti allo svolgimento dei rispettivi compiti di controllo, supervisione e acquisizione dati. Questi sono denominati come:

- sistema di elaborazione dati, adibito al controllo del processo;
- sistema di trasmissione dati, destinato allo svolgimento delle funzionalità di acquisizione dati;
- sistema di acquisizione, a cui è assegnato il compito di supervisione.

Come è possibile notare in figura 3.1, nella zona più periferica è presente il sistema di acquisizione, mentre quella centrale è destinata al sistema di elaborazione. La comunicazione tra questi due sottosistemi è garantita grazie al sistema di trasmissione dati.

Come per il caso del sistema SCADA in generale, l'implementazione di questi tre sottosistemi è specifica per il tipo di processo a cui sono destinati (e può risultare a volte molto complessa), ma se ne può dare comunque una definizione generica, individuando gli elementi che caratterizzano ognuno di essi.

3.2.1 Sistema di elaborazione

In 3.2 è riportato uno schema a blocchi dei componenti presenti in un elaboratore. Come è possibile vedere, in generale i vari componenti di un elaboratore possono essere raggruppati in tre macro-blocchi: quello preposto alla gestione dei dati, quello il cui compito è di garantire la disponibilità delle informazione e il blocco elaborativo vero e proprio. Di seguito verranno analizzati in maniera più dettagliata le funzionalità che caratterizzano ognuno dei blocchi.

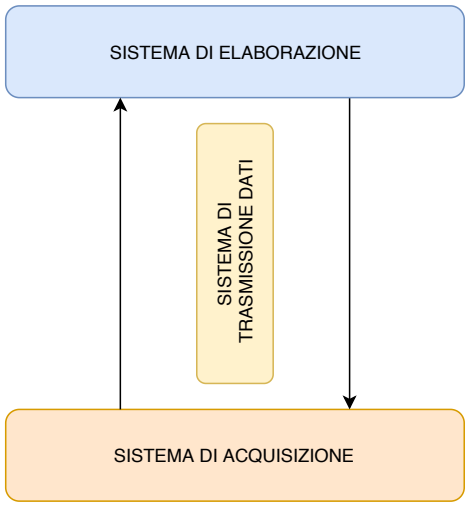

Figura 3.1. Rappresentazione grafica dell'architettura di un generico sistema SCADA

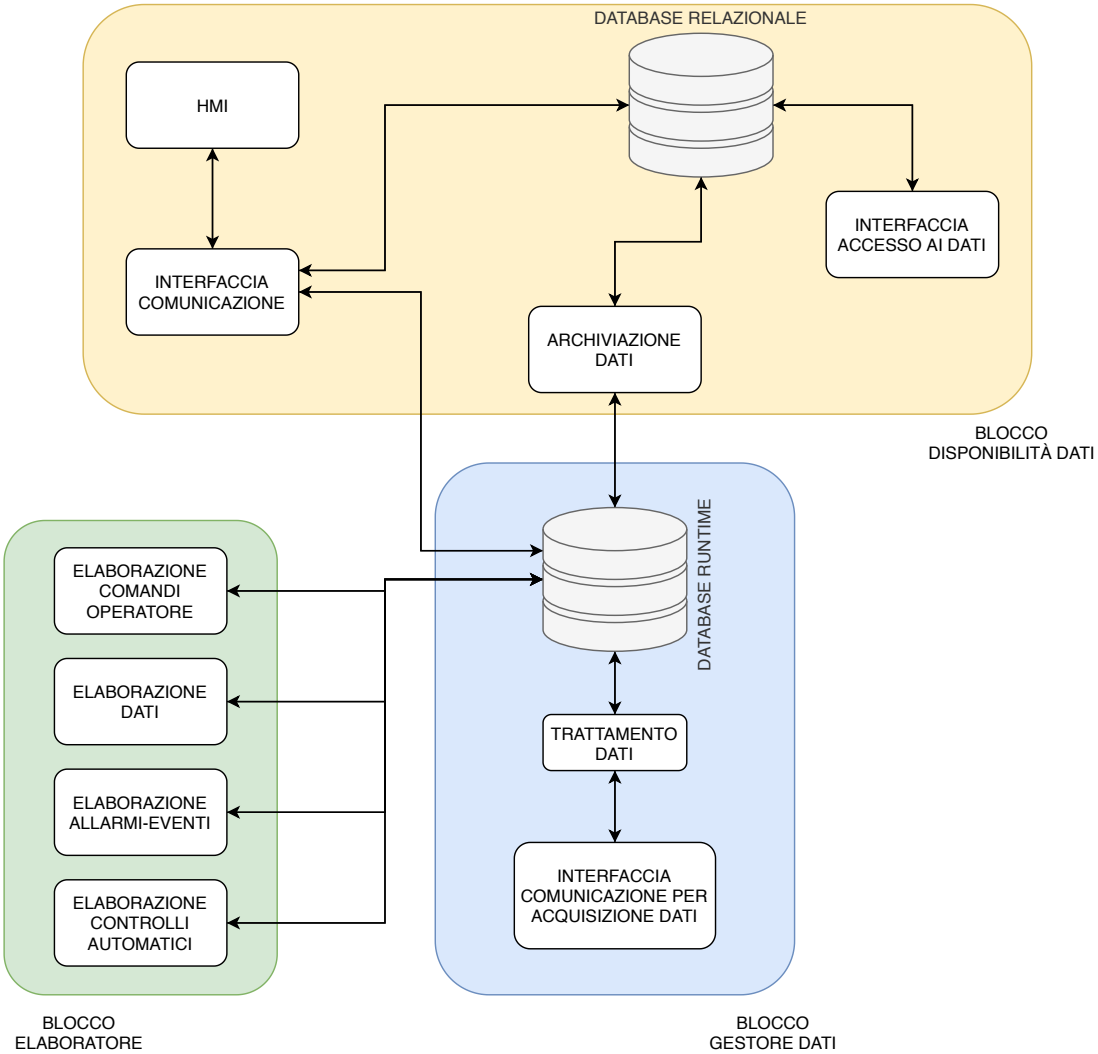

Figura 3.2. Rappresentazione grafica dell'elaboratore di un sistema SCADA

Gestore dati Il compito delle componenti per la gestione dei dati è quello di comunicare con le apparecchiature periferiche (sia per raccogliere i dati necessari all'elaborazione che per inviare i dati elaborati necessari per le azioni di controllo), trattare i dati per renderli interpretabili dal sistema e archiviare le informazioni, sia grezze che già elaborate. Tutte queste operazioni costituiscono il cuore di tutte le alte funzionalità del sistema SCADA.

I dati riguardanti lo stati del processo da controllare sono ricevuti dalle apparecchiature di controllo e sono subito tradotti nel formato di riferimento del sistema interno (ovviamente, nel momento in cui occorre comunicare con l'esterno avviene il procedimento inverso, in cui le informazioni sono tradotte nel formato adeguato affinché siano utilizzabili per effettuare le varie azioni di controllo). Il flusso di queste informazioni, sia in entrata che in uscita, ha come punto di raccolta il "database runtime", che viene chiamato in questo modo proprio per il fatto che opera in tempo reale con la dinamica evolutiva del processo controllato. È implementabile in vari modi (ad esempio, può essere costituito da aree di memoria condivisa tra i vari processi di controllo), ma l'importante è che soddisfi tutte le richieste ricevute da parte del sistema di elaborazione in tempi ridotti. Tramite ciò vengono, quindi, garantite le funzionalità di controllo e supervisione del sistema SCADA. Ovviamente, per motivi di sicurezza e per permettere lo svolgimento di azioni di supporto al controllo del processo, è consigliabile mantenere questi dati e salvarli per un eventuale utilizzo successivo. A tale scopo viene utilizzato, parallelamente al "database runtime", un "database relazionale", che rappresenta il componente principale del blocco adibito a garantire la disponibilità dei dati.

Disponibilità dati Come detto in precedenza, al fine di permettere lo svolgimento di azioni di supporto a quelle di controllo, spesso si rivela utile utilizzare un "database relazionale" in cui salvare tutti i dati, si quelli ricevuti dall'esterno che quelli elaborati dal sistema stesso. Le azioni di supporto possono essere le più svariate, dalle analisi legate ai dati descrittivi degli stati del processo, in modo da poter svolgere funzioni di correzione preventiva, alle gestione e consultazione di dati dal carattere economico. Ovviamente, in questo caso non è richiesto che il database lavori in tempo reale. Ha molta più importanza che le informazioni abbiano un'elevata intelligibilità, vi siano gli strumenti di accesso necessari per garantire la disponibilità dei dati e infine che il database riesca a gestire facilmente le grandi quantità di informazioni salvate al suo interno. Tutto ciò permette agli operatori ed ai sistemi esterni di fare accesso ai dati senza coinvolgere direttamente il sistema di controllo e supervisione.

Elaboratore Il blocco dell'elaboratore è quello preposto alla manipolazione ed interpretazione dei dati descrittori dello stato evolutivo del processo. Questo blocco, una volta ricevute le informazioni dall'esterno (fornite dal "database runtime"), effettua un'analisi su di essi ed invia come risposta una serie di comandi per il controllo del processo. Ovviamente questa analisi può essere fatta dagli operatori umani, ma spesso la grande quantità di dati da analizzare, corrispondente di solito a processi da controllare dall'elevata complessità, rendono necessario l'utilizzo di procedure di supporto. Queste producono, sostanzialmente, una serie di informazioni aggregate insieme ad una sintesi dei vari comandi da poter inviare al processo controllato. Ovviamente vi sono anche altre esigenze che questi processi elaborativi di supporto devono poter soddisfare, tra queste le più importanti sono:

- generare segnalazioni di eventuali anomale presenti nell'evoluzione del sistema;
- generare rappresentazioni sintetiche dello stato attuale e dell'evoluzione del processo;
- interpretare i comandi forniti dall'operatore;
- realizzare procedure di controllo automatiche.

Quest'ultimo punto si rende necessario nei sistemi SCADA adibiti al controllo di processi che non sono direttamente controllabili dagli operatori umani. In questo caso, il sistema compie delle azioni predefinite di controllo automatico con una cadenza periodica.

3.2.2 Sistema di acquisizione

Il sistema di acquisizione dati in un impianto SCADA rappresenta lo strumento tramite cui dialogare con l'esterno. Il suo compito principale è fare da traduttore tra il sistema centrale (l'elaboratore) e quello periferico, convertendo le informazioni analogiche, quali temperatura, pressione e

tutte le altre grandezze che descrivono lo stato del processo, in informazioni binarie. Al fine di permettere una comunicazione corretta, è necessario, oltre a stabilire un linguaggio di comunicazione unico per tutto il sistema SCADA, anche definire le modalità di comunicazione e la codifica da applicare alle informazioni scambiate. Ovviamente, le tipologie di sistemi di acquisizione possono essere delle più disparate, a seconda delle caratteristiche considerate per definire l'architettura del sistema SCADA.

Al fine di individuare l'apparato di acquisizione dati più indicato per le proprie esigenze, è utile capire ed individuare la tipologia di informazioni che l'impianto dovrà gestire. Questa analisi può essere svolta in base ai seguenti criteri:

Direzione delle informazioni Sulla base di questo criterio è possibile effettuare una duplice distinzione. Abbiamo infatti le "informazioni in ingresso", ossia i dati che riceve, sia dal sistema centrale che da quello periferico, che le "informazioni in uscita", di nuovo, informazioni che possono essere dirette o al sistema centrale o agli apparati esterni. Ovviamente, è chiaro che vi è una relazione di equivalenza tra i vari dati gestiti. Le informazioni in ingresso dal sistema centrale, infatti, sono le stesse che, opportunamente tradotte, sono dirette in uscita verso l'impianto periferico (vale, chiaramente, anche l'equivalenza opposta, ossia informazioni in ingresso dal sistema periferico diventano quelle in uscita verso l'elaboratore centrale);

Caratteristiche elettriche delle informazioni Questo criterio viene applicato quando sono prese in analisi le informazioni relative agli apparati periferici, che siano in entrata o in uscita. Affinché il sistema riesca ad interpretare le informazioni, è necessario tradurre le grandezze fisiche in un segnale elettrico adeguato. Ciò è reso possibile grazie al lavoro svolto dai trasduttori, mentre per l'operazione inversa è richiesto l'utilizzo degli attuatori. Ovviamente è necessario che tutti gli apparati di acquisizione utilizzino lo stesso tipo di rappresentazione elettrica dei dati.

Vi sono molteplici standard industriali per ognuno dei segnali considerati, che siano digitali, analogici, in ingresso o in uscita. In generale, per i dati digitali, sono utilizzati dei segnali elettrici dal voltaggio variabile (il range va dai 24 V ai 220 V) a corrente sia continua che alternata, a seconda di ciò che è più adatto allo scenario in analisi. Per le informazioni analogiche, invece, i segnali utilizzati sono differenziati sulla base delle grandezze elettriche. Abbiamo infatti:

- misure in tensione;
- misure in corrente;
- misure in resistenza;
- misure in termo resistenza;
- misure in termo coppia.

Tutte queste informazioni sono rilevanti quando si va ad eseguire l'analisi per il corretto dimensionamento del circuito elettrico, in quanto descrivono le caratteristiche elettriche, come ad esempio l'assorbimento, dei vari trasduttori e/o attuatori.

Qualità delle informazioni In questo caso, il termine "qualità" sta ad indicare quale è la tipologia dell'informazione gestita ed è necessario definirla al fine di garantire il loro corretto trattamento. È possibile eseguire questa distinzione in base a quattro macro-aree:

- **informazioni digitali:** queste informazioni sono rappresentate come un insieme di dati binari. Sono associate alle grandezze fisiche che descrivono lo stato di un processo;
- **informazioni analogiche:** questa tipologia di dati è utilizzata per rappresentare le grandezze fisiche tramite una serie di valori oscillanti all'interno di un certo range. Sono richieste delle opportune conversioni analogico/digitale affinché siano utilizzabili dall'elaboratore per svolgere le funzioni di supervisione e controllo;
- **informazioni impulsive:** questo tipo di informazione non è interpretabile in tempo reale. Piuttosto se ne richiede la conoscenza in un determinato arco temporale, in modo da ottenere una corretta rappresentazione della grandezza a cui sono associate;

- **informazioni complesse:** questo tipo di informazione è prodotto, usualmente, da dispositivi complessi con cui il sistema SCADA si interfaccia. Un esempio è fornito dai contatori elettrici di ultima generazione, che producono un'insieme di informazione relativamente a tensione, corrente, potenze, dati economici ed altro ancora, il tutto in relazione a diversi periodi temporali, più o meno lunghi. In questo caso, piuttosto che analizzare ed acquisire ognuno di quei dati in maniera indipendente come delle informazioni analogiche, è preferibile utilizzare interfacce apposite, in grado di stabilire la comunicazione con il dispositivo tramite protocolli di comunicazione ad hoc. Questo permette al sistema di acquisizione di comunicare con tutti i dispositivi esterni che utilizzano quel particolare protocollo, aumentando di fatto la compatibilità del sistema stesso. Una volta stabilita la comunicazione, questa avviene in maniera autonoma, permettendo l'acquisizione delle informazioni. Tra i vari protocolli utilizzati, si segnalano il "ModBus", il "ProfiBus", il "CanBus" ed il "LonWorks", che risultano essere quelli più diffusi.

A livello pratico, questa analisi si traduce nei parametri di programmazioni da applicare ai PLC, "Programmable Logic Controller". Questi sono dei veri e propri computer componibili, la cui struttura hardware è adattata al processo da controllare. Il loro compito è gestire i segnali digitali ed analogici che transitano nella rete costituita dai sensori, gli attuatori e il sistema di elaborazione centrale. Negli ultimi anni, grazie anche alle progressive migliorie tecnologiche che ne hanno permesso una riduzione delle componenti fisiche e, conseguentemente, dei costi, si è cominciato ad utilizzare i PLC anche in ambiti domestici. Un esempio è la loro applicazione nei quadri elettrici delle abitazioni per gestire automaticamente i vari impianti presenti nelle case: riscaldamento, irrigazione, rete internet, ecc.

3.2.3 Sistema di trasmissione dati

I componenti sopra elencati necessitano di interfacce adeguatamente implementate per comunicare correttamente tra di loro. In un sistema SCADA occorre garantire la comunicazione tra:

- sistema di elaborazione e sistema di acquisizione dati;
- sistema di elaborazione e sistema di gestione dati;
- processo controllato e dispositivi di interazione (attuatori);
- dispositivi di interazione e sistema di acquisizione dati.

Può, inoltre, rivelarsi utile implementare delle interfacce per comunicare con altri elementi esterni, quali sistemi gestionali dell'azienda o sistemi informativi in generale. Ognuna di queste interfacce dovrà essere implementata secondo le caratteristiche più adeguate per lo scopo a cui è destinata. Il rischio è quello di compromettere le normali funzionalità del sistema SCADA o, addirittura, la sua realizzabilità. Inoltre, in fase di progettazione, è consigliabile tenere conto anche dei possibili sviluppi futuri a cui potrebbe essere soggetto il sistema SCADA. Di seguito sono riportate le caratteristiche su cui basare l'analisi dei protocolli da applicare alle interfacce comunicative che si vogliono implementare.

Velocità Uno degli aspetti più cruciali dei canali di comunicazione è quello di garantire una velocità sufficientemente elevata, tale da rendere possibile che l'azione di controllo del processo avvenga in tempi ridotti ed adeguati. I vincoli imposti, in questo senso, sono spesso molto restringenti e ciò può costringere all'impiego di sistemi periferici ad intelligenza distribuita per compiere le azioni di controllo. Alcuni dei casi in cui questi problemi si verificano con costanza sono i sistemi SCADA con notevoli dimensioni geografiche. In questo caso occorre utilizzare le infrastrutture di comunicazione dei gestori telefonici, che normalmente sono destinate ad un impiego "general purpose". Ciò può rendere il servizio non sufficientemente adeguato per lo scopo finale e occorre, quindi, ricorrere a sistemi periferici con intelligenza di controllo distribuita, in modo da svolgere le azioni nei tempi richiesti.

Nel caso della comunicazione tra il sistema e le HMI, invece, la comunicazione deve avvenire sempre in "real-time", in modo da rendere più efficace il lavoro degli operatorio. Questo deve avvenire sia per la visualizzazione dei parametri descrittivi dello stato del processo, che per la risposta alle azioni di controllo attuate dagli operatori.

Infine, la comunicazione con i sistemi esterni è vincolata dal tipo di sistema con cui occorre interfacciarsi. Se è anch'esso un sistema di controllo, allora i vincoli sono gli stessi visti nel caso di comunicazione con il processo, con annesse limitazioni. Se, d'altra parte, si vuole comunicare con un sistema non di controllo, l'interfaccia da implementare non presenta delle particolari restrizioni da rispettare, anzi possono essere considerate come trascurabili. È importante sottolineare come, di solito, non sia possibile soddisfare tutti i requisiti imposti dal tipo di comunicazione che si vuole implementare, scendendo di fatto ad un compromesso tra protocolli implementati e tecnologia utilizzata.

Sicurezza La sicurezza è un aspetto altrettanto importante da considerare al pari della velocità, soprattutto nel caso in cui le probabilità di intrusione da parte di soggetti indesiderati siano abbastanza alte. Chiaramente, nel caso di un sistema chiuso, i tentativi di intrusione a cui si è soggetti diventano esigui, ma non bisogna scordare che è sempre presente la possibilità di un errore umano da parte degli operatori. Si rende necessario, quindi, ricorrere ai ripari preventivamente, cercando di evitare il presentarsi di queste spiacevoli situazioni, che siano causati da attacchi intenzionali o errori in buona fede.

La gestione della sicurezza deve riguardare sia le comunicazione tra elaboratore e sistema periferico di acquisizione dati, sia tra un sistema SCADA ed un altro, in quanto i entrambi i casi l'alterazione delle informazioni trasportate può provocare un comportamento anomalo da parte dell'impianto di controllo.

Tra le soluzioni adottabili, la più gettonata è la separazione delle diverse aree di lavoro accessibili al sistema ed agli operatori. Questa separazione può avvenire sia fisicamente, così come dal punto di vista dell'implementazione logica, ed in generale è dipendete dal tipo di tecnologia implementata per l'interfaccia di comunicazione.

In ogni caso, il passo più importante da compiere è quello di definire un'adeguata politica di sicurezza fin dalle fasi progettuali, cosa che negli ultimi anni è diventata imprescindibile, ma che nel passato era trattato con molta superficialità. Questo perché mentre prima i sistemi SCADA erano isolati completamente dal mondo esterno, oggi fanno largo impiego di tecnologie comunicative pubbliche, dal basso costo, ma dalla sicurezza più debole. Si è reso necessario, quindi, un cambio di approccio al tema della sicurezza.

Intelligibilità L'intelligibilità è un parametro importante nel momento in cui si vuole realizzare un sistema che interagisca costantemente con apparecchiature esterne per la supervisione ed il controllo. In questo caso è possibile applicare diverse soluzioni, ma le più adatte sono quelle che fanno riferimento a degli standard comunicativi predefiniti. Ciò è facilmente comprensibile sia dal punto di vista tecnologico, funzionale ed economico.

Tecnologico e funzionale perché un protocollo proprietario (ossia non standard) non è detto che sia utilizzato dai dispositivi con cui si vuole comunicare, restringendo quindi la gamma di apparecchi utilizzabili. Inoltre, non è detto che tra quelli effettivamente utilizzabili vi sia il dispositivo adatto a soddisfare tutte le esigenze di comunicazione richieste. Infine, economicamente parlando, un protocollo standardizzato ha dietro di sé il supporto di un'intera comunità scientifica, mentre nel caso di uno proprietario si è costretti a sottostare alle esigenze di mercato dell'azienda padrona del protocollo.

Affidabilità In generale è richiesto che i dati trasmessi all'elaboratore del sistema SCADA mantengano un alto gradi di integrità. Ciò per rendere possibile una corretta valutazione dello stato evolutivo del processo. Una soluzione potrebbe essere quella di applicare meccanismi di validazione dati nei dispositivi periferici, ma ciò rallenterebbe la risposta generale del sistema all'evoluzione del processo. L'unica opzione rimanente è integrare questi meccanismi nel canale di comunicazione, in modo da riuscire a garantire l'integrità delle informazioni trasportate.

La soluzione ideale per rispondere a questo problema prevede l'implementazione di tre diverse funzionalità:

- rilevazione degli errori;
- richiesta di ritrasmissione in caso di errori rilevati;
- ordinamento delle informazione all'interno del flusso dati.

Questi meccanismi sono implementabili nelle maniere più differenti, facendo ricorso agli algoritmi che più si ritengono adatti. Purtroppo, però, l'introduzione di queste funzioni va ad intaccare la velocità di comunicazione, richiedendo, quindi, il raggiungimento di un compromesso in grado di garantire un adeguato rapporto tra la rapidità di scambio delle informazioni e il mantenimento della loro integrità. Inoltre, per ottimizzare l'efficacia della soluzione adottata, è consigliabile lo svolgimento di un'attenta analisi del sistema di comunicazione in questione. È ovvio infatti che nel caso in cui la comunicazione avvenga su di un mezzo di per sé già affidabile, come la fibra ottica, si preferisce evitare l'implementazione di tecnologie per gestire gli errori di trasmissione, in quanto sarebbe praticamente inutilizzato. Se, invece, la comunicazione avviene su di un mezzo meno affidabile, come dei trasmettitori wireless, è fortemente consigliato l'impiego di funzionalità di rilevamento e correzione degli errori.

Infine, nel caso in cui la comunicazione da stabilire sia tra più sistemi SCADA, l'aspetto che si prende più in considerazione è l'ottimizzazione dei costi di trasmissione, in quanto, come detto, sono presenti molti meno vincoli sulla velocità e sull'affidabilità dei dati scambiati.

Disponibilità In stretta correlazione con l'aspetto dell'affidabilità vi è quello della disponibilità dei dati trasmessi. In casi di disservizio del sistema di comunicazione, infatti, anche le operazioni di controllo, direttamente collegate alle comunicazioni, rischiano di subire dei malfunzionamenti. Occorre, quindi, garantire la continuità della disponibilità delle informazioni in quanti necessaria per svolgere correttamente le attività di controllo (per le quali è fondamentale conoscere in tempo reale lo stato del processo da controllare o, nel caso di attuazione di una politica di controllo, occorre prontamente informare l'attuatore dell'azione che dovrà intraprendere).

Le soluzioni più adatte partono tutte dall'assunzione di un principio molto importante: anche un sistema ad alta disponibilità può interrompere il suo normale servizio per un guasto. In questi casi, per prevenire e combattere il verificarsi di una situazione del genere, si richiede l'utilizzo di protocolli comunicativi opportunamente scelti e la realizzazione di sistemi ridonanti, in cui i dispositivi restano a riposo fino al verificarsi di un guasto. In quel momento saranno prontamente messi in azione per sostituire le componenti ordinarie fuori servizio. Questa soluzione è adottata per tutti i dispositivi presenti nel canale di comunicazione.

Supporto dei servizi Un'ulteriore aspetto da considerare nell'analisi del sistema di comunicazione è la tipologia delle informazione scambiate. È stato dimostrato, infatti, che a parità di tipo di dati trasmesso, diverse tecnologie e protocolli offrono prestazioni diverse da loro. Occorre, quindi, analizzare se l'interfaccia scelta è adeguata a garantire una qualità di comunicazione sufficientemente elevata per il tipo di dato che si vuole trasmettere.

3.3 Differenze tra sistemi DCS e sistemi SCADA

Le componenti appena analizzate non sono una prerogativa dei soli sistemi SCADA, ma sono utilizzati nei vari sistemi di controllo, tra cui i DCS. Ciò che li differenzia, quindi, non è quali strumenti sono utilizzati, ma piuttosto come vengono implementati. In particolar modo la distinzione è sul grado di distribuzione dell'intelligenza di controllo. I DCS, acronimo che sta ad indicare i "Ditributed Control System", come suggerisce già il nome stesso, sono basati sull'impiego di strutture di acquisizione dati, ma con anche un'elevata capacità elaborative, creando di fatto un paradigma tecnologico contiguo delle funzioni di controllo e acquisizione, Negli SCADA, invece, come abbiamo visto, le due funzioni sono ben distinte ed affidate ad impianti separati, fisicamente e tecnologicamente.

Nel caso DCS, quindi, non si parla di apparecchiature di acquisizione, ma di unità di elaborazione periferiche, in grado non solo di ricevere dati, ma anche di interpretarli, analizzarli per

individuare lo stato attuale in cui il processo si trova e, se necessario, eseguire delle azioni di controllo sul processo. LA complessità architetturale di queste unità periferiche, così come le funzioni che sono in grado di svolgere, è basata sul tipo di processo che si deve controllare. L'unità centrale di elaborazione, nel contesto di un sistema DCS, ha quindi il compito di acquisire sia informazioni grezze che informazioni già elaborate che danno indicazioni sullo stato delle strutture di controllo.

Osservando, però, l'evoluzione tecnologica avvenuta negli ultimi anni che ha interessato anche l'ambiente dei sistemi di controllo, è possibile fare un riflessione sulla necessità di mantenere o meno questa distinzione tra sistemi DCS e SCADA. Inizialmente, la distinzione era dettata dalle differenti caratteristiche tecnologiche che venivano implementate, spesso frutto di scelte obbligatorie per risolvere un particolare problema di controllo. Con lo sviluppo delle infrastrutture di comunicazione e delle tecnologie computazionali la scelta tra un sistema di elaborazione centralizzato con periferiche di acquisizione pure e un sistema con controllo ed elaborazione distribuito anche nelle apparecchiature di acquisizione è diventata sempre più una questione legata a fattori quali scalabilità dell'impianto, grado di manutenzione da garantire e altre caratteristiche non inerenti alla effettiva realizzabilità del sistema. A tal proposito, si prevede che a causa del continuo progresso tecnologico, la distinzione tra sistemi DCS e SCADA si farà sempre più assottigliata fino a sfociare nella creazione di un'unica categoria di classificazione, di cui entrambi faranno parte.

3.3.1 Evoluzione tecnologica dei sistemi SCADA: integrazione con sistemi informativi aziendali

L'evoluzione tecnologica che ha colpito e continua ad interessare i sistemi SCADA ha portato a due principali conseguenze:la prima è stata, come detto, la diminuzione del divario tra la categoria SCADA e quella dei DCS, creando dei sistemi ibridi in cui sono integrate funzionalità di entrambe le parti. La seconda, altrettanto importante, è stata la definizione di nuove architetture con lo scopo di creare non più una paradigma che descriva un sistema fisico, ma piuttosto una funzionalità. In tal senso, è utile vedere le diverse linee evolutive che sono state applicate per realizzare questa idea.

La prima è stata quella che permettesse l'integrazione tra un sistema SCADA con uno gestionale concorrente. In questo modo è possibile interagire direttamente con un sistema informatico aziendale, con il fine di fornire supporto automatizzando la gestione interna, sia organizzativa che contabile. In figura 3.3 è possibile vedere un esempio di questo caso. Come è possibile osservare,le varie modalità di comunicazione del sistema SCADA convergono tutte nel sistema informatico aziendale.

Una volta progettata l'architettura, con le entità che ne faranno parte, e le comunicazioni tra di esse, occorre analizzare di quali informazione il sistema aziendale ha bisogno per svolgere correttamente le sue funzioni. Il supporto che deve fornire, infatti, è ottimale solo se il sistema SCADA è in grado di fornire le informazioni corrette sullo stato del processo controllato. Un esempio pratico di quanto descritto può essere dato da un sistema SCADA integrato con il sistema gestionale di un magazzino. In questa situazione ipotetica, le decisioni su cosa ordinare e quanto sono prese basandosi sulle informazioni che sono fornite dal sistema SCADA riguardo lo stato del processo controllato. Se queste non fossero corrette, vi sarebbero delle decisioni prese in maniera errata che alla lunga porterebbero ad un danneggiamento economico alla stessa azienda. Questo elencato è solo uno dei molteplici casi in cui è possibile effettuare l'integrazione tra sistema aziendale e quello SCADA, automatizzando la gestione interna. Si possono fare considerazioni simili anche nel caso in cui ci si trova a fornire dei servizi tecnologici tramite web.

Un'altra linea evolutiva scaturita dal processo tecnologico riguarda lo sviluppo di sistema SCADA con larga estensione geografica. In questo caso, un grosso aiuto è stato dato dallo sviluppo delle comunicazioni e le tecnologie legate ad esse. Grazie alle reti ethernet e la tecnologia TCP/IP è stato possibile creare delle infrastrutture di comunicazione sempre più affidabili e soprattutto facilmente modificabili secondo le proprie esigenze. Queste, come detto, sono largamente utilizzate nelle strutture di comunicazione di un sistema SCADA. Inoltre si stanno integrando anche funzionalità legate alla comunicazione radio: sempre più spesso si utilizzano comunicazione wireless per le trasmissioni locali, così come molti sistemi di controllo utilizzano la rete cellulare sia

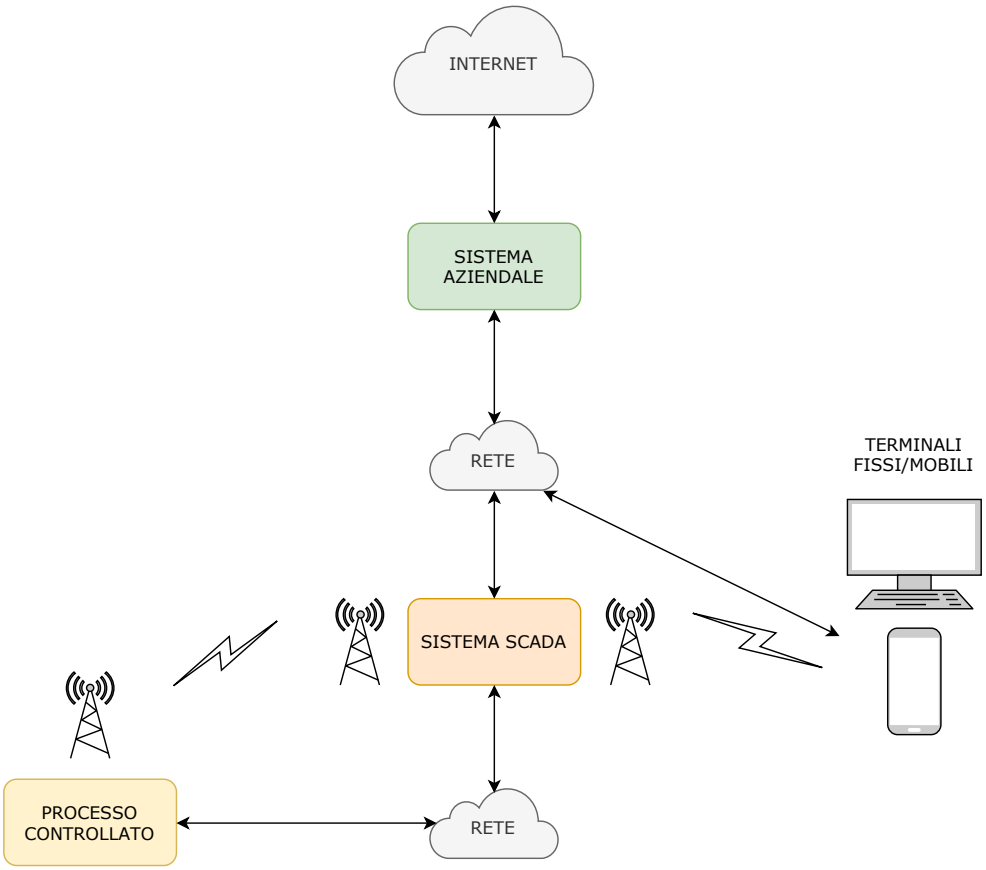

Figura 3.3. Architettura di un sistema SCADA integrato con un sistema aziendale

per comunicare con strutture difficilmente raggiungibili da una rete cablata che per avvisare gli stessi operatori nel caso in cui ci sia bisogno di un intervento urgente.

3.3.2 Un nuovo paradigma: la creazione di servizi SCADA

Una delle ultime evoluzioni scaturite dal miglioramento tecnologico è stato un cambio di approccio alla soluzione del problema della supervisione. Tradizionalmente, come visto, sono realizzati dei sistemi di controllo ad hoc la cui gestione è a carico degli operatori responsabili del processo controllato. Questa soluzione prevede prima una fase di analisi per individuare le esigenze di controllo, definire successivamente i requisiti del sistema e la sua architettura per poi, alla fine, passare alla fase di installazione hardware ed implementazione software. Il tutto chiaramente comporta un lavoro molto lungo e oneroso.

Negli ultimi tempi, invece, si è applicata un diverso approccio per la ricerca della soluzione al problema sopra citato: il focus è stato spostato sulla realizzazione di servizi SCADA che rispondano a esigenze di tipo funzionale. In questo modo il servizio coinvolge due tipologie di operatori, in maniera diversa. Chi crea il servizio, ossia il fornitore, gestisce e controlla direttamente il sistema SCADA e dovrà occuparsi dei problemi architetturali e progettuali. Il committente, ovverosia il fruitore del servizio, non sarà coinvolto nella realizzazione dell'impianto, ma avrà il compito di gestire i problemi organizzativi nati dall'introduzione del sistema di controllo nella rete aziendale. Una configurazione tipica di un servizio SCADA è mostrata nella figura 3.4, dove è possibile notare la separazione tra il fornitore ed il fruitore del servizio stesso.

Come si può notare dallo schema riportato, il sistema di elaborazione viene gestito dal fornitore. Questo sarà messo in comunicazione con le restanti componenti tramite una rete realizzata nelle vicinanze del processo ed accessibile tramite dei canali di comunicazioni simili a quelli utilizzati

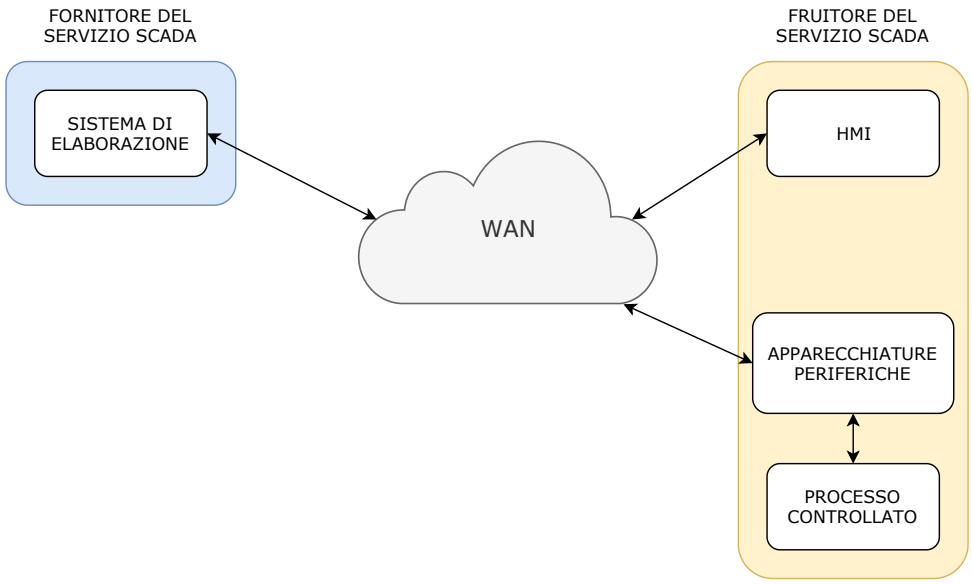

Figura 3.4. Architettura di un servizio SCADA

nei sistemi distribuiti. Il fruitore potrà accedere alle funzionalità di supervisione e controllo del sistema SCADA tramite le interfacce HMI messe a disposizione dal fornitore.

Ovviamente, occorre tenere anche in considerazione gli aspetti legati all'affidabilità del servizio, alle prestazione e, soprattutto, alla sicurezza. Se prima ci trovavamo spesso al cospetto di un sistema chiuso, creato ad hoc per rispettare le proprie esigenze, ora l'adozione di un servizio SCADA comporta il condividere molti dati con una terza parte esterna, rappresentata in questo caso dal fornitore. Questi dati, molto spesso, sono cruciali poiché permettono la realizzazione del sistema di controllo. Una loro eventuale perdita o manomissione comporterebbe dei danni sia economici che operativi molto elevati.

Capitolo 4

Analisi dei rischi in un sistema SCADA

In questo capitolo verrà analizzata la sicurezza di un sistema di controllo SCADA e di come i paradigmi di difesa siano cambiati con il progredire delle tecnologie utilizzate. Nella sezione 4.1 si provvederà a dare una panoramica sui concetti di sicurezza informatica che hanno particolare rilievo nei sistemi industriali e quali siano le differenze con la sicurezza IT domestica. Nella sezione 4.2, invece, verrà trattato l'argomento dei dispositivi mobile utilizzati in ambiente SCADA e di come occorre preoccuparsi anche della sicurezza di questi dispositivi, molto spesso sottovalutata.

Successivamente, nella seconda parte del capitolo sarà affrontato l'argomento degli scenari di attacco che possono presentarsi: nella sezione 4.3 verranno mostrate una serie di minacce, analizzando le loro modalità di impiego e quali difese è possibile adottare contro di esse. Infine, nella sezione 4.4.1, verrà effettuato il case study di una delle minacce più pericolose nell'ambiente SCADA: il worm Stuxnet con cui, nel 2010, è stato effettuato uno dei più grandi attacchi ai danni dei sistemi di controllo delle centrali nucleari iraniane da parte del governo USA [51].

4.1 Sicurezza informatica in ambito industriale

Come visto nel capitolo precedente, l'evoluzione tecnologica che ha investito i sistemi di controllo industriali ha portato all'utilizzo di numerose tecnologie IT, sia per questioni di efficienza funzionale che economica. Questa scelta ha comportato non solo la presenza di aspetti positivi, ma anche l'introduzione di tutte le criticità tipiche del mondo IT. Di conseguenza, è aumentata l'apprensione per i temi di sicurezza, quali accessi non autorizzati, dati manomessi/perduti e attacchi alla rete o al sistema con conseguente malfunzionamento degli stessi; in altre parole tutte le criticità che precedentemente avevano poca importanza poiché ci si trovava molto spesso in un sistema chiuso non accessibile dall'esterno, ora assumono una rilevanza molto alta.

Tradizionalmente, per l'ambiente domestico, esistono molte soluzioni ai suddetti problemi IT, tutte costantemente testate e modificate per migliorare la loro efficienza. Purtroppo, però, queste soluzioni non possono essere applicate nello stesso modo agli ambienti industriali, in quanto di origine tecnologica diversa e soprattutto le criticità da considerare sono diverse. Per questo motivo, nel 2002, fu istituito il comitato dell'"International Society for Automation" (ISA) formato da esperti di sicurezza informatica industriale, con l'intento di stabilire delle linee guida per indicare quali metodi di difesa e "best practices" sono da adottare per mantenere la sicurezza dei sistemi di controllo industriale.

Il comitato stilò, come prima cosa, una lista al cui interno sono state raccolte le principali preoccupazioni che possono scaturire da un attacco informatico ad un sistema di controllo:

- danneggiamento della sicurezza pubblica e/o personale;

- perdita di fiducia da parte del pubblico;
- violazione di requisiti normativi;
- perdita di informazione proprietarie e/o confidenziali;
- perdita economica; item impatti sulla sicurezza nazionale.

Finalmente, nel 2007, il comitato rilasciò la prima serie di schemi di certificazioni che furono pubblicati dall'"American National Standards Institute" (ANSI) con il nome di protocollo ANSI/ISA99. Questo è stato ampliato e aggiornato nel coso degli anni fino al 2010, anno in cui il nome fu cambiato in ISA/IEC-62443. Il cambio fu effettuato per mantenere coerenza nella numerazione dei protocolli rilasciato dall'ANSI e il corrispettivo standard dell'"International Electrotechnical Commission". Come è possibile notare dalla figura 4.1, i report tecnici che compongono lo standard ISA/IEC-62443 sono suddivise in quattro macro categorie:

1. la prima (partendo dall'alto) è la categoria che include le informazioni di base, quali terminologie, modelli e concetti fondamentali;
2. la seconda è rivolta ai proprietari degli asset. Sono affrontati vari aspetti legati alla creazione e mantenimento di programmi per la sicurezza;
3. il terzo gruppo include una serie di documenti che tracciano delle linee guida per la progettazione dei sistemi di difesa. Sono inclusi anche dei modelli di riferimento per il design del sistema;
4. l'ultima categoria descrive lo sviluppo ed i requisiti tecnici da rispettare per i sistemi di sicurezza degli impianti di controllo.

C'è da sottolineare che l'intero standard ISA/IEC-62443 è sottoposto periodicamente a revisioni ed aggiornamenti, sia nella sua interezza che solamente in alcuni dei suoi moduli. Chiaramente le nozioni basilari restano invariate nel tempo, sono integrate da informazioni aggiornate ed in linea con il progresso tecnologico. È utile quindi vedere quali siano queste nozioni e come si riflettano ad un livello più funzionale e pragmatico.

In generale, dai moduli ISA/IEC-62443, si evince l'esistenza di vulnerabilità comuni a tutti i sistemi di controllo e possono essere sia di natura strumentale che organizzativa. La prima, una delle più importanti, è la sottovalutazione delle politiche di sicurezza. Molto spesso si registrano casi di attacchi in cui il motivo primario del problema è stato il mancato rispetto, o addirittura l'assenza, di comuni norme di sicurezza. Questo tema e i problemi che porta con sé, purtroppo, vengono frequentemente trattati come meno importanza di quanto ne richiedano e alle volte non si rileva la presenza neanche degli strumenti organizzativi basilari, quali le documentazioni dei sistemi o delle reti. Anche nel caso in cui siano presenti, spesso non sono aggiornati. Inoltre il monitoraggio delle attività di accesso o di registrazione degli incidenti non sono adeguatamente supportate, creando conseguenti difficoltà nel fornire i dati relativi all'affidabilità della rete e del sistema di controllo. Altri fattori organizzativi critici sono l'assenza di procedure di "disaster recovery", backup dei dati o delle configurazioni del sistema e, più in generale, piani per la gestione delle emergenze.

Da un punto di vista architetturale, invece, la vulnerabilità più cruciale è la segmentazione e separazione della rete informatica aziendale da quella industriale di controllo del processo (a tale scopo si è cercato di fornire un supporto con il protocollo ISA/IEC-62443 tramite schemi di architetture di riferimento, come il modello PERA rappresentato fig. 4.2). Il problema di una rete informatica piatta, ossia non adeguatamente segmentata, è che semplifica estremamente la circolazione di una minaccia al suo interno, per finire con il suo arrivo fino ai PLC. Ciò significa che, in caso di un attacco positivo sapientemente programmato, le conseguenze derivanti potrebbero mettere a rischio la sicurezza non solo degli asset aziendali, ma anche l'incolumità fisica di oggetti e/o persone (i PLC, infatti, si trovano a ridosso dei macchinari controllati ed una loro manomissione causerebbe il malfunzionamento dell'intero sistema di produzione).

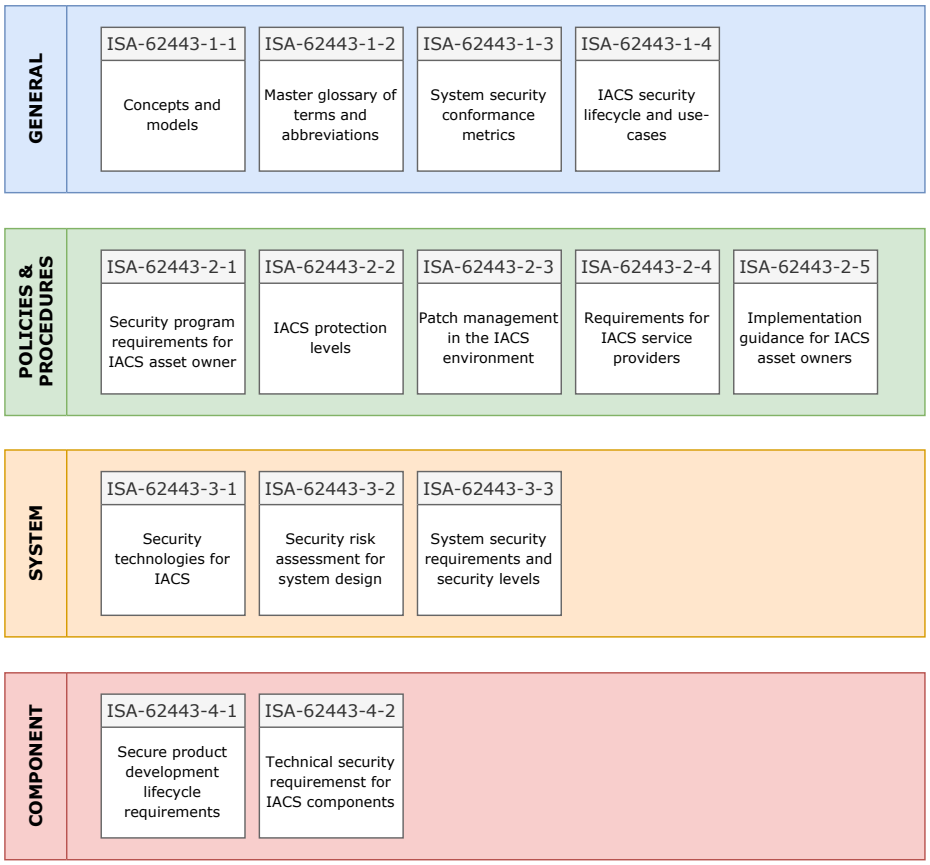

Figura 4.1. Schema dei moduli della famiglia ISA/IEC-62443

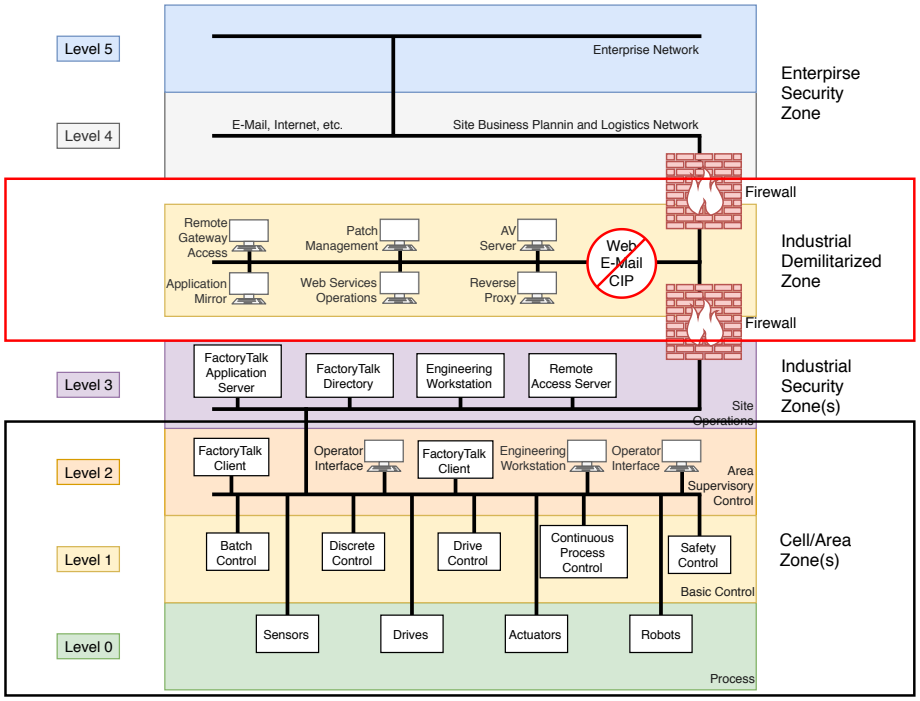

Figura 4.2. Purdue Enterprise Reference Architecture (PERA), modello concettuale per la segmentazione della rete di un ICS

Altre vulnerabilità architetturali sono anche fornite dall'assenza o il mancato aggiornamento di comuni software di sicurezza, come antivirus o strumenti di rilevamento dei malware, dettati spesso dall'incompatibilità tra questi programmi e i software real-time utilizzati nei sistemi di acquisizione dati. Inoltre, capita che i computer su cui operano gli applicativi di controllo non siano correttamente configurati, mantenendo configurazioni standard facilmente attaccabili, o installano sistemi operativi a cui non è stata fatta alcuna operazione di hardening o, per lo meno, di aggiornamento. Infine, altra vulnerabilità è data dalla mancanza di monitoraggio degli accessi remoti: negli ultimi tempi, infatti, l'utilizzo di dispositivi mobile per gestire questi sistemi si è ampiamente diffuso grazie alla comodità che li contraddistingue. Questo, però, ha portato con sé anche numerose criticità che se non correttamente affrontate possono rappresentare delle vulnerabilità alla sicurezza del sistema.

4.1.1 Differenze tra sicurezza IT e ICS

Come precedentemente detto, sebbene le tecnologie applicate ai sistemi ICS sia derivante da quella dei sistemi IT domestici, le soluzioni adottate per la sicurezza devono essere differenti a cause delle differenze dovute all'ambiente di lavoro ed alle differenti esigenze. Mentre nel caso della sicurezza IT le proprietà di confidenzialità e riservatezza dei dati ha maggiore importanza rispetto alla disponibilità, nei sistema di controllo, data la loro natura operativa continua, è necessario porre la disponibilità e l'integrità dei dati in cima alle proprietà da mantenere, anche quando ci si trova sotto attacco. Il rischio, in caso contrario, è di causare gravi perdite economiche o, peggio, danni materiali al sistema o mettere in pericolo l'incolumità degli operatori del sistema di controllo [25].

Di seguito sarà esposta una lista di differenze, evidenziate anche dal comitato ISA99 al momento dello studio del nuovo protocollo di sicurezza, tra sistemi di sicurezza ICS e quelli IT [45].

Requisiti operativi e temporali Di norma, negli ambienti ICS i tempi delle comunicazioni sono fondamentali: devono essere effettuate il più possibile in real-time, con le soglie dei livelli di tolleranza dei ritardi e disturbi ammessi dettate dallo scopo a cui sono destinate. In alcuni casi, le risposte del sistema alle interazioni degli operatori sono così critiche che si utilizzano sistemi operativi specifici per lo scopo denominati real-time operating system, o RTOS (ovviamente l'unità di misura per il real-time dipende molto dalle applicazioni a cui è destinato il sistema ed è di norma specificata in fase di progettazione dell'ICS). Molti sistemi effettuano trasmissioni rapide e ripetute, quindi ha molta importanza l'affidabilità del mezzo di comunicazione, mentre non si rende necessaria una larghezza di banda estesa. Nelle comunicazioni IT, invece, si hanno necessità diametralmente opposte: il real-time è un'esigenza che può essere messa in secondo piani (disturbi e ritardi nelle comunicazioni hanno soglie di tolleranza molto più alte), mentre è assolutamente primaria la necessità di una banda di comunicazione sufficientemente larga per trasportare tutte le comunicazioni del sistema.

Requisiti di disponibilità Molti sistemi ICS, come precedentemente detto, sono in esecuzione ininterrottamente. Interruzioni improvvise del sistema, quindi, non sono accettabili e spesso devono essere pianificati con largo anticipo (giorni o, addirittura, settimane prima). Inoltre è fondamentale effettuare numerosi test intensivi prima di eseguire le operazioni di interruzione, in modo da garantire la corretta affidabilità del sistema durante tutto il processo. È quindi facilmente intuibile come le tipiche strategie del mondo IT, quali il riavvio di un componente in caso di malfunzionamenti, non possano essere attuate nell'ambiente ICS, dato l'enorme impatto negativo che avrebbe sulla intero sistema di controllo. Spesso, infatti, è più importante mantenere la disponibilità del processo controllato, rispetto alle informazioni su cui si basa la produzione. A tale scopo, si è già visto come in molti ICS l'architettura sia ridondante, per garantirne la funzionalità anche in caso di componenti non disponibili.

Requisiti di gestione dei rischi In un sistema IT, in caso di minaccia andato a segno, le preoccupazioni maggiori sono rivolte all'integrità e la riservatezza dei dati colpiti dall'attacco.

In un ambiente del genere, questi rappresentano l'asset principale ed una loro manomissione si tramuterebbe in gravi danni economici e d'immagine per l'azienda attaccata. In un sistema ICS, invece, le conseguenze sono del tutto imprevedibili e portare alle situazioni più disparate: si va dalle perdite di materiali e proprietà intellettuali fino al provocare danni ben più gravi che posso mettere a rischio la salute degli operatori o, in caso estremi, della salute pubblica. Quindi, chi si occupa dell'implementazione dei metodi di sicurezza di un impianto ICS deve tenere bene a mente l'importanza del legame esistente tra sicurezza del sistema e sicurezza delle persone. Qualsiasi rimedio che non ne tenga conto è da considerarsi inaccettabile.

Effetti fisici Come detto, i PLC interagiscono direttamente e fisicamente con il processo da controllare. Spesso le interazioni possono essere molto complesse e generare dei particolari eventi fisici. Questo rappresenta, ovviamente, un potenziale pericolo che in un ambiente IT non sia ha per niente. Per gestire questi eventi fisici è necessario che gli operatori del sistema di controllo comunichino costantemente con gli esperti del campo fisico in questione.

Sistemi operativi I sistemi operativi implementati negli ambienti ICS sono abbastanza differenti dalle loro controparti utilizzate in ambienti IT. E anche qualora si utilizzi un sistema operativo comune al mondo IT (come Windows o Linux), le configurazioni ed i software utilizzati per il controllo sono completamente diverso. Chi deve gestire una rete di un ICS, quindi, deve possedere delle conoscenze e delle capacità specifiche per quel tipo di ambiente. In caso contrario vorrebbe dire non riconoscere le differenze tra i due ambienti, con il rischio di incappare in gravi conseguenze.

Limitazioni delle risorse Molto spesso i sistemi operativi ICS girano su macchine dalle risorse hardware e software limitate, con la conseguenza che alcune delle feature di sicurezza tipiche dei sistemi IT non sono incluse. Caratteristiche come la capacità di criptare e decriptare dati, log testuale che riporti gli errori o strumenti per la protezione delle password spesso sono assenti dalle macchine di un sistema ICS, sia per questioni hardware che software. E alle volte aggiungere delle nuove componenti aggiornate per implementare le suddette funzionalità non è una soluzione percorribile, in quanto ne andrebbe della stabilità dell'intero sistema di controllo.

Comunicazioni In un ICS i protocolli di comunicazione utilizzati dai dispositivi periferici, ossia i PLC, sono molto spesso proprietari e messi a disposizione direttamente dalle aziende produttrici. Questo rappresenta un problema, dato che in questo modo si viene a creare un sistema con protocolli misti e da gestire singolarmente in modo da garantire la corretta funzionalità del sistema. Nei sistemi IT, invece, si fa affidamento a protocolli di comunicazione standard, quali TCP/IP, le cui funzionalità sono oramai ben conosciute e consolidate.

Gestione degli aggiornamenti In generale, sia per gli ambienti IT che per quelli ICS, gli aggiornamenti dei software utilizzati sono importantissimi e ne garantiscono il loro corretto funzionamento. Purtroppo però, la gestione dell'applicazione di questi aggiornamenti differisce notevolmente quando si prendono in esame i due casi separatamente. Nei sistemi IT, le patch di sicurezza e le migliorie software sono applicate periodicamente e molto spesso in maniera del tutto autonoma, senza richiedere l'intervento di chi opera con la macchina in questione (nelle grandi reti IT aziendali sono presenti dei server adibiti al solo scopo di gestire automaticamente l'aggiornamento di tutte le macchine collegate a quella rete). Nei sistemi ICS, purtroppo, non si può eseguire la stessa procedura. Innanzitutto gli aggiornamenti devono essere testati a lungo e in maniera intensiva, sia dai produttori dell'applicazione che dagli operatori del sistemi di controllo, prima di poter essere applicati in maniera definitiva. Inoltre la procedura deve essere pianificata per tempo, in quanto, come visto, l'interruzione improvvisa dell'operatività del sistema di controllo causerebbe numerosi ed ingenti danni economici all'azienda produttrice. Un altro inconveniente è che spesso i sistemi operativi implementati negli ICS sono obsoleti e non più supportati dagli stessi venditori del software. Questo può comportare che alcune patch rilasciate non siano compatibili con questi sistemi operativi ed applicarle vorrebbe dire rendere instabile l'intero ICS. Un esempio è stato il caso di Windows XP ed il Service Pack 3 rilasciato da Microsoft che rendeva instabile numerosi sistemi di controllo, tant'è che ad oggi molti di essi ancora utilizzano il Service Pack 2.

Assistenza tecnica Tipicamente nei sistemi IT il sistema di assistenza tecnica può essere erogato in maniere differenti ed è in grado di gestire tutte le diverse componenti dell'architettura, anche se appartengono a venditori diversi tra loro. Nel caso di un ICS, invece, l'assistenza viene effettuata spesso da parte di una singola azienda e potrebbe applicare delle soluzioni differenti da quelle che applicherebbe un altro venditore. Inoltre, in alcuni casi non è permesso l'utilizzo di soluzioni per la sicurezza messe a disposizione da eventuali terze parti esterne, in quanto impedito dal contratto che l'azienda gestore dell'ICS ha redatto con il venditore delle apparecchiature. Nel caso in cui si venga meno a questa clausola, si rischierebbe di annullare la garanzia offerta dal venditore portando il sistema ad un'esposizione molto più diretta a dei potenziali rischi.

Tempo di vita dei componenti Nei sistemi IT, le componenti hardware e software hanno una durata garantita per un tempo che va dai 3 ai 5 anni. Questo è dettato dalle rapide e continue evoluzioni tecnologiche, che rendono obsoleti i dispositivi molto velocemente. In un sistema di controllo, invece, si fa largo uso di componenti specificamente studiati per quello scopo. Questo fa sì che la vita di questi strumenti si attesti tra i 10 e i 15 anni, circa il triplo rispetto al caso IT.

Disposizione fisica dei componenti I sistemi IT sono di norma situati in aree commerciali e facilmente accessibili, per facilitare un eventuale intervento in loco. Questo è il caso anche dei sistemi di controllo situati a ridosso del processo controllato. Le uniche componenti che potrebbero essere situate in luoghi più lontani sono i server utilizzati per il backup dei dati. Come abbiamo visto, però, numerosi ICS, a causa delle caratteristiche geografiche del processo controllato, sono disposti in luoghi molto vasti e raggiungibili solo con mezzi appositi. Inoltre, dato l'alto rischio rappresentato da alcune componenti dei sistemi di controllo , spesso si utilizzano luoghi di installazione più isolati proprio per motivi di sicurezza.

In conclusione, è evidente che le differenze operative e di sicurezza di un ICS crei la necessità di creare delle strategie di sicurezza molto più complesse rispetto ad un sistema IT. È necessaria la presenza un team composto sia da operatori di sistemi di controllo che da esperti di sicurezza informatica per individuare e capire le possibili conseguenze dovute alle normali azioni di assistenza: che sia l'installazione di un aggiornamento, operazioni di manutenzione o l'implementazione di una nuova soluzioni per la sicurezza. Tutto ciò deve essere studiato prima di effettuare l'azione in maniera definitiva.

4.2 SCADA e dispositivi mobile

4.2.1 Applicazioni locali e remote

Nel corso degli ultimi 2/3 anni si è cominciato a diffondere sempre di più anche in ambiente SCADA l'utilizzo di applicazioni su dispositivi mobile per la gestione dei sistemi di controllo.Inoltre, l'idea di affidarsi a piattaforme cloud per svolgere le normali procedure di gestione del sistema, quali monitoraggio degli accessi, attività di supervisione e quant'altro, non sembra più così assurda come poteva apparire qualche anno fa. Tutte queste comodità, però, portano ovviamente con loro anche delle criticità che, se non correttamente risolte, a loro volta potrebbero portare a subire degli attacchi, con annesse perdite economiche.

L'infrastruttura di un ICS, come precedentemente esposto, può essere di natura molto eterogenea, seguendo anche il principio di segmentazione del protocollo ISA99. In figura 4.3 è riportato un esempio di architettura di un sistema in cui si fa utilizzo di applicazioni mobile per il monitoraggio e la gestione dell'impianto. Come è possibile notare, i terminali mobile, a seconda del livello dell'architettura in cui sono posti, si possono raggruppare in due categorie:

Applicazioni locali Queste applicazioni sono installate su dispositivi che si interfacciano direttamente con i dispositivi periferici dell'ICS, usualmente tramite connessioni Wifi o Bluetooth. un esempio sono le applicazioni installate su tablet usate dagli ingegneri di controllo per osservare lo stato del processo, anche durante eventuali pause o momenti in cui sono lontani

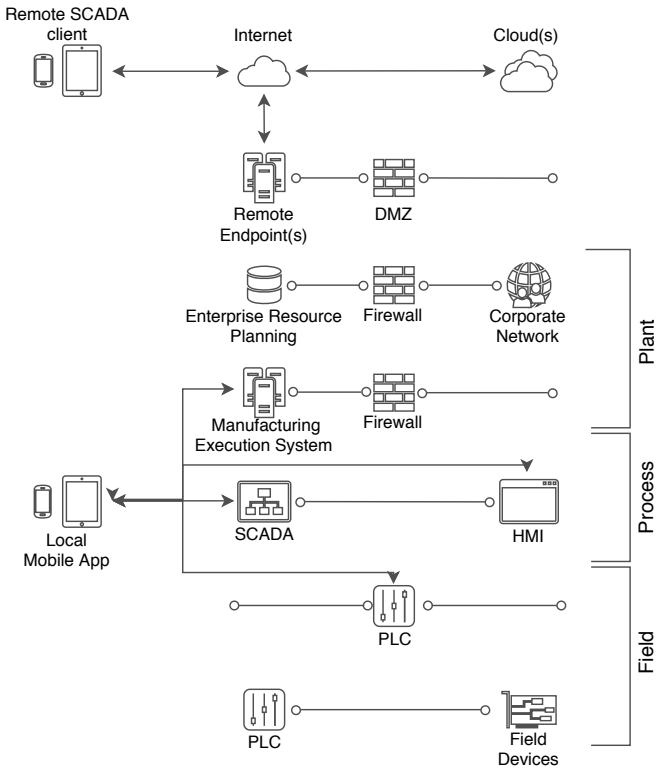

Figura 4.3. Architettura di un sistema SCADA integrato con applicazioni mobile

dai terminali fissi. Queste applicazioni posso essere utilizzate anche per altri scopi, come ad esempio configurare le operazioni di un PLC o configurare una schermata HMI, in sostituzione di laptop e pc. In base all'utilizzo, quindi, le applicazioni locali possono essere suddivise a loro volta in:

- Configuratori di PLC, ossia applicazioni per monitorare, ma anche programmare i vari dispositivi della rete del sistema di controllo, quali PLC, RTU (Remote Terminal Unit, dispositivi elettronici di controllo con cui è possibile monitorare i parametri fisici di un processo e trasmetterli al sistema di elaborazione dell'impianto SCADA) ed altro ancora;
- Client SCADA, applicazioni tramite cui gli ingegneri di controllo monitorano e supervisiono lo stato del processo. Queste applicazioni si interfacciano con i server SCADA;
- Pannelli HMI Mobili, ossia delle applicazioni che trasformano il proprio dispositivo mobile in un pannello HMI vero e proprio. Alcuni software mettono a disposizione dei modelli di interfacce con già alcune logiche di controllo implementate, ma è possibile, sempre tramite le stesse applicazioni o le loro controparti desktop, creare dei pannelli HMI personalizzati e programmarli a seconda delle proprie esigenze. Una volta create, le interfacce vengono caricati sui server SCADA ed eseguite sui dispositivi mobile, trasformandoli in veri e propri pannelli HMI (i quali possono essere completamente sostituiti, causando anche un notevole risparmio economico).

In generale, per quanto riguarda questo tipo di applicazioni, bisogna effettuare alcune considerazioni, basate sul fattore "ambientale": queste applicazioni sono utilizzate in un segmento dell'architettura considerato sicuro, in quanto separato tramite firewall dall'esterno. Ciò significa che la mancanza di funzioni per la crittografia dei dati e per svolgere le procedure di autenticazione e autorizzazione non è da considerare come fonte di rischi elevati. Il problema, però, si pone nell'eventualità di manomissioni, anche parziali, di queste applicazioni.

In questo caso gli attaccanti possono interagire direttamente con il sistema di controllo, influenzando lo stato del processo a proprio piacimento. È chiaro quindi come sia importante che queste applicazioni non vengano utilizzate al di fuori di questi ambienti sicuri, altrimenti il rischio di minacce diverrebbe troppo alto. La conseguenza è che i dispositivi in cui sono installate i software mobile non lasciano mai le stanze di controllo e, soprattutto, non sono connesse con la rete esterna.

Applicazioni remote Questo tipo di applicazione permette agli operatori di controllo di connettersi ai server del sistema SCADA tramite canali di comunicazioni remoti, quali rete internet, reti VPN ed altre soluzioni simili. Tipicamente sono utilizzate per eseguire azioni di monitoraggio dei sistemi, ma esistono anche delle applicazioni che permettono di svolgere delle azioni di controllo sul processo. In questo caso sono integrate anche delle funzionalità di avvertimento tramite degli alert remoti in modo da avvisare l'operatore della necessità di un suo intervento.

Confrontandole con le applicazioni locali si può subito notare la grossa differenza tra le due categorie: le applicazioni remote utilizzano i vari protocolli Internet per comunicare con il sistema SCADA e spesso sono installate sui dispositivi personali degli operatori. Quindi, l'ambiente in cui operano non può essere considerato chiuso e isolato come nel caso delle applicazioni locali. Questo significa che le applicazioni remote sono esposte ad un numero maggiore di rischi ed attacchi provenienti sia dalla rete di comunicazione che da software maligni installati sui dispositivi.

4.2.2 Analisi delle vulnerabilità mobile

Sebbene le tipologie di applicazioni mobile utilizzate in ambiente SCADA possano essere molteplici, ognuna con le proprie funzionalità, è possibile stilare una lista di minacce comuni a tutti i suddetti software. Il motivo è da ricercare nelle caratteristiche del mezzo su cui vengono utilizzate: i dispositivi mobile, sebbene molto pratici e comodi, sono di natura molto vulnerabili, non solo agli attacchi software, ma anche a minacce fisiche, come il furto o lo smarrimento. In base al tipo di attacco, quindi, è possibile raggruppare le varie minacce in tre categorie:

Accesso non autorizzato ai dati In questo scenario, l'attaccante riesce ad ottenere l'accesso al dispositivo dove è installata l'applicazione e, di conseguenza, a tutti i dati che sono salvati al suo interno. Può avvenire in diversi modi, ma in tutti i casi le conseguenze sono molto gravi.

Uno dei modi che può dare il via al suddetto scenario di attacco è la perdita fisica del dispositivo. In questa situazione l'attaccante ha modo di appropriarsi dei dati dell'applicazione sia con attacchi software che hardware. Se il dispositivo è destinato ad un uso locale, l'attaccante otterrebbe delle preziose informazioni riguardo il processo controllato, l'infrastruttura del sistema SCADA, schemi di rete ed altro ancora. Queste informazioni, poi, potranno essere usate per eseguire successivamente un attacco più mirato e con più alte probabilità di successo. Nel caso in cui, invece, il dispositivo perso sia destinato ad un uso da remoto, le conseguenze sarebbero ben più gravi: gli attaccanti, infatti, potrebbero ottenere tutte le informazioni necessarie per l'autenticazione ed utilizzarle per connettersi direttamente ai server del sistema SCADA.

Un altro modo con cui può avvenire un accesso non autorizzato è nel caso in cui venga lasciato incustodito un dispositivo mobile senza alcuna protezione. In questo caso l'attaccante ha a disposizione una finestra temporale in cui può sia ottenere delle informazioni cruciali, ma anche alterarle senza che nessuno se ne accorga. Similmente, si può verificare la stessa situazione quando i dati sono salvati su partizione esterne (tipicamente schede di memoria SD) e queste non vengono protette adeguatamente. Anche in questo caso l'attaccante ha la possibilità di manipolare e/o accedere ai dati presenti nella partizione senza particolari problemi.

Compromissione del canale di comunicazione Questo scenario ha più probabilità di manifestarsi nel caso delle applicazioni remote, ma ciò non esclude del tutto dal pericolo quelle

locali, anch'esse comunque a rischio. Generalmente, il canale di comunicazione che le applicazioni utilizzano per dialogare con i server del sistema SCADA deve garantire il mantenimento delle proprietà di riservatezza ed integrità. Purtroppo però, i dispositivi su cui sono installate gli applicativi possono connettersi alla rete Internet attraverso dei canali non sicuri. Questo può provocare una serie di pericoli e minacce quali:

- Connessione ad access point clandestini, sia per reti WiFi che per quelle cellulari;
- Connessione ad access point pubblici che non integrano funzionalità per la sicurezza;
- Compromissione delle rete privata, sia aziendale che domestica;
- Compromissione della rete VPN.

Quale che sia la minaccia, le conseguenze sono sempre le stesse: l'attaccante può pianificare un attacco Man in the Middle (sarà esposto più nel dettaglio nella sezione 4.3), ottenendo la possibilità di osservare il traffico dati in chiaro che ha luogo nel canale di comunicazione in questione. In questa maniera può effettuare può effettuare lo sniffing delle informazioni scambiate tra server SCADA e applicazione, ma anche manipolarle a proprio piacimento senza essere scoperto.

Compromissione dell'applicazione Anche se le applicazioni operano in un ambiente sicuro, con canali di comunicazioni protetti e gli accessi sono controllati, è possibile che vi siano delle vulnerabilità nel loro codice, esponendole di fatto a pericoli inaspettati. Queste vulnerabilità del codice possono essere sia dal lato del server (il backend) che nelle applicazioni client. I pericoli che possono scaturire da questo scenario sono diversi, come ad esempio una manomissione delle liste di controllo per gli accessi, l'esecuzione di codice malevolo da remoto o la compromissione di dati riservati.

Basandosi sugli scenari delle minacce appena esposte, si possono individuare due categorie di attacchi mobile ognuno con obiettivi diversi. La prima categoria è formata dagli attacchi il cui scopo è quello di influenzare, in maniera più o meno diretta, lo stato del processo controllato. Generalmente può avvenire quando l'attaccante ha libero accesso alle credenziali di un operatore e modifica le informazioni cruciali per in controllo di un processo. La seconda tipologia di attacchi è quella che mira ad influenzare le azioni di un operatore SCADA, in modo da fargli compiere delle azioni dannose sul sistema senza che se ne accorga. In questo caso, l'attaccante crea un ambiente di lavoro fittizio, del tutto simile a quello reale, in cui potrà ad esempio cambiare a proprio piacimento i dati sullo stato del processo o scambiare le informazioni da mostrare nei vari pannelli della schermata HMI. Basandosi su queste informazioni alterate, l'operatore si troverà a effettuare in buona fede delle azioni di controllo, ma che in realtà finiranno per danneggiare il sistema SCADA ed il processo controllato.

La sicurezza dei dispositivi mobili, inizialmente sottovalutata, è tornata sotto i riflettori negli ultimi anni. Numerosi sono i stati i lavori di studio effettuati per attirare l'attenzione sui problemi nascosti in questi software che hanno sicuramente reso l'ambiente degli ICS di più facile utilizzo, ma al contempo più vulnerabile. Un lavoro dettagliato in questo senso è quello svolto dagli esperti di sicurezza informatica Alexander Bolshev e Ivan Yushkevich. I due esperti, basandosi sulla classificazione delle applicazioni e delle minacce precedentemente fornita, nel corso degli anni hanno analizzato nel dettaglio le applicazioni per mobile presenti sul mercato. I dati ottenuti sono poi stati esposti in rapporti tecnici dettagliati. Nel loro ultimo lavoro [5], pubblicato nel gennaio dello scorso anno, hanno analizzato software creati da oltre 30 venditori. In queste hanno riscontrato un totale di 147 falle della sicurezza , ognuna delle quali è stata categorizzata seguendo le linee guida fornite dal "Open Web Application Security Project" (OWASP), un ente no-profit istituito nel 2001 con lo scopo di aiutare a migliorare la sicurezza dei software. L'OWASP prevede un totale di 10 categorie per i rischi delle applicazioni mobile:

- **utilizzo improprio della piattaforma**, ossia l'uso improprio o il mancato impiego delle funzionalità di sicurezza messe a disposizione dello sviluppatore;
- **archiviazione impropria dei dati**, che ricopre l'errata gestione dei dati utilizzati dall'applicazione;

- **comunicazione non sicura**, in cui rientrano tutti i problemi nel creare un canale di comunicazione adeguato;
- **autenticazione non sicura**, ossia vulnerabilità dovute alla errata gestione della sessione o all'autenticazione dell'end user;
- **crittografia insufficiente**, che include problemi dovuti ai tentativi di crittografia errati;
- **autorizzazione non sicura**, in cui sono compresi i tentativi errati nel gestire le autorizzazioni;
- **qualità del codice client**, che raggruppa gli errori di codice presenti nell'applicazione;
- **manomissione del codice**, ossia i problemi dovuti alle modifiche che è possibile apportare al codice dell'applicazione una volta installata;
- **reverse engineering**, che identifica la capacità di ottenere il codice sorgente partendo dal software;
- **funzionalità non pertinenti,** ossia funzionalità aggiuntive utilizzate in fase di sviluppo erroneamente mantenute nell'applicazione finale.

A queste categoria ne hanno aggiunta una specifica per l'ambiente di lavoro SCADA, ossia **errori presenti nella programmazione del codice implementato nel backend**.

OWASP ID	Categoria OWASP	N° di errori	% di Applicazioni
M1	Utilizzo Improprio della Piattaforma	5	6%
M2	Archiviazione Impropria dei Dati	20	47%
M3	Comunicazione Non Sicura	11	38%
M4	Autenticazione Non Sicura	6	18%
M5	Crittografia Insufficiente	8	24%
M6	Autorizzazione Non Sicura	20	59%
M7	Qualità del Codice Client	12	35%
M8	Manomissione del Codice	32	94%
M9	Reverse Engineering	18	53%
M10	Funzionalità Non Pertinenti	8	24%
	Errori Lato Backend	7	12%

Tabella 4.1. Statistiche vulnerabilità applicazioni SCADA per mobile [5].

Nella tabella 4.1 sono riportati i risultati dell'analisi effettuata dai due studiosi. È possibile notare come qualche categoria presenti una bassa percentuale di errori, mentre alte sono moto elevate. Da notare come la categoria inerente alla manomissione del codice abbia una percentuale molto elevata, segno che alcuni problemi spesso sono ancora sottovalutati.

4.3 Scenari di attacco

Esistono molti metodi per attaccare un obiettivo, una volta che questa è stato identificato. I vari attacchi usati in ambiente, Man in the Middle, Denial of Service, attacchi replay, phishing, infezione dei sistemi con malware, sono molto efficaci anche in ambiente ICS. I motivi primari, come visto nei precedenti paragrafi, sono da ricercarsi in una combinazione di fattori legati a protocolli di comunicazione non sicuri, autenticazione tra dispositivi molto scarna e architetture di rete fragili.

Se una rete venisse penetrata ed installato un malware al suo interno, tramite dei tool specifici si potrebbe garantire l'accesso remoto al sistema, installare dei keylogger o intraprendere qualsiasi altra azione distruttiva. In alcuni casi le informazioni possono essere estrapolate ed utilizzate per

effettuare un ulteriore attacco. Inoltre, se un attacco riesce ad andare a segno senza lasciare alcuna traccia, l'attaccante può rimanere all'interno del sistema indefinitamente. Nel caso in cui ciò accada nella rete di un sistema collegato ad altri sistema (come può essere una rete di un server di una stanza di controllo SCADA), a questo punto si ha la possibilità di iniziare un ulteriore attacco con obiettivo altre parti della rete industriale, aumentando l'ammontare dei danni provocati.

Di seguito verranno esposti i tipi di attacchi informatici più comuni che possono essere effettuati, le modalità in cui essi si svolgono e quali difese possono essere implementate nell'architettura di un ICS per contrastarli [24] [45].

4.3.1 Attacco Man in the Middle

Con Man in the Middle (abbreviato in MitM) ci si riferisce ad un attacco in cui l'attaccante riesce a frapporsi fra due dispositivi in comunicazione senza che se ne accorgano, riuscendo in questo modo a catturare tutti i dati e le informazioni scambiate durante il dialogo. L'attaccante, per fare ciò, si connette ad entrambi i dispositivi ed inizia a gestire tutto il traffico dati tra di loro in modo da far sembrare che stiano comunicando direttamente tra loro, quando in realtà la comunicazione si sta sviluppando attraverso un terzo dispositivo in ascolto.

Al fine di eseguire un attacco MitM, l'attaccante deve essere in grado di intercettare i traffico tra i due obiettivi. Se la connessione da attaccare risulta mancante di funzionalità per la criptazione dei dati e l'autenticazione dei dispositivi connessi, come spesso accade nei protocolli di comunicazione industriali, il procedimento è molto diretto e veloce. Ma, anche nel caso in cui queste funzionalità siano presenti, è comunque possibile effettuare un attacco, ad esempio intercettando lo scambio delle chiavi di autenticazione e sostituirle con delle copie personali.

Un altro metodo per effettuare un attacco MitM è quello di agire sulle tabelle di risoluzione degli indirizzi e modificarle per ottenere l'accesso al flusso delle informazioni scambiate. Questo può essere fatto sia nelle tabelle del Address Resolution Protocol (ARP), effettuando un attacco di tipo ARP Poisoning, che in quelle del Domain Name System (DNS), dando luogo in questo caso ad un attacco DNS Poisoning. Prima di mostrare le modalità con cui si svolgono i due attacchi, è bene effettuare una panoramica sulle funzionalità per cui sono utilizzati i protocolli ARP e DNS.

L'ARP è un protocollo di servizio, incluso nel protocollo IP, che agisce a livello di accesso alla rete ed il cui compito è quello di effettuare una mappatura tra indirizzi IP e indirizzi MAC dei dispositivi connessi alla rete locale. Il DNS, invece, è un sistema di indirizzamento, presente sempre nel protocollo IP, che agisce a livello applicazione ed è utilizzato per tradurre il nome di un nodo della rete o host in un indirizzo IP. Iin entrambi i casi, per velocizzare le operazioni di traduzione, si fa utilizzo di tabelle di cache, solitamente memorizzate all'interno dei server aziendali. Queste tabelle sono l'obiettivo degli attacchi ARP Poisoning e DNS Poisoning: ad un attaccante basterà modificare opportunamente i valori delle cache inserendo quelli del proprio indirizzo IP o MAC, a seconda che si agisca rispettivamente sulla cache di un server DNS o di uno ARP, in modo da riuscire ad instradare tutto il traffico verso la propria macchina senza che gli interlocutori se ne accorgano.

Una volta che l'attaccante è riuscito ad intromettersi nel flusso delle informazioni, avrà il completo controllo sulle stesse dando l'inizio all'attacco vero e proprio. Uno scenario tipico è quello di un attacco replay in cui l'attaccante, una volta intercettate le informazioni ricevute dal pannello HMI, le modifica per far compiere al dispositivo di controllo delle azioni dannose. Nel frattempo, i dati inizialmente catturati vengono ritrasmessi al pannello HMI, in modo da non far accorgere l'operatore di nulla. Esiste un altro modo per effettuare un attacco replay, complementare a quello appena esposto e prevede che sia l'operatore stesso ad effettuare delle azioni dannose per il sistema. In questo caso, l'attaccante modificherà i dati da trasmettere al pannello HMI, in modo da indicare la presenza di un problema nell'impianto. L'operatore, quindi, inizierà al routine di correzione del problema, causando però degli eventi dannosi e imprevedibili.

Esistono dei metodi da applicare con cui potersi difendere dagli attacchi MitM o, in generale, che permettono la mitigazione degli effetti negativi. Oltre, ovviamente, ad utilizzare tecniche di cifratura e autenticazione robuste(o comunque implementarne di migliori laddove già esistano),

in modo da creare un ostacolo difficile da superare per gli attaccanti, nel caso in cui l'attaccante faccia uso di metodi di ARP Poisoning o DNS Poisoning vi sono delle ulteriori soluzioni.

Per contrastare in partenza un attacco ARP Poisoning si possono utilizzare diversi metodi, i più comuni sono quelli del MAC Address Locking (o port security) e delle tabelle di indirizzamento statiche. Con il primo metodo si va a creare un'associazione univoca tra un indirizzo MAC ed una specifica porta dello switch di comunicazione. Se l'indirizzo MAC della macchina connessa alla porta non combacia con quella associata, la comunicazione viene interrotta e l'intruso non potrà più raggiungere il suo obiettivo. Inoltre con questo metodo si garantisce un livello di protezione fisica ulteriore, in quanto può impedire l'utilizzo di macchine non aggiornate e più vulnerabili all'interno di una rete protetta. Il secondo metodo si basa sul presupposto che la rete informatica di un ICS non subisce grosse variazioni di composizione e può essere considerata relativamente statica. Partendo da ciò è possibile codificare gli indirizzi MAC in maniera statica nelle tabelle ARP presenti in ogni computer, cosicché si impedisca all'attaccante di cambiarne i valori. Chiaramente questa tecnica diventa difficile da implementare quando ci si trova nel caso di reti molto estese e/o caratterizzata da un elevato dinamismo delle macchine connesse.

Anche nel caso di DNS Poisoning esistono metodi di prevenzione e mitigazione. Il primo ed il più efficace è quello di utilizzare il protocollo DNSSEC (Domain Name System Security Extensions). Questo prevede l'utilizzo di una firma crittografata per ogni record inserito nelle tabelle di cache che sarà utilizzata dal DNS resolver per verificare l'integrità e l'autenticità dei dati ricevuti. Se così non fosse, i dati verrebbero scartati a priori e sarebbe lanciato un messaggio d'errore all'operatore. Ulteriori metodi prevedono l'utilizzo di tecniche di validazione end-to-end a livello di trasporto una volta stabilita la connessione tra i due dispositivi. Ad esempio, utilizzando il protocollo HTTPS, è possibile controllare che il certificato digitale del server sia valida ed appartenga al legittimo proprietario.

Infine è consigliabile l'uso di programmi adibiti all'individuazione di questi attacchi. Questi software agiscono ispezionando i dati prima che questi vengano trasmessi: se questi sembrano essere stati manipolati, il programma blocca la trasmissione dei dati ed avverte l'operatore della presenza di un tentativo di attacco.

4.3.2 Attacco Denial of Service

Si ha un attacco Denial of Service (Dos) quando si scatena un evento malevolo con l'intento di rendere una risorsa non più disponibile all'utente. Come si può immaginare, è una categoria molto vasta ed eterogenea che può includere qualsiasi tipo di indisponibilità, dalla perdita di comunicazione con un dispositivo fino all'inibizione o manomissione di particolari funzionalità del sistema di controllo, come l'archiviazione, l'elaborazione continua ed altro ancora. Un attaccoDos prevede l'utilizzo, da parte dell'attaccante, di una singola macchina con cui inviare un grossa quantità di richieste da elaborare al sistema da attaccare. Per amplificare gli effetti di questo attacco e diminuire i tempi di esecuzione, ne è stata concepita una versione che agisce su larga scala, denominato Distributed Denial of Service (DDos). In questo caso, l'attaccante sfrutta una rete di macchine, creata appositamente per lo scopo e chiamata "botnet", per attaccare il sistema vittima da più nodi di comunicazione. In questo caso l'attaccante avrà l'ulteriore possibilità di nascondere la propria identità in maniera più efficace, poiché, una volta inviato il segnale di attacco alla botnet, potrà scollegarsi dalla rete e far sparire le proprie traccie, mentre l'attacco verrà svolta in maniera completamente autonoma. A parte la differenza dovuta ai tempi di esecuzione dell'intera operazione, gli effetti di un attacco DDos sono pressoché gli stessi di un attacco Denial of Service. Per brevità, quindi, in questo paragrafo si parlerà solo di attacchi Denial of Service, ma il discorso si estende anche a quelli Distributed Denial of Service.

Normalmente gli attacchi Dos rivolti contro i tradizionali sistemi IT non hanno un così forte impatto economico negativo, se ovviamente sono presi e gestiti per tempo. Alcune tra le più comuni conseguenze possono essere rallentamenti nell'accesso ad una pagina web o nella gestione delle e-mail, ma alla risoluzione del problema le normali funzionalità vengono ripristinate. In questi sistemi raramente l'interruzione dei servizi porta al verificarsi di conseguenze fisiche, ma ciò non toglie che un attacco Dos ben strutturato possa causare lo spegnimento delle diverse macchine collegate alla rete informatica presa di mira.

In uni sistema SCADA, invece, le conseguenze possono essere ben più gravi, come ad esempio nel caso in cui l'obiettivo dell'attacco sia un ICS automatizzato, che opera in maniera continua e controlla un determinato processo fisico (il flusso di un liquido in una conduttura di distribuzione, la produzione di energia elettrica, la produzione di parti meccaniche, ecc.) tramite dei controller hardware e software. L'impossibilità di uno dei controller, a seguito di un attacco Dos, di eseguire la normale routine di azioni programmate, è denominata "Loss of Controll (LoC)" e può portare ad interrompere istantaneamente l'operatività del sistema di controllo mentre il processo viene messo in uno stato di sicurezza (ossia avviene lo spegnimento dell'impianto). Ciò chiaramente rappresenta un pericolo da evitare assolutamente perché, come detto in precedenza, quando un sistema SCADA cessa improvvisamente di funzionare, le conseguenze che ne scaturiscono possono essere molto gravi e portare anche a gravi danni fisici e ambientali.

Allo stesso modo, nel caso in cui l'obiettivo dell'attacco Dos sia una scherma HMI, anche se questa non è direttamente connessa con il sistema di controllo meccanico, le conseguenze che ne possono scaturire non sono di certo meno gravi. Tipicamente in questo scenario l'attaccante rivolge i propri sforzi cercando di bloccare il sistema di autenticazione remoto, o RADIUS (Remote Authentication Dial In User Service). In questa maniera l'operatore viene espulso dalla rete e non può più monitorare lo stato del processo controllato. Questa situazione, analogamente al caso precedente, è denominata "Loss of View (LoV)" e, se non si riesce ristabilire il contatto in tempi brevi, l'univo modo per contrastarla è un riavvio dell'intero sistema di controllo, con le solite conseguenze viste dal grave impatto negativo.

Per contrastare questa tipologia di attacchi, purtroppo non esistono dei metodi definitivi, ma piuttosto una serie di rimedi, accortezze e linee guida da seguire per mitigare le conseguenze negative. Innanzitutto è molto importante prendere consapevolezza dei rischi a cui si è esposti quando si subisce un attacco Dos ed organizzare le risorse su cui si fa affidamento più spesso in modo da ridurre l'impatto negativo che l'attacco stesso può avere su di esse. Inoltre è molto importante conoscere i servizi messi a disposizione dal proprio provider di servizi internet (ISP) per contrastare un attacco Dos, come ad esempio il cambio degli indirizzi IP della propria rete o il blocco di quelli da cui proviene l'attacco.

A livello funzionale, invece, si può agire sul proprio sistema sovradimensionandolo rispetto alle normali esigenze operative. In questo modo si rende il sistema in grado di sopportare più a lungo un attacco Dos mentre si cerca di identificare la sorgente o comunque prepararlo per lo spegnimento senza causare gravi danni all'impianto di controllo. Un ulteriore metodo operativo per contrastare questi attacchi è quello di utilizzare meccanismi di controllo e osservazione del traffico dati sulla rete in modo da individuare quando è in atto un attacco Dos o DDos. Un esempio è quello fornito in [19] che sfrutta le tecnologie "Software-defined networking (SDN)". Questa rappresenta un nuovo approccio con cui si cerca di integrare il cloud computing nell'architettura di una rete, separando le funzioni di forwarding e di routing dei pacchetti di rete.

L'esempio fornito da Béla Genge in [19] prevede la presenza di un controller SDN che fornisce continuamente informazioni sul traffico dati ad un'applicazione di monitoraggio. Questa, allo stesso tempo, dialoga con il controller fornendogli aggiornamenti sullo stato della rete. Non appena viene individuato l'inizio di un attacco Denial of Service, il controller SDN viene informato e subito devia il flusso di dati incriminato in modo da proteggere la rete del sistema di controllo

4.3.3 Attacchi Phishing

Nonostante il progresso tecnologico abbia portato alla creazione di metodi di attacco sempre più complesse e difficile da contrastare, spesso le più efficaci risultano essere quelle legate a tecniche di social engineering. La loro caratteristica principale è l'utilizzo di metodi a bassa tecnologia per coinvolgere la vittima nell'azione d'attacco, senza che questa se ne renda conto.

Da recenti ricerche svolte sugli attacchi informatici subiti dalle aziende emerge che più del 90% di essi ha avuto inizio per mezzo di tecniche di phishing [49]. Questa è una delle tecniche preferite dagli hacker per creare delle minacce di tipo "Advanced Persistent Threat (APT)", ossia un attacco con cui si ottiene l'accesso non autorizzato nella rete informatica della vittima, rimanendo nascosti per un lungo periodo di tempo.

Esistono numerosi versioni di attacchi phishing, ma tipicamente la metodologia rimane sempre la stessa: viene inviata una mail alla vittima con cui si cerca di convincerla a fornire dati sensibili con l'inganno, ad esempio facendo compilare loro dei finti questionari, o, installare dei software malevoli, spesso inviati sotto forma di documenti allegati al messaggio, con cui si ottiene accesso alla rete della vittima. Una volta guadagnato l'accesso, si dà inizio all'attacco vero e proprio. Nel contesto industriale la tipologia di phishing che è utilizzata più spesso è lo spear phising, consistente nell'includere nel messaggio i dati personali della vittima, rivolgendosi a lui con nome, cognome, titolo ed altre informazioni private, in modo da aumentarne la credibilità ed avere più probabilità che venga letto. Tipicamente gli obiettivi a cui sono rivolti i messaggi di spear phishing sono persone che ricoprono incarichi particolari, in quanto spesso in possesso di informazioni cruciali per l'azienda.

Essendo basati su tecniche di social engineering, non esistono dei rimedi assoluti agli attacchi di phishing. L'arma più potente a disposizione è sicuramente quella di instillare nel personale della compagnia una cultura di consapevolezza dei pericoli: un dipendente deve essere in grado di distinguere quali sono i messaggi ricevuti legittimi e quali, invece, nascondono dei tentativi di intrusione. Inoltre non deve mai condividere informazioni riservate e/o personali, a meno che l'interlocutore non sia affidabile e la sua identità sia già stata confermata. Sempre riguardo i dati, è consigliabile che anche quelli meno sensibili, come appunto nome, cognome e indirizzi e-mail, non vengano divulgati con troppa facilità. Nella maggior parte dei casi, infatti, gli attaccanti riescono a personalizzare i messaggi di spear phishing con informazioni facilmente reperibili sul web.

Ma la sola consapevolezza, spesso, non è sufficiente come arma. Ecco perchè è sempre consigliabile fare utilizzo di programmi per bloccare i tentativi di attacco di phishing. Usualmente questi software scansioni tutti i messaggi sospetti in entrata alla ricerca di una serie di parametri che mostrino l'eventuale natura malevola dell'email. Altre azioni eseguite da questi programmi sono la scansione sia di eventuali URL inclusi nei messaggi che di file allegati, eliminandoli nel caso in cui risultino infetti.

4.3.4 Attacchi con software malevoli (malware e worm)

Come detto nei precedenti paragrafi, molto spesso alcune tecniche sono utilizzate per predisporre tutti gli strumenti a disposizione prima di effettuare l'attacco vero e proprio. Questi strumenti molto spesso sono programmi appositamente creati per svolgere dei compiti specifici e in alcuni casi possono riprodursi anche in maniera autonoma all'interno della rete attaccata, in modo da diffondersi su più macchine possibili ed amplificare l'effetto distruttivo per cui sono stati creati.

I software utilizzati più frequentemente per questi scopi sono malware e worm, termini spesso usati come sinonimi ma che in realtà identificano due tipologie di programmi diverse tra di loro. Un malware, termine abbreviativo per "malicious software" (letteralmente "programma malintenzionato"), è una qualsiasi programma utilizzato per disturbare le normali operazioni svolte al computer, rubare informazioni o permettere di accedere al sistema senza autorizzazione. In un sistema SCADA i malware possono essere utilizzati in diverse maniere: se installato in una macchina HMI, posso cominciare a generare istruzioni sbagliate da far svolgere al sistema SCADA, provocando malfunzionamenti e errori nel sistema di controllo. Un altro utilizzo è quello di agire come Man in the Middle, occupandosi del dirottamento dei messaggi in modo che siano inviati alla macchina dell'attaccante. Inoltre esistono numerosi malware utilizzati come keylogger, ossia programmi che registrano qualsiasi cosa venga digitata sulla tastiera del dispositivo, come backdoor, tramite cui è possibile avere un punto di accesso al sistema non autorizzato, ed altro ancora.

I worm sono una particolare categoria di malware, il cui tratto distintivo è la capacità di auto replicarsi su tutte le macchine in cui riesce a diffondersi. Spesso viene erroneamente identificato come virus, ossia un malware che sfrutta lo scambio di file tra computer per potersi diffondere il più possibile. Un worm, invece, sfrutta le connessioni di una rete per potersi riprodurre su tutti i sistemi collegati ad essa. Questo rappresenta un grosso vantaggio per l'attaccante dato che, tipicamente, in un sistema SCADA, il passaggio di file tra computer è molto limitato. Al contrario, esistono moltissime connessioni tra i dispositivi di un impianto di controllo e ciò permette ad

un worm di arrivare a colpire direttamente i PLC, andando ad influenzare in maniera attiva il controllo di un processo.

L'infezione iniziale può avvenire in modi diversi: come visto gli attacchi di phishing possono essere uno dei vettori utilizzati per penetrare all'interno del sistema, ma non è da escludere che tutto possa partire da un semplice dispositivo di archiviazione esterno, come una chiavetta USB o una scheda di memoria SD. Al momento dell'inserimento in una macchina, il worm si attiva, viene installato in maniera silenziosa nel computer e inizia la sua attività, cercando contemporaneamente di riprodursi su altre macchine collegate a quella che lo ospita.

Worm e malware rappresentano delle minacce molto pericolose. Più di un volta sono stati registrati casi in cui, a causa di essi, sono stati riportati danni molto seri ad impianti di controllo. Fra tutti, i più diffusi sono sicuramente il worm Stuxnet ed il malware Dragonfly, i cui metodi operativi verranno esposti più in dettaglio nella sezione 4.4.

Anche in questo caso non esistono dei metodi di difesa efficaci al 100%: spesso questi programmi sfruttando bug dei sistemi operativi che non sono ancora stati segnalati o di cui ancora non è disponibile un patch per la correzione. Per prevenire una loro intrusione, quindi, occorre come sempre accortezza da parte degli operatori: controllare sempre i file che si sta per utilizzare, in modo da accertarsi della loro integrità. Inoltre, se è necessario fare uso di dispositivi rimovibili, evitare che questi vengano esposti all'ambiente esterno. Ovviamente, come nei casi precedenti, esistono programmi anti-malware che permettono l'individuazione e l'eliminazione nelle macchine di software dannoso, ma hanno dei lati negativi:

- per individuare il software è necessario eseguire una scansione che analizzi tutti i file presenti in una macchina, andando però ad impattare inevitabilmente sulle prestazione della stessa;
- se il malware viene programmata basandosi su una vulnerabilità al "zero-day", ossia che non è ancora stata scoperta, il software anti-malware non riuscirà ad intervenire, in quanto incapace di riconoscere la minaccia.

4.3.5 Attacchi combinati

Molti dei più grandi attacchi effettuati ai danni di un ICS non utilizza un'unica vulnerabilità presente su un unico obiettivo, ma, al contrario sono attacchi altamente sofisticati che combinano elementi di diversi tipi di malware, utilizzando molteplici vettori d'attacco, in modo da aumentare la gravità del danno causato e la velocità con cui il software riesce a diffondersi.

Originariamente gli attacchi combinati utilizzavano diverse tipologie di malware, il cui dispiego avveniva in maniera sequenziale. Un esempio può essere un attacco phishing con cui si ottiene l'accesso nel sistema, scavalcando il firewall. Da qui poi viene avviato un keylogger, con cui si acquisiscono le credenziali di accesso alla rete interna, rendendo possibile un attacco diretto al sistema di controllo o ottenere dati importanti e protetti. Al giorno d'oggi, il concetto di attacco combinato si è evoluto fino ad arrivare ad un grado di complessità veramente elevato. Il primo caso fu proprio registrato con Stuxnet (4.4.1), un worm in grado di variare il suo comportamento a seconda dell'ambiente in cui veniva eseguito. Questo concetto è stato chiaramente portato avanti, sfociando nella creazione di altri malware molto pericolosi e altrettanto complessi, come ad esempio Flame, una delle numerose varianti create a partire dallo steso Stuxnet.

Alla fine di questo capitolo è stata riportata una tabella in cui sono specificati diversi tipi di attacchi, a seconda del dispositivo o sottosistema che si intende attaccare (4.3).

4.4 Case study: il worm Stuxnet ed il malware Dragonfly

Gli attacchi informatici diretti alle reti di sistemi ICS, fino a una decina di anni fa, era puramente teorici. Il primo caso di un malware creato per danneggiare un sistema SCADA è stato documentato nel 2010, quando venne scoperto per la prima volta il worm Stuxnet. A distanza di quattro anni è stata resa nota l'esistenza di un'altra minaccia molto sofisticata che colpiva il mondo ICS,

Dargonfly, un malware utilizzato per scopi di spionaggio. Ovviamente, si sono succeduti molti altri attacchi nel corso degli anni successivi, molti dei quali sfruttavano il paradigma operativo che era stato implementato per lo stesso Stuxnet. Si possono citare, come esempio, Shamoon e Flame (quest'ultimo conosciuto anche come Flamer o Skywiper), due worm creati a partire da alcune delle vulnerabilità portate alla luce con Stuxnet.

In questa sezione sono presi come case study Stuxnet e Dragonfly, in quanto considerati tutt'ora i due software che gettato le basi per tutti i successivi attacchi creati nel corso degli anni.

4.4.1 Stuxnet

Secondo molti esperti di sicurezza informatica, Stuxnet è uno dei malware più sofisticati attualmente esistente [51]. Si pensa, nonostante non vi sia mai stata una conferma da parte degli organi interessati, che abbiano contribuito alla sua creazione un tema americano dell'NSA, in collaborazione con un team della corrispettiva agenzia di sicurezza nazionale israeliana [38].

Il worm fu scoperto nel giugno del 2010 in un impianto nucleare situato a Natanz, in Iran, per merito di alcuni ispettori dell'International Atomic Energy Agency (IAEA). Questi si accorsero, mentre osservavano dei filmati dell'impianto di sorveglianza, che i tecnici dell'impianto avevano sostituito un gran numero di centrifughe per il trattamento dell'uranio, in seguito a dei malfunzionamenti. Sebbene all'inizio la cosa fu vista come normale prassi (non è raro che queste componenti si guastino con il passare degli anni), quello che fece scattare il campanello d'allarme fu la cadenza con cui si verificavano i malfunzionamenti, molto elevata, e l'ulteriore fatto che lo stesso rateo con cui avvenivano i guasti incrementasse con il passare del tempo. Inizialmente gli esperti erano molto fiduciosi sui sistemi di sicurezza implementati nella centrale ed esclusero l'idea di un attacco hacker (la rete informatica della centrale nucleare era completamente isolata dall'esterno, era impossibile accedervi via Web). Ma alla fine, dopo una serie di indagini, si capì che il malfunzionamenti non erano dovuti né a fattori fisici dovuti a componenti mal progettati, né all'errore umano degli operatori, ma erano tutti imputabili alla presenza di un software malevolo nel sistema, ossia Stuxnet, che agiva infettando l'hardware delle centrifughe. L'inizio della propagazione, molto probabilmente, è da imputare ad una chiavetta USB infetta che, una volta connesso ad uno dei terminali SCADA diede inizio a tutto l'attacco.

Stuxnet, originariamente, è stato creato basandosi su quattro vulnerabilità al zero-day, rendendolo capace di infettare macchine con sistema operativo Windows di diverse generazione: da Windows 2000 fino a Windows 7/Server 2008R2. L'obiettivo primario della sua programmazione è quello di compromettere il software HMI della Siemens (l'azienda produttrice dei PLC utilizzati per controllare le centrifughe della centrale iraniana) WinCC e PCS7, così come compromettere degli specifici modelli PLC S7, sfruttando il bus di comunicazione Profibus.

Tenendo a mente l'obiettivo principale, le azioni che Stuxnet poteva intraprendere variavao in base ai software installati nel sistema che lo ospita. Se all'interno della macchina infettata non era presente nessuno dei programmi obiettivi, allora veniva installato un rootkit, ossia una collezione di software, affinché il malware venisse eseguito silenziosamente ad ogni avvio della macchina. Dopodiché iniziava ad applicare i metodi per cercare di propagarsi su altre macchine, sfruttando fino a sette differenti mezzi. Addirittura, per quelli che sfruttavano l'utilizzo di un dispositivo USB rimovibile, il worm aveva implementato un protocollo tramite cui si cancellava una volta che il dispositivo USB aveva infettato tre nuovi sistemi. Una volta installato in una macchina in cui era presente il software obiettivo, Stuxnet iniziava l'esecuzione di un'altra serie di metodi con cui riusciva ad installarsi nel database di WinCC o, addirittura, copiarsi nei file dei progetti utilizzati per programmare i PLC S7. Inoltre poteva anche sovrascrivere un determinato driver di comunicazione utilizzato per connettersi con il PLC, creando a tutti gli effetti un attacco Man in the Middle. Fortunatamente ormai la minaccia di Stuxnet è rientrata, grazie all'operato della stessa Siemens che, una volta messa al corrente della minaccia, creò un software per l'individuazione e la rimozione del worm.

Il processo di infezione attuato da Stuxnet è riportato in figura 4.4 e come è possibile notare è molto esteso e dettagliato. Non esistono schemi, invece, in cui è sintetizzata l'intera gamma delle azioni che Stuxnet era in grado di eseguire e della portata dei potenziali danni che era in grado di

arrecare. Attualmente, di Stuxnet, si sa solo ciò che è stato dedotto dai dati raccolti durante lo studio effettuato per eliminare la minaccia. Oltre alle azioni descritte in precedenza, ce ne sono molte altre che confermano quanto questo malware fosse sofisticato nella sua implementazione. Ad esempio:

- prima di eseguire qualsiasi azione di corruzione dati, controllava che la macchina ospitante eseguisse una versione compatibile di Windows, se era stata già infettata e, soprattutto, si accertava della presenza o meno di software anti-virus;
- tentava di oltrepassare i sistemi di protezione contro gli intrusi, installati sulla macchina vittima, infettando processi fidati preesistenti;
- eseguiva il processo di infezione iniettando le proprie DLL per intero in altri processi in esecuzione sul computer;
- si metteva alla ricerca di specifiche configurazioni di sistema. Una volta individuate, iniettava il proprio codice nei file di progetto dei PLC, in modo da agire come un rootkit nascosto in grado di inviare qualsiasi comando al controller;
- se il PLC su si installava operava a determinate frequenze, era in grado di modificarle in maniera improvvisa, sabotando e danneggiando il sistema di centrifugazione;
- era in grado di rimuoversi dai sistemi non compatibili con la sua programmazione, mettersi in uno stato dormiente per non essere individuato, infettare nuovamente sistemi che erano stati ripuliti e comunicare con qualsiasi nodo della rete infettata in modo da rimanere costantemente aggiornato sul proprio stato.

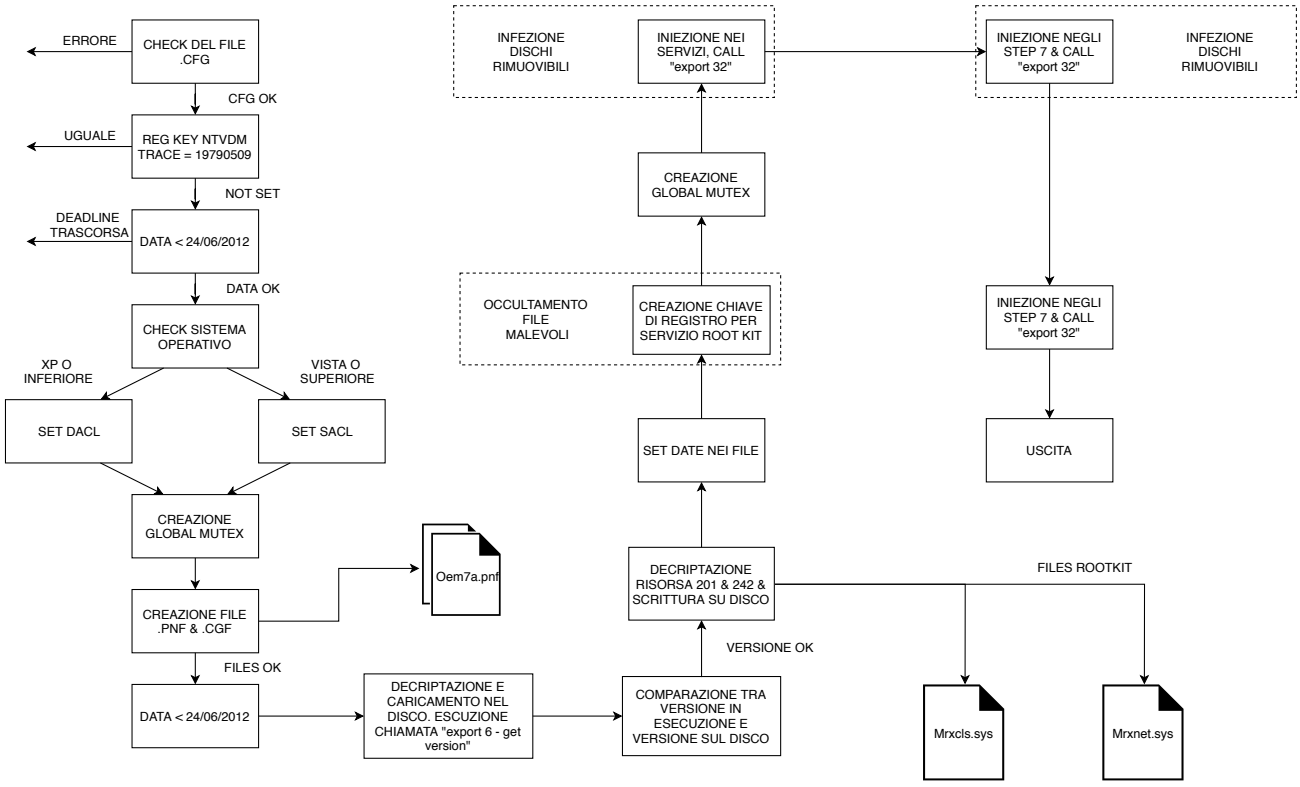

Figura 4.4. Processo di infezione di Stuxnet (fonte [24])

Ma, nonostante i vari problemi che Stuxnet ha provocato nel periodo in cui è stato operativo, la sua comparsa ha permesso in realtà di comprendere molti limiti e difetti dei sistemi di sicurezza che fino al quel momento erano stati applicati per proteggere un sistema SCADA. Grazie agli

studi condotti su Stuxnet, si è capito che molti dei paradigmi di sicurezza su cui venivano basate le architetture dei sistemi ICS erano errati e dovevano essere rivisti. Inoltre si è reso necessario concepire nuove misure di sicurezza per difendersi dalle nuove minacce basate sullo stesso paradigma operativo di Stuxnet. Nella tabella 4.2 sono riportati nel dettaglio alcune delle differenze tra i paradigmi di sicurezza prima e dopo l'avvento di Stuxnet.

PARADIGMI PRE-STUXNET	**PARADIGMI POST-STUXNET**
I sistemi possono essere isolati efficacemente dalle reti esterne, eliminando il rischi di incidenti informatici	I sistemi di controllo sono ancora soggetti alla natura umana: un forte perimetro difensivo può sempre venire oltrepassato a causa della superficialità nel trattare la sicurezza
I PLC che non eseguono sistemi operativi moderni non sono vulnerabili agli attacchi informatici	I PLC possono essere degli obiettivi concreti per i malware
I macchinari utilizzati per operazioni speciali beneficiano del paradigma "security through obscurity". Dato che la documentazione degli ICS non è disponibile, è impossibile creare un attacco contro di loro	Le risorse per creare un attacco di successo ed altamente specializzato contro un ICS sono tutte a disposizione degli hacker
Per proteggere la rete di un ICS da un attacco è sufficiente l'utilizzo di firewall e IDS	L'utilizzo di molteplici vulnerabilità allo zero-day per effettuare un attacco indicano che i metodi difensivi basati su blacklist non sono più sufficienti ed occorre considerare tecnologie basate su whitelist per difendersi contro delle vulnerabilità ancora sconsociute

Tabella 4.2. Differenze dei paradigmi difensivi SCADA prima e dopo l'avvento di Stuxnet (fonte: [24])

4.4.2 Dragonfly

Nel giugno del 2014, la compagnia finlandese di sicurezza informatica F-Secure postò, all'interno del proprio blog, la notizia della scoperta di un malware, descrivendo come lo stesso era stato utilizzato fino ad allora per scopi di sabotaggio ed acquisizione dati sensibili [15]. Furono condotti subito un serie di studi da parte della Symantec, altra azienda dedita alla sicurezza informatica, culminati nella creazione di un report tecnico sulle modalità operative del malware [36]. La prima cosa scoperta fu che il malware era il frutto di una campagna di spionaggio iniziata nel 2010. Inizialmente gli obbiettivi erano industrie legate all'aviazione e alla difesa localizzate negli USA e in Canada, ma successivamente la campagna venne rivota contro compagnie del settore energetico, propagandosi anche al di fuori del continente americano [33]. Ma la scoperta più significativa fu che, come nel caso di Stuxnet, Dragonfly era stato appositamente programmato per colpire dispositivi ICS, rappresentando il secondo caso di un malware creato esclusivamente per colpire i sistemi informatici degli impianti industriali.

La campagna di attacco ai danni delle industrie energetiche di Dargonfly si è sviluppata in tre fasi, sfruttando in ognuna di essa un vettore d'attacco differente:

1. nella prima fase, iniziata a febbraio 2013, fu impiegata la tecnica di spear phishing per infettare i computer delle vittime ed ottenere informazioni sensibili riguardanti la rete informatica dell'azienda da colpire. Tramite l'invio di file .PDF corrotti, veniva installato nelle macchine delle vittime un "Remote Access Trojan Horse (RAT)", ossia un malware con cui è possibile ottenere il controllo amministrativo da remoto del computer;

2. la seconda fase cominciò subito dopo la prima, nel maggio 2013, e si protrasse per circa 11 mesi, fino ad aprile 2014. Durante questo periodo furono utilizzati attacchi di watering hole.

Questo attacco consiste nel dirottare la connessione di un utente verso un sito, portandolo a vistarne un altro, senza che la vittima se ne accorga. Il gruppo gestore di Dragonfly riuscì a dirottare il traffico internet delle aziende attaccate verso siti gestiti direttamente da loro. All'interno di questi siti veniva caricato il software malevolo che veniva installato sulla macchina di chiunque visitasse la pagina in questione;

3. per la terza ed ultima fase cominciò nel giugno 2013 e venne portata avanti in contemporanea con la seconda. In questa fase, gli hacker di Dragonfly riuscirono a attaccare i siti di supporto di alcune note aziende produttrici di componenti per ICS. In questa maniera riuscirono a sostituire i software legittimi delle compagnie con programmi creati da loro e contenenti sempre il codice malware che era stato diffuso con gli altri due metodi.

La complessità di questo attacco fa ben capire della profonda organizzazione che ci fu alle spalle.Il malware utilizzato serviva ad ottenere informazioni sui dispositivi ICS presenti nella rete, per oi inviarlo agli stessi creatori di Dragonfly. Anche in questo caso, come per Stuxnet, il sfotware malevolo era stato programmato per ricercare delle specifiche configurazioni hardware, di cui sfruttava alcune vulnerabilità zero-day. Per farlo, utilizzava lo stesso protocollo industriale di comunicazione impiegato dai dispositivi ICS ricercati.

La scoperta di Dragonfly ha avuto un impatta nel mondo della sicurezza industriali paragonabile a quello che provocò Stuxnet quattro anni prima. La campagna di attacco attuata dalla compagnia creatrice del malware ha sottolineato, di nuovo, alcuni limiti nell'organizzazione della difesa:

- la strategia "Offense in Depth" ha sicuramente giocato un ruolo importante nella diffusione del malware. Le organizzazioni vittime sono state attaccate su più livelli della loro infrastruttura;
- le macchine colpite avevano installato tutte la stessa versione di Windows XP e sfruttava alcune vulnerabilità del sistema operativo al zero-day;
- Dragonfly finì per essere installato sui dispositivi mobile del personale, che devono essere ormai considerati dei potenziali punti d'ingresso per le minacce;
- nella campagna furono coinvolte anche le aziende produttrice, considerate fonte di software fidato. È necessario, quindi, controllare anche il traffico rivolto verso queste aziende, in quanto anche loro possono essere potenziali punti d'ingresso per delle minacce;
- il malware ha operato indisturbatamente sulle macchine infettate per circa un anno senza mai venire scoperto dai software che monitoravano il traffico internet.

Inoltre c'è da sottolineare come le aziende colpite, attualmente, rischino di subire degli ulteriori attacchi che sfruttano tutte le informazioni raccolte grazie a Dragonfly. A causa di tutto ciò si è reso necessario sviluppare un paradigma di sicurezza che tenesse conto dell'integrità non solo dei dispositivi ICS ma anche della rete informatica tra gli stessi, sfruttando, ad esempio, schemi per la segmentazione di rete che permettono un controllo del traffico dati più accurato e completo.

Tabella 4.3: Tabella delle possibili minacce che possono colpire un sistema SCADA, specificati a partire dall'obiettivo da attaccare (fonte: [24])

Target	**Possible Attack Vectors**	**Possible Attack Methods**	**Possible Consequences**
Access control system	• Identification cards • Closed-circuit television (CCTV) • Building management network • Software vendor support portal	• Exploitation of unpatched application (building management system) • RFID spoofing • Network access through unprotected access points • Network pivoting through unregulated network boundaries	• Unauthorized physical access • Lack of (video) detection capabilities • Unauthorized access to additional ICS assets (pivoting)
Analyzers/analyzer management system	• Subcontractor Laptop • Maintenance Remote Access • Plant (analyzer) network	• Exploitation of unpatched application • Network access via insecure access points (analyzer shelters) • Remote Access VPN via stolen or compromised subcontractor laptop • Remote Access VPN via compromise of maintenance vendor site • Insecure implementation of OPC (communication protocol)	• Product quality • spoilage, loss of production, loss of revenue • Reputation • product recall, product reliability
Application servers	• Remote user access (interactive sessions) • Business application integration communication channel • Plant network • Software vendor support portal	• Exploitation of unpatched application • Installation of malware via unvalidated vendor software • Remote access via "interactive" accounts • Database injection • Insecure implementation of OPC (communication protocols)	• Plant upset / shutdown • Credential leakage (control) • Sensitive / confidential information leakage • Unauthorized access to additional ICS assets (pivoting)
Asset management system	• Plant Maintenance Software / ERP • Database integration functionality • Mobile devices used for device configuration • Wireless device network - Software vendor support portal	• Exploitation of unpatched application • Installation of malware via unvalidated vendor software • Remote access via "interactive" accounts • Database injection • Installation of malware via mobile devices • Access via insecure wireless infrastructure	• Calibration errors • product quality • Credential leakage (business) • Credential leakage (control) • Unauthorized access to additional business assets like plant maintenance / ERP (pivoting) • Unauthorized access to additional ICS assets (pivoting)

Tabella 4.3: Tabella delle possibili minacce che possono colpire un sistema SCADA, specificati a partire dall'obiettivo da attaccare (fonte: [24])

Target	**Possible Attack Vectors**	**Possible Attack Methods**	**Possible Consequences**
Condition monitoring system	• Subcontractor Laptop • Maintenance Remote Access • Plant (maintenance) network • Software vendor support portal	• Exploitation of unpatched application • Installation of malware via unvalidated vendor software • Network access via unsecure access points (compressor / pump house) • Remote Access VPN via stolen or compromised subcontractor laptop • Remote Access VPN via compromise of maintenance vendor site • Remote access via "interactive" accounts • Database injection • Insecure implementation of OPC (communication protocols)	• Equipment damage / sabotage • Plant upset / shutdown • Unauthorized access to additional ICS assets (pivoting)
Controller (PLC)	• Engineering workstation • Operator HMI • Standalone engineering tools • Rogue device in Control Zone • USB / removable media • Controller network • Controller (device) network	• Engineer / technician misuse • Network exploitation of industrial protocol • known vulnerability • Network exploitation of industrial protocol • known functionality • Network replay attack • Network DoS via communication buffer overload • Direct code / malware injection via USB • Direct access to device via rogue network (local / remote) PC with appropriate tools / software	• Manipulation of controlled process(es) • Controller fault condition • Manipulation / masking of input / output data to / from controller • Plant upset / shutdown - Command-and-control
Data historian	• Business network client • ERP data integration communication channel • Database integration communication channel • Remote user access (interactive session) • Plant network • Software vendor support portal	• Exploitation of unpatched application • Installation of malware via unvalidated vendor software • Remote access via "interactive" accounts • Database injection • Insecure implementation of required communication protocols • Exploitation of unnecessary / excessive openings on perimeter defense (firewall) due to insecure communication infrastructure between applications	• Manipulation of process / batch records • Credential leakage (business) • Credential leakage (control) • Unauthorized access to additional business assets like MES, ERP (pivoting) • Unauthorized access to additional ICS assets (pivoting)

Tabella 4.3: Tabella delle possibili minacce che possono colpire un sistema SCADA, specificati a partire dall'obiettivo da attaccare (fonte: [24])

Target	**Possible Attack Vectors**	**Possible Attack Methods**	**Possible Consequences**
Directory services	• Replication services • Print spooler services • File sharing services • Authentication services • Plant network • Software vendor support portal	• Exploitation of unpatched application(s) • Installation of malware via unvalidated vendor software • DNS spoofing • NTP Reflection attack • Exploitation of unnecessary / excessive openings on perimeter defense (firewall) due to replication requirements between servers - Installation of malware on file shares	• Communication disruptions via DNS • Authentication disruptions via NTP • Authentication disruptions via LDAP / Kerberos • Credential leakage • Information leakage • file shares • Malware distribution • Unauthorized access to ALL domain-connected ICS assets (pivoting) • Unauthorized access to business assets (pivoting)
Engineering workstations	• Engineering tools and applications • Non-engineering client applications • USB / Removable media • Elevated privileges (engineer / administrator) • Control network - Software vendor support portal	• Exploitation of unpatched applications • Installation of malware via unvalidated vendor software • Installation of malware via removable media • Installation of malware via keyboard • Exploitation of trusted connections across security perimeters • Authorization to ICS applications without sufficient access control mechanisms	• Plant upset / shutdown • Delay plant startup • Mechanical damage / sabotage • Unauthorized manipulation of operator graphics • inappropriate response to process action • Unauthorized modification of ICS database(s) • Unauthorized modification of critical status / alarms • Unauthorized distribution of faulty firmware • Unauthorized startup / shutdown of ICS devices • Process / plant information leakage • ICS design / application credential leakage • Unauthorized modification of ICS access control mechanisms • Unauthorized access to most ICS assets (pivoting / own) • Unauthorized access to business assets (pivoting)
Environmental controls	• HVAC control • HVAC (building management) network • Software vendor support portal	• Exploitation of unpatched application (building management system) • Installation of malware via unvalidated vendor software • Network access through unprotected access points • Network pivoting through unregulated network boundaries	• Disruption of cooling / heating • Equipment failure / shutdown

Tabella 4.3: Tabella delle possibili minacce che possono colpire un sistema SCADA, specificati a partire dall'obiettivo da attaccare (fonte: [24])

Target	**Possible Attack Vectors**	**Possible Attack Methods**	**Possible Consequences**
Fire detection and suppression system	• Fire alarm / evaluation • Fire suppressant system • Building management network • Software vendor support portal	• Exploitation of unpatched application (building management system) • Installation of malware via unvalidated vendor software • Network access through unprotected access points • Network pivoting through unregulated network boundaries	• Unauthorized release of suppressant • Equipment failure / shutdown
Master and/or slave devices	• Unauthorized / Unvalidated firmware • Weak communication problems • Insufficient authentication for "write" operations • Control network • Device network	• Distribution of malicious firmware • Exploitation of vulnerable industrial protocols via rogue PC on network (local / remote) • Exploitation of vulnerable industrial protocols via compromised PC on network (local) • Exploitation of industrial protocol functionality via rogue PC on network (local / remote) • Exploitation of industrial protocol functionality via compromised PC on network (local) • Communication buffer overflow via rogue PC on network (local / remote) • Communication buffer overflow via compromised PC on network (local)	• Plant upset / shutdown • Delay plant start • Mechanical damage / sabotage • Inappropriate response to control action • Suppression of critical status / alarms
Operator workstation (HMI)	• Operational applications (HMI) • non-SCADA client applications • USB / Removable media - Elevated privileges (administrator) • Control network - Software vendor support portal	• Exploitation of unpatched applications • Installation of malware via unvalidated vendor software • Installation of malware via removable media • Installation of malware via keyboard • Authorization to ICS HMI functions without sufficient access control mechanisms	• Plant upset / shutdown • Suppression of critical status / alarms • Product quality • Plant / process efficiency • Credential leakage (control) • Plant / operational information leakage • Unauthorized access to ICS assets (pivoting) • Unauthorized access to ICS assets (communication protocols)

Tabella 4.3: Tabella delle possibili minacce che possono colpire un sistema SCADA, specificati a partire dall'obiettivo da attaccare (fonte: [24])

Target	**Possible Attack Vectors**	**Possible Attack Methods**	**Possible Consequences**
Patch management servers	• Software patches / hotfixes • Patch management software • Vendor software support portal • Business network • Plant network • Software vendor support portal	• Insufficient checking of patch "health" before deployment • Alternation of automatic deployment schedule • Installation of malicious software via trusted (supplier) media • Installation of malware via unvalidated vendor software	• Malware distribution server • Unauthorized modification of patch schedule • Credential leakage • Unauthorized access to ICS assets (pivoting)
Perimeter protection (firewall/IPS)	• Trusted connections (Business-to-Control) • Local user account database • Signature / rule updates	• Untested/unverified rules • Exploitation of unnecessary / excessive openings on perimeter defense (firewall) • Insecure office and industrial protocols allowed to cross security perimeter • Reuse of credentials across boundary	• Unauthorized access to business network • Unauthorized access to DMZ network • Unauthorized access to control network • Local credential leakage • Unauthorized modification of rulesets / signatures • Communication disruption across perimeter / boundary
SCADA servers	• Non-SCADA client applications • Application integration communication channels • Data historian • Engineering Workstation • Control network • Software vendor support portal	• Exploitation of unpatched applications • Installation of malware via unvalidated vendor software • Remote access via "interactive" accounts • Installation of malware via removable media • Exploitation of trusted connections within control network • Authorization to ICS applications without sufficient access control mechanisms	• Plant upset / shutdown • Delay plant startup • Mechanical damage / sabotage • Unauthorized manipulation of operator graphics • inappropriate response to process action • Unauthorized modification of ICS database(s) • Unauthorized modification of critical status / alarms • Unauthorized startup / shutdown of ICS devices • Credential leakage (control) • Plant / operational information leakage • Unauthorized modification of ICS access control mechanisms • Unauthorized access to most ICS assets (pivoting / own) • Unauthorized access to ICS assets (communication protocols) • Unauthorized access to business assets (pivoting)

Tabella 4.3: Tabella delle possibili minacce che possono colpire un sistema SCADA, specificati a partire dall'obiettivo da attaccare (fonte: [24])

Target	**Possible Attack Vectors**	**Possible Attack Methods**	**Possible Consequences**
Safety systems	• Safety engineering tools • Plant / emergency shutdown communication channels (DCS / SCADA) • Control (safety) network • Software vendor support portal	• Exploitation of unpatched applications • Installation of malware via unvalidated vendor software • Installation of malware via removable media • Installation of malware via keyboard • Authorization to ICS applications without sufficient access control mechanisms	• Plant shutdown • Equipment damage / sabotage • Environmental impact • Loss of life • Product quality • Company reputation
Telecommunications systems	• Public key infrastructure • Internet visibility	• Disclosure of private key via external compromise • Exploitation of device "unknowingly" connected to public networks • Network access through unmonitored access points • Network pivoting through unregulated network boundaries	• Credential leakage (control) • Information leakage • Unauthorized remote access • Unauthorized access to ICS assets (pivoting) • Command and control
Uninterruptible power systems (UPS)	• Electrical management network • Vendor / subcontractor maintenance	• Exploitation of unpatched application (building management system) • Installation of malware via unvalidated vendor software • Network access through unprotected access points • Network pivoting through unregulated network boundaries	• Equipment failure / shutdown • Plant upset / shutdown • Credential leakage • Unauthorized access to ICS assets (pivoting)

Tabella 4.3: Tabella delle possibili minacce che possono colpire un sistema SCADA, specificati a partire dall'obiettivo da attaccare (fonte: [24])

Target	**Possible Attack Vectors**	**Possible Attack Methods**	**Possible Consequences**
User-ICS engineer	• Social engineering • Corporate assets • Social engineering • Personal assets • E-mail attachments • File shares	• Introduction of malware through watering hole or spear-phishing attack on business PC • Introduction of malware via malicious email attachment on business PC from trusted source • Introduction of malware on control network via unauthorized / foreign host • Introduction of malware on control network via shared virtual machines • Introduction of malware via inappropriate use of removable media between security zones (home-business-control) • Propagation of malware due to poor segmentation and "full visibility" from EWS • Establishment of C2 via inappropriate control-to-business (outbound) connections	• Process / plant information leakage • ICS design / application credential leakage • Unauthorized access to business assets (pivoting) • Unauthorized access to ICS assets (pivoting / own)
User-ICS technician	• Social engineering • Corporate assets • Social engineering • Personal assets • E-mail attachments • File shares	• Introduction of malware on control network via connection of unauthorized / foreign host • Introduction of malware on control network via shared virtual machines • Introduction of malware via inappropriate use of removable media between security zones (home-business-control) • Exploitation of applications due to unnecessary use of administrative rights • Network disturbances resulting from connection to networks with poor segmentation	• Plant upset / shutdown • Delay plant startup • Mechanical damage / sabotage • Unauthorized manipulation of operator graphics • inappropriate response to process action • Unauthorized modification of ICS database(s) • Unauthorized modification of status / alarms settings • Unauthorized download of faulty firmware • Unauthorized startup / shutdown of ICS devices • Design information leakage • ICS application credential leakage • Unauthorized access to most ICS assets (pivoting / own)

Tabella 4.3: Tabella delle possibili minacce che possono colpire un sistema SCADA, specificati a partire dall'obiettivo da attaccare (fonte: [24])

Target	**Possible Attack Vectors**	**Possible Attack Methods**	**Possible Consequences**
User-plant operator	• Keyboard • Removable media • USB • Removable Media • CD / DVD	• Introduction of malware on control network via unauthorized / foreign host • Introduction of malware via inappropriate use of removable media between security zones (home-business-control) • Exploitation of applications due to unnecessary use of administrative rights	• Plant upset / shutdown • Mechanical damage / sabotage • Unauthorized startup/-shutdown of mechanical equipment • Process / plant operational information leakage • Credential leakage • Unauthorized access to ICS assets (pivoting) • Unauthorized access to ICS assets (communication protocols)

Capitolo 5

SimSCADA: design e implementazione

In questo capitolo verranno esposte le motivazioni che hanno portato allo sviluppo di SimSCADA. Inizialmente verranno illustrate le idee che si trovano alla base dello sviluppo del gioco. In seguito verranno mostrate le scelte di design e progettazione prese durante l'implementazione del progetto. Verranno descritte alcune delle dinamiche base presenti nel gioco e come sono state sviluppate. Verrà illustrato quali lezioni sono state inserite nel gioco e come possono essere apprese dal giocatore. Infine verrà mostrato come è stato realizzata la raccolta dati per monitorare e valutare le prestazioni del giocatore.

5.1 Scelte di game design

5.1.1 Perché i sistemi SCADA

Il gioco è stato sviluppato come parte pratica di una tesi magistrale. Vista le difficoltà che possono nascere durante la creazione di un videogioco, si è cercato principalmente di creare un concept di ciò che potrebbe essere uno strumento di formazione in ambito di sicurezza informatica. Questa è una materia che può coinvolgerne molte altre. È stata necessaria, quindi, un'attenta analisi iniziale sia dei serious games a tema di sicurezza informatica già presenti nel mercato che di altri lavori precedentemente sviluppati (anche non commercializzati). Da questa analisi è scaturita l'assenza di uno strumento del genere applicabile in un contesto industriale e che andasse a coinvolgere i sistemi SCADA. Si è deciso quindi di rivolgere l'attenzione verso questo mondo che, a causa di una percezione influenzata ancora dalle idee del passato, viene ancora considerato come sicuro e privo di minacce, anche se gli episodi registrati negli ultimi anni descrivono un quadro ben diverso.

Il mondo dei sistemi SCADA è caratterizzato da una elevata varietà di fini applicativi: si va dal sistema di controllo per impianti industriali a quelli per il controllo dei trasporti automatizzati, passando per impianti di produzione energetica (centrali nucleari, gasdotti, ecc.). Piuttosto che focalizzarsi solo su una delle molte applicazioni per sistemi del genere, si è preferito mantenere un'ambientazione più generica. Questo è stato fatto principalmente per due motivi:

- innanzitutto riprodurre solo un determinato campo in maniera fedele avrebbe limitato in qualche modo il bacino d'utenza finale;
- l'intento principale per cui è stato sviluppato il gioco è quello di cercare di educare il giocatore a prendere delle decisioni mirate ed oculate sia in maniera preventiva che al momento dell'emergenza, cercando di dare maggior importanza alle azioni che l'utente deve attuare rispetto agli effettivi strumenti utilizzati, in quanto differenti da situazione a situazione;

- per aiutare a mantenere il tipo di ambientazione scelto, si è evitato di inserire rimandi a software e tecnologie specifici utilizzati in ambienti SCADA, in quanto questi si differenziano da situazione a situazione, a seconda del processo da controllare, l'impianto su cui si agisce e tutti gli altri elementi che sono stati elencati in 3.2.

Ai fini di non ridurre in maniera eccessiva l'utenza finale, si sono inserite all'interno del gioco delle nozioni su cosa sia un sistema SCADA ed altre informazioni relativi ad essi, oltre a tutte le lezioni riguardanti i pericoli e gli attacchi informatici a cui un sistema di questi tipo può essere sottoposto. Sono poi presenti le lezioni sugli attacchi informatici che sistemi di questo tipo possono subire e quali azioni si possono effettuare per mitigare delle situazioni potenzialmente pericolose.

5.1.2 La tipologia di gioco: tycoon

Per ottemperare all'intento prefissato di dar risalto alle azioni che l'utente deve intraprendere durante la propria partita, si è scelto di sviluppare un gioco di genere gestionale o tycoon, ispirandosi in particolar modo alla serie di titoli di questo genere sviluppati nel corso degli anni, quali "Roller Coaster Tycoon" e "SimCity". Lo scenario tipico di questi giochi prevede che l'utente si trovi a capo di una determinata realtà economica (un parco di divertimento, una città e così via, a seconda dell'ambientazione) e deve cercare di gestirla nel migliore dei modi (portandola ad esempio ad aumentare gli incassi o completando altri obiettivi imposti dal sistema), cercando di far fronte con le proprie azioni ai vari problemi che si presentano nel corso della partita.

Da questo punto di vista, si è cercato di riprenderne anche l'impostazione grafico, adottando uno stile retro, di rimando alle versioni dei suddetti titoli sviluppati a cavallo degli anni '80 e '90 (periodo caratterizzato da un massiccio sviluppo di questa tipologia di giochi). Inoltre questo stile permette di mantenere il gioco leggero, graficamente parlando, rendendolo utilizzabile anche su macchine con basse performance.

5.1.3 La piattaforma di destinazione: WebGL

La piattaforma su cui far eseguire il gioco una volta completato è stata oggetto di diverse analisi. Nella fase iniziale di sviluppo, si era deciso di implementare il gioco per piattaforma Windows, essendo questo il sistema operativo più diffuso, soprattutto in ambito di videogames. Durante lo sviluppo del gioco, però, è stato considerato che, sebbene diffusa, sviluppare il gioco solo per piattaforma Windows sarebbe stato comunque limitante per lo scopo finale del progetto. È stato deciso, quindi, di cambiare la piattaforma di destinazione, affidandosi all'utilizzo delle API WebGL.

Queste interfacce di programmazione, scritte in linguaggio JavaScript, permettono l'esecuzione di applicazioni 2D e 3D tramite web browser, senza richiedere l'utilizzo di alcun tipo di plug-in da parte dell'utente, facendo eseguire il codice relativo alla componente grafica direttamente alla GPU del sistema utente. Sono state rilasciate inizialmente nel 2011 e da allora si sono diffuse largamente, venendo integrate da altri standard web. Recentemente diversi motori grafici hanno iniziato a supportarne lo sviluppo (Unity e Unreal Engine tra i più famosi) permettendo ulteriormente il diffondersi di questa tecnologia.

Le ragioni per cui è stato effettuato il cambio della piattaforma sono diverse, quelle che hanno avuto un peso maggiore nella decisione sono state:

- la tecnologia WebGL può essere eseguita su qualsiasi sistema, a prescindere dal sistema operativo installato su di esso. Basta, infatti, l'utilizzo di un web browser che ne supporti l'utilizzo (ovviamente il sistema deve essere sufficientemente potente per garantire il corretto funzionamento del gioco, ma nel caso in questione non ci si è soffermati più di tanto su questo aspetto, in quanto il gioco presenta una grafica molto minimale che non richiede un utilizzo eccessivo di risorse da parte del sistema);

- dovendo effettuare una raccolta dati sulle performance del giocatore, in modo da poter estrapolare successivamente i dati che permettano di capire se l'apprendimento del giocatore sta avvenendo in maniera corretta, utilizzare una tecnologia web, come appunto WebGL, permette una gestione più semplice di questo aspetto. Infatti, mentre il codice grafico è eseguito in locale, il codice di controllo del gioco viene eseguito dal server su cui è stato caricato. Per la raccolta dati, quindi, è stato sufficiente creare i file necessari all'interno del server e collezionarli, senza richiedere alcun tipo di azione da parte del giocatore.

5.1.4 Il motore di gioco: Unity

La scelta della piattaforma di sviluppo è stata presa senza troppi dubbi, contrariamente agli altri aspetti fin'ora esposti. Subito, infatti, è stato deciso di sviluppare il gioco grazie all'ausilio del software Unity, un motore grafico multipiattaforma, che permette la creazione di applicazioni grafiche sia 2D che 3D. Inizialmente si è tentato di approcciare lo sviluppo del gioco tramite l'utilizzo del framework Phaser.io, ma questa idea è stata rapidamente scartata per diversi motivi tra cui:

- Phaser.io è un framework di gioco nato di recente e creato unicamente per applicazioni 2D. Unity, d'altro canto, è un motore grafico in circolazione da molto più tempo e che permette di sviluppare applicazioni sia in 2D che in 3D. Dato che lo stile grafico del gioco è stato deciso solamente in un secondo momento, avere una piattaforma di sviluppo che non ponesse dei limiti da questo punto di vista è stato decisivo;
- il fatto che Unity abbia un tempo di vita maggiore, fa si che esistano molti più asset grafici, sviluppati da terze parti, liberamente utilizzabili e che hanno permesso di velocizzare lo sviluppo in alcuni momenti;
- Unity permette di cambiare la piattaforma di destinazione, a prescindere dal codice che si è prodotto fino a quel momento. Questa funzionalità è stata presa in considerazione nella scelta ed è stata, come si è visto, sfruttata durante lo svolgimento del lavoro;
- Unity permette di sviluppare codice in diversi linguaggi di programmazione (C, C++, C#) mentre Phaser.io è scritto solamente per linguaggio JavaScript. Data la maggior familiarità dello sviluppatore con il linguaggio C# (utilizzato poi nello sviluppo del codice), Unity è sembrata la scelta più adatta per iniziare in maniera più veloce lo sviluppo del gioco.

5.1.5 L'IDE: Visual Studio

Per quanto riguarda l'IDE di sviluppo, la scelta di Unity come motre grafico ha condizionato in maniera decisiva la scelta di quest'ultimo. Sebbene, infatti, sia possibile utilizzare qualsiasi IDE che permette lo sviluppo di codice C#, Visual Studio, a differenza degli altri software, possiede diverse integrazioni con il motore in questione che sono di supporto nello sviluppo, rendendo di fatto questa scelta quasi obbligata.

5.2 La raccolta dati

Prima di iniziare lo sviluppo del gioco, è stata affrontata una fase di raccolta dati. Lo scopo era quello di individuare quale fosse il modo migliore per rappresentare l'ambiente SCADA all'interno di un videogioco. In particolare si voleva evitare di rendere il gioco troppo tecnico e complesso da punto di vista operativo, in quanto avrebbe limitato, come detto, il bacino di utenza finale e sarebbe stato eventualmente di difficile comprensione per il giocatore.

L'idea di gioco iniziale era quella di creare un simulatore di un sistema SCADA. Il giocatore avrebbe interagito con esso tramite l'interfaccia HMI e avrebbe dovuto fronteggiare diversi tipi di attacchi informatici. Per questo motivo sono stati presi in esame una serie di software per interfacce HMI, tra cui WinLog CC e STEP7. Quest'ultimo, in particolare è sviluppato dalla

Siemens e fa parte della famiglia di prodotti Simatic, comprendente sia software che hardware per il controllo di sistemi di automazione. Analizzati i software, però si è capito che emulare un sistema SCADA sarebbe stato complesso, in quanto era necessario dover programmare i diversi PLC che avrebbero controllato il processo (emulato anch'esso tramite computer).

Scartata quindi l'ipotesi dell'emulatore, si è deciso di porre il focus del gioco sulle diverse minacce che possono coinvolgere un sistema SCADA. Dalle ricerche svolte, infatti, sugli attacchi informatici effettuati ai danni di un sistema di controllo, è emerso che esiste ancora l'errata convinzione di ritenere l'ambiente SCADA sicuro dagli attacchi esterni in quanto tecnologicamente diverso, senza rendersi conto che, in realtà, con il passare degli anni le tecnologie del mondo domestico sono entrate a far parte di quello industriale, rendendoli molto più simili di quanto si pensi.

Partendo da ciò, è stata formulata l'idea di un gioco gestionale in cui l'utente si trova a gestire la sicurezza di un ambiente SCADA, fronteggiando diversi tipi di attacchi e cercando di prevenire il fallimento economico per colpa di essi.

5.3 La storia nel gioco

Il gioco si sviluppa all'interno di una società fornitrice di servizi SCADA. Il giocatore è stato appena messo a capo del reparto di sicurezza e dovrà gestire i vari aspetti legati ad essa. Quindi non dovrà solo evitare di subire attacchi informatici, ma avrà il compito anche di scovare eventuali infiltrati tra i dipendenti che si troveranno a lavorare all'interno dei locali da proteggere.

Ma mano che riuscirà a difendere i sistemi aziendali, il giocatore guadagnerà soldi e aumenterà la propria reputazione. Questo gli permetterà di acquistare migliorie difensive per aiutarlo nel proprio compito. Una volta raggiunto un livello di reputazione e soldi sufficientemente alto, lo scenario si concluderà con la vittoria e si potrà accedere la secondo livello, in cui dovrà fronteggiare minacce più pericolose. Se, invece, non dovesse riuscire a proteggere l'azienda dagli attacchi e subisse troppi attacchi, con relative perdite economiche, il gioco terminerà con una sconfitta e il giocatore dovrà ricominciare lo scenario dall'inizio.

La storia è stata mantenuta molto basilare in quanto, tipicamente, nei giochi gestionali non ha una particolare rilevanza. In questo tipo di giochi vi sono, usualmente, degli obiettivi che il giocatore può completare nel corso del gioco (raggiungere un determinato livello di introiti economici, effettuare un determinato numero di azioni positive, ecc.), ma in questo caso sono stati estromessi nella programmazione per questioni legate al tempo necessario per idearle ed implementarle.

5.4 Struttura del gioco

Il gioco è ambientato in un'unica mappa, al cui interno si svolgono tutte le azioni del gioco. La mappa è divisa in due locali principali: la stanza server e la stanza degli operatori. Durante lo svolgimento della partita verranno generati automaticamente dei personaggi, raffiguranti i dipendenti dell'azienda. Questi avranno dei lavori da svolgere, la cui durata è indicata da una progress bar presente sopra la testa del personaggio, e il giocatore dovrà impedire che venga interrotto. Se ci riuscirà, riceverà un bonus economico, altrimenti subirà una perdita di soldi.

Non tutti i lavoratori, però, sono reali: fra di essi, infatti, possono nascondersi dei personaggi ostili, il cui compito sarà quello di effettuare un attacco nei confronti dell'azienda. In questo caso il giocatore dovrà fermarli prima che riescano nel loro intento, altrimenti subirà una perdita economica e, se necessario, dovrà rimettere in funzione il sistema. Per individuare le minacce, il giocatore potrà interagire con i vari personaggi presenti nello scenario e avrà la possibilità di avvalersi di aiuti acquistabili tramite lo store interno.

Oltre alle minacce fisiche, il giocatore dovrà difendersi anche dalle minacce remote, il cui attacco può avvenire in qualunque momento. Per evitarle, sarà compito del giocatore attivare le

difese da remoto (firewall, ids, ecc.) e migliorarne il tasso di efficacia, sempre tramite acquisti nello store.

Il giocatore, oltre che con i personaggi, potrà interagire anche con altre attrezzature (pc, server, ecc.) tramite cui eseguire diverse azioni, sia preventive, ma anche di riparazione, nel caso in cui non sia riuscito a sgominare una minaccia in tempo.

Un'altra delle componenti che sarà d'aiuto durante la partita è, come anticipato, lo store: tramite esso, il giocatore potrà acquistare nuove difese, migliorare quelle che possiede già e, inoltre, eseguire campagne di assunzione del personale, in modo da aumentare gli introiti dell'azienda.

5.5 Game tutorial

Prima di iniziare a giocare, viene presentata l'opportunità di svolgere un tutorial. Questo serve, come in questo tipo di giochi, a presentare le dinamiche base del gioco, mostrare con quali oggetti è possibile interagire, oltre a fornire nozioni sulle minacce da cui dovrà difendersi.

Il tutorial è parte integrante dell'esperienza di formazione fornita dal gioco, in quanto sono presenti anche delle informazioni sui sistemi SCADA e cosa siano. Questo è stato necessario in quanto non conoscendo in anticipo la formazione del giocatore sull'argomento è importante che i concetti base siano esposti. Questi poi saranno ampliati nel corso della partita, grazie anche alle lezioni inserite nel gioco.

5.6 Le lezioni

Il gioco è suddiviso in due livelli, creati in modo che al passaggio dal primo al secondo la difficoltà aumenti. Durante la partita al giocatore verranno impartite delle lezioni in relazione alle minacce che subirà. Ognuna di esse, infatti, potrà essere effettuata attraverso un diverso tipo di attacco: attacco Man in the Middle, phishing, malware e così via, fino ad arrivare anche ad attacchi complessi e/o combinati. Ogni volta che il giocatore subirà un nuovo attacco, verrà subito mostrata la lezione corrispondente in cui saranno anche esposte le misure difensive che dovrà intraprendere.

Essendo diviso in due livelli, nel primo gli attacchi che subirà saranno più semplici, mentre nel secondo sono introdotti anche attacchi complessi, a cui il giocatore dovrà reagire con una combinazione di diverse azioni difensive.

Oltre alle lezioni sugli attacchi, sono fornite anche lezioni più approfondite su cosa sia un sistema SCADA, sulle componenti che possono trovarsi in un'architettura tipica e sulle difese che è possibile implementare per proteggerlo.

Le lezioni sono accessibili in ogni momento, sia all'interno del gioco (interagendo con il telefono sarà possibile aprire il quaderno al cui interno sono raccolte tutte le lezioni), sia dal menù principale del gioco. In questo modo, se ne dovesse avere bisogno, l'utente potrà rivedere le nozioni di teoria durante la partita, ma anche ripassarle in un secondo momento.

5.7 Gestione degli asset di gioco

Tutti gli asset grafici e audio del gioco sono gestiti automaticamente da Unity, così come il loro caricamento. Una volta importati dentro l'editor, infatti, è possibile gestirne il caricamento tramite la classe `Resources`, in grado di gestire diversi tipi di file multimediali e testuali. Tra gli asset utilizzati sono ad annoverare anche i file di testo al cui interno sono trascritte le lezioni, quelli con i messaggi di sistema, così come i file `.json` utilizzati per creare le varie classi create per il gioco.

Altri asset caricati sempre tramite la classe `Resources` sono le immagini utilizzate per raffigurare i personaggi all'interno della mappa e quelle necessarie per raffigurare le loro animazioni di movimento.

5.8 Raccolta dati giocatore

Durante il gioco, sono raccolte delle statistiche sulle azioni del giocatore. Principalmente sono monitorati due parametri: i tempi di reazione nel momento in cui l'utente intraprende delle azioni di risposta all'esecuzione di una minaccia e le azioni di prevenzione eseguite durante il corso di tutta la partita.

Nel primo caso, lo scopo è di osservare innanzitutto se il giocatore riesce a migliorare i tempi di reazione man mano che la partita procede. Questi tempi sono registrati e illustrati in una tabella, osservabile sia durante la partita che alla fine della stessa. L'andamento dei tempi di risposta, inoltre, può fornire un'informazione sulla qualità dell'apprendimento. Se i fatti i tempi per una determinata tipologia di attacco rimangono invariati nel corso del tempo (o addirittura aumentano) si può presupporre che il giocatore abbia difficoltà nel capire in fondo quel tipo di attacco e saprà quale lezione rivedere per colmare le proprie lacune.

La registrazione delle azioni di risposta, invece, viene effettuata per capire in quale modo il giocatore si comporta nel momento di gioco che intercorre tra una minaccia e l'altra (ossia quei momenti in cui gli attacchi sono in preparazione). Questa registrazione è effettuata in relazione alla "minaccia in tendenza". Periodicamente, infatti, sarà scelto un particolare tipo di attacco che sarà di tendenza per un determinato lasso di tempo. Se il giocatore effettuerà delle azioni preventive (acquisti, attivazione o disattivazione di determinati sistemi di difesa) utili a contrastare quel tipo di minaccia è lecito presupporre che abbia recepito la lezione inerente ad essa, altrimenti è probabile che debba rivedere le informazioni presenti all'interno del gioco.

Un'altra tipologia di azione registrata è quella di "contrasto" alle minaccia già eseguite. In questa situazione, a seconda dell'attacco subito, verranno analizzate le scelte effettuate dal giocatore per porre rimedio al danno. Anche in questo caso, se le scelte effettuate risultano non in linea con quelle richieste per fermare quel determinato tipo di attacco, è probabile che il giocatore debba rivedere le lezioni inerenti ad esso.

Tutte queste informazioni sono raccolte e scritte nel log del giocatore, creato e salvato all'interno del server. Questo file di testo viene costantemente aggiornato sia con queste informazioni, ma anche con gli altri eventi generati dal gioco.

Capitolo 6

Manuale del programmatore

In questo capitolo sono inserite tutte le informazioni necessarie per chiunque voglia modificare la struttura del gioco, aggiungendo livelli, minacce, messaggi e quant'altro. Nella prima parte è esposta la modalità di installazione del gioco, in modo da riuscire ad eseguirlo in locale con fini di sviluppo 6.1. Successivamente, nei paragrafi 6.2, 6.3 e **??** sono fornite informazioni riguardo sia i file di codice che non, utilizzati all'interno del gioco e su cui eventualmente agire per apportare modifiche riguardo la logica del gameplay. In **??** è mostrata la struttura delle cartelle di gioco, esposta così come appare all'interno dell'editor di Unity. Infine, nell'ultima sezione (6.4) è esposto come è possibile aggiungere dei livelli al gioco.

6.1 Guida d'installazione

Il gioco, al momento della stesura di questo report, non è da installare ed è disponibile all'indirizzo `http://simscada.sfcoding.com/SIMSCADA/`. Se il sito di hosting non dovesse essere più raggiungibile, è possibile eseguirne una versione in un server locale. Questo può essere utile anche per effettuare test in caso di modifiche del codice di controllo, aggiunte di livelli e così via. Per far sì che il gioco funzioni in locale, però, sono necessari alcuni passaggi.

Installazione di un server di gioco Per eseguire il gioco, è sufficiente avere un web server Apache installato sulla propria macchina e copiare i file prodotti da Unity al momento della build, ossia il contenuto delle cartelle `Build` e `TemplateData`, più il file `index.html`, all'interno della cartella `htdocs` del server. Esistono diversi metodi per installare un server Apache, a seconda anche del sistema operativo su cui si sta lavorando, ma una soluzione veloce, semplice e soprattutto valida per tutti i casi è quella fornita dal software `XAMPP`.

Questo software, disponibile al link `https://www.apachefriends.org/it/download.html`, è disponibile per Windows, MacOS e Linux e permette di installare velocemente tutte le componenti necessarie per eseguire un web server sulla propria macchina in locale.

Una volta terminata la procedura d'installazione di `XAMPP`, sarà necessario individuare la cartella dove sono stati salvati i file. Normalmente, quelle predefinite sono:

Windows: C:/xampp

Linux: /opt/xampp

MacOS: /Applications/XAMPP/xamppfiles

Creazione della cartella di gioco Trovata la cartella di installazione di `XAMPP`, sarà necessario individuare la cartella`htdocs`. Al suo interno è consigliato creare un'ulteriore cartella, denominata ad esempio `SIMSCADA`. Qui dovranno essere copiati tutti i file prodotti da Unity al momento della build.

Download del progetto Tutti i file del progetto Unity sono salvati nella repository GitHub raggiungibile all'indirizzo https://github.com/serranda/SecuritySeriousGame. È possibile ottenerne una copia sia utilizzando il comando `git clone https://github.com/serranda/SecuritySeriousGame` (per utilizzare questo comando è necessario installare il client Git nel proprio sistema, per questo passaggio si rimanda alle guida ufficiali presenti nel sito di Git, https://git-scm.com/downloads), ma anche tramite il pulsante download presente nella pagina della repository (in questo caso verrà effettuato il download di un file .zip. Una volta ottenuta, sarà necessario estrarne il contenuto)

Installazione di Unity Per installare l'editor di Unity sul proprio sistema, il modo più semplice è quello di utilizzare lo Unity Hub creato per aiutare nella gestione delle installazioni e dei progetti creati con l'editor. Questo programma può essere ottenuto vistando il forum ufficiale al link https://forum.unity.com/threads/unity-hub-v2-0-0-release.677485/. Una volta installato Unity Hub, è possibile scegliere la versione dell'editor che si vuole installare. Per questo progetto è consigliato utilizzare le versioni 2018.4.x LTS, in quanto sono quelle che garantiscono la compatibilità con tutti i file del progetto. Versioni precedenti non sono utilizzabili, mentre per quanto riguarda quelle successive non se ne garantisce il corretto funzionamento con il progetto intero. Al momento dell'installazione dell'editor è importante selezionare di voler installare anche il modulo aggiuntivo denominato `WebGL Build Support`, con cui sarà possibile creare la build effettiva del gioco. Una volta installato l'editor, è possibile importarvi al suo interno i file del progetto, contenuti all'interno della cartella `SIMSCADA`.

Creazione della build Una volta importati i file del progetto, è possibile aprirlo all'interno dell'editor. Se si vogliono apportare delle modifiche, è possibile farlo. Una volta terminate, è necessario creare la build finale, da caricare successivamente nel server locale creato tramite XAMPP. Per fare ciò, bisogna selezionare la voce `Build Settings`, contenuta all'interno del tab `File`. A questo punto si aprirà una finestra in cui sarà necessario impostare quali scene si vogliono includere nella build finale e per quale piattaforma effettuarla. Occorrerà quindi selezionare la piattaforma WebGL dal menù posto sulla sinistra della finestra e premere sul tasto `Build`. A questo punto, dopo aver scelto la cartella all'interno di cui verranno generati tutti i file, la compilazione inizierà e sarà necessario attenderne il termine.

Copia della build nel server Terminata la compilazione, è possibile prenderne il contenuto e spostarlo all'interno della cartella `SIMSCADA`, creata precedentemente nella directory `htdocs` di `XAMPP`. È necessario, inoltre, spostare anche la cartella `PHP` (situata all'interno della directory del progetto ottenuta tramite GitHub), contenente i file degli script PHP necessari per eseguire operazioni sul server. Una volta eseguite queste azioni è possibile utilizzare un qualsiasi web browser che supporti l'utilizzo di WebGL (per controllare che il proprio browser supporti l'esecuzione di applicazioni WebGL è possibile collegarsi al sito https://get.webgl.org/ ed effettuare il test) e collegarsi all'indirizzo localhost/SIMSCADA/. Il gioco partirà non appena terminato il caricamento della pagina.

6.2 Contenuti del gioco senza codice

Diverse risorse del gioco sono definite attraverso file di testo e json in modo da agevolare le operazioni di gestione durante la partita e/o eventuali loro modifiche durante lo sviluppo. Questi file sono contenuti all'interno della cartella `Resources` della directory del progetto. Questo si è reso necessario in modo da poter sfruttare i metodi della classe `Resources` e caricare velocemente il contenuto dei file all'interno del gioco. Di seguito saranno descritte le strutture di queste risorse più nel dettaglio.

Di queste risorse sono stati creati anche dei template a cui far riferimento, nel caso in cui se ne voglia aggiungere di nuovi. Questi sono situati sempre nella directory del progetto Unity.

```
{
        "head": <MessageHead>,
        "bodyPath": <MessageBodyPath>,
        "backBtn": <MessageBackBtn>,
        "nextBtn": <MessageNextBtn>
}
```

Figura 6.1. Contenuto del file json da cui sono estratte le proprietà della classe `DialogBoxMessage`.

6.2.1 Messages

All'interno del gioco sono presenti diversi messaggi di sistema rivolti all'utente, in modo da informarlo sugli avvenimenti della partita, se sta commettendo qualche azione errata o, ancora, per fornire dei suggerimenti. Vi sono poi i messaggi del tutorial, tramite cui si danno al giocatore le informazioni necessarie per apprendere le dinamiche basilari del gameplay. Ognuno di questi messaggi è creato partendo dal contenuto di due file, un file json ed un file txt.

Nel file json sono contenuti i valori delle proprietà della classe con cui si definiscono i messaggi all'interno del codice (un esempio di file json di questo genere è mostrao nella figura 6.1). La classe è stata denominata `DialogBoxMessage` ed il suo contenuto è riportato nella figura 6.2.

Come è possibile vedere, un oggetto della classe `DialogBoxMessage` è composto da cinque proprietà. Le prime quattro 4 sono fornite dal file json e sono:

- `head`, ossia il testo riportato nel titolo del messaggio;
- `bodyPath`, ossia il percorso dove è salvato il testo del messaggio;
- `backBtn`, ossia il testo che viene riportato in uno dei due pulsanti con cui il giocatore può interagire (usualmente con questo valore si vuole indicare il pulsante che permette di tornare indietro o annullare un'eventuale azione proposta dal sistema);
- `nextBtn`, ossia il testo che viene riportato nell'altro pulsante (con questo valore, al contrario, si vuole indicare il pulsante che permette procedere con l'azione proposta dal gioco).

Queste proprietà sono assegnate durante la creazione dell'oggetto, resa possibile grazie al metodo `FromJson` della classe `JsonUtility`, messa a disposizione dalle API di Unity (6.3).

La quinta proprietà, `body`, come si può vedere la codice della classe riportato in figura 6.2, viene creata automaticamente a partire dalla proprietà `bodyPath`. Come anticipato, `bodyPath` contiene il valore del percorso in cui è salvato il file txt in cui è riportato il testo completo del messaggio. È stato deciso, infatti, di non inserire l'intero contenuto del messaggio nel file json, ma di salvarlo in un file a parte, tenendo nota però della sua posizione (relativa al working tree del progetto di Unity).

Questa strategia, applicata anche per i file delle `Lesson` (come si vedrà successivamente in 6.2.2) è stata ideata per evitare di incappare in errori durante la fase di deserializzazione del file json. In questo modo, inoltre, è possibile modificare il contenuto del messaggio con un qualsiasi editor di testo (è importante che la codifica del file txt sia impostata in `UTF-8`, altrimenti l'editor di Unity non riconsocerà il contenuto e darà un errore al momento della creazione dell'oggetto) senza dover cambiare la struttura del file json.

6.2.2 Lesson

Anche per le lezioni è stato utilizzato un approccio simile a quello visto per i `DialogBoxMessage`. In questo caso le proprietà definite per la classe sono solamente tre (6.5), di cui quelle riportate nel file json sono `id` e `textPath` (6.4).

```
using UnityEngine;

public class DialogBoxMessage
{
    public string head;
    public string bodyPath;
    public string backBtn;
    public string nextBtn;

    public string body =>
        string.IsNullOrEmpty(bodyPath)
            ? string.Empty
            : Resources.Load<TextAsset>(bodyPath).text;
}
```

Figura 6.2. Contenuto della classe `DialogBoxMessage`.

```
DialogBoxMessage dialogBoxMessage =
    JsonUtility.FromJson<DialogBoxMessage>(jsonFile.text);
```

Figura 6.3. Codice tramite cui sono creati gli oggetti della classe `DialogBoxMessage`.

Come nel caso dei `DialogBoxMessage`, anche il testo delle `Lesson` è salvato in un file di testo a parte, la cui posizione è memorizzata con la proprietà `textPath`. Questa verrà poi utilizzata per generare, al momento della creazione dell'oggetto, il valore della proprietà `textBody`.

L'altra proprietà, `id`, è utilizzata per identificare l'argomento associato alla lezione in questione. Il suo valore, inoltre, è impiegato per rinominare i pulsanti tramite cui è possibile selezionare la lezione a cui si vuole accedere (6.6)

6.2.3 ItemStore

`ItemStore` è la classe tramite cui sono definiti gli oggetti acquistabili tramite il negozio. L'approccio è lo stesso utilizzato per gli altri oggetti e sfrutta, come visto, un file json per creare l'oggetto della classe `ItemStore` e un file txt contenente la descrizione dell'oggetto da visualizzare nel gioco. Gli `ItemStore`, però, posseggono ulteriori proprietà (figure 6.7 e 6.8), qui di seguito analizzate:

- `name`, ossia il nome con cui è identificato un oggetto. Al momento della creazione della lista di `ItemStore`, è la proprietà utilizzata per ordinare la collezione di oggetti. Questo è possibile grazie all'implementazione di un classe a parte, `NameRelationalComparer`. Questa è una classe, derivata dall'interfaccia `IComparer<T>`, esposta tramite l'utilizzo di una proprietà statica (6.9), che permette di ordinare una lista di `ItemStore` tramite il metodo `List<T>.Sort` (6.10). Per il caso in questione è stata scelta come termine di paragone il solo nome dell'oggetto, mo ovviamente è possibile implementare altre classi che tengano conto di più proprietà;

- `descriptionPath`, stringa tramite cui è memorizzato il percorso di salvataggio del fil txt contenete la descrizione dell'oggetto;

```
{
  "id":"LessonID",
  "textPath":"LessonBodyPath"
}
```

Figura 6.4. Contenuto del file json da cui sono estratte le proprietà della classe `Lesson`.

```
using UnityEngine;

public class Lesson
{
    public string id;
    public string textPath;

    public string textBody =>
        string.IsNullOrEmpty(textPath)
            ? string.Empty
            : Resources.Load<TextAsset>("LessonsBody/" + textPath +
                "_IT").text;
}
```

Figura 6.5. Contenuto della classe `Lesson`.

- `imageName`, stringa contenete il nome dell'immagine associata all'oggetto. L'immagine, successivamente, verrà caricata sempre tramite la classe `Resources`. Questa proprietà, attualmente, non è ancora utilizzata, ma lo sarà in un possibile futuro sviluppo del gioco (si rimanda alla sezione 8.2);
- `price`, valore tramite cui è impostato il prezzo iniziale dell'oggetto. Ad ogni acquisto subirà un incrementato del 10%. La modifica del prezzo non è memorizzata nel file json originale, si rimanda alla sezione 6.3.6 per ulteriori spiegazioni;
- `finalLevel`, ossia il valore del livello finale a cui può arrivare un oggetto dopo essere stato potenziato al massimo. Per alcune tipologie di oggetti, come ad esempio la campagna di assunzione per aumentare il numero di dipendenti dell'azienda, non è previsto un limite di livello. In questo caso il valore della proprietà sarà di -1;
- `currentLevel`, il livello di potenziamento a cui si trova attualmente l'oggetto. Il valore di partenza è 0 ed è aumentato ad ogni acquisto, a meno che non si tratti di oggetti senza limite di potenziamento, nel cui caso rimarrà fisso al valore iniziale;
- `effect`, il valore tramite cui è identificato quale effetto produce l'acquisto dell'oggetto. Ad ogni oggetto è assegnato un valore univoco di questa proprietà. L'applicazione dell'effetto è gestita tramite l'utilizzo di uno statement `switch` che, in base al valore assegnato all'effetto, esegue il codice associato al `case` corrispondente;
- `itemObject`, ossia l'istanza del `GameObject` associato ad un `ItemStore`. Questa proprietà non viene sfruttata in fase di creazione dell'oggetto, ma è in realtà necessaria per permettere la corretta memorizzazione degli `ItemStore` durante la fase di salvataggio. Per approfondimenti si rimanda alla sezione 6.3.6;
- `threatAffinity`, ossia una lista contenente i tipi di minaccia contro di cui l'`ItemStore` in questione garantisce protezione. Questa proprietà è utilizzata durante la fase di analisi delle performance del giocatore (si rimanda alla sezione 6.3.7).

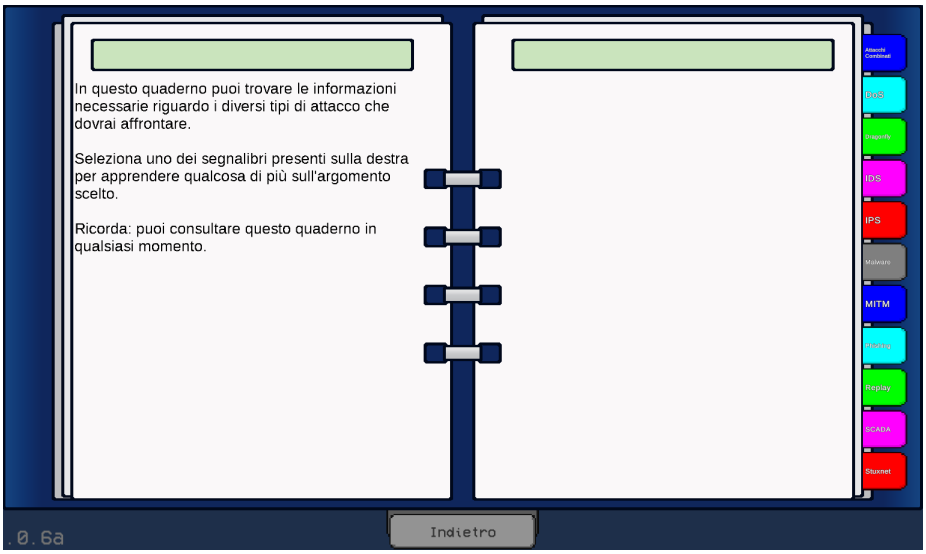

Figura 6.6. Screenshot del quaderno presente nel gioco, tramite cui è possibile accedere alle `Lesson`. Sulla destra sono visibili i pulsanti, raffigurati come dei segnalibri, tramite cui è possibile acceder al testo inerente alla `Lesson` desiderata.

```
{
        "name": "",
        "descriptionPath": "",
        "imageName": "",
        "price": 0,
        "finalLevel": 0,
        "currentLevel" : 0,
        "effect" : 0,
        "itemObject" : {},
        "threatAffinity" : []
}
```

Figura 6.7. Contenuto del file json da cui sono estratte le proprietà della classe `ItemStore`.

6.3 Classi e script di controllo

La programmazione di un'applicazione tramite Unity si basa sull'utilizzo di script di codice, eventualmente associabili a dei `GameObject`, che vengono eseguiti durante il gioco. Questo è possibile tramite le varie funzioni delle API messe a disposizione da Unity. Per l'elenco completo di tutte le funzioni si rimanda al manuale disponibile al link `https://docs.unity3d.com/ScriptReference/index.html`. È comunque possibile accedere tramite l'editor ad una copia di questo manuale, specifica per la versione di Unity che si sta utilizzando: è sufficiente aprire il tab `Help` e fare click sulla voce `Scripting Reference`.

Tutti gli script creati con Unity devono derivare dalla classe base `MonoBehaviour`, in modo da rendere possibile l'utilizzo delle funzioni evento associate ad essa. Ad ogni frame, Unity scorre tutti gli script attivi in scena, e se trova una di queste funzioni predefinite, la chiama (passando quindi il controllo alla funzione). Al termine dell'esecuzione, il controllo viene restituito a Unity.

Nelle prossime sezioni verranno analizzati alcuni degli script creati per le gestione delle componenti di gioco con cui l'utente può interagire durante una partita (6.3.2, 6.3.4, 6.3.1) e quelli utilizzati per gestire la logica di gioco (6.3.5), il salvataggio dei dati (6.3.6) e l'analisi delle prestazioni del giocatore (6.3.7).

```
using System;
using System.Collections.Generic;
using UnityEngine;

[System.Serializable]
public class ItemStore
{
    public string name;
    public string descriptionPath;

    public string descriptionBody =>
        string.IsNullOrEmpty(descriptionPath)
            ? string.Empty
            : Resources.Load<TextAsset>("ItemStoreBody/"+descriptionPath +
                "_IT").text;

    //public string imageName;
    public int price;
    public int finalLevel;
    public int effect;
    public int currentLevel;
    public GameObject itemObject;
    public List<string> threatAffinity;
}
```

Figura 6.8. Contenuto della classe `ItemStore`.

```
private sealed class NameRelationalComparer : IComparer<ItemStore>
{
    public int Compare(ItemStore x, ItemStore y)
    {
        if (ReferenceEquals(x, y)) return 0;
        if (ReferenceEquals(null, y)) return 1;
        if (ReferenceEquals(null, x)) return -1;
        return string.Compare(x.name, y.name, StringComparison.Ordinal);
    }
}

public static IComparer<ItemStore> nameComparer { get; } = new
    NameRelationalComparer();
```

Figura 6.9. Definizione della classe `NameRelationalComparer` derivata dall'interfaccia `IComparer<T>`.

6.3.1 `InteractiveSprite` e `IObjectListener`

Questo script è associato a tutti gli oggetti della scena con cui il giocatore può interagire (fig. 6.11). Esegue diverse funzioni, in base agli eventi che hanno luogo durante lo svolgimento della partita. La prima avviene in automatico ad ogni avvio del gioco. Questo è possibile grazie al metodo `Start`, eseguito all'attivazione dell'oggetto a cui lo script è associato. Di norma il metodo è utilizzato per impostare uno stato iniziale. In questo caso, sono eseguiti dei controlli per impostare alcune variabili e, soprattutto, per stabilire se l'oggetto deve essere attivato o meno.

```
//POPULATE ITEMLIST
if (itemList.Count == 0)
{
    foreach (TextAsset textAsset in itemAsset)
    {
        ItemStore itemStore = ItemStoreFromJson(textAsset);
        itemList.Add(itemStore);
    }
}

//SORT ITEMLIST
itemList.Sort(ItemStore.nameComparer);
```

Figura 6.10. Creazione della lista di `ItemStore` e successivo ordinamento tramite il metodo `List<T>.Sort`.

Inizialmente, infatti, gli oggetti con cui è possibile effettuare delle azioni sono limitati nel numero. Per incrementarlo è necessario acquistare l'upgrade, specifico per ogni oggetto, dal negozio. Tramite il metodo `Start`, quindi, dopo aver impostato i valori di alcune variabili, si richiama il metodo `CheckOperativeItem`. Tramite esso si controlla il nome dell'oggetto, si ricava il numero massimo di quel tipo di oggetti che possono essere attivi in scena e, in base al confronto tra il valore del limite ed il numero associato ad ogni oggetto (sono denominati in maniera sequenziale, in modo da poter ricavare facilmente il numero dell'oggetto dal suo nome), viene deciso se renderlo interattivo o meno.

Un'altra serie di funzioni svolte nello script sono quelle che servono ad interagire effettivamente con l'oggetto. In questo caso vengono sfruttate le interfacce `IPointerUpHandler`, `IPointerDownHandler`, `IPointerEnterHandler`, `IPointerExitHandler` e `IPointerClickHandler` che permettono l'implementazione dei corrispettivi metodi `OnPointerUp`, `OnPointerDown`, `OnPointerEnter`, `OnPointerExit` e `OnPointerClick` utilizzati per gestire le interazioni dell'oggetto con il puntatore del mouse. Queste funzioni rimangono inizialmente in attesa e sono eseguite a seguito di un particolare evento del mouse. Nello specifico, le azioni controllate in questo caso sono il click, l'ingresso del cursore all'interno dell'immagine dell'oggetto e la sua uscita.

Allo scatenarsi dei citati eventi, lo script svolge due azioni: la prima consiste nel cambio dell'immagine associata all'oggetto, in modo da dare l'impressione del click su di un pulsante. La seconda prevede la comparsa del menù contestuale associato all'oggetto, specifico per ognuna delle categorie di `InteractiveSprite`. Il menù è composto semplicemente da dei pulsanti, denominati nel codice come `ActionButton`, tanti quante sono le azioni effettuabili con l'oggetto selezionato. È necessario, però, impostare anche i `Listener` per ognuno di essi, in modo da permettere lo svolgimento effettivo delle azioni proposte. Per agevolare questa procedura, si è ricorso all'utilizzo di un'interfaccia, denominata `IObjectListener` che, come si può veder in 6.13, prevede l'implementazione di un unico metodo, `SetListeners`. Per ogni categoria degli oggetti (`Telephone`, `ServerPc`, `RoomPc` e `Security`) è stato implementato uno script apposito, derivato dall'interfaccia `IObjectListener`. In questo modo, al click del mouse, è possibile richiamare il metodo `SetListeners` esposto tramite l'interfaccia, senza la necessità di conoscere su quale oggetto si sta cliccando (vedi fig. 6.14).

6.3.2 `Threat`

Con la classe `Threat` si va a definire il codice relativo alle minacce da cui difendersi. Queste istanze vengono create periodicamente dal `LevelManager` in maniera casuale (per ulteriori spiegazioni vedere 6.3.5, per poi essere associate al `GameObject` di un `AI` (vedi 6.3.4).

La classe `Threat` presenta diverse proprietà (fig. 6.15) esposte qui di seguito più nel dettaglio:

Figura 6.11. Screenshot dello scenario di gioco. Le `InteractiveSprite` sono facilmente riconoscibili grazie al bordo verde che le circonda.

```
// Start is called before the first frame update
private void Start()
{
    manager = SetLevelManager();

    interactiveSprite = GetComponent<SpriteRenderer>();
    boxCollider2D = GetComponent<BoxCollider2D>();

    //get the real name fo the sprite without the index. necessary in order
        to swap the sprite
    int pos = interactiveSprite.sprite.name.IndexOf("_",
        StringComparison.Ordinal);
    realName = interactiveSprite.sprite.name.Substring(0, pos);

    spriteMaxIndex =
        Resources.LoadAll<Sprite>(Path.Combine(StaticDb.rscIntSpriteFolder,
        realName)).Length - 1;

    //SET OPERATIVE
    CheckOperativeItem();
}
```

Figura 6.12. Metodo `Start` della classe `InteractiveSprite`.

- `id`, int utilizzato per identificare in maniera univoca ogni nuova `Threat` generata. È assegnato al momento della creazione di una nuova `Threat` ed è incrementato in maniera sequenziale;

- `threatType`, enum creato per identificare il tipo di `Threat` e può assumere uno tra cinque valori predefiniti. Questi sono `local`,`remote`,`fakeLocal`,`hybrid` e `timeEvent`. Ognuno di essi contraddistingue un tipo di minaccia diverso, ad esempio `local` è utilizzato per gli attacchi che vengono effettuati dall'interno dell'azienda mentre `remote` per quelli condotti dall'esterno;

```
interface IObjectListener
{
    void SetListeners();
}
```

Figura 6.13. Codice dell'interfaccia `IObjectInterface`.

```
private void SetObjectListener()
{
    IObjectListener objectListener =
        gameObject.GetComponent<IObjectListener>();

    objectListener.SetListeners();
}
```

Figura 6.14. Metodo `SetObjectListener` implementato nello script `InteractiveSprite`.

- `deployTime`, valore che identifica il tempo (in termini di gioco) necessario perché l'attacco venga effettuato. Si può considerare anche come il tempo a disposizione del giocatore per evitare di subire quella minaccia;
- `threatAttacker`, enum utilizzato per distinguere la tipologia dell'attaccante. Può assumere i valori di `internal`, utilizzato per gli attacchi effettuati da dipendenti corrotti interni all'azienda, `external`, con cui si identificano i normali attaccanti, e `timeEvent`, utilizzato per identificare le minacce associate ai `TimeEvent` generici (vedi6.3.3);
- `threatDanger`, enum che definisce il livello di pericolosità della minaccia. In particolare, viene utilizzato per simulare il tentativo di corruzione nei confronti di un impiegato dell'azienda. Se il livello di pericolosità della minaccia è più alto del grado di resistenza del dipendente si subirà l'attacco, altrimenti verrà fermato;
- `threatAttack`, enum che identifica quale tipo di attacco è associato alla nuova `Threat` generata. I valori che definiscono gli attacchi sono i seguenti: `dos`, `phishing`, `replay`, `mitm`, `stuxnet`, `dragonfly`, `malware`, `createRemote`, `fakeLocal` e `timeEvent`. I primi sette, come si evince dai nomi dati, identificano tipi di attacchi reali, `createRemote` è utilizzato per gli attacchi `local` al cui termine però non viene effettuato alcun attacco, ma se ne genera uno di tipo `remote`, `fakeLocal` identifica le attività locali legittime (inizialmente era utilizzato per identificare degli attacchi finti, successivamente, dopo aver cambiato le modalità con cui è effettuato un attacco si è riutilizzata la definizione per le attività locali delle AI che generano profitti) e `timeEvent`, infine, è associato alle minacce utilizzate per i `TimeEvent` generici (vedi6.3.3);
- `fromCreateRemote`, bool utilizzato per identificare quelle minacce `remote` che hanno avuto origine da un attacco `createRemote` (è necessario per mostrare successivamente il messaggio relativo a questo tipo di attacco);
- `aiController` e `serializableAiController`, proprietà entrambe utilizzate per salvare le informazioni relative all'`AI` associata alla `Threat` generata. Sono utilizzate entrambe, nonostante rappresentino sostanzialmente la stessa variabile, in quanto di `aiController`, un'istanza dello script responsabile del controllo delle `AI` (vedi 6.3.4), non è possibile effettuare la serializzazione in fase di salvataggio. Viene, quindi, sostituita dalla classe `serializableAiController` con cui vengono salvat i valori delle proprietà di `aiController`, permettendone successivamente il ripristino in fase di caricamento della partita (vedi 6.3.6.

```
public class Threat
{
    public int id;
    public StaticDb.ThreatType threatType;
    public float deployTime; //expressed in hour
    public StaticDb.ThreatAttacker threatAttacker;
    public StaticDb.ThreatDanger threatDanger;
    public StaticDb.ThreatAttack threatAttack;
    public float moneyLossPerMinute;
    public bool fromCreateRemote;

    //values for ai controller prefab; the serializable one is created when
        game is saved
    public AiController aiController;
    public SerializableAiController serializableAiController;
}
```

Figura 6.15. Proprietà della classe `Threat`.

6.3.3 `TimeEvent`

Un'altra classe di particolare importanza è quella dei `TimeEvent`, responsabile della definizione di tutti gli eventi a tempo che possono avvenire durante la partita. Tra questi vengono annoverate le `Threat`, ma anche tutti gli eventi a tempo che sono effettuabili tramite gli oggetti. Le proprietà (visibili in fig. 6.16), sono:

- `id`, int che identifica in maniera univoca ogni `TimeEvent`. È incrementato sequenzialmente ad ogni evento generato;
- `duration`, float che esprime, in termini di tempo di gioco, la durata di un evento. Se l'evento è associato ad una `Threat`, coincide con il valore di `deployTime`;
- `percentagePerMin`, float, calcolato al momento della generazione dell'evento, che esprime il progresso percentuale compiuto ogni minuto;
- `currentPercentage`, float che esprime l'attuale percentuale di completamento dell'evento;
- `progressBar`, riferimento al `Canvas` utilizzato per rappresentare la barra di progresso associata all'evento;
- `visible`, bool con il quale si definisce la visibilità della barra di progresso (alcuni eventi a tempo, come le minacce da remoto, non sono visibili e necessitano l'occultamento della barra di progresso);
- `threat`, la `Threat` associata all'evento. Se il `TimEvent` generato è una nuova minaccia, questa proprietà imposterà tutti i valori specificamente per il tipo di minaccia generata, altrimenti verranno utilizzati valori standard per ogni proprietà della minaccia, in modo da renderla identificabile dalle reali `Threat` in corso di svolgimento;
- `stoppable`, bool con cui si identifica se il `TimeEvent` possa essere fermato dall'utente o meno;
- `progressBarParentName`, stringa per memorizzare il nome dell'oggetto a cui è attaccata la `progressBar`;
- `routine`, stringa utilizzata per memorizzare la routine da eseguire al termine del completamento del `TimeEvent`. È utilizzata al momento del ripristino degli oggetti dopo il caricamento di una partita (vedi 6.3.6).

```
public class TimeEvent
{
    public int id;
    public float duration; //expressed in minute
    public float percentagePerMin;
    public float currentPercentage;
    public Canvas progressBar;
    public bool visible;
    public Threat threat;
    public bool stoppable;
    public string progressBarParentName;
    public string routine;
}
```

Figura 6.16. Proprietà della classe `TimeEvent`.

6.3.4 AI

Le `AI` sono dei prefab utilizzati per rappresentare nella scena del gioco i personaggi umani, ossia i dipendenti interni dell'azienda e gli attaccanti che tentano di camuffarsi tra loro. I prefab sono dei `GameObject` di Unity che è possibile salvare, insieme a tutte le componenti e le proprietà necessarie. In questo modo è possibile utilizzarli in un secondo momento come template per creare delle istanze nella scena del `GameObject` in questione.

Come si può vedere in 6.17, il prefab utilizzato per le `AI` comprende diverse componenti, tutte dipendenti tra di loro al fine di garantirne il corretto funzionamento. Alcune di esse sono delle componenti base di Unity, necessarie affinché il `prefab` sia visibile e possa interagire sia con le azioni dell'utente che con le altre componenti presenti nella scena. Tra queste componenti si possono annoverare`Rigidbody 2D`, `Box Collider 2D`, `Animator`, `Sprite Renderer`, `Canvas Renderer` e `Canvas Group`.

Le rimanenti tre componenti, invece, sono state implementate appositamente per asserire alle funzionalità di gestione del `prefab`, animazione e risposta agli input del giocatore d. In particolare:

- `AiController` è la classe responsabile per la gestione generale del prefab. Tramite questo script sono effettuate le azioni del movimento nella scena dell'oggetto, oltre ad un serie di funzioni di controllo relative alla partita in corso. Il movimento è gestito in tre fasi: durante la creazione di una nuova istanza del `prefab` viene impostato un oggetto della scena come obiettivo da raggiungere al termine del `TimeEvent` associato all'`AI`. Dopodiché viene creata una destinazione temporanea da raggiungere, situata in un intorno dell'obiettivo (l'intorno ha un raggio che si restringe con il progredire del `TimEvent` associato). Una volta creata la destinazione temporanea, viene calcolato se è raggiungibile, tramite un algoritmo di pathfinding. Se il calcolo dà un riscontro positivo, fornendo un percorso da seguire, lo script inizia a spostare la posizione dell'oggetto verso la destinazione, seguendo il percorso fornito precedentemente, e, contemporaneamente, imposta l'immagine da visualizzare dallo script responsabile dell'animazione (`SpriteSheet Animator`). Una volta raggiunta la sua destinazione, o nel caso in cui l'algoritmo di pathfinding non si riuscito a trovare un percorso, l'`AI` viene fermata per alcuni secondi, prima di rieseguire le precedenti azioni.

  La routine di movimento continuerà fin quando il `TimeEvent` associato all'`AI` non sarà completato. Nell'istante in cui ciò avviene, è impostato un flag con cui si dà il via all'esecuzione della routine che porterà alla distruzione del `prefab`. Viene fornita una destinazione (corrispondente al punto della mappa in cui inizialmente sono create) e, al suo raggiungimento, il `prefab` viene eliminato dalla scena, permettendo il rilascio delle risorse grafiche utilizzate.

- `AiListener` è lo script in cui sono definite le azioni che è possibile effettuare con le `AI` presenti in scena. In particolare, tramite `Ai Controller` sono implementati i metodi

per abilitare l'interazione del `prefab` con gli input dell'utente (come è stato fatto per le `InteractiveSprite` viste in precedenza in 6.3.1). Anche in questo caso, una volta effettuato un click sull'immagine di una `AI`, si aprirà il menù con i pulsanti relativi alle azioni che si possono svolgere. I metodi artefici di queste azioni sono racchiusi, appunto, all'interno della classe `AiListener`.

- `SpriteSheet Animator` è la classe con cui è resa possibile l'animazione delle `AI`. Dato che le immagini relative al `prefab` sono scelte casualmente al momento della creazione, creare per ognuna di esse un'animazione specifica avrebbe richiesto molto tempo. Inoltre ciò avrebbe reso necessario la costruzione di `prefab` multipli per la `AI` (uno per ogni personaggio che si vuole raffigurare). Per questo si è deciso di tentare un approccio più elastico, che potesse essere applicato con facilità al caso in questione e rimanesse valido anche in caso di eventuali futuri sviluppi.

  Il metodo attualmente in uso prevede l'utilizzo di un `Animator` in cui sono creati due stati, `Idle`, relativo a quando il personaggio è fermo, e `Walk`, per i movimenti dell'`AI`. Per ognuno degli stati sono state programmate otto animazioni, grazie all'utilizzo dei `Blend Trees` messi a disposizione dall'editor di Unity. I `Blend Trees` sono uno strumento che permette di miscelare animazioni multiple. tramite l'utilizzo di un parametro numerico. In questo caso sono stati sfruttati per riprodurre le animazioni in base alla direzione di cammino (e di arresto) del personaggio: otto, infatti, sono le direzioni lungo cui una `AI` può muoversi (nord, sud, est, ovest e quelle intermedie tra queste quattro). Quindi, durante lo spostamento del personaggio, viene calcolata la direzione e il `Blend Tree` permette la visualizzazione dell'animazione corretta. Questo, però, non è sufficiente per risolvere completamente il problema, dato che persiste la questione legata alle differenti immagini di cui riprodurre l'animazione. Per ovviare a ciò, è stata sfruttata la possibilità di creare delle animazioni in cui ad ogni cambio di frame viene eseguito un metodo. Nel caso in questione inizialmente si procede ad identificare il gruppo di immagini da riprodurre, partendo dal nome dell'immagine iniziale. Una volta effettuata l'identificazione, si procede con il caricare ognuna delle immagini in un array. Successivamente, tramite il metodo richiamato dall'`Animator` ad ogni frame, si indica quale visualizzare specificandone l'indice dell'array.

  La soluzione adottata è stata creata ad hoc per il set di immagini con cui deve interagire. I diversi set, infatti, presentano tutte lo stesso numero immagini, disposte secondo le stesse modalità. Una volta importati nell'editor di Unity, grazie allo `Sprite Editor` è stato possibile dividere ogni set, creando singole immagini indicizzate in maniera sequenziale. Dato che ad ogni valore dell'indice corrisponde lo stesso tipo di immagine da visualizzare, è stato semplice arrivare alla soluzione attualmente in uso. In figura 6.18 è possibile vedere un esempio dei diversi set di immagini utilizzate.

6.3.5 `LevelManager` e `ILevelManager`

Gli script analizzati fin'ora sono quelli implementati per il controllo di singoli elementi di gioco, ma, affinché tutto queste componenti possano interagire tra di loro in maniera ordinata e controllata, è necessario di uno script d gestione globale. Da questo bisogno si è arrivati alla creazione del `LevelManager` e della relativa interfaccia di classe `ILevelManager`.

Il `LevelManager` è il vero nucleo del gameplay, senza di cui il gioco non sarebbe nemmeno utilizzabile. Al suo interno sono svolte diverse funzioni di creazione, gestione degli elementi di gioco e controllo dei dati relativi alla partita in corso di svolgimento. Tra le più significative si annoverano:

- metodi per la simulazione e gestione temporale, fondamentali per la gestione dei `TimeEvent`. Tramite queste funzioni, gli eventi a tempo possono essere completati e le azioni per cui sono stati creati possono essere eseguite. Inoltre, correlato alla simulazione temporale vi è anche la creazione e gestione delle minacce. Ogni `Threat` viene generata con una cadenza temporale variabile nel corso del gioco. Sia la creazione di una nuova minaccia che la frequenza con cui ciò avviene sono gestite dal `LevelManager` (vedi 6.19);

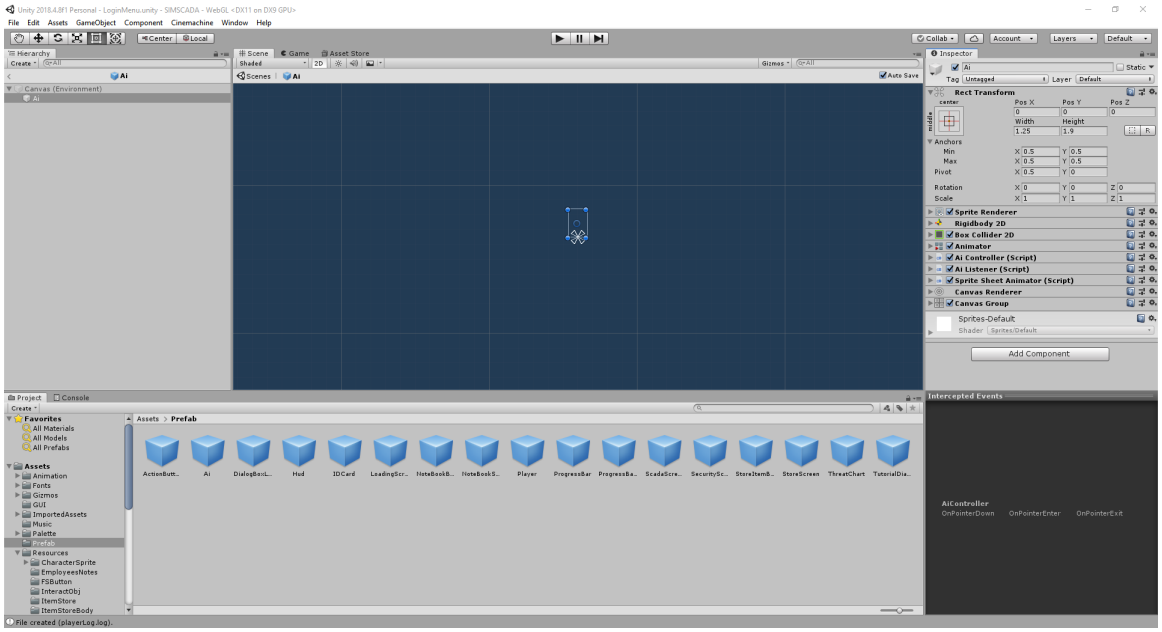

Figura 6.17. Screenshot del prefab utilizzato per le `AI`. Sulla destra si possono notare le varie componenti utilizzate per il prefab, tra cui gli script `AiController` e `AiListener`, utilizzati per gestire i movimenti e le azioni delle `AI`.

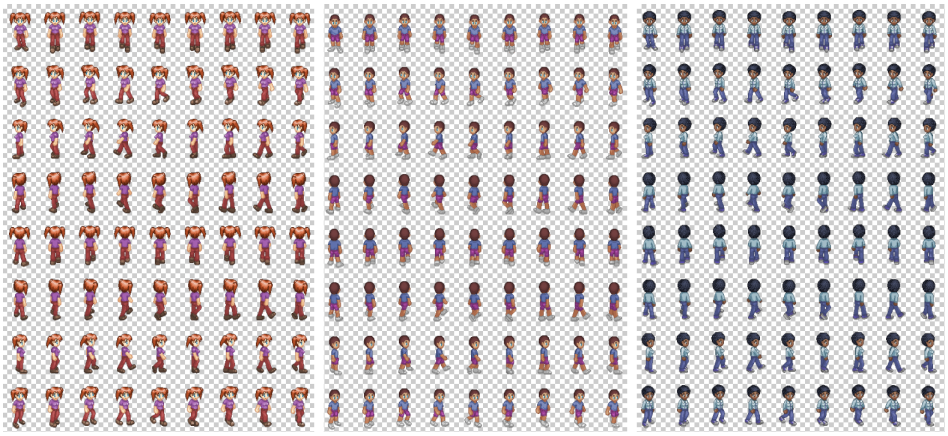

Figura 6.18. Esempio di set di immagini utilizzate per le `AI`.

- legate alla creazione della minaccia, all'interno del `LevelManager` sono contenuti anche i metodi necessari per la loro gestione. In particolare le funzioni `BeforeDeployThreat`, `DeployThreat` e `AfterDeploThreat`, che gestiscono rispettivamente ciò che avviene prima, durante e dopo il concretizzarsi di un attacco. Nel primo metodo sono svolti dei controlli per accertare che la minaccia possa essere concretizzata in un attacco, in base anche alle caratteristiche del tipo di `Threat` in questione. Se il riscontro è negativo, viene richiamato il metodo per mostrare il messaggio corrispondente, la minaccia è tolta dalla coda degli eventi ed il `prefab` eliminato dalla scena. In caso positivo, invece, viene richiamato il metodo `DeployThreat` in cui, sempre in base al `ThreatType` vengono effettuate le azioni specifiche per concretizzare l'attacco. Una volta fatto ciò (e mostrato anche il messaggio all'utente che lo notifica di un attacco avvenuto), viene eseguito il metodo `AfterDeployThreat`, in cui sono aggiornati i valori relativi alle statistiche di gioco (attacchi ricevuti, numero di dipendenti e reputazione del giocatore).Il metodo è richiamato anche in caso di attacco non andato a segno, ma chiaramente i valori saranno aggiornati tenendo conto del mancato attacco;

```
gameData.threatSpawnRate = gameData.threatSpawnBaseTime / (float)
    gameData.totalEmployees;

public IEnumerator CreateNewThreat()
{
    //yield return new WaitForSeconds(5);
    for (;;)
    {
        gameData.threatSpawnTime = 0;

        yield return new WaitUntil(() => gameData.threatSpawnTime >
            gameData.threatSpawnRate);

        yield return new WaitWhile(() => gameData.hasThreatDeployed);

        NewThreat();
    }
}
```

Figura 6.19. Metodo per la creazione di una nuova minaccia. Nella prima riga di codice è possibile vedere la definizione della frequenza con cui le minacce sono create.

- una volta subito un attacco, il giocatore dovrà intraprendere determinate azioni, a seconda del tipo, attraverso gli oggetti presenti in gioco (le `InteractiveSprite` analizzate in 6.3.1). I metodi relative alle azioni sono contenute nei `IObjectListener` di ogni oggetto, ma al fine della raccolta dati, nel `LevelManager`, durante questa fase, viene eseguito il metodo `ThreatManagementResultData` con cui si analizzano le mosse del giocatore, per capire come sta procedendo per risolvere la situazione di crisi dovuta all'attacco. Questo aspetto verrà approfondito in 6.3.7;

- un'altra serie di metodi, legata sempre al gameplay, sono quelli necessari per attivare le difese (firewall, IDS e sicurezza locale) in grado di prevenire gli attacchi. Queste sono attivabili tramite gli oggetti in scena, ma il loro codice di controllo è stato inserito nel `LevelManager` poiché agiscono sulle variabili di gioco definite al suo interno. In particolare, effettuano dei controlli sulle code di minacce attive e, nel caso in cui abbiano successo lanciano un messaggio all'utente. La probabilità di efficacia delle componenti è fissa e può essere incrementata tramite gli acquisti del negozio, tuttavia il controllo in sé consiste nel confronto tra il suddetto valore e un numero calcolato casualmente. Se il numero è più alto della probabilità l'attacco non viene intercettato, altrimenti la difesa ha avuto successo e si procede con l'informare l'utente.

Tutti i metodi appena analizzati, oltre a diversi altri non elencati ma che svolgono funzioni di supporto nelle azioni elencate, sono implementati in ogni `LevelManager`. Per rendere possibile il loro utilizzo anche da classi esterne, senza ogni volta dover effettuare controlli su quale livello si sta giocando per recuperare la giusta istanza del corretto `LevelManager`, si è fatto utilizzo anche in questo caso di un'interfaccia, `ILevelManager`. In essa sono definiti tutti i metodi che devono essere presenti in un `LevelManager` affinché la partita si svolga correttamente. In questo modo la procedura per richiamare un metodo da una classe esterna risulta più veloce in quanto è sufficiente ottenere l'istanza dell'interfaccia, invece di quella della classe specifica (vedi fig. 6.20). Inoltre, essendo definiti al suo interno tutti i metodi necessari affinché la partita si svolga in maniera corretta, risulta più semplice aggiungere dei nuovi livelli al gioco. Per questo punto si rimanda alla sezione 6.4.

```
private ILevelManager SetLevelManager()
{
    ILevelManager iManager;
    if (SceneManager.GetActiveScene().buildIndex ==
        StaticDb.level1SceneIndex)
        iManager = FindObjectOfType<Level1Manager>();
    else
        iManager = FindObjectOfType<Level2Manager>();

    return iManager;
}
```

Figura 6.20. Metodo con cui si richiama l'istanza dell'interfaccia `ILevelManager` nelle classi esterne. Utilizzando la variabile `iManager` è possibile richiamarne i metodi.

6.3.6 Script per il salvataggio

Come in tutti i giochi, è presente durante la partita la possibilità di salvare i dati, interrompendola, per poi continuare in un secondo momento. I file dei dati salvati sono mantenuti nel server, all'interno della cartella creata per ogni giocatore. Al fine di inviare le informazioni sulla partita in maniera veloce e in un formato facilmente gestibile, si è scelto di fare ricorso, nuovamente, alla classe di Unity `JsonUtility`, serializzando tutto in un unico file json da inviare al server. Per rendere questo processo ancora più veloce e semplice da eseguire, è stato deciso di implementare una classe, denominata `GameData`, al cui interno sono contenuti i valori di tutte le variabile relative al gioco:

- liste dei `TimeEvent` e delle `Threat` in corso di svolgimento;
- flag relativi agli stati delle difese (attivate o disattivate);
- valori relativi a data, soldi a disposizione, minacce sventate, minacce subite e altre statistiche del gioco;
- flag con cui registrare l'acquisto di determinati potenziamenti dal negozio;
- valori necessari per il corretto ripristino dei `prefab` al momento del caricamento di una partita.

Tutti questi valori sono di tipo primitivo, in modo da poter essere serializzati correttamente in un file json. Sono presenti anche variabili di tipo complesso e/o personalizzato (come ad esempio la lista dei `TimeEvent` in corso di svolgimento), ma si è potuto utilizzarle perché anch'esse, a loro volta, sono state rese possibili da serializzare.

I metodi per il salvataggio ed il caricamento del gioco sono contenuti nella classe `GameDataManager` e sono richiamati due classi esterne, risepttivamente `PauseManager`, ossia lo script utilizzato per mettere in pausa il gioco e da cui è possibile caricare il menù iniziale, e `LevelManager`. In quest'ultima, infatti, è presente il campo dell'istanza di `GameData` utilizzato per memorizzare i valori del livello in corso di svolgimento. Al momento del salvataggio, è richiamato il metodo `SaveLevelGameData` in cui si copiano i dati dell'istanza di `GameData` in una variabile locale. Successivamente vengono impostati i valori di tutti i `SerializableAiController`, partendo da quelli degli `AiController` presenti in scena, in modo da poter ripristinare durante il caricamento i corretti `prefab` dei personaggi. Svolto questo passaggio, si procede con al serializzazione del valore locale di `GameData` e, sfruttano la classe `UnityWebRequest`, viene effettuata una richiesta web indirizzandola allo script php contenuto nel server responsabile per la gestione dei dati di gioco. Quando la procedura termina, il giocatore è rimandato nel menù principale.

Per quanto riguarda il caricamento, la procedura applicata è sostanzialmente l'inversa di quella vista per il salvataggio. Non appena viene caricato il livello di gioco, il `LevelManager` richiama il

metodo `LoadLevelGameData`. Qui viene subito effettuata una richiesta online indirizzata al server per ottenere una copia del file json presente al suo interno. Se la richiesta ha successo, si effettua prima un controllo sui dati ricevuti, per prevenire errori successivi che possono presentarsi al momento della deserializzazione. Se il controllo ha successo, i dati sono deserializzati e salvati una istanza locale di `GameData`. I valori della variabile vengono successivamente assegnati all'istanza di `GameData` implementata nel `LevelManager` che, una volta riottenuto il controllo, procede a ripristinare tutti i `prefab` e le variabili della partita in modo da permetterne il proseguimento. Questa procedura è eseguita ogni volta che viene avviata una partita. Se si tratta del primo avvio di un determinato livello, per evitare di creare inavvertitamente dei `prefab` errati ed utilizzare dei valori sbagliati dell'istanza di `GameData`, il `LevelManager` utilizza un flag che permette di distinguere se si tratta di una nuova od una vecchia partita da caricare. In caso affermativo, inizializza l'istanza di `GameData` con i valori predefiniti e fa cominciare la partita, mostrando il messaggio di benvenuto.

6.3.7 Script per l'analisi delle prestazioni del giocatore

Ai fini di poter analizzare la partita del giocatore, e stabilire se sta effettivamente apprendendo i concetti inseriti nel gioco, sono stati implementati degli script per poter monitorare le azioni dell'utente sia quando si trova a fronteggiare le conseguenze di un attacco, sia nei momenti di attesa in cui deve applicare una tattica di prevenzione da nuove minacce. Per quanto riguarda la prima situazione, si è deciso di monitorare i tempi con cui il giocatore reagisce ad un attacco subito, registrando anche le azioni intraprese in quel lasso temporale. Questi dati sono poi salvati in un file di log presente nel server, in modo da poterlo analizzare in un secondo momento. La statistica relativa ai tempi di risposta è resa disponibile anche al giocatore, tramite un grafico in cui sono raccolti i valori registrati nel corso della partita e che può essere visionato in qualsiasi momento durante lo svolgimento del gioco. Il motivo per cui si è deciso di analizzare la statistica dei tempi si basa sull'idea che, se il giocatore sta apprendendo in maniera corretta le lezioni, l'andamento generale delle curve dei tempi sarà in diminuzione con il passare del numero di attacchi fronteggiati. Inoltre, registrare le azioni effettuate nel lasso di tempo in questione permette di capire se il giocatore sta procedendo in maniera ragionata e specifica per il tipo di minaccia che sta affrontando o, più semplicemente, sta sta affidando al caso (in quest'ultima situazione le curve dei tempi di reazione avranno un andamento costante o, addirittura, in salita).

Per quanto riguarda, invece, l'analisi delle azioni di prevenzione attuate dal giocatore è stato necessario implementare una variabile che potesse essere d'aiuto in questo compito. Si è scelto, quindi, di implementare un trend delle minacce: all'avvio di una nuova partita viene scelta, casualmente, un tipo di minaccia che sarà più probabile subire rispetto alle altre. In particolare, al momento della creazione di una nuova minaccia, il tipo di `Threat` scelto per essere in trend avrà una probabilità di generazione del 60% superiore rispetto agli altri tipi. Il trend delle minacce è poi ricalcolato ogni tre giorni (in termini di tempo di gioco), evitando che venga scelto di nuovo la stessa tipologia già presente in trend. Sfruttando questa dinamica è stato possibile implementare un metodo che analizzasse le azioni di prevenzione relativamente al trend del momento. In particolare, le azioni monitorate sono l'attivazione o disattivazione dei servizi di sicurezza (firewall, IDS e controlli locali), oltre agli acquisti di potenziamenti dal negozio (sfruttando la proprietà `ThreatAffinity` presente per ogni `ItemStore`, come esposto in 6.2.3. Il metodo di analisi viene richiamato ogni volta in cui è effettuata una delle azioni elencate precedentemente e, in base al trend del momento, ne dà una valutazione, registrandola all'interno del file di log presente nel server.

L'analisi delle azioni del giocatore può essere, ovviamente, ampliata e strutturata in maniera più dettagliata, ma per i fini del progetto e per questioni legate al tempo necessario per uno sviluppo più dettagliato, si è preferito fornire un'idea di cosa è possibile analizzare e in quale maniera. Per un approfondimento sul tema in questione si rimanda alla sezione 8.2.

6.4 Come aggiungere un livello

Il gioco può essere ampliato, aggiungendo nuovi livelli, in modo da poter inserire nuove nozioni all'interno del gioco o, addirittura, creare livelli specifici per un tipo di minaccia in particolare, rendendo l'esperienza di gioco più focalizzata e differenziata. Per aggiungere un nuovo livello è possibile procedere da zero, creando una nuova scena al cui interno inserire i diversi **??** con allegati gli script analizzati in precedenza. Ma, per evitare di incorrere in problemi durante l'esecuzione del gioco, legati alla mancanza di qualche componente, è consigliabile selezionare una delle scene relative ai due livelli già implementati e duplicarlo. In questo modo si avrà una nuova scena, su cui poter agire senza problemi. A questo punto è importante aprire i `Build Settings` del progetto (la voce è presente nel tab `File`). Una volta fatto click sulla voce, si aprirà una nuova finestra. Qui occorre inserire la scena appena creata all'interno del riquadro `Scenes In Build` (per farlo è sufficiente trascinare l'icona della scena dentro il riquadro). Una volta effettuato questo passaggio, sarà possibile notare che accanto al nome della scena sarà comparso un numero, corrispondente all'indice della scena stessa. Questo indice rappresenta il valore da utilizzare negli script di caricamento per poter permettere l'accesso alla nuova scena.

Come detto, si può agire su qualsiasi aspetto del gioco:

- per modificare, ad esempio, la generazione delle minacce nel nuovo livello occorre creare un nuovo `LevelManager`, facendolo derivare dall'interfaccia `ILevelManager`. Successivamente è necessario implementare un nuovo metodo nella classe `ThreatManager`, riprendendo la struttura generale dei metodi `NewRandomLevel#Manager` (con il numero del livello al posto di #) già implementati e richiamarlo nel metodo `NewThreat` presente nel nuovo `LevelManager`.
- per aggiungere nuove azioni eseguibili tramite le `InteractiveSprite` è possibile agire sui singoli `Listener` di ogni categoria. Naturalmente, in questo caso, occorre tenere a mente che se si vuole agire su variabili di gioco presenti nella classe `GameData` è necessario recuperare i giusti valori tramite la corretta istanza presente nel `LevelManager`. Per fare ciò si può utilizzare il metodo riportato in 6.20, aggiungendo eventualmente i casi relativi ai nuovi livelli implementati.
- per implementare nuovi `ItemStore`, `Lesson` e `Message` è sufficiente seguire il pattern già esistente. In particolare è necessario un file json e un file txt con le caratteristiche fornite nei template e salvarli nelle relative cartelle (le cartelle sono denominate secondo il pattern NomeOggetto e NomeOggettoBody e raccolgono, rispettivamente, il file json dell'oggetto ed il file txt contenente il testo più lungo relativo al messaggio/lezio/descrizione dell'oggetto da acquistare). Successivamente, in base all'oggetto aggiunto, possono essere necessarie alcune ulteriori azioni.

  Nel caso dell'`ItemStore` sarà necessario aggiungere un nuovo `case` nello `switch` del metodo `ApplyItemEffect` relativo all'effetto che si vuole associare al nuovo potenziamento (il metodo è implementato all'interna dello script `StoreManager`).

  Nel caso di un messaggio, invece, è necessario aggiungere il metodo da richiamare per visualizzare il messaggio. Per fare ciò si può agire nello script `LevelMessageManager`, rispettando il pattern presentato negli altri casi: un metodo `public` con cui richiamare una `coroutine`. Il primo metodo sarà quello che verrà richiamato per dar via alla visualizzazione del messaggio, la quale avverrà grazie al secondo metodo. Questo è stato fatto perché trattandosi di `coroutine`, metodi la cui esecuzione può essere messa in pausa, si è cercato di rispettare i design pattern proposti dallo stesso manuale di Unity.

  Se invece è stata aggiunta una nuova lezione non c'è bisogno di ulteriori azioni. C'è però una precauzione da prendere in questo caso: per evitare che il gioco possa dare problemi di caricamento della lezione al momento della visualizzazione, è consigliato creare file txt non eccessivamente lunghi. Il rischio è quello di scatenare un'eccezione di tipo `OutOfMemory` nel browser, richiedendo il refresh della pagina web in cui il gioco è eseguito.

Per testare che le nuove funzionalità aggiunte vengano eseguite in maniera corretta, è possibile sfruttare il player nell'editor di Unity. In questo modo si possono effettuare test senza dover ogni

volta creare una build. Nel momento in cui si vuole effettuare una build da caricare in un server occorre selezionare di nuovo la voce `Build Settings`, controllare che la scena sia selezionata all'interno del riquadro `Scenes In Build` e lanciare il build dell'applicativo finale.

Capitolo 7

Risultati

7.1 Test per l'efficacia

Capitolo 8

Conclusioni

8.1 Considerazioni sullo stato attuale del gioco

8.2 Futuri sviluppi

Bibliografia

[1] E. Aarseth, A. M. Bean, H. Boonen, M. C. Carras, M. Coulson, D. Das, J. Deleuze, E. Dunkels, J. Edman, C. J. Ferguson, M. C. Haagsma, K. H. Bergmark, Z. Hussain, J. Jansz, D. Kardefelt-Winther, L. Kutner, P. Markey, R. K. L. Nielsen, N. Prause, A. Przybylski, T. Quandt, A. Schimmenti, V. Starcevic, G. Stutman, J. V. Looy e A. J. V. Rooij. "Scholars' open debate paper on the World Health Organization ICD-11 Gaming Disorder proposal". *Journal of Behavioral Addictions*. Vol.6, n.3. Settembre 2017, pp. 267–270. DOI: 10.1556/2006.5.2016.088. URL: https://doi.org/10.1556%2F2006.5.2016.088.

[2] C. Abt. *Serious Games*. Viking Press, 6 marzo 1970. ISBN: 978-0-670-63490-3.

[3] C. A. Anderson, A. Shibuya, N. Ihori, E. L. Swing, B. J. Bushman, A. Sakamoto, H. R. Rothstein e M. Saleem. "Violent video game effects on aggression, empathy, and prosocial behavior in Eastern and Western countries: A meta-analytic review." *Psychological Bulletin*. Vol.136, n.2. 2010, pp. 151–173. DOI: 10.1037/a0018251. URL: https://doi.org/10.1037%2Fa0018251.

[4] I. Bogost. *Persuasive Games: The Expressive Power of Videogames*. The MIT Press, 2007. ISBN: 978-0-262-02614-7.

[5] A. Bolshev e I. Yushkevich. *SCADA and Mobile Security in the IoT Era*. Rapp. tecn. 11 Gennaio 2018. URL: https://ioactive.com/pdfs/SCADA-and-Mobile-Security-in-the-IoT-Era-Embedi-FINALab%20(1).pdf.

[6] P. Bonato. "Advances in wearable technology and applications in physical medicine and rehabilitation". *Journal of NeuroEngineering and Rehabilitation*. Vol.2, n.2. 25 Febbraio 2005. DOI: 10.1186/1743-0003-2-2. URL: https://jneuroengrehab.biomedcentral.com/articles/10.1186/1743-0003-2-2.

[7] K. Clarke. "State Department Deploys Anti-Phishing 'Phil' Game Training". *DARKReading*. 19 Ottobre 2009. URL: https://www.darkreading.com/risk/state-department-deploys-anti-phishing-phil-game-training/d/d-id/1132218.

[8] K. Corti. "Games-based Learning; a serious business application". *PIXELearning*. 9 Febbraio 2006. URL: https://www.cs.auckland.ac.nz/courses/compsci777s2c/lectures/Ian/serious%20games%20business%20applications.pdf.

[9] M. Csikszentmihalyi. *Flow: The Psychology of Optimal Experience*. HarperCollins Publishers, 1991. ISBN: 978-0-060-92077-7.

[10] M. Csikszentmihalyi e I. S. Csikszentmihalyi, cur. *Optimal Experience: Psychological Studies of Flow in Consciousness*. Cambridge University Press, 26 agosto 1988. ISBN: 978-0-521-34288-9. DOI: 10.1017/cbo9780511621956. URL: https://doi.org/10.1017%2Fcbo9780511621956.

[11] *CyLab Usable Privacy and Security Laboratory. Anti-Phishing Phil*. URL: https://cups.cs.cmu.edu/antiphishing_phil/.

[12] *[d0x3d!] Website*. URL: http://d0x3d.com/d0x3d/about.html.

[13] A. D'Angour. "Plato and Play: Takin Education Seriously in Ancient Greece". *American Journal of Play*. Vol.5, n.3. 2013. URL: http://www.journalofplay.org/sites/www.journalofplay.org/files/pdf-articles/5-3-article-plato-and-play.pdf.

[14] L. Elliott, A. Golub, M. Price e A. Bennett. "More than Just a Game? Combat-Themed Gaming Among Recent Veterans with Posttraumatic Stress Disorder". *Games for Health Journal*. Vol.4, n.4. Agosto 2015, pp. 271–277. DOI: 10.1089/g4h.2014.0104. URL: https://doi.org/10.1089%2Fg4h.2014.0104.

[15] *F-Secure Labs. Havex Hunts For ICS/SCADA Systems*. 23 Giugno 2014. URL: https://www.f-secure.com/weblog/archives/00002718.html.

[16] P. Felicia. *Handbook of Research on Improving Learning and Motivation through Educational Games: Multidisciplinary Approaches*. IGI Global, aprile 2011. ISBN: 978-1-609-60495-0. DOI: 10.4018/978-1-60960-495-0.

[17] J. Fisher. "How "Assassin's Creed" is Helping Students Learn about History". *Geek & Sundry*. 13 Marzo 2018. URL: https://geekandsundry.com/how-assassins-creed-is-helping-students-learn-about-history/.

[18] D. Frye. "Could This Be the First FDA-Approved Video Game for ADHD?" *ADDitude Magazine*. 18 Dicembre 2017. URL: https://www.additudemag.com/therapeutic-video-game-adhd-symptoms-children/?tos=accepted.

[19] B. Genge, A. Beres e I. Kiss. "Using Software-Defined Networking to Mitigate Cyberattacks in Industrial Control Systems". In: *Securing Cyber-Physical Systems*. CRC Press, settembre 2015, pp. 305–330. DOI: 10.1201/b19311-13. URL: https://doi.org/10.1201%2Fb19311-13.

[20] M. Gondree e Z. N. Peterson. "Valuing Security by Getting [d0x3d!]: Experiences with a Network Security Board Game". In: *Presented as part of the 6th Workshop on Cyber Security Experimentation and Test*. USENIX, 12 agosto 2013. URL: https://www.usenix.org/conference/cset13/workshop-program/presentation/Gondree.

[21] M. Hendrix, A. Al-Sherbaz e V. Bloom. "Game Based Cyber Security Training: are Serious Games suitable for cyber security training?" *International Journal of Serious Games*. Vol.3, n.1. Marzo 2016. DOI: 10.17083/ijsg.v3i1.107. URL: http://journal.seriousgamessociety.org/index.php/IJSG/article/view/107/pdf_41.

[22] J. Herrington e R. Oliver. "An instructional design framework for authentic learning environments". *Educational Technology Research and Development*. Vol.48, n.3. Settembre 2000, pp. 23–48. DOI: 10.1007/bf02319856. URL: https://ro.uow.edu.au/cgi/viewcontent.cgi?referer=https://www.google.it/&httpsredir=1&article=1031&context=edupapers.

[23] J. Jones, X. Yuan, E. Carr e H. Yu. "A comparative study of CyberCIEGE game and Department of Defense Information Assurance Awareness video". In: *Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon)*. IEEE, marzo 2010. DOI: 10.1109/secon.2010.5453895. URL: https://doi.org/10.1109%2Fsecon.2010.5453895.

[24] E. D. Knapp e J. T. Langill. "Hacking Industrial Control Systems". In: *Industrial Network Security*. Elsevier, 2015, pp. 171–207. DOI: 10.1016/b978-0-12-420114-9.00007-1. URL: https://doi.org/10.1016%2Fb978-0-12-420114-9.00007-1.

[25] E. D. Knapp e J. T. Langill. "Industrial Cyber Security History and Trends". In: *Industrial Network Security*. Elsevier, 2015, pp. 41–57. DOI: 10.1016/b978-0-12-420114-9.00003-4. URL: https://doi.org/10.1016%2Fb978-0-12-420114-9.00003-4.

[26] A. L. Krassmann, A. Falcade, L. E. G. da Silva e R. D. Medina. "Serious Games to Computer Networks Learning With CyberCIEGE: A Case Study in Brazilian Higher Education". In: *2015 WEI - Workshop sobre Educação em Computação*. Luglio 2015. URL: https://www.researchgate.net/publication/284644533_Serious_Games_to_Computer_Networks_Learning_With_CyberCIEGE_A_Case_Study_in_Brazilian_Higher_Education.

[27] R. Kulman. "Video Games Can Help Kids with ADHD - If You Choose Wisely". *ADDitude Magazine*. 2014. URL: https://www.additudemag.com/video-games-help-adhd/?tos=accepted.

[28] J. Lave e E. Wenger. *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press, settembre 1991. ISBN: 978-0-521-42374-8. DOI: 10.1017/cbo9780511815355.

[29] K. Lohse, N. Shirzad, A. Verster, N. Hodges e H. F.M. V. der Loos. "Video Games and Rehabilitation". *Journal of Neurologic Physical Therapy*. Vol.37, n.4. Dicembre 2013, pp. 166–175. DOI: 10.1097/npt.0000000000000017. URL: https://doi.org/10.1097%2Fnpt.0000000000000017.

[30] Ludoscience. *Serious Game Classification*. 2011. URL: http://serious.gameclassification.com/EN/index.html.

[31] D. Michael e S. Chen. *Serious Games: Games That Educate, Train, and Info*. Course Technology PTR, 10 ottobre 2015. ISBN: 978-1-592-00622-9.

[32] S. Morgan. *2019 Official Annual Cybercrime Report*. Rapp. tecn. Herjavec Group, 2019. URL: https://www.herjavecgroup.com/wp-content/uploads/2018/12/CV-HG-2019-Official-Annual-Cybercrime-Report.pdf.

[33] N. Nelson. *The Impact of Dragonfly Malware on Industrial Control Systems*. Rapp. tecn. 22 Gennaio 2016. URL: https://www.sans.org/reading-room/whitepapers/ICS/impact-dragonfly-malware-industrial-control-systems-36672.

[34] D. Parlett. *Oxford History of Board Games*. Oxford University Press, 3 giugno 1999. ISBN: 978-0-192-12998-7.

[35] R. Raman, A. Lal e K. Achuthan. "Serious games based approach to cyber security concept learning: Indian context". In: *2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*. IEEE, marzo 2014. DOI: 10.1109/icgccee.2014.6921392. URL: https://doi.org/10.1109%2Ficgccee.2014.6921392.

[36] S. S. Response. *Dragonfly: Cyberespionage Attacks Against Energy Suppliers*. Rapp. tecn. 7 Luglio 2014. URL: https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/Dragonfly_Threat_Against_Western_Energy_Suppliers.pdf.

[37] M. Rouse. "Definition of Cybersecurity". *Security threats and countermeasures glossary*. Maggio 2018. URL: https://searchsecurity.techtarget.com/definition/cybersecurity.

[38] D. E. Sanger. *Confront and Conceal: Obama's Secret Wars and Surprising Use of American Power*. Random House, 5 giugno 2012. ISBN: 978-0-307-71802-0.

[39] T. Saraiva, P. Gamito, J. Oliveira, D. Morais, M. Pombal, L. Gamito e M. Anastácio. "The use of VR exposure in the treatment of motor vehicle PTSD: A case-report". *Annual Review of CyberTherapy and Telemedicine*. Vol.5. 2007, pp. 199–205.

[40] B. Sawyer e P. Smith. "Serious Games Taxonomy". In: Game Developer Conference 2008. San Francisco (CA, USA), 19 febbraio 2008. URL: https://thedigitalentertainmentalliance.files.wordpress.com/2011/08/serious-games-taxonomy.pdf.

[41] D. W. Shaffer, K. R. Squire, R. Halverson e J. P. Gee. "Video Games and the Future of Learning". *Phi Delta Kappan*. Vol.87, n.2. Ottobre 2005, pp. 105–111. DOI: 10.1177/003172170508700205. URL: https://website.education.wisc.edu/kdsquire/tenure-files/23-pdk-VideoGamesAndFutureOfLearning.pdf.

[42] D. Shamah. "Stuxnet, gone rogue, hit Russian nuke plant, space station". *The Times of Israel*. 11 Novembre 2013. URL: http://www.timesofisrael.com/stuxnet-gone-rogue-hit-russian-nuke-plant-space-station/.

[43] S. Sheng, B. Magnien, P. Kumaraguru, A. Acquisti, L. F. Cranor, J. Hong e E. Nunge. "Anti-Phishing Phil". In: *Proceedings of the 3rd symposium on Usable privacy and security - SOUPS '07*. ACM Press, luglio 2007. DOI: 10.1145/1280680.1280692. URL: https://doi.org/10.1145%2F1280680.1280692.

[44] *Steam Workshop. [d0x3d!]* URL: https://steamcommunity.com/sharedfiles/filedetails/?id=590440835.

[45] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams e A. Hahn. *Guide to Industrial Control Systems (ICS) Security*. Rapp. tecn. Giugno 2015. DOI: 10.6028/nist.sp.800-82r2. URL: https://doi.org/10.6028%2Fnist.sp.800-82r2.

[46] K. Subrahmanyam e P. M. Greenfield. "Effect of video game practice on spatial skills in girls and boys". *Journal of Applied Developmental Psychology*. Vol.15, n.1. Gennaio 1994, pp. 13–32. DOI: 10.1016/0193-3973(94)90004-3. URL: https://doi.org/10.1016%2F0193-3973%2894%2990004-3.

[47] *TableTop Security. Website*. URL: https://www.tabletopsecurity.com/about.

[48] N. Trépanier. "The Assassin's Perspective: Teaching History with Video Games". *Perspective on History*. 1 Maggio 2014. URL: https://www.historians.org/publications-and-directories/perspectives-on-history/may-2014/the-assassins-perspective.

[49] S. Widup, M. Spitler, D. Hylender e G. Bassett. *2018 Data Breach Investigations Report*. Rapp. tecn. Aprile 2018. URL: https://enterprise.verizon.com/resources/reports/2018/DBIR_2018_Report.pdf.

[50] World Health Organization. *ICD-11 for Mortality and Morbidity Statistics*. 18 Giugno 2018.

[51] K. Zetter. *Countdown to Zero Day: Stuxnet and the Launch of the World's First Digital Weapon*. Crown Pub, settembre 2015. ISBN: 978-0-770-43617-9.