

POLITECNICO DI TORINO

Laurea Magistrale in Ingegneria Informatica



Tesi Magistrale

Modellazione e analisi del comportamento di un attaccante mediante la costruzione di grafi

Relatore:

Prof. Cataldo Basile

Candidato:

Marco Barca

Anno Accademico 2023/2024
Torino

Abstract

Nel contesto digitale altamente interconnesso di oggi, la sicurezza informatica rappresenta una priorità cruciale, poiché gli attacchi informatici diventano sempre più sofisticati e pervasivi. La comprensione del comportamento degli attaccanti è fondamentale per prevedere le loro azioni e sviluppare strategie di difesa efficaci. Questa tesi si propone di esplorare un nuovo approccio all'analisi del comportamento degli attaccanti, sfruttando l'uso di modelli di linguaggio avanzati, come il Generative Pre-trained Transformer (GPT) sviluppato da OpenAI. L'obiettivo centrale di questo lavoro è la modellazione e la successiva analisi di attacchi informatici derivanti da competizioni Capture The Flag (CTF) attraverso la costruzione di grafi che collegano i passi effettuati evidenziando le tecniche sfruttate e i Digital Artefacts ovvero dati, file o elementi di qualsiasi tipo che si generano come risultato delle azioni umane, dei processi o delle interazioni all'interno di un ambiente digitale.

Indice

Elenco delle figure	VIII
1 Introduzione	1
1.1 Approccio innovativo con l'utilizzo di GPT	2
1.2 Obiettivi della tesi	2
1.3 Stato dell'arte e lavori precedenti	3
1.4 Approfondimento sulla comprensione del comportamento di un attaccante .	3
1.4.1 Fasi di analisi	4
1.4.2 Dinamiche durante le fasi di attacco	4
1.4.3 Orchestrare l'attacco come processo	4
1.4.4 Motivazioni dell'attaccante	5
1.5 Struttura della tesi	5
2 Fondamenti delle tecniche di intelligenza artificiale	6
2.1 Natural Language Processing (NLP)	6
2.2 L'evoluzione dei Trasformers	7
2.3 OpenAI GPT (Generative Pre-trained Transformer)	8
3 Design ed implementazione del sistema	9

3.1	Sviluppo di una metodologia automatizzata	10
3.2	GPT: ingegnerizzazione del prompt	11
3.3	Creazione del Dataset: il problema dei dati non strutturati	14
3.3.1	Raccolta dati	15
3.4	Ruolo della modellazione	16
3.4.1	AttackModel	16
3.4.2	Popolare l'AttackModel	18
3.4.3	StepModel template	20
3.5	StepModel	20
3.5.1	Summarization	21
3.6	Etichettatura del dataset di StepModel	22
3.6.1	Scomposizione in sottostep	23
3.6.2	Tassonomia	24
3.6.3	Le Tecniche	26
3.7	Digital Artefacts	26
3.7.1	Creazione della lista di Digital Artefacts	27
3.7.2	Attributi	29
3.8	Estrazione dei 'path Nascosti'	30
4	AttackModeler: un framework per la mappatura e l'analisi di StepModel	33
4.1	Struttura del Framework	34
4.1.1	Contenuto della directory principale	35
4.2	Classi e Moduli	35
4.2.1	Attack	36
4.2.2	Node	37

4.2.3	DigitalArtefact	38
4.2.4	digital_artefact_extractor	39
4.2.5	gpt_api_call	41
4.3	Workflow	42
4.4	Utilizzo pratico: example.py	43
4.4.1	Caricamento dell'Attacco dallo StepModel	44
4.4.2	Estrazione degli Artefatti Digitali	44
4.4.3	Salvataggio dell'Attacco come file binario	44
4.4.4	Caricamento dell'Attacco da file binario	44
4.4.5	Visualizzazione del Grafo dell'Attacco	45
4.4.6	Salvataggio del NodeModel in Formato JSON	45
4.4.7	Analisi della Sequenza (report)	45
4.5	Generazione del grafo e report	46
4.6	Generazione delle classifiche	47
5	Rappresentazione completa di un attacco e valutazione empirica	50
5.1	AttackModel	50
5.2	Summary e StepModel	52
5.3	Etichettatura dello StepModel	53
5.4	Estrazione dei Path Nascosti	53
5.5	AttackModeler	54
5.5.1	Grafo, report e classifiche	55
5.6	Rappresentazione completa	56
5.7	Validazione empirica	57
6	Conclusioni e Prospettive di Sviluppo	58

A	Writeup, Tassonomia e Tecniche	60
A.1	Esempio di writeup	60
A.2	Tassonomia Completa	62
A.3	Tecniche	64
B	Rappresentazione completa di un attacco - Dati	67
B.1	Writeup	67
B.2	StepModel etichettato con la Tassonomia	70
B.3	Report	75
	Bibliografia	81

Elenco delle figure

3.1	Architettura del sistema.	10
3.2	Overview della directory contenente i repositories.	15
3.3	Overview di un repository.	16
3.4	Costruzione tassonomia - Fase 1.	25
3.5	Costruzione tassonomia - Fase 2.	26
3.6	Fase 1 - Creazione lista di Digital Artefacts.	28
3.7	Fase 2 - Creazione lista di Digital Artefacts.	28
4.1	Struttura del progetto.	34
4.2	Collegamento tra i moduli.	36
4.3	Processo di estrazione della lista di artefatti da un nodo.	40
4.4	Processo di arricchimento degli attributi.	41
4.5	Panoramica del workflow.	43
4.6	Esempio di grafo.	46
4.7	Breve estratto del report.	47
5.1	Grafo dell'attacco.	55
5.2	Rappresentazione completa dell'attacco.	57

Capitolo 1

Introduzione

Nel contesto sempre più interconnesso e digitalizzato del mondo odierno, la sicurezza informatica è diventata una preoccupazione centrale per individui, aziende e organizzazioni di ogni dimensione. La crescente dipendenza dalla tecnologia e l'accumulo di dati sensibili in ambienti virtuali rendono gli attacchi informatici una minaccia sempre più pervasiva e sofisticata.

Gli attacchi informatici possono assumere forme diverse, dalle intrusioni nell'infra struttura critica, come reti elettriche o sistemi finanziari, alle violazioni dei dati personali, con gravi implicazioni per la privacy degli individui. Inoltre, il cybercrime è diventato un business redditizio per i criminali informatici, che cercano costantemente nuovi metodi per eludere le difese e ottenere vantaggi illeciti.

Il costo di tali attacchi, in termini di danni finanziari e reputazionali, è elevato, e la necessità di proteggere l'ambiente digitale è diventata una priorità globale. Per affrontare questa sfida, è essenziale acquisire una profonda comprensione del comportamento degli attaccanti. Gli esperti di sicurezza informatica devono essere in grado di anticiparne le mosse, identificare le nuove tattiche e sviluppare misure di difesa più robuste.

La ricerca e l'analisi del comportamento degli attaccanti possono svolgere un ruolo fondamentale nell'individuazione delle minacce e nella formulazione di strategie di difesa efficaci. La presente tesi si concentra su questo aspetto, cercando di portare una nuova prospettiva all'analisi del comportamento degli attaccanti attraverso l'uso di dati derivanti da CTF writeup e l'applicazione di modelli di linguaggio AI come GPT.

1.1 Approccio innovativo con l'utilizzo di GPT

Gli attacchi variano ampiamente in termini di metodi, obiettivi e complessità, e il vasto volume di dati rende difficile un'analisi dettagliata e sistematica.

L'utilizzo di modelli di linguaggio avanzati, come GPT (Generative Pre-trained Transformer) sviluppati da OpenAI, apre nuove possibilità per l'analisi del comportamento degli attaccanti. Questi modelli, addestrati su enormi quantità di testo, sono in grado di comprendere e generare testi in linguaggio naturale con notevole precisione. Questa capacità offre un vantaggio significativo nella comprensione di dettagli nascosti nei dati.

In questo contesto, GPT funge da strumento chiave nella comprensione del comportamento degli attaccanti. Tramite l'analisi dei testi e delle tracce linguistiche presenti nei writeup, attraverso una corretta modellazione, GPT può semplificare l'identificazione di pattern, strategie e motivazioni degli attaccanti.

L'impiego di GPT ed in generale dei modelli di linguaggio avanzati apre nuove opportunità per comprendere come gli attaccanti pianificano e conducono le loro operazioni. Questo approccio non solo accelera l'analisi dei dati, ma consente anche di rilevare dettagli e sfumature che potrebbero sfuggire all'osservazione umana.

1.2 Obiettivi della tesi

L'obiettivo principale di questa tesi è modellare e mappare gli attacchi attraverso un approccio basato sull'estrazione di informazioni da dati non strutturati, al fine di generarne una rappresentazione che ne semplifichi l'analisi. Il problema principale che si pone è relativo alla mancanza di conoscenza sulla composizione dei dati, poiché la raccolta di questi non è stata effettuata mediante un processo di selezione. L'unica informazione disponibile riguardante tali dati è il fatto che rappresentino un attacco.

Per superare questa limitazione, è necessario modellare l'attacco in modo da standardizzare la rappresentazione di tutti i dati. Una volta sviluppato un modello adeguato e riportati i dati nella suddetta forma, l'obiettivo sarà standardizzare ulteriormente le informazioni al loro interno, definendo un dominio finito di azioni che possono comporre l'attacco.

Con una tale rappresentazione, sarà possibile mappare le informazioni e renderle accessibili per un'analisi approfondita.

1.3 Stato dell'arte e lavori precedenti

Diversi studi sono stati compiuti nel campo della modellazione di attacchi, sviluppando metodologie come il Diamond Model Caltagirone et al. [1], la Kill Chain Yadav e Rao [2] e l'Attack Graph Phillips e Swiler [3]. La comprensione dettagliata di queste metodologie si è rivelata essenziale per anticipare le minacce e migliorare le strategie di difesa (Al-Mohannadi et al. [4]). Questi modelli forniscono una prospettiva approfondita sui comportamenti degli attaccanti, contribuendo a delineare le migliori pratiche per la protezione dei sistemi.

Il Modello Diamond per l'Analisi delle Intrusioni (Caltagirone et al. [1]) rappresenta un approccio caratterizzato da elementi atomici fondamentali come avversario, infrastruttura, capacità e vittima, offrendo una visione strutturata delle attività di intrusione. Attraverso l'interconnessione di questi elementi, il modello Diamond fornisce un'analisi approfondita delle attività offensive, migliorando l'efficacia e l'efficienza dell'analisi delle intrusioni.

La ricerca condotta da Phillips e Swiler [3] si concentra sull'approccio basato su grafi per l'analisi delle vulnerabilità di rete. Questo metodo impiega grafi per identificare percorsi di attacco con elevata probabilità di successo, integrando informazioni sulla configurazione della rete e profili degli attaccanti. La flessibilità di questo approccio risiede nella sua capacità di analizzare attacchi provenienti da diverse direzioni, contribuendo a una comprensione più approfondita delle vulnerabilità di rete e risultando particolarmente utile nell'analisi dettagliata e nella mappatura dei possibili scenari di attacco.

L'articolo di Yadav e Rao [2] si concentra sugli aspetti tecnici della Cyber Kill Chain, offrendo una panoramica delle metodologie, tecniche e strumenti utilizzati negli attacchi. Attraverso l'analisi delle fasi della Cyber Kill Chain, il documento offre una guida tecnica per comprendere le opzioni dell'attaccante in ogni fase. Questo contributo si distingue per la sua attenzione agli aspetti pratici e tecnici della Cyber Kill Chain, offrendo una risorsa preziosa per gli analisti che desiderano approfondire la comprensione delle dinamiche degli attacchi mirati.

L'ispirazione tratta da questi lavori si manifesta nella concreta implementazione delle soluzioni presenti in questo lavoro, evidenziando come le metodologie proposte abbiano guidato e modellato in modo significativo l'approccio adottato per affrontare le sfide legate alla comprensione del comportamento degli attaccanti.

1.4 Approfondimento sulla comprensione del comportamento di un attaccante

Nel contesto di questa tesi, il concetto di "comportamento" riveste un'importanza fondamentale e richiede una definizione approfondita. Il comportamento, in un senso generale, rappresenta l'insieme di azioni, reazioni e modalità attraverso cui un individuo o in generale

un soggetto, interagisce con il proprio ambiente. La comprensione di questo termine è essenziale per comprendere come sia possibile modellare un attacco, poiché su questo fondamento si basa tutto il lavoro successivo.

Nel perseguire una definizione più approfondita di questo concetto, è cruciale considerare la sua complessità e la molteplicità di sfaccettature che lo caratterizzano. La figura dell'attaccante non è semplicemente limitata a una serie di azioni malevole, ma coinvolge un processo più ampio che comprende fasi di preparazione, analisi e esecuzione.

1.4.1 Fasi di analisi

Contrariamente alla percezione comune, il comportamento dell'attaccante non si limita ad un insieme di azioni aggressive, al contrario, spesso è maggiormente composta da fasi di analisi del sistema durante le quali l'attaccante raccoglie informazioni. Questa analisi dettagliata è essenziale per identificare possibili vulnerabilità e pianificare un attacco efficace. Le fasi di attacco in senso stretto sono spesso le meno interessanti se lo scopo è quello di comprendere il quadro generale dell'attacco, poiché una volta scoperta la vulnerabilità del sistema, il modo in cui questa viene sfruttata è generalmente ben documentato.

1.4.2 Dinamiche durante le fasi di attacco

Mentre le fasi di attacco sono spesso ben documentate, è importante sottolineare che l'aspetto dinamico dell'azione può variare notevolmente. L'attaccante può adattare la sua strategia in base alle risposte del sistema difensivo o alle nuove informazioni acquisite durante l'esecuzione. L'analisi dettagliata di queste dinamiche può rivelare approcci sofisticati che sfuggono a una visione superficiale del comportamento dell'attaccante.

1.4.3 Orchestrare l'attacco come processo

Il susseguirsi delle fasi di un attacco non è casuale; è, piuttosto, un processo orchestrato che può coinvolgere diverse entità e tecniche. Comprendere questa orchestrazione richiede non solo una visione lineare delle azioni, ma anche la capacità di identificare correlazioni e relazioni tra le diverse fasi. L'utilizzo dell'intelligenza artificiale in questo contesto diventa cruciale, poiché come vedremo sarà possibile cercare nelle sequenze di azioni dei path non immediatamente visibili.

1.4.4 Motivazioni dell'attaccante

Oltre alla sequenza di azioni, è di particolare interesse comprendere le motivazioni che guidano l'attaccante. L'intelligenza artificiale può giocare un ruolo chiave nell'analizzare dati provenienti da fonti varie per identificare pattern comportamentali che suggeriscono le ragioni dietro specifiche azioni. Questa comprensione più approfondita delle motivazioni può informare strategie difensive più efficaci.

1.5 Struttura della tesi

Nel Capitolo 2, si analizzano dettagliatamente le principali tecniche di Intelligenza Artificiale, con particolare attenzione alla tecnologia dei Transformers e ad OpenAI GPT.

Il Capitolo 3 si concentra sul design e sull'implementazione del sistema proposto, che comprende la creazione di modelli come l'AttackModel e lo StepModel per la rappresentazione e la classificazione degli attacchi.

Nel Capitolo 4, si presenta il framework "AttackModeler", sviluppato per la mappatura e l'analisi di StepModel, con funzionalità chiave come la visualizzazione grafica dell'attacco e la generazione di report.

Nel Capitolo 5 viene illustrato il processo completo mediante l'applicazione di tutte le tecniche sviluppate nei capitoli precedenti. Ciò condurrà alla generazione di un insieme di dati rappresentanti l'attacco i quali saranno validati empiricamente.

Infine, nel Capitolo 6, vengono esplorate le prospettive di sviluppo della metodologia e dei dati generati, delineando possibili applicazioni in diversi ambiti.

Capitolo 2

Fondamenti delle tecniche di intelligenza artificiale

L'Intelligenza Artificiale è un campo in costante evoluzione che trae ispirazione dall'abilità umana di apprendere, adattarsi e risolvere problemi complessi. Una delle principali caratteristiche dell'IA è la sua capacità di apprendere da dati, un concetto fondamentale nel *machine learning*. Diamo in questo capitolo una breve occhiata alle tecniche di Intelligenza Artificiale che hanno portato allo sviluppo di modelli di linguaggio avanzati come GPT.

2.1 Natural Language Processing (NLP)

Il Natural Language Processing (NLP) rappresenta il cuore pulsante dell'Intelligenza Artificiale, offrendo alle macchine la capacità di comprendere, interpretare e generare il linguaggio umano in modo simile all'interazione umana. Questo campo si estende ben oltre la semplice manipolazione di parole e frasi, immergendosi nei complessi meccanismi che regolano la comunicazione linguistica.

Nel contesto dell'NLP, l'analisi semantica gioca un ruolo fondamentale, permettendo alle macchine di comprendere il significato sottostante alle parole. Questo implica la capacità di riconoscere relazioni, sentimenti e concetti impliciti nei testi, rendendo l'interpretazione dei dati linguistici più profonda e accurata. L'analisi semantica è cruciale per applicazioni come la ricerca semantica su motori di ricerca, l'analisi del sentiment nei social media e la classificazione del testo.

La comprensione del contesto è un altro elemento chiave dell'NLP. Le parole possono assumere significati diversi a seconda del contesto in cui sono utilizzate, e gli algoritmi di

NLP devono essere in grado di discernere queste sfumature per interpretare correttamente il significato di una frase. Questa capacità è fondamentale per applicazioni come l'assistenza virtuale, in cui la macchina deve rispondere in modo appropriato alle domande degli utenti.

Inoltre, l'NLP si occupa della generazione di testo, consentendo alle macchine di produrre testo coerente e comprensibile. Questo aspetto è centrale in scenari come la scrittura automatica di email, la creazione di contenuti generati automaticamente e la composizione di risposte basate su dati precedentemente appresi.

2.2 L'evoluzione dei Trasformers

L'evoluzione dei Trasformers rappresenta un capitolo entusiasmante nella storia dell'Intelligenza Artificiale, con impatti significativi soprattutto nel campo del Natural Language Processing (NLP). Introdotta nel 2017 (Radford et al. [5]), il modello Transformer ha rivoluzionato l'approccio alla gestione del contesto e delle relazioni a lungo raggio nei dati sequenziali.

L'architettura dei Trasformers si basa sull'attenzione, una meccanica che consente al modello di attribuire pesi differenti a diverse parti di una sequenza in base alla loro rilevanza. Questo approccio consente ai Trasformers di catturare correlazioni complesse in dati sequenziali, superando le limitazioni dei modelli precedenti che facevano affidamento su strutture ricorrenti o convoluzionali.

La chiave del successo dei Trasformers risiede nella loro capacità di parallelizzare l'elaborazione durante l'addestramento. A differenza dei modelli precedenti, che richiedevano l'elaborazione sequenziale dei dati, i Trasformers possono processare simultaneamente diverse parti di una sequenza, accelerando notevolmente i tempi di addestramento. Ciò ha reso possibile l'impiego di dataset più ampi e complessi, contribuendo al miglioramento delle prestazioni generali.

Un punto culminante dell'evoluzione dei Trasformers è stato l'introduzione di varianti come BERT (Bidirectional Encoder Representations from Transformers) e GPT (Generative Pre-trained Transformer). BERT ha portato miglioramenti significativi alla comprensione del contesto bidirezionale, mentre GPT ha spinto oltre i confini della generazione di testo pre-addestrato, aprendo la strada a applicazioni creative e interattive.

La flessibilità dei Trasformers si estende oltre il NLP, trovando applicazioni in campi come la computer vision, la traduzione automatica e il riconoscimento di pattern complessi. La loro capacità di apprendere rappresentazioni ricche e contestualmente significative ha reso i Trasformers una scelta predominante in molteplici settori dell'IA.

2.3 OpenAI GPT (Generative Pre-trained Transformer)

GPT, sviluppato da OpenAI [6], rappresenta un trionfo nell'applicazione dei Trasformers nell'Intelligenza Artificiale. La sua architettura basata sull'attenzione, insieme all'approccio pre-addestrato, ha sollevato l'efficienza e le prestazioni dei modelli di linguaggio a un livello senza precedenti.

La fase pre-addestrata di GPT coinvolge l'allenamento su grandi quantità di testo da fonti varie, esponendo il modello a una vasta gamma di strutture linguistiche, stili di scrittura e conoscenza generale. Ciò consente a GPT di acquisire una comprensione approfondita della struttura del linguaggio naturale, catturando relazioni complesse e modelli di contesto.

La caratteristica distintiva di GPT risiede nella sua capacità di generare testo coerente e significativo in risposta a input specifici. Questo lo rende un potente strumento per compiti di generazione di testo automatico, come la stesura di articoli, la creazione di contenuti creativi e la risposta a domande complesse. La flessibilità di GPT si manifesta nel fatto che può essere adattato a una vasta gamma di compiti specifici, mediante l'approccio di affinamento esponendolo a dataset ristretti e focalizzati. Questa 'personalizzazione' ottimizza il modello per produrre risultati rilevanti in un determinato contesto, rendendo GPT un alleato ideale per applicazioni specializzate.

Capitolo 3

Design ed implementazione del sistema

Un attacco è considerato come una sequenza finita di azioni eseguite in modo sequenziale. Da questa premessa, è possibile suddividere tali fasi (*step*) in azioni elementari (sottostep), cioè non ulteriormente scomponibili. In questo modo, è possibile rappresentare un attacco come una sequenza di step (e sottostep) da cui è possibile estrarre le informazioni che lo caratterizzano. Il modello sarà il template che ospiterà questi dati e potrà essere esteso per includere dati estratti dagli step e dati generati da GPT. Queste informazioni aggiuntive sono utili per contestualizzare le azioni e migliorare la comprensione dell'attacco.

Ci riferiremo al modello che ospita step e sottostep con il nome di StepModel.

Una volta modellato l'attacco, è possibile etichettarlo utilizzando una tassonomia generata in modo semi-automatico frutto di un lavoro parallelo. Questa tassonomia è stata creata utilizzando GPT su un dataset composto da sottostep estratti da attacchi modellati. L'etichettatura dei sottostep consente di mappare l'azione utilizzando una "Tecnica", ossia una categoria che mira a rappresentare l'azione nel modo più specifico possibile.

Un'operazione necessaria è l'estrazione dei Digital Artefacts, cioè qualsiasi oggetto che possa avere un significato all'interno di un'azione. Per estrarre i Digital Artefacts, viene utilizzato un approccio ricorsivo di estrazione sugli step che porta alla creazione di una lista di questi oggetti.

Verrà introdotto il concetto di Path Nascosti, vale a dire collegamenti tra azioni non immediatamente evidenti, estratti mediante l'impiego di GPT-4-Turbo.

Lo StepModel costituirà il dato su cui lavorerà il framework "AttackModeler". Da questo modello, sarà possibile mappare l'attacco utilizzando classi specifiche (Attack, Node,

DigitalArtefact), portandoci verso una rappresentazione diversa dell'attacco chiamata NodeModel. Un Nodo è l'equivalente di un sottostep nello StepModel e conterrà informazioni rilevanti per l'analisi.

Il NodeModel verrà quindi rappresentato mediante una struttura a grafo con a supporto un report basato sull'analisi della sequenza dei nodi, il quale consente di visualizzare le Tecniche connesse tra loro e i Digital Artefacts coinvolti.

In Figura 3.1 è possibile vedere una rappresentazione semplificata del sistema.

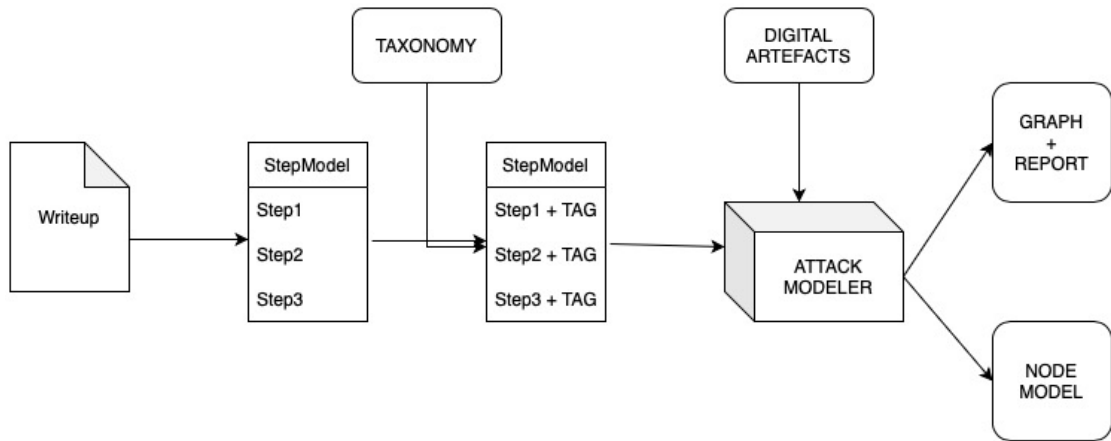


Figura 3.1: Architettura del sistema.

3.1 Sviluppo di una metodologia automatizzata

L'obiettivo di sviluppare una metodologia automatizzata è garantire che il lavoro possa adattarsi facilmente a diversi tipi di dati senza richiedere modifiche sostanziali nella sua struttura di base. Ciò significa che il sistema deve essere progettato per essere altamente flessibile e in grado di gestire in modo efficiente nuovi tipi di dati senza richiedere una riconfigurazione significativa.

Per raggiungere questo obiettivo, l'approccio modulare è fondamentale. Ogni fase del processo, dall'estrazione iniziale dei dati alla creazione del grafo, deve essere suddivisa in moduli autonomi e interconnessi. Questi moduli devono essere progettati per essere facilmente sostituibili o estendibili, consentendo così al sistema di integrare nuovi tipi di dati senza compromettere l'integrità dell'intero lavoro.

La struttura del modello, deve essere progettata con un'attenzione particolare all'adattabilità. Ciò implica che la rappresentazione dei dati deve essere sufficientemente generica da poter incorporare informazioni specifiche di diversi contesti senza richiedere modifiche

sostanziali. La flessibilità del modello consente una maggiore portabilità del lavoro su diverse tipologie di dati.

Il sistema deve essere progettato con l'idea di essere versatile e generale. La sua implementazione deve consentire l'estensione e l'adattamento a nuovi scenari senza richiedere una riprogettazione completa. Questa caratteristica è essenziale per garantire che il framework mantenga la sua utilità nel tempo, rimanendo all'avanguardia anche di fronte a nuovi tipi di problematiche.

3.2 GPT: ingegnerizzazione del prompt

La potenza di uno strumento come GPT emerge chiaramente fin dalle prime interazioni, evidenziando una notevole precisione nella gestione di richieste complesse e dettagliate. Al fine di illustrare questa precisione, esaminiamo la capacità di GPT nell'estrarre il "comportamento dell'attaccante" da un writeup fornito come input:

- Prompt:

```
"I'll give you a CTF writeup, I want you to extract from it the
  attacker's behaviour:"
[writeup-bodycount]
```

Listing 3.1: GPT Test 1 - prompt.

- gpt3.5-turbo:

```
## Attacker's Behavior

1. **Initial Reconnaissance:**
   - The attacker observed the initial behavior of the website by
     visiting it and noticed a redirection to '/?file=wc.php'.

2. **File Inclusion Vulnerability:**
   - Identified a potential file inclusion vulnerability, allowing
     the inclusion of files from the server using the '?file='
     parameter.

3. **Discovery of Robots.txt:**
   - Discovered the existence of a 'robots.txt' file which revealed
     the disallowed file 'checkpass.php'.

4. **Analysis of Checkpass.php:**
   - Attempted to access 'checkpass.php' and found a message
     indicating that the page is under development and has a secret.

5. **Base64 Encoding to View Source:**
```

```
- Used 'php://filter' to view the base64-encoded source of '
checkpass.php' and decoded it, revealing PHP code that checked a
password and had a secret message.

6. **Identification of Suspicious Password:**
- Identified a suspicious '$password' variable in the code of '
checkpass.php'.

7. **Analysis of WC.php:**
- Analyzed the source code of 'wc.php' and noticed the existence
of a password check using cookies.

8. **Remote Code Execution (RCE) Exploitation:**
- Leveraged the discovered RCE vulnerability by injecting a
payload (''; ls #') to list the files in the current directory.

...

15. **Access to Flag:**
- Successfully accessed the flag located at './system/of/a/down/
flag.txt' using the elevated privileges.

16. **Flag Retrieval:**
- Printed the contents of the flag, revealing the challenge flag
'csictf{1nj3ct10n_15_p41nfu1}'.

In summary, the attacker followed a systematic approach, starting
with the identification of vulnerabilities, exploiting them for
initial access, and then progressively escalating privileges to
achieve the final goal of retrieving the challenge flag.
```

Listing 3.2: GPT Test 1 - risposta.

Come si può notare, la risposta fornita è estremamente dettagliata e coerente con ciò che immaginiamo possa essere il “comportamento dell’attaccante”. Tuttavia, è importante notare che nel prompt non è stata fornita alcuna definizione di tale concetto. Inoltre, è da sottolineare che il contesto iniziale fornito nel prompt è minimale e ciò mette in luce la notevole precisione della risposta in relazione alla qualità del quesito posto.

Ripetendo la stessa domanda, si ottiene una risposta concettualmente simile, sebbene non identica:

- gpt3.5-turbo:

```
# Attacker's Behavior Summary

## 1. Initial Discovery:
- Identified a PHP code injection vulnerability on the target
website.
```

```
## 2. File Inclusion Exploitation:
- Discovered that the website redirects to '/?file=wc.php'.
- Exploited the file inclusion vulnerability by checking files like
  '/etc/passwd' and 'robots.txt'.

## 3. Robots.txt Analysis:
- Discovered the existence of 'checkpass.php' through 'robots.txt'.
- Attempted to access 'checkpass.php' directly, revealing a message
  about the page being under development.

## 4. Source Code Analysis - checkpass.php:
- Used 'php://filter' to view the base64-encoded source code of '
  checkpass.php'.
- Revealed the existence of a '$password' variable with the value '
  w0rdc0unt123'.

## 5. Exploiting Password Protection:
- Set the 'password' cookie to the value obtained from '$password'
  ('w0rdc0unt123').
- Bypassed the password protection, gaining access to the restricted
  page.

## 6. Source Code Analysis - wc.php:
- Analyzed the source code of 'wc.php' and identified the use of the
  'exec' function with user input ('$text').
- Identified a potential remote code execution vulnerability.

## 7. Remote Code Execution:
- Attempted remote code execution by injecting the payload ';' ls
  '#'.
- Successfully executed commands on the server and obtained a shell.

## 8. Exploration of File System:
- Navigated through the file system and discovered a directory '/ctf'
  with various subdirectories.

## 9. Locating Flag:
- Located the 'flag.txt' file using the 'findaas' script or the '
  find' command.

## 10. Privilege Escalation - ctf User:
- Discovered a password hash for the user 'ctf' in the 'README' file
  ('6f246c872cbf0b7fd7530b7aa235e67e').
- Successfully brute-forced the password ('csictf') and switched to
  the 'ctf' user.

## 11. Flag Retrieval:
- Accessed the flag file ('./system/of/a/down/flag.txt') using the '
  ctf' user.

## 12. Flag Obtained:
- Retrieved the flag: 'csictf{1nj3ct10n_15_p41nfu1}'.
```

```
In summary, the attacker identified and exploited a PHP code injection vulnerability, bypassed password protection, achieved remote code execution, explored the file system, and eventually obtained the flag by escalating privileges and accessing restricted files.
```

Listing 3.3: GPT Test 2 - risposta.

Questo comportamento indesiderato è da evitare. Per guidare in modo più efficace questo strumento e ottenere risposte coerenti, è consigliabile suddividere l'interazione in più fasi. Pur riconoscendo la straordinaria capacità di GPT nel comprendere l'input, è importante sottolineare che, per applicazioni che richiedono un elevato grado di determinismo, è essenziale un'ingegnerizzazione del prompt. Questo processo permette di definire in modo più specifico il contesto e le aspettative, contribuendo così a minimizzare il rischio di informazioni non conformi alla richiesta.

È dunque opportuno stabilire un flusso di lavoro che favorisca più interazioni semplici rispetto a un singolo prompt complesso. Questo approccio, pur richiedendo più passaggi, consente di affinare progressivamente la richiesta e di ottenere risposte più precise e mirate, riducendo la probabilità di ambiguità o di informazioni non pertinenti.

3.3 Creazione del Dataset: il problema dei dati non strutturati

La creazione del dataset è avvenuta attraverso la ricerca manuale di CTF writeup. Una CTF (Capture The Flag) è una competizione composta da sfide, in cui i partecipanti devono risolvere enigmi legati alla sicurezza informatica per ottenere la Flag, confermando così il successo nella sfida. Di solito, a tali competizioni è permessa la partecipazione in team formati da più persone, e l'accesso è libero e gratuito, il che porta a un'affluenza massiccia nelle competizioni più importanti.

I team che risolvono le sfide tendono spesso a pubblicare su blog personali o su GitHub le strategie adottate per superare le challenge, i cosiddetti writeup. Da ciò risulta evidente che questi individui godono di piena libertà nella strutturazione e nella scrittura di tali documenti, proprio come hanno libertà nella risoluzione degli enigmi.

Questa situazione rende estremamente complessa l'analisi di tali documenti, poiché potrebbero contenere informazioni ridondanti che generano rumore o, come spesso accade, potrebbero omettere passaggi necessari per la comprensione dell'attacco in generale. Ciò si verifica perché tali documenti sono prodotti senza alcuna linea guida, su spontanea iniziativa dell'individuo.

D'altra parte, il quantitativo di questi dati che è possibile recuperare online è enorme, motivo per cui rappresentano una ghiotta opportunità per chi ha come obiettivo quello di comprendere come un attaccante ragiona.

E' possibile osservare un esempio di writeup dal dataset nell'Appendice A.1

3.3.1 Raccolta dati

La raccolta dei dati è avvenuta principalmente attraverso GitHub il quale ospita una vasta quantità di writeup. Tutti i file raccolti sono formattati in markdown. Esaminiamo in Figura 3.2 la directory che contiene i repository da cui sono stati estratti i dati:

```
~/Desktop/CTF_writeups_NLP_analysis/dataset_writeups/humans/github (Web Exploitation) main* > tree -L 1
├── DanArmor (Web Exploitation)
├── DavidRotert (Web Exploitation)
├── HHousen (Web Exploitation)
├── InfoSecIITR (Web Exploitation)
├── VoidHack (Web Exploitation)
├── csivitu (Web Exploitation)
├── infosec-ucalgary (Web Exploitation)
├── karma9874 (Web Exploitation)
├── m3ssap0 (Web Exploitation)
├── perfectblue (Web Exploitation)
├── terjanq (Web Exploitation)
├── thewhitecircle (Web Exploitation)
├── wani-hackase (Web Exploitation)
├── xiosex (Web Exploitation)
└── zeyu2001 (Web Exploitation)
```

Figura 3.2: Overview della directory contenente i repositories.

Nell'esplorare una delle directory, è possibile notare che i vari writeup sono suddivisi in base alle competizioni (Figura 3.3):

```
~/Desktop/CTF_writeups_NLP_analysis/dataset_writeups/humans/github (Web Exploitation) [doing]/m3ssap0 (Web Exploitation) [done] main* > tree
├── 35C3 Junior CTF
├── ASIS CTF Quals 2020
├── AUCTF 2020
├── Access Denied 1.2 2018
├── CONFidence CTF Teaser 2019
├── DCTF DEF CAMP Qualif 2018
├── DEF CON CTF Qualifier 2020
├── DarkCON CTF 2021
├── DarkCTF 2020
├── DawgCTF 2020
├── DefCamp CTF 2020
├── Evlz CTF 2019
├── FireShell CTF 2019
├── Google Capture The Flag 2020
├── Houseplant CTF 2020
├── Inferno CTF 2019
├── Insomni'hack teaser 2020
├── OverTheWire Advent Bonanza 2019
├── Quals Saudi and Oman National Cyber Security CTF 2019
├── RIITSEC CTF 2019
├── STEM CTF Cyber Challenge 2019
├── Securinets Prequals CTF 2019
├── TAMUctf 2020
├── TUCTF 2017
├── UTCTF 2020
├── VolgaCTF Qualifier 2020
├── X-MAS CTF 2019
├── angstromCTF 2020
├── b01lers CTF 2020
├── m0leCon CTF 2019
├── zer0pts CTF 2021
32 directories, 0 files
```

Figura 3.3: Overview di un repository.

3.4 Ruolo della modellazione

Come evidenziato dalle prove di interazione condotte con GPT nelle sezioni precedenti, emerge chiaramente che questo strumento dimostra una notevole abilità nell'interpretare le richieste, anche quando la loro qualità risulta essere scarsa. È stato dimostrato che GPT è in grado di comprendere in modo estremamente preciso il significato di istruzioni come "Estrarre il comportamento dell'attaccante" da un writeup, senza la necessità di fornire una definizione esplicita di tale concetto.

Sebbene una domanda semplice, formulata come nell'esempio, possa rivelarsi utile per ottenere un'idea generale dell'attacco, è fondamentale stabilire un approccio che assicuri una coerenza nelle risposte fornite dal modello.

3.4.1 AttackModel

Di seguito osserviamo la struttura json che è stata adottata per rappresentare un attacco:

```
{
  "AttackModel": {
    "AttackerCapabilities": [],
    "Target": {
```

```

        "Type": "",
        "Vulnerabilities": [],
        "SecurityMeasures": []
    },
    "Steps": [
        {
            "StepNumber": "",
            "Description": "",
            "StepDetails": {
                "Objective": "",
                "Action": "",
                "Result": "",
                "Success": "",
                "FailingReason": ""
            }
        }
    ]
}

```

Listing 3.4: Template dell'AttackModel.

Analizziamo dunque ogni parametro del modello:

- **AttackerCapabilities:** Capacità o competenze possedute dall'attaccante.
- **Target**
 - **Type:** Tipologia di target.
 - **Vulnerabilities:** Lista delle vulnerabilità presenti nel sistema target.
 - **SecurityMeasures:** Lista delle misure di sicurezza o configurazioni del sistema target.
- **Steps**
 - **StepNumber:** Descrizione dello step.
 - **Description:** Descrizione dello step.
 - **StepDetails**
 - * **Objective:** Definizione dell'obiettivo dell'attaccante, inclusa la sua ipotesi o convinzione riguardo allo scopo di questo passo e uno scenario ipotetico di ciò che l'attaccante intende raggiungere nei passaggi successivi se questo passo ha successo.
 - * **Action:** Descrizione delle azioni o tecniche specifiche utilizzate dall'attaccante per portare a termine questo passo
 - * **Result:** Esito o risultato delle azioni dell'attaccante durante questo passo
 - * **Success:** Indicatore di successo o fallimento per questo passo (ad esempio, Successo o Fallito)

- * **FailingReason:** Spiegazione del motivo del fallimento nel caso in cui il passo non abbia avuto successo.

3.4.2 Popolare l'AttackModel

L'AttackModel fornisce una rappresentazione dell'attacco nel suo insieme, questo vuol dire che per popolarlo è necessario fornire a GPT l'intero writeup insieme al template del modello. Vediamo il prompt utilizzato per popolare l'AttackModel:

```
Considering the following CTF writeup:

[bodycount_writeup]

I want you to fill the following AttackModel:

{
  "AttackModel": {
    "AttackerCapabilities": [
      "List the capabilities or skills possessed by the attacker."
    ],
    "Target": {
      "Type": "Specify the type of target.",
      "Vulnerabilities": [
        "List the vulnerabilities present in the target system."
      ],
      "SecurityMeasures": [
        "List the security measures or configurations of the
target."
      ]
    },
    "Steps": [
      {
        "StepNumber": "",
        "Description": "",
        "StepDetails": {
          "Objective": "The attacker's objective, including
their assumption or belief regarding the purpose of this step and a
hypothetical scenario of what the attacker intends to achieve in
future steps if this step is successful",
          "Action": "The specific actions or techniques used
by the attacker to carry out this step",
          "Result": "The outcome or result of the attacker's
actions during this step",
          "Success": "Success or failure indicator for this
step (e.g., Success or Failed)",
          "FailingReason": "If the step fails, the reason for
failure",
          "Success": "Success or failure indicator for this
step (e.g., Success or Failed)",
```

```
        "FailingReason": "If the step fails, the reason for
failure"
      }
    ]
  }
}
```

Listing 3.5: Prompt di popolamento AttackModel.

Esaminiamo un esempio di AttackModel popolato:

```
{
  "AttackModel": {
    "AttackerCapabilities": [
      "Knowledge of PHP code injection",
      "Ability to identify and exploit file inclusion
vulnerabilities",
      "Understanding of base64 encoding and decoding",
      "Exploitation of command injection vulnerabilities",
      "Bruteforcing password hashes"
    ],
    "Target": {
      "Type": "Web server running a PHP application",
      "Vulnerabilities": [
        "File inclusion vulnerability allowing access to
sensitive files",
        "Command injection vulnerability in the 'wc.php' script
",
        "Insufficient file permission on the 'flag.txt' file"
      ],
      "SecurityMeasures": [
        "Password protection on sensitive pages",
        "Code execution restrictions in PHP configuration",
        "File permission restrictions to prevent unauthorized
access"
      ]
    },
    "Steps": [...]
  }
}
```

Listing 3.6: AttackModel.

L'AttackModel è impiegato per ottenere una visione generale dell'attacco. Nel prompt utilizzato per generarlo, non vengono fornite indicazioni specifiche se non all'interno del template del modello.

Questo modello comprende una suddivisione in step, effettuata però in piena libertà da GPT. Essendo questa un'operazione delicata, sarà oggetto del prossimo modello, pertanto

ci riferiamo all'AttackModel come a un modello essenziale per ottenere una panoramica dell'attacco.

3.4.3 StepModel template

Lo StepModel rappresenta una versione più leggera dell'AttackModel, concentrandosi sulla rappresentazione dell'attacco come una sequenza di step senza l'aggiunta di ulteriori informazioni.

Di seguito, esaminiamo la struttura JSON adottata per rappresentare lo StepModel:

```
{
  "StepModel": {
    "Steps": [
      {
        "StepNumber": 1,
        "StepString": ""
      },
      {
        "StepNumber": 2,
        "StepString": ""
      },
      ...
    ]
  }
}
```

Listing 3.7: Template dello StepModel.

Il processo di popolamento dello StepModel si differenzia da quello dell'AttackModel poiché richiede una specifica attenzione alla suddivisione in step. Questa distinzione implica che tale operazione venga eseguita come un task isolato.

3.5 StepModel

Popolare lo StepModel rappresenta un'operazione che non viene eseguita direttamente tramite GPT, e tale scelta è motivata da due principali ragioni:

- **Automatizzazione del processo e Limite dei Token:** GPT impone un vincolo sull'utilizzo dei token, con un limite massimo di 4096. Questi token devono essere distribuiti tra la domanda e la risposta. Ad esempio, se la richiesta occupa 100 token, ma la risposta supera i 3996 token, GPT restituirà un errore ($3997 + 100 > 4096$). Ciò diventa problematico quando l'obiettivo è automatizzare l'intero processo,

dunque è necessario minimizzare le informazioni scambiate tra domanda e risposta, considerando che la dimensione in token del writeup non è nota.

- **Limitare la complessità del task:** Come già discusso, GPT eccelle maggiormente in compiti considerati “banali”. Il popolamento dell’AttackModel potrebbe sembrare un compito banale, ma non lo è. La richiesta in quel caso non si limita solo alla suddivisione in step, ma anche all’estrazione di numerose informazioni da essi, aumentando il rischio di errore.

3.5.1 Summarization

Il compito che GPT svolge per estrarre l’elenco degli step dal writeup rappresenta una semplice attività di summarization. Il writeup viene utilizzato come input senza subire alcuna operazione di pulizia preventiva. La richiesta che viene fatta a GPT è di rispondere con una lista di azioni fornendo un semplice esempio di output:

```
I'll share a Capture The Flag (CTF) writeup detailing a cybersecurity
challenge. Your task is to respond with a list of the key steps
executed by the attacker. Ensure the list maintains the original
wording from the writeup, and format it as follows:

""
- Step 1
- Step 2
- Step 3
...
""

[bodycount_writeup]
```

Listing 3.8: Prompt di summarization.

Di seguito osserviamo la risposta di GPT:

```
- Visit the website and observe the redirection to '/wc.php' indicating
  possible file inclusion.
- Explore 'robots.txt' to find the disallowed file 'checkpass.php'.
- Investigate the content of 'checkpass.php' using 'php://filter/convert
  .base64-encode/resource=checkpass.php'.
- Decode the base64-encoded content of 'checkpass.php' to reveal a
  suspicious '$password' variable.
- Check the source code of 'wc.php' to identify the password check based
  on the 'password' cookie.
- Set the cookie with the discovered password ('w0rdc0unt123') to access
  the webpage.
- Identify the potential for remote code execution by analyzing the
  source code of 'wc.php'.
```

```

- Exploit the remote code execution by injecting the payload ''; ls #'
  to list files in the current directory.
- Discover limitations in output display and choose to use a reverse
  shell for better exploration.
- Inject the reverse shell payload ''; bash -c "bash -i >& /dev/tcp/your
  .server.ip.address/8000 0>&1" #' replacing 'your.server.ip.address'
  with the actual IP.
- Gain a shell on the server and navigate to the '/ctf' directory.
- Explore folders in '/ctf' and discover the 'findaas' script, which
  reveals the location of 'flag.txt'.
- Attempt to read 'flag.txt', realizing that permission is denied.
- Find the password hash for user 'ctf' in the 'README' file ('6
  f246c872cbf0b7fd7530b7aa235e67e').
- Bruteforce the password, discover it is 'csictf', and switch to user '
  ctf'.
- Read the flag from 'flag.txt' using the elevated privileges.
- Celebrate! The flag is: csictf{1nj3ct10n_15_p41nful}

```

Listing 3.9: Summary del writeup.

Gli steps vengono quindi prelevati ed inseriti all'interno dello StepModel:

```

{
  "StepModel": {
    "Steps": [
      {
        "StepNumber": 1,
        "StepString": "Visit the website and observe the
redirection to '/wc.php' indicating possible file inclusion."
      },
      {
        "StepNumber": 2,
        "StepString": "Explore 'robots.txt' to find the
disallowed file 'checkpass.php'."
      },
      ...
    ]
  }
}

```

Listing 3.10: StepModel.

3.6 Etichettatura del dataset di StepModel

L'obiettivo del processo di etichettatura degli step consiste nell'attribuire un'etichetta a ciascuna azione in modo tale da classificarla all'interno di una categoria specifica. Il caso ideale sarebbe ottenere un insieme finito di azioni, tale per cui ciascuno step possa essere mappato su un'azione nota.

3.6.1 Scomposizione in sottostep

Il primo passo da fare è quello di suddividere gli step in azioni elementari, cioè non ulteriormente scomponibili. Questa operazione viene effettuata direttamente sullo StepModel tramite il seguente prompt:

```

Given the provided StepModel, which represents the action executed by an
attacker in the context of a cyber attack, I need you to break down
each step into its most granular substeps, further dividing them
until they are no longer divisible.

Please provide a detailed breakdown of each step, ensuring that the
substeps are as specific and detailed as possible.

Fill in the template below:

{
  "StepModel": {
    "Steps": [
      {
        "StepNumber": 1,
        "StepString": "",
        "Substeps": [
          {
            "SubstepNumber": 1.1,
            "SubstepString": ""
          },
          ...
        ]
      },
      ...
    ]
  }
}

Here's the StepModel:
[StepModel]
""

[bodycount_writeup]

```

Listing 3.11: Prompt di scomposizione in substep.

Osserviamo di seguito una scomposizione in sottostep:

```

{
  "AttackModel": {
    "Steps": [
      {
        "StepNumber": 1,

```

```

        "StepString": "Visited the website and got redirected to
'/?file=wc.php'",
        "Substeps": [
            {
                "SubstepNumber": 1.1,
                "SubstepString": "Visited the website."
            },
            {
                "SubstepNumber": 1.2,
                "SubstepString": "Observed redirection to '/?
file=wc.php'."
            }
        ]
    },
    {
        "StepNumber": 2,
        "StepString": "Visited '/?file=/etc/passwd' to confirm
server-side file inclusion",
        "Substeps": [
            {
                "SubstepNumber": 2.1,
                "SubstepString": "Visited '/?file=/etc/passwd'."
            },
            {
                "SubstepNumber": 2.2,
                "SubstepString": "Confirmed server-side file
inclusion."
            }
        ]
    },
    ...
]
}
}

```

Listing 3.12: Scomposizione in Substep.

Questo genere di operazione può essere eseguita concedendo maggiore libertà a GPT, in modo che possa riformulare i periodi suddividendoli in substeps o richiedere la conservazione dell'originale formulazione delle frasi. Nel prompt di esempio, è stata adottata la prima variante.

3.6.2 Tassonomia

L'obiettivo della tassonomia, nel caso ideale, consiste nel mappare ogni possibile azione dello StepModel su una specifica azione ben definita denominata "Tecnica". Se consideriamo la tassonomia come una struttura ad albero, le tecniche costituiranno le foglie, ossia i livelli più estremi della gerarchia.

La tassonomia è stata generata attraverso l'utilizzo di GPT seguendo un processo ricorsivo, il quale verrà delineato in maniera concisa:

- Costruzione del dataset di sottosteps.
- Suddivisione del dataset in chunks di circa 50-60 sottosteps ciascuno.
- Applicazione di una procedura di clustering con GPT su ciascun chunk, con l'assegnazione di un nome rappresentativo del contenuto a ciascun cluster risultante (Categoria).
- Suddivisione dell'insieme di tutte le categorie in chunks, seguita da un processo di accorpamento semantico e rimozione duplicati. Quest'ultimo comporta l'aggregazione di categorie con nomi differenti ma dal medesimo significato.
- Etichettatura di tutti i sottosteps con le rispettive categorie.
- Ripetizione del processo di clustering per ogni categoria al fine di estrarre le corrispondenti sottocategorie.

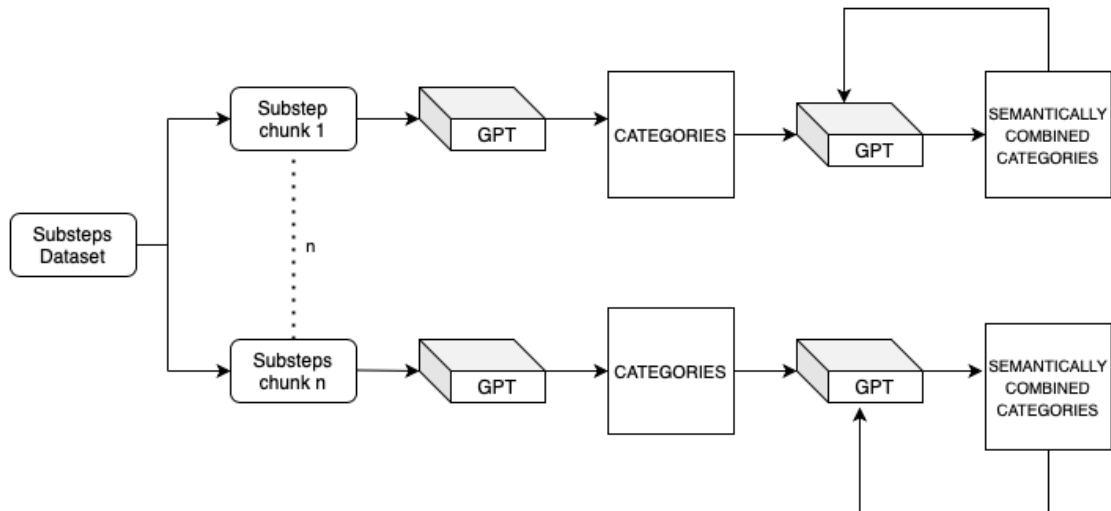


Figura 3.4: Costruzione tassonomia - Fase 1.

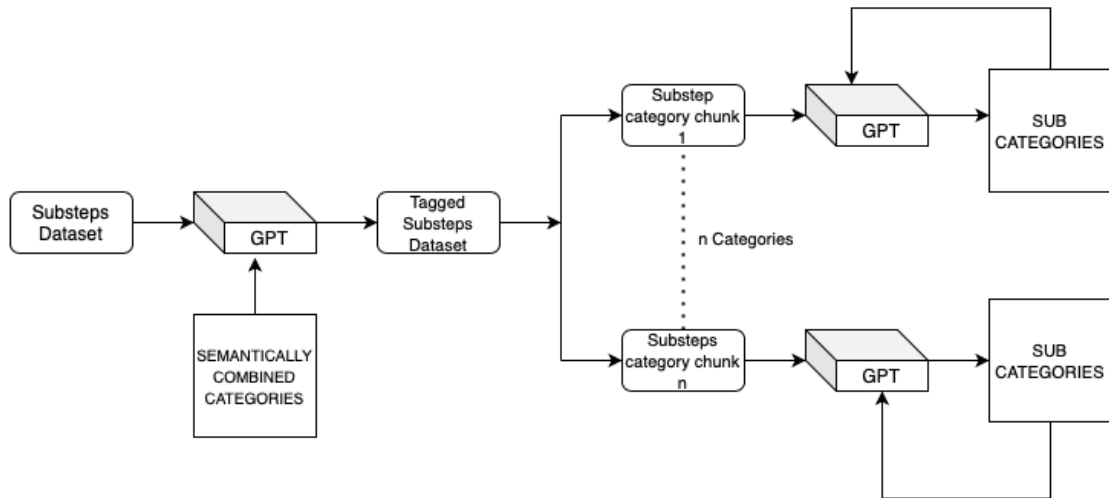


Figura 3.5: Costruzione tassonomia - Fase 2.

È importante notare che in questo elaborato saranno forniti i dettagli ed una descrizione completa del processo solo per le parti che sono stati oggetto di questo lavoro di tesi. Per maggiori approfondimenti sui passi del processo non coperti qui, è possibile far riferimento al lavoro di tesi di Lonardo [6].

Nell'Appendice A.2 è possibile osservare la tassonomia completa.

3.6.3 Le Tecniche

Le Tecniche sono l'ultimo livello della tassonomia e possono essere interpretate come un astrazione delle azioni che andranno a mappare. Idealmente potremmo dire che una tecnica rappresenta un metodo o un processo utilizzato dall'attaccante per eseguire attacchi o condurre attività mirate. Le tecniche possono variare ampiamente, e coprono le varie fasi da cui può essere composto l'attacco, quindi la raccolta di informazioni, l'analisi e l'esecuzione dell'attacco.

Nell'Appendice A.3 è possibile osservare l'insieme di tutte le tecniche.

3.7 Digital Artefacts

I Digital Artefacts sono dati, file o elementi di qualsiasi genere che vengono creati o modificati a seguito di attività umane, di processi o di qualsiasi tipo di interazione. Questi artefatti possono includere documenti, file multimediali, codice qualsiasi altra forma di

contenuto digitale che consenta l'analisi, l'interpretazione o la ricostruzione di eventi e interazioni.

L'individuazione di questi elementi all'interno delle azioni, insieme alle tecniche, ha lo scopo di descrivere quanto più possibile l'azione stessa.

3.7.1 Creazione della lista di Digital Artefacts

Si è deciso di generare una lista di Digital Artefacts con l'utilizzo di GPT, seguendo un processo simile a quello che è stato impiegato per la creazione della tassonomia, in modo tale che questa lista possa in futuro essere aggiornata.

Vediamo il processo che è stato eseguito per la generazione della lista di Digital Artefacts:

- Costruzione del dataset di sottosteps.
- Suddivisione del dataset in chunks di circa 50-60 sottosteps ciascuno.
- Applicazione di una procedura di estrazione di Digital Artefacts con GPT su ciascun chunk.
- Accorpamento di tutti i Digital Artefacts.
- Suddivisione in chunks della lista di Digital Artefacts.
- Accorpamento semantico e rimozione duplicati.

Osserviamo il prompt utilizzato per la prima estrazione:

```
Digital Artefacts are data, files or elements of any kind that are
  created or modified as a result of human activities, processes or
  any type of interaction.
Try to extract, from each of the following actions the Digital Artifacts
, if possible.
Examples of Digital Artefacts can be Path, File, IP Address, etc.
Respond exclusively with the identified list of Digital Artefacts.
[Substeps chunk]
```

Listing 3.13: Prompt di scomposizione in substeps.

Il risultato di questo prompt sarà una lista di Digital Artefacts. Considerato che il dataset comprende circa 7000 substeps, ed ogni chunk fornito a GPT è di 50 sottostep, l'intero processo di estrazione richiede circa 140 interazioni. Le liste risultanti, andranno fuse e sottoposte nuovamente a GPT per la rimozione dei duplicati e l'accorpamento semantico.

Questa operazione deve avvenire in maniera ciclica finquando GPT restituirà una lista della stessa dimensione di quella data in input.

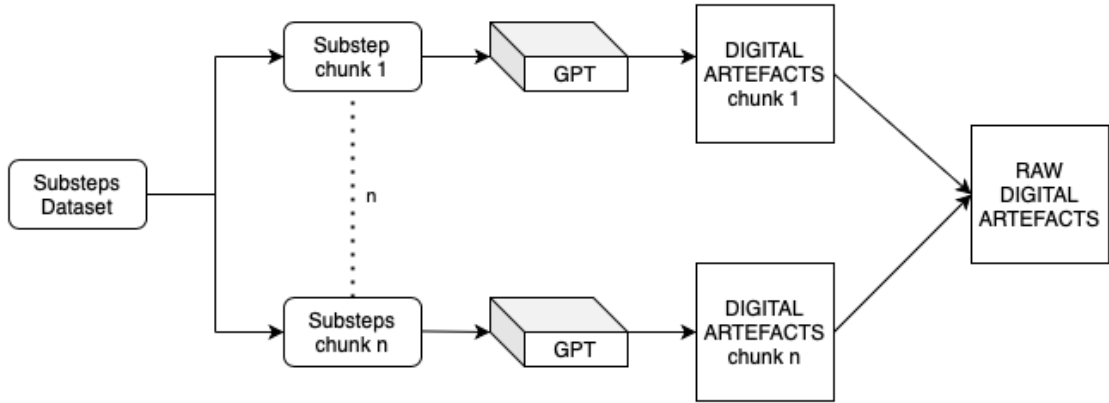


Figura 3.6: Fase 1 - Creazione lista di Digital Artefacts.

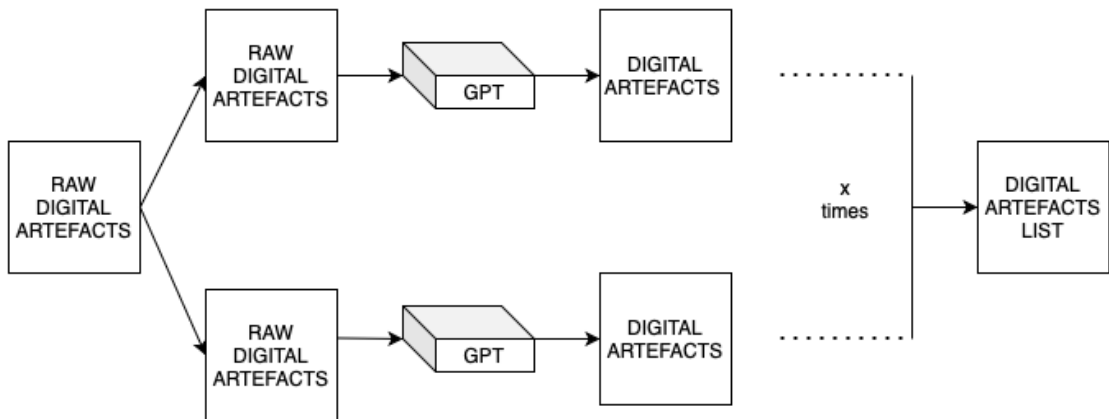


Figura 3.7: Fase 2 - Creazione lista di Digital Artefacts.

Vediamo di seguito la lista dei Digital Artefacts, risultato del processo descritto:

```
- file
- log
- path
- folder
- url
- cookie
- credentials
- key
- token
- query
- database_record
- ip_address
- http_request
- http_response
- hash_value
- digital_signature
- metadata
- encoded_data
- shell_command
- json_data
- input_data
- flag
- system_information
- form_field
- code
- library
- tool
- vulnerability
```

Listing 3.14: Lista dei Digital Artefacts.

3.7.2 Attributi

Gli attributi rappresentano le informazioni fondamentali associate a un artefatto, risultando cruciali per l'estrazione di dati dalle azioni. Ad esempio, quando l'attaccante interagisce con un file, è importante caratterizzare tale artefatto mediante le informazioni legate ad esso, come ad esempio il nome del file o la sua estensione. Gli attributi possono essere estratti molto semplicemente con l'ausilio di GPT partendo dalla lista degli artefatti.

```
- file (filename="", extension="", description="")
- log (filename="", description="", events="")
- path (directory="", description="")
- folder (name="", description="")
- url (value="", description="", purpose="", parameters="")
- cookie (name="", value="", expiration_time="", description="")
- credentials (username="", password="", service="")
- key (key="", description="", service="")
```

```
- token (token="", token_type="", description="", expiration="",
  verification_algorithm="")
- query (query="", description="", purpose="")
- database_record (data="", table="", description="")
- ip_address (ip="", location="", description="")
- http_request (data="", method="", parameters="", headers="", route="",
  description="")
- http_response (data="", headers="", content="", format="", description
  = "", status="")
- hash_value (hash="", algorithm="", description="")
- digital_signature (signature="", description="", purpose="")
- metadata (file="", metadata="", description="")
- encoded_data (data="", encoding="", description="")
- shell_command (command="", description="", usage="")
- json_data (data="", description="", structure="")
- input_data (fields="", description="", context="")
- flag (flag="", description="", context="")
- system_information (info="", description="", platform="")
- form_field (field="", value="", description="", purpose="")
- code (language="", description="", context="")
- library (library_name="", description="", language="", documentation
  = "")
- tool (name="", description="", purpose="")
- vulnerability (type="", cve_number="", description="")
```

Listing 3.15: Lista dei Digital Artefacts con attributi.

L'attributo "Description" è stato intenzionalmente incluso per ciascun artefatto, poiché gli attributi saranno dati che GPT compilerà. Pertanto, nel caso in cui l'azione non contenga le informazioni ricercate, GPT potrà popolare il campo "Description" in modo autonomo per fornire una panoramica dell'oggetto se necessario.

3.8 Estrazione dei 'Path Nascosti'

Possiamo concepire l'attacco come una sequenza ordinata di azioni, connesse principalmente dal fatto che vengono eseguite in successione. Naturalmente, queste azioni presentano anche una dipendenza temporale e semantica, poiché, ad esempio, non è possibile sfruttare una vulnerabilità se questa non viene prima rilevata. Considerata tale rappresentazione, la correlazione tra azioni non adiacenti non è immediatamente evidente.

Gli elementi denominati "Path Nascosti" indicano quindi correlazioni che non emergono dalla rappresentazione sequenziale utilizzata. Identificare questi "Path Nascosti" implica la ricerca di correlazioni tra i nodi, basata su qualsiasi informazione in essi contenuta.

Sono state condotte numerose prove al fine di elaborare un prompt che consentisse a gpt-3.5-turbo di eseguire l'operazione necessaria per estrarre tali percorsi. Nonostante la

sua potenza questo strumento si è rivelato inadatto, costringendo così all'utilizzo del suo successore, GPT-4.

Il primo passo nell'estrazione dei percorsi consiste nella rimozione degli step dallo StepModel, ottenendo così una successione di sottostep. Questa operazione è essenziale, poiché gli step non sono necessari per il procedimento e, anzi, introducono inutilmente della complessità.

Di seguito abbiamo un esempio di uno StepModel, senza la struttura Step{Sottostep}:

```
{
  "AttackModel": {
    "Substeps": [
      {
        "SubstepNumber": 1.1,
        "SubstepString": "Visited the website."
      },
      {
        "SubstepNumber": 1.2,
        "SubstepString": "Observed redirection to '/?file=wc.php'."
      },
      {
        "SubstepNumber": 2.1,
        "SubstepString": "Visited '/?file=/etc/passwd'."
      },
      {
        "SubstepNumber": 2.2,
        "SubstepString": "Confirmed server-side file inclusion."
      },
      ...
    ]
  }
}
```

Listing 3.16: Rimozione Steps dallo StepModel.

Osserviamo quindi il prompt utilizzato per l'estrazione dei path nascosti:

```
Consider the following attack model, which is structured as an ordered
sequence of substeps.

Your task is to identify and extract the hidden path between substeps. A
path is considered hidden when it involves correlations between
substeps that are not directly adjacent, requiring at least one
intermediary substep between them. When replying, provide the links
between substeps; for example, if Substep 1.1 is linked to Substep
4.2, your response should be {1.1, 4.2}.

[StepModel]
```

Listing 3.17: Prompt di ricerca path nascosti v1.

E' stata sviluppata una seconda versione di questo prompt che richiede di inserire informazioni per motivare ogni correlazione.

```
Consider the following attack model, which is structured as an ordered
sequence of substeps.

Your task is to identify and extract the hidden path between substeps. A
path is considered hidden when it involves correlations between
substeps that are not directly adjacent, requiring at least one
intermediary substep between them. When replying, provide the links
between substeps; for example, if Substep 1.1 is linked to Substep
4.2, your response should be {1.1, 4.2}.
Highlight correlations, ensuring that they involve a maximum of two
nodes.

For each link {substep x, substep y}, elucidate the correlation using
the following template:

{substep x, substep y} - info

[StepModel]
```

Listing 3.18: Prompt di ricerca path nascosti v2.

Questa seconda versione del prompt è essenziale per comprendere le ragioni alla base della selezione di specifiche correlazioni.

Capitolo 4

AttackModeler: un framework per la mappatura e l'analisi di StepModel

Al centro di questa analisi si trovano le decisioni progettuali che hanno guidato lo sviluppo del framework, delineando le fondamenta su cui si basa il suo funzionamento. Prima di immergersi nei dettagli implementativi, è essenziale comprendere il contesto e gli obiettivi che hanno ispirato lo sviluppo di questo strumento.

Il motivo principale che ha spinto verso lo sviluppo dell'AttackModeler è la creazione di un dataset di attacchi direttamente analizzabili e navigabili a partire dai dati disponibili, ossia dagli StepModel.

Data la giovane natura del progetto, è stato imperativo considerare, durante il processo di sviluppo, la necessità di renderlo modulare e scalabile, altrimenti con l'incremento dei dati, la configurazione di quest'ultimo avrebbe richiesto delle sostanziose modifiche.

Esaminiamo di seguito le funzionalità del framework:

- Mappatura di uno StepModel e conversione in NodeModel;
- Salvataggio di un NodeModel in formato binario;
- Caricamento di un NodeModel da file binario;
- Visualizzazione di un NodeModel;
- Rappresentazione grafica di un NodeModel secondo una struttura a grafo;

- Generazione di un report informativo sul NodeModel.

Il NodeModel rappresenta un'evoluzione dello StepModel, orientata maggiormente verso una struttura a grafo. Gli Step vengono eliminati e all'interno del NodeModel troviamo i Node, corrispondenti ai sottostep dello StepModel.

Un attacco può essere concepito come una sequenza di nodi (sottostep) interconnessi, presentando i dati in una maniera alternativa rispetto al modello a Step.

Avere un dataset di NodeModel risulta essere di grande aiuto, poichè questo rappresenta una fonte importante per l'analisi di attacchi. Considerato che il framework è stato pensato secondo un'architettura modulare, è possibile creare nuovi moduli per effettuare nuovi tipi di analisi sui dati.

4.1 Struttura del Framework

Vediamo in Figura 4.1 la struttura dell'interno progetto:

```
.
├── data
│   ├── input_data
│   └── output_data
├── example.py
├── src
│   ├── __init__.py
│   ├── input_data
│   │   └── digital_artefacts.txt
│   └── lib
│       ├── Attack.py
│       ├── DigitalArtefact.py
│       ├── Node.py
│       ├── digital_artefact_extractor.py
│       ├── gpt_api_call.py
│       ├── prompts
│       │   └── digital_artefact
│       │       ├── attributes
│       │       └── digital_artefact_extractor_prompt.txt
│       ├── ranking_analysis.py
│       ├── sequence_analysis.py
│       └── stepModel_loader.py
```

Figura 4.1: Struttura del progetto.

4.1.1 Contenuto della directory principale

La directory principale del framework è strutturata come segue:

- **data:** contiene i dati di input (StepModel + Hidden path) necessari per l'esecuzione del framework e i dati di output cioè rappresentazioni serializzate (.pkl) e report (.txt).
- **“src: ”** contiene la directory **“lib”** la quale fornisce moduli e librerie che collaborano per il flusso di lavoro del framework, dalla mappatura dell'attacco all'estrazione degli artefatti digitali, all'analisi della sequenza e alla generazione di report. La directory **“src”** contiene anche **“input_data”** la quale ospita file di input aggiuntivi per il framework, come la lista dei digital artefacts.

4.2 Classi e Moduli

Nel contesto della progettazione del framework, si è adottato un approccio meticoloso nella strutturazione del codice per garantire chiarezza, manutenibilità e modularità. La directory chiave in questo contesto è **“lib”**, che raggruppa i moduli e le classi essenziali per la struttura del sistema.

All'interno di questa directory, troviamo diversi moduli che svolgono compiti specifici nel flusso di lavoro:

- **stepModel_loader:** Gestisce il caricamento degli StepModel, fornendo una base solida per avviare il processo di analisi.
- **gpt_api_call:** Si occupa delle chiamate API al modello gpt-3.5-turbo, assicurando un'interazione efficiente e affidabile.
- **Attack:** Ospita l'implementazione della classe omonima, centrata sulla mappatura dell'attacco. La sua logica è fondamentale per comprendere e rappresentare le fasi iniziali del processo.
- **Node:** Implementa la classe Node, rappresentando nodi nel modello e fornendo una base per navigare e capire la struttura del grafo di attacco.
- **DigitalArtefact:** Contiene l'implementazione della classe DigitalArtefact, responsabile dell'estrazione e della gestione degli artefatti digitali. Questo modulo è cruciale per acquisire e analizzare informazioni rilevanti attraverso il modello gpt-3.5-turbo.
- **digital_artefact_extractor:** Gestisce l'estrazione degli artefatti digitali tramite il modello gpt-3.5-turbo, contribuendo alla fase chiave di acquisizione delle informazioni.

- **sequence_analysis:** Implementa la logica per analizzare la sequenza dei nodi, offrendo una visione dettagliata delle interazioni e delle relazioni nel grafo.

Questa struttura modulare è pensata per semplificare la comprensione del codice, agevolando una gestione efficiente delle responsabilità specifiche di ciascun componente.

In Figura 4.2 è possibile vedere come i vari moduli siano collegati tra loro:

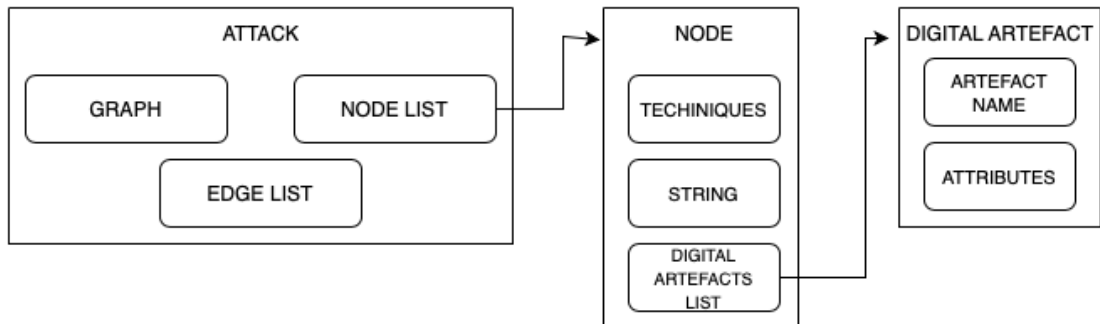


Figura 4.2: Collegamento tra i moduli.

Di seguito verrà analizzato il contenuto dei moduli più interessanti.

4.2.1 Attack

Il modulo Attack gestisce la rappresentazione e la manipolazione di un attacco. Tra i suoi attributi figurano:

- **G:** un oggetto grafo NetworkX.
- **node_list:** una lista di nodi.
- **edge_list:** una lista di archi.
- **legend_labels:** una lista di stringhe rappresentanti le etichette utili per rappresentare la legenda.

Da un punto di vista funzionale, la classe offre i seguenti metodi:

- **from_stepMode(stepModel_path, links_file_path):** questo metodo consente la creazione di un oggetto Attack a partire da uno StepModel caricato da file JSON ed un file contenente i path nascosti. L'implementazione utilizza la funzione

`load_stepModel()` dal modulo `stepModel_loader()` al fine di ottenere il grafo, le etichette di legenda, la lista di nodi e la lista di archi.

- **`from_pickle(pickle_file_path)`**: questo metodo consente il caricamento di un oggetto Attack da un file pickle, restituendo una nuova istanza di Attack con le stesse caratteristiche di quella precedentemente salvata.
- **`save_as_pickle(attack_name, output_directory)`**: consente di salvare l'oggetto Attack in un file pickle, il cui percorso è specificato dal nome dell'attacco e dalla directory di output. Nel caso in cui la directory di output non esista, viene creata.
- **`save_as_json(filename, output_directory)`**: permette di esportare l'oggetto Attack in un file JSON (NodeModel).
- **`plot()`**: questo metodo genera una rappresentazione grafica dell'attacco utilizzando le librerie `Matplotlib` e `NetworkX`. Il grafo risultante è caratterizzato da nodi, archi, etichette e una legenda che fornisce informazioni sui path nascosti.

4.2.2 Node

La classe `Node` è progettata per incapsulare le informazioni essenziali associate a un nodo dell'attacco. Tra i suoi attributi figurano:

- **`node_number`**: un intero che identifica univocamente il numero del nodo.
- **`node_techniques`**: una lista di tecniche associate al nodo.
- **`node_string`**: una stringa che rappresenta il nodo.
- **`node_digital_artefacts`**: una lista di artefatti digitali associati al nodo.

Da un punto di vista funzionale, la classe offre i seguenti metodi:

- **`to_json_string()`**: restituisce una rappresentazione JSON dell'oggetto nodo, escludendo gli artefatti digitali. Questo metodo mira a fornire una visione compatta e leggibile del nodo senza dettagli specifici sugli artefatti.
- **`to_json_string_with_digital_artefacts()`**: restituisce una rappresentazione JSON dell'oggetto nodo, includendo gli artefatti digitali.
- **`to_json_string_with_digital_artefacts_template(artefact_list)`**: ritorna una rappresentazione JSON del nodo senza compilare gli attributi degli artefatti digitali, accettando una lista di artefatti digitali così da crearne un template che potrà successivamente essere compilato.
- **`from_json_string(json_string)`**: restituisce un oggetto `Node` da una rappresentazione JSON del nodo.

4.2.3 DigitalArtefact

Il modulo DigitalArtefact ha lo scopo di fornire una rappresentazione strutturata e manipolabile degli artefatti digitali, consentendo la gestione degli attributi associati. I campi di un Digital Artefact sono i seguenti:

- **artefact_name:** Il nome dell'artefatto digitale.
- **artefact_attributes:** Un dizionario contenente gli attributi dell'artefatto digitale.

La classe offre i seguenti metodi:

- **to_json_string():** un metodo che converte l'oggetto DigitalArtefact in una stringa JSON, facilitando la rappresentazione serializzata del Digital Artefact.
- **from_json_string(cls, input_str):** un metodo di classe che crea un oggetto DigitalArtefact da una stringa JSON fornita in input.

Il codice esegue il caricamento iniziale dei Digital Artefacts utilizzando la funzione `load_digital_artefacts_list()` che accede al file (`src/lib/prompts/digital_artefact/digital_artefacts.txt`) successivamente utilizza la funzione `prompt_generator` per generare un prompt specializzato per ogni artefatto.

All'interno dello stesso modulo possiamo trovare la definizione di altre funzioni:

- `prompt_generator(artefact)`

La lista degli artefatti viene estratta a partire da ogni Nodo dell'attacco. Ogni artefatto però richiederà un'interazione separata per la compilazione dei suoi attributi, dunque un prompt per ogni tipologia di artefatto è necessaria. Questa funzione si assicura che modificando la lista di Digital Artefacts disponibili, i prompt mancanti vengano creati. Il nuovo prompt file viene creato nella directory (`src/lib/prompts/digital_artefact/attributes/`) e contiene informazioni sul nome dell'artefatto e sui suoi attributi.

- `get_digital_artefacts_list()`

La funzione `get_digital_artefacts_list()` restituisce la lista degli artefatti digitali caricati dal file txt.

- `get_digital_artefact(artefact_name)`

La funzione `get_digital_artefact(artefact_name)` restituisce un oggetto `DigitalArtefact` con il nome specificato, o `None` se non trovato. Utilizza la lista degli artefatti digitali ottenuta da `get_digital_artefacts_list()`.

4.2.4 `digital_artefact_extractor`

Il modulo `digital_artefact_extractor` rappresenta un componente dedicato all'estrazione e all'arricchimento di Digital Artefacts da nodi specifici all'interno di un attacco. Questo modulo interagisce direttamente con il modello `gpt-3.5-turbo` per acquisire informazioni dettagliate e significative.

La sua struttura è progettata per gestire il processo iterativo di estrazione degli artefatti digitali e il riempimento dei relativi attributi. Il modulo si compone dei seguenti elementi chiave:

- **DIGITAL_ARTEFACT_EXTRACTOR_PROMPT_PATH**: percorso utilizzato per caricare il prompt necessario per l'estrazione della lista degli artefatti dal nodo.
- **DIGITAL_ARTEFACT_ATTRIBUTE_DIRECTORY_PATH**: percorso utilizzato per salvare i prompt generati per ogni tipologia di artefatto, necessari per l'estrazione degli attributi.
- **process_node(node)**: Questa funzione accetta un oggetto `Node` e procede con l'estrazione e l'arricchimento degli artefatti digitali associati a quel nodo. L'approccio iterativo con la gestione delle eccezioni consente una maggiore robustezza in caso di eventuali errori di elaborazione.
- **digital_artefact_extraction(attack)**: Questa funzione si occupa di orchestrare l'estrazione degli artefatti digitali per tutti i nodi presenti in un oggetto `Attack`. Attraverso l'utilizzo di `process_node`, il modulo elabora ciascun nodo nel contesto dell'intero attacco, aggiornando l'oggetto `Attack` con i dati estratti.

In Figura 4.3 possiamo osservare il grafico che riguarda il processo di estrazione della lista degli artefatti dal nodo e successivamente il prompt utilizzato:

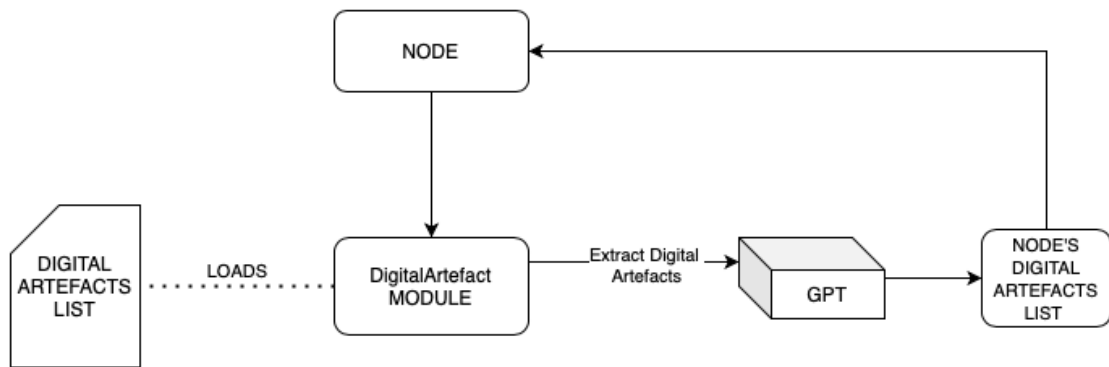


Figura 4.3: Processo di estrazione della lista di artefatti da un nodo.

```
I will give you a step extracted from a CTF writeup represented as a
JSON. Starting from the string, if you find any of the following
artefacts within it, you should insert them into the appropriate
field of the JSON.
Here is the list of digital artifacts you need to look for:

[artefacts list]

Reply just with the artefacts you find and put them inside a list ["
  artifact1", "artifact2", ...].
Here's the json:

[node]
```

Listing 4.1: Prompt di estrazione degli artefatti.

In Figura 4.4 abbiamo il processo di arricchimento degli attributi e successivamente un prompt di esempio che riguarda l'estrazione degli attributi relativi al Digital Artefact "Cookie":

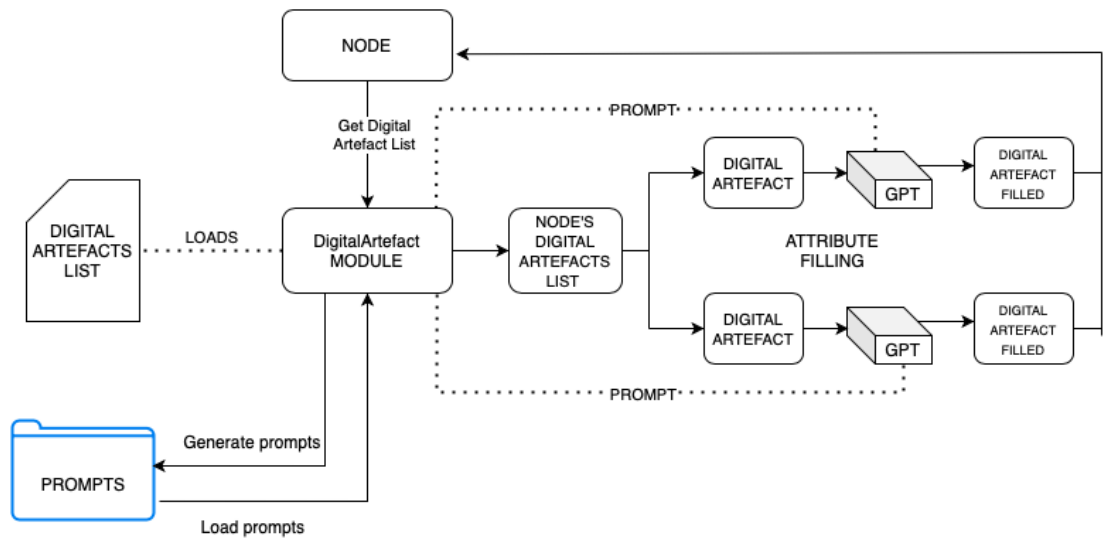


Figura 4.4: Processo di arricchimento degli attributi.

```

Considering the following digital artefact and its attributes:
Artefact Name: cookie
Attributes:
  - name
  - value
  - expiration_time
  - description
Please fill the following attributes if they are mentioned in the
node_string:
[node]

```

Listing 4.2: Prompt di arricchimento degli attributi.

All'interno di “lib” è presente la directory “prompts” dove vengono organizzati tutti i prompts necessari per l'utilizzo del modulo `gpt_api_call`.

4.2.5 `gpt_api_call`

Il modulo `gpt_api_call` rappresenta un componente fondamentale all'interno del framework, avendo il compito di orchestrate le comunicazioni con l'API di GPT-3.5-turbo fornita da OpenAI. Esso si articola principalmente attraverso la funzione `gpt_api_call`, la quale consente di invocare l'API con parametri configurabili quali il prompt, i dati di input, il numero massimo di token nella risposta, il numero massimo di tentativi di chiamata API in caso di fallimento, e il limite di tempo massimo per la risposta.

La funzione `gpt_api_call()` utilizza il paradigma di multiprocessing per gestire in parallelo la chiamata API e il monitoraggio del timeout. La chiamata API in sé è gestita dalla funzione `api_call`, la quale costruisce un registro di messaggi seguendo una struttura di dialogo e interagisce con l'API di GPT-3.5-turbo attraverso la libreria OpenAI. Una volta ottenuta la risposta dall'assistente virtuale, il risultato viene inserito in una coda condivisa.

Contemporaneamente, la funzione `timeout_check()` viene eseguita in un processo separato per monitorare il tempo trascorso. Nel caso in cui la chiamata API non sia completata entro il limite di tempo specificato, questa funzione termina la chiamata API e notifica il processo principale tramite la coda condivisa.

Il ciclo principale della funzione `gpt_api_call` gestisce iterativamente la chiamata API e il controllo del timeout per un numero massimo di tentativi definito dal parametro `max_retries`. In caso di successo, la risposta ottenuta viene restituita. Nel caso in cui il numero massimo di tentativi sia raggiunto senza successo, viene sollevata un'eccezione segnalando che il limite massimo di tentativi è stato superato.

Infine, il modulo offre una funzione ausiliaria `load_prompt()`, che consente di caricare un prompt da un file specificato nel percorso `prompt_path`. Questa funzione restituisce il contenuto del file come una stringa o segnala un'eccezione se il file specificato non è presente.

4.3 Workflow

L'utilizzo del framework può essere riassunto in 4 fasi:

- **Mappatura dell'attacco:** prendendo in input uno StepModel come file JSON, il framework lo mappa, utilizzando le classi `Attack` e `Node` e lo salva in un formato pkl (`NodeModel`), in modo da poterlo caricare e navigare.
- **Estrazione dei Digital Artefacts:** utilizzando un file `attack.pkl` (`NodeModel`) come input, il framework consente l'estrazione dei Digital Artefacts per ciascun nodo. La gestione dei Digital Artefacts avviene attraverso l'impiego della classe `DigitalArtefact`. Questi artefatti vengono estratti per ogni nodo attraverso l'utilizzo del modello `gpt-3.5-turbo`.
- **Riempimento attributi Digital Artefacts:** ciascun Digital Artefact viene preso singolarmente, e i suoi attributi vengono compilati attraverso l'uso del modello `gpt-3.5-turbo`.
- **Visualizzazione dell'attacco:** il `NodeModel` viene presentato su schermo mediante una rappresentazione a grafo, accompagnato da una legenda che semplifica l'analisi dei percorsi nascosti.

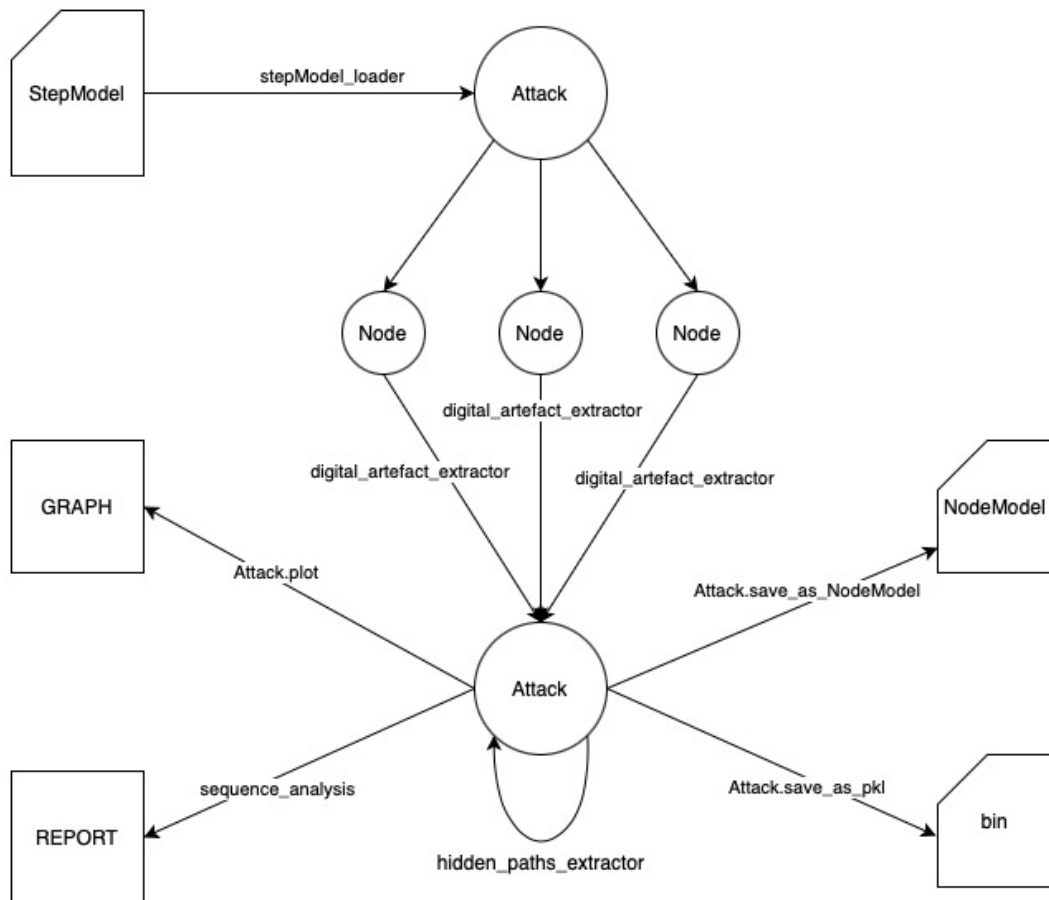


Figura 4.5: Panoramica del workflow.

- **Generazione del report di analisi:** il framework consente la creazione di un report dettagliato inerente all'analisi della sequenza dei nodi. Tale analisi permette di osservare l'interazione tra le diverse tecniche e i Digital Artefacts coinvolti nel processo.

In Figura 4.5 è possibile osservare una panoramica del workflow.

4.4 Utilizzo pratico: example.py

In questa sezione verrà esaminato come il framework possa essere utilizzato osservando il file example.py.

4.4.1 Caricamento dell'Attacco dallo StepModel

L'attacco viene caricato utilizzando la classe `Attack` del modulo `src.lib.Attack`, sfruttando il metodo `from_stepModel()`:

```
# Specify the file paths
stepModel_path = './data/input_data/StepModel2.json'
hidden_paths_file = './data/input_data/links.txt'

# Load attack from stepModel and links file
attack = Attack.from_stepModel(stepModel_path, hidden_paths_file)
```

4.4.2 Estrazione degli Artefatti Digitali

La funzione `digital_artefact_extraction()` dal modulo `src.lib.digital_artefact_extractor` viene utilizzata per integrare le informazioni estratte tramite il modello GPT con l'attacco:

```
# Extract digital artefacts (GPT)
attack_with_digital_artefacts = digital_artefact_extraction(attack)
```

4.4.3 Salvataggio dell'Attacco come file binario

L'attacco, arricchito con gli artefatti digitali, viene salvato come file pickle utilizzando il metodo `save_as_pickle()` nella directory `./data/output_data`:

```
# Save attack as pickle file
attack_with_digital_artefacts.save_as_pickle("bodycount", "./data/output_data")
```

4.4.4 Caricamento dell'Attacco da file binario

L'attacco precedentemente salvato viene ricaricato dal file pickle, rappresentando ora un `NodeModel`:

```
# Load attack (with digital artefacts) from pickle file as a NodeModel
attack = Attack.from_pickle("./data/output_data/bodycount.pkl")
```

4.4.5 Visualizzazione del Grafo dell'Attacco

Viene eseguito il plotting del grafo dell'attacco utilizzando il metodo `plot` della classe `Attack`. Questa rappresentazione visiva facilita la comprensione della struttura dell'attacco e delle relazioni tra i nodi, mostrando anche una leggenda che mostra le tecniche legate ai nodi coinvolti dai path nascosti:

```
## Plot the graph
attack.plot()
```

4.4.6 Salvataggio del NodeModel in Formato JSON

L'attacco, ora considerato un `NodeModel`, viene salvato come file JSON utilizzando il metodo `save_as_json()`. Il file viene memorizzato nella directory (`./data/output_data`):

```
# Save attack as NodeModel json file
node_model = attack.save_as_json("bodycount_nodeModel_short", "./data/
output_data")
```

4.4.7 Analisi della Sequenza (report)

Viene eseguita un'analisi della sequenza dell'attacco utilizzando la funzione `sequence_analysis()` dal modulo `src.lib.sequence_analysis`. Questa analisi fornisce un report dettagliato sull'interazione tra le diverse tecniche e gli artefatti digitali coinvolti nel processo:

```
# Print the the sequence analysis
report = sequence_analysis(attack)
print(report)
```

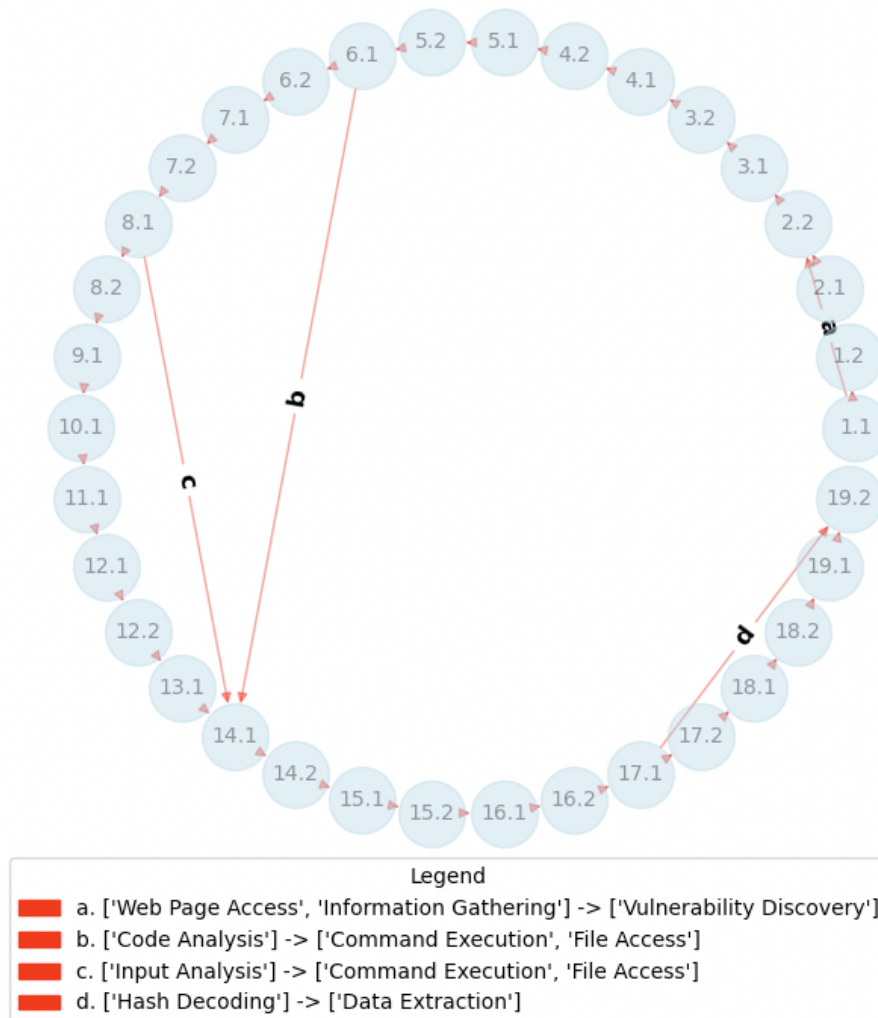


Figura 4.6: Esempio di grafo.

4.5 Generazione del grafo e report

In Figura 4.6 possiamo osservare un esempio di grafo generato da un attacco. I nodi sono collegati in sequenza, quindi ogni nodo è direttamente connesso al successivo. Tuttavia, è possibile notare, contrassegnati da etichette, i percorsi nascosti tra i nodi. La legenda illustra le tecniche impiegate all'interno dei nodi collegati tramite i percorsi nascosti, cioè una breve anticipazione delle informazioni contenute nel report.


```
Node: 1.1
* String: Visited the website.
* Technique: Web Page Access
* Digital Artefacts:
  - No digital artefacts
* Hidden Neighbors:
  - Node: 2.2
    * String: Confirmed server-side file inclusion.
    * Technique: Vulnerability Discovery
    * Digital Artefacts:
      - file | Attributes: {'filename': '', 'extension': '', 'description': 'Confirmed server-side file inclusion.'}

Node: 1.2
* String: Observed redirection to '/?file=wc.php'.
* Technique: HTTP Response Inspection
* Digital Artefacts:
  - file | Attributes: {'filename': 'wc', 'extension': 'php', 'description': ''}
* No Hidden Neighbors

Node: 2.1
* String: Visited '/?file=/etc/passwd'.
* Technique: Web Page Access
* Digital Artefacts:
  - file | Attributes: {'filename': '/etc/passwd', 'extension': '', 'description': ''}
* No Hidden Neighbors

Node: 2.2
* String: Confirmed server-side file inclusion.
* Technique: Vulnerability Discovery
* Digital Artefacts:
  - file | Attributes: {'filename': '', 'extension': '', 'description': 'Confirmed server-side file inclusion.'}
* No Hidden Neighbors
```

Figura 4.7: Breve estratto del report.

In Figura 4.7 possiamo apprezzare un breve estratto dal report relativo al grafo appena visto.

Per ciascun nodo viene mostrato il suo contenuto (String, Technique, DigitalArtefacts) e i nodi con i quali ha una correlazione non direttamente visibile (Hidden Neighbors). Come si può osservare, nel nostro caso il nodo 1.1 ha una correlazione con il nodo 2.2, pertanto viene mostrato il contenuto del nodo con cui è collegato.

È altresì possibile notare l'arricchimento degli attributi dei digital artefacts e come il campo 'Description' venga riempito solo nel caso in cui gli altri attributi non abbiano alcuna corrispondenza all'interno della stringa.

4.6 Generazione delle classifiche

Attraverso il modulo `ranking_analysis` si ha la possibilità di generare classifiche relative a un attacco specifico o classifiche globali, aggiornabili ogni volta che si analizza un attacco.

Sia nel caso della classifica globale che di un attacco specifico, vengono mostrate due tipologie di classifiche:

- **Techniques Ranking:** mostra quali tecniche sono le più utilizzate, dalla più frequente alla meno frequente.
- **Techniques-DigitalArtefact Ranking:** mostra tutte le coppie Tecnica-Artefatto, ordinandole dalla più frequente alla meno frequente.

Osserviamo di seguito come stampare le classifiche relative ad un attacco singolo:

```
# Print attack rankings
print_attack_rankings(attack)
```

Di seguito un esempio di output:

```
### Technique Ranking ###
* TECHNIQUE RANKING:
1. Web Page Access: 5 occurrences
2. Command Execution: 5 occurrences
3. File Discovery: 4 occurrences
4. Data Extraction: 4 occurrences
5. Page Behavior Analysis: 3 occurrences
6. File Access: 3 occurrences
7. Code Analysis: 3 occurrences
8. Data Analysis: 3 occurrences
9. Authentication: 3 occurrences
10. Information Gathering: 2 occurrences
...

### Technique-Digital_Artefact Pair Ranking ###
* TECHNIQUE-DIGITAL_ARTEFACT Pair Ranking:
1.Web Page Access - file: 3 occurrences
2.File Discovery - file: 3 occurrences
3.Command Execution - shell_command: 3 occurrences
4.File Access - file: 2 occurrences
5.Code Analysis - code: 2 occurrences
6.File Discovery - folder: 2 occurrences
7.HTTP Response Inspection - file: 1 occurrences
8.Vulnerability Discovery - file: 1 occurrences
9.Web Page Access - url: 1 occurrences
10.Web Page Access - path: 1 occurrences
...
```

Listing 4.3: Classifiche di un attacco.

Per quanto riguarda invece le classifiche globali, prima di essere stampate, vanno aggiornate inserendo le informazioni relative all'attacco che si sta analizzando. Le classifiche vengono salvate in un file json (`data/output_data/global_ranking.json`) e ricaricate ogni volta che vanno aggiornate o stampate.

```
# Update global rankings
update_global_ranking(attack)

# Print the global rankings
print_global_ranking()
```

Di seguito un esempio di output:

```
### Global Ranking ###
### Technique Ranking (Based on 3 attacks) ###
* TECHNIQUE RANKING:
1. Web Page Access: 5 occurrences
2. Command Execution: 5 occurrences
3. File Discovery: 4 occurrences
4. Data Extraction: 4 occurrences
5. Page Behavior Analysis: 3 occurrences
6. File Access: 3 occurrences
7. Code Analysis: 3 occurrences
8. Data Analysis: 3 occurrences
9. Authentication: 3 occurrences
10. Information Gathering: 2 occurrences
...

### Technique-Digital_Artefact Pair Ranking (Based on 3 attacks) ###
* TECHNIQUE-DIGITAL_ARTEFACT Pair Ranking:
1.Web Page Access - file: 3 occurrences
2.File Discovery - file: 3 occurrences
3.Command Execution - shell_command: 3 occurrences
4.File Access - file: 2 occurrences
5.Code Analysis - code: 2 occurrences
6.File Discovery - folder: 2 occurrences
7.HTTP Response Inspection - file: 1 occurrences
8.Vulnerability Discovery - file: 1 occurrences
9.Web Page Access - url: 1 occurrences
10.Web Page Access - path: 1 occurrences
...
```

Listing 4.4: Classifiche globali.

Questa tipologia di dati offre una panoramica su un dataset di NodeModel, risultando utile nel caso in cui l'analisi non sia mirata a un singolo attacco, bensì miri a individuare pattern all'interno di una vasta gamma di attacchi.

Capitolo 5

Rappresentazione completa di un attacco e valutazione empirica

Come già discusso, l'obiettivo di questo lavoro è fornire una rappresentazione del comportamento dell'attaccante. A tale scopo, sono state sviluppate diverse tecniche, dalla costruzione dell'AttackModel alla creazione del Grafo. È importante notare che queste tecniche non devono essere considerate singolarmente, piuttosto coesistono all'interno della stessa idea, mirando a generare un prodotto finale che fornisca gli strumenti necessari per un'analisi sia manuale che automatica. In questo capitolo viene mostrato il processo completo, dunque l'insieme di tutte le tecniche necessarie al fine di produrre i dati finali.

E' possibile analizzare il writuep che verrà utilizzato per l'esempio nell'Appendice B.1.

5.1 AttackModel

Generiamo dunque l'AttackModel ed osserviamone la prima parte:

```
{
  "AttackModel": {
    "AttackerCapabilities": [
      "Knowledge of PHP language and common web vulnerabilities",
      "Ability to analyze and understand source code",
      "Skills in scripting (e.g., Python)"
    ],
    "Target": {
      "Type": "Web application",
```

```
"Vulnerabilities": [
  "Directory listing enabled on the web server",
  "Insecure cookie handling",
  "Type juggling vulnerability in PHP"
],
"SecurityMeasures": [
  "Use of MD5 hashing for password comparison",
  "Attempt to obfuscate the MD5 hash through bitwise AND
operation"
]
...
```

Questa è focalizzata sul definire le informazioni necessarie ad una prima lettura dell'attacco. Ci informa sulla tipologia di Target e Vulnerabilità che esso possiede, sulle Misure di Sicurezza che implementa e sulle Conoscenze necessarie all'attaccante per eseguire l'attacco.

La seconda parte dell'AttackModel si concentra sulla divisione in step dell'attacco (non guidata):

```
"Steps": [
{
  "StepNumber": "1",
  "Description": "Identify the directory listing on the target web
server",
  "StepDetails": {
    "Objective": "Discover the files present on the server and
potentially access sensitive information.",
    "Action": "Exploit directory listing to find relevant files.",
    "Result": "Identification of 'login.php' and 'login.php.bak' files."
  }
},
{
  "StepNumber": "2",
  "Description": "Analyze the 'login.php.bak' source code",
  "StepDetails": {
    "Objective": "Understand the authentication mechanism and potential
vulnerabilities.",
    "Action": "Review PHP source code for 'login.php.bak'.",
    "Result": "Identification of the 'otadmin' cookie requirement and
potential type juggling vulnerability."
  }
},
...
]
```

Come si può osservare, le informazioni fornite sono estremamente dettagliate. La potenza di questo modello non risiede semplicemente nell'enumerare le azioni compiute, ma nell'analizzare le ragioni sottostanti a tali comportamenti. Il modello è progettato

per includere anche azioni che conducono a esiti negativi o divergenti dalle aspettative. Quest'aspetto assume un'importanza cruciale, poiché consente di comprendere le intenzioni dell'attaccante, nonostante l'esito sfavorevole dell'operazione.

5.2 Summary e StepModel

Il passo successivo è quello di generare la lista di step:

- ```
- The website has directory listing active, connecting to it two files
 are listed: login.php and login.php.bak.
- The login.php.bak file contains the PHP source code.
- Analyzing the PHP source code, it is revealed that a cookie named
 otadmin must be passed, and its format must follow the regex: /^{"
 hash": [0-9A-Z\\"]+}$/.
- A correct value for the otadmin cookie is identified: {"hash": "T35T
 "}.
- If the passed value for the otadmin cookie is different from the MD5
 hash of the $cfg_pass variable, a hint is given.
- The hint is obtained by a PHP snippet that performs an AND operation
 between the ASCII value of each character in the MD5 hash and the
 constant 0xC0 (i.e., 11000000).
- The script identifies that the first three characters of the hash are
 0, indicating that the original characters were numbers.
- The potential vulnerability is located in a type juggling in the
 comparison: $session_data['hash'] != strtoupper(MD5($cfg_pass)).
- To exploit the vulnerability, the attacker must guess the numbers at
 the beginning of the MD5 string.
- A Python script (my-admin-panel.py) is provided to automate the
 process of checking different cookie values and finding the correct
 one.
- With the cookie otadmin={"hash": 389}, the flag is obtained: - p4{
 wtf_php_comparisons_how_do_they_work}.
```

Listing 5.1: Steps.

Gli step andranno dunque inseriti all'interno dello stepModel e successivamente scomposti in substeps. Per semplicità viene riportata semplicemente la lista dei substeps:

- ```
- Connect to the website with active directory listing.
- Observe the presence of two files: login.php and login.php.bak.
- Access the content of the login.php.bak file.
- Identify and analyze the PHP source code within login.php.bak.
- Examine the PHP source code for the specified cookie requirement.
- Identify the regex pattern /^{"hash\\": [0-9A-Z\\"]+}$ for the otadmin
  cookie.
- Determine a valid value for the otadmin cookie: {"hash\\": \\\"T35T\\\"}.
- Check if the provided otadmin cookie value differs from the MD5 hash
  of $cfg_pass.
```

```
- Provide a hint if the values do not match.
- Execute a PHP snippet to perform AND operation on MD5 hash characters
  and 0xC0.
- Analyze the script output to determine the characteristics of the
  first three characters of the hash.
- Identify the potential vulnerability in the type juggling comparison.
- Examine the comparison: $session_data['hash'] != strtoupper(MD5(
  $cfg_pass)).
- Understand the requirement for exploiting the vulnerability: guessing
  the initial numbers of the MD5 string.
- Review the provided Python script: my-admin-panel.py.
- Understand the purpose of the script: automating the search for the
  correct otadmin cookie value.
- Utilize the identified correct otadmin cookie value: {"hash": 389}.
- Retrieve the flag: p4{wtf_php_comparisons_how_do_they_work}.
```

Listing 5.2: Substeps.

Come possiamo notare in questo caso è stato utilizzato un prompt che permette a GPT di riformulare i periodi. Con la scomposizione in sottosteps, si è passati da una rappresentazione ad 11 step ad una con 18 sottostep.

5.3 Etichettatura dello StepModel

Lo StepModel suddiviso in Sottosteps ed etichettato con la tassonomia è osservabile nell'Appendice B.2.

5.4 Estrazione dei Path Nascosti

Di seguito si ha l'estrazione dei path nascosti, compresi di una breve spiegazione, eseguita con gpt-4-turbo:

- **{1.1, 2.1}** - The connection to the website with active directory listing (Substep 1.1) enables the attacker to access the content of the login.php.bak file (Substep 2.1) by observing the directory contents and structure.
- **{1.2, 2.2}** - Observing the presence of the files login.php and login.php.bak (Substep 1.2) leads to the identification and analysis of the PHP source code within login.php.bak (Substep 2.2), as the attacker would target the backup file for potential vulnerabilities or sensitive information.
- **{2.1, 3.1}** - Accessing the content of login.php.bak (Substep 2.1) precedes examining the PHP source code for specific cookie requirements (Substep 3.1), as understanding the file's content is necessary to identify its code structure and functionality.

- **{2.2, 3.2}** - Identifying and analyzing the PHP source code within login.php.bak (Substep 2.2) is a precursor to identifying the regex pattern for the otadmin cookie (Substep 3.2), as the source code would contain the patterns used for cookie validation.
- **{3.1, 4.1}** - Examining the PHP source code for the specified cookie requirement (Substep 3.1) leads to determining a valid value for the otadmin cookie (Substep 4.1), which is essential for bypassing authentication mechanisms.
- **{3.2, 5.1}** - Identifying the regex pattern for the otadmin cookie (Substep 3.2) is related to checking if the provided otadmin cookie value differs from the MD5 hash of `cfg_pass` (Substep 5.1), as the regex pattern can provide insights into the expected format of the hash used in the cookie.
- **{4.1, 5.2}** - Determining a valid value for the otadmin cookie (Substep 4.1) would logically lead to providing a hint if the values do not match (Substep 5.2), as the hint may guide the attacker towards the correct cookie value if the initial guess was wrong.
- **{9.1, 10.2}** - Understanding the requirement for exploiting the vulnerability (Substep 9.1) is related to understanding the purpose of the Python script for automating the search for the correct admin cookie value (Substep 10.2), as exploiting the vulnerability would require automation to efficiently guess the correct values.
- **{10.1, 11.1}** - Reviewing the provided Python script (Substep 10.1) is a necessary step before utilizing the identified correct otadmin cookie value (Substep 11.1), as the script would be used to perform the necessary actions to exploit the vulnerability.

5.5 AttackModeler

Lo StepModel etichettato e il file contenente i path nascosti, costituiscono l'input per il framework che produrrà:

- **node_model.pkl** NodeModel in formato binario
- **node_model.json** NodeModel in formato json
- **graph.png** Grafo
- **report.txt** Report
- **rankings.txt** Classifiche

5.5.1 Grafo, report e classifiche

Osserviamo in Figura 5.1 il grafo dell'attacco prodotto dal framework:

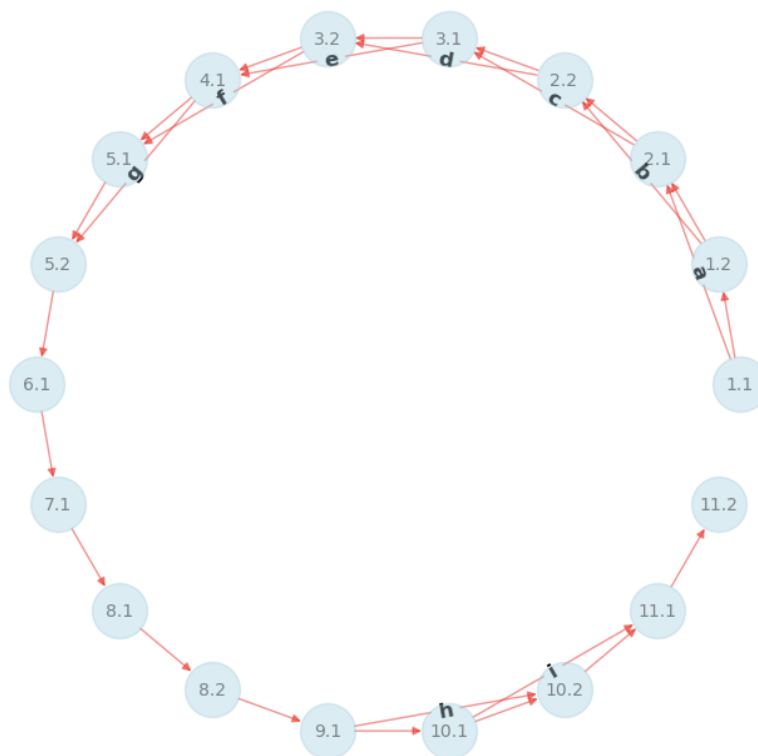


Figura 5.1: Grafo dell'attacco.

Dal grafo si evince che tutti i percorsi nascosti si verificano tra nodi distanti 1. Nelle linee guida per l'estrazione di tali connessioni, è specificato che è possibile collegare esclusivamente nodi non adiacenti, ovvero a distanza ≥ 1 . Si può ipotizzare che questi percorsi “nascosti” siano il risultato della suddivisione in sottosteps, poiché, suddividendo le azioni in passi più piccoli, rimane comunque evidente una forte dipendenza tra di essi. Ciò potrebbe indicare che l'attacco non presenta effettivamente dei veri e propri percorsi nascosti, bensì una dipendenza sequenziale tra i nodi.

Il report generato è osservabile all'interno dell'Appendice B.3.

Date le considerazioni fatte sul grafo, non è necessario un'approfondito esame del report, poiché esso trova il suo impiego nel comprendere meglio i percorsi nascosti.

Infine è possibile esaminare le classifiche per osservare la distribuzione delle tecniche nell'attacco:

```
### Technique Ranking ###
* TECHNIQUE RANKING:
1. Token and Cookie Management: 3 occurrences
2. Code Review and Analysis: 2 occurrences
3. Data Encoding/Decoding: 2 occurrences
4. Webpage Access: 1 occurrences
5. Webpage Elements Interaction and Testing: 1 occurrences
6. File and Directory Discovery: 1 occurrences
7. Gameplay Strategy Observation and Understanding: 1 occurrences
8. Cryptographic Technique Application: 1 occurrences
9. Vulnerability Identification and Analysis: 1 occurrences
10. Cryptographic Algorithm Identification and Analysis: 1 occurrences
11. Cryptographic Exploitation Techniques: 1 occurrences
12. Tool Configuration and Setup: 1 occurrences
13. Script Development: 1 occurrences
14. Flag Data Leakage and Extraction: 1 occurrences

### Technique-Digital_Artefact Pair Ranking ###
* TECHNIQUE-DIGITAL_ARTEFACT Pair Ranking:
1.Token and Cookie Management - cookie: 3 occurrences
2.Code Review and Analysis - code: 2 occurrences
3.Data Encoding/Decoding - encoded_data: 2 occurrences
4.Token and Cookie Management - hash_value: 2 occurrences
5.Webpage Elements Interaction and Testing - file: 1 occurrences
6.File and Directory Discovery - file: 1 occurrences
7.Code Review and Analysis - cookie: 1 occurrences
8.Data Encoding/Decoding - cookie: 1 occurrences
9.Cryptographic Technique Application - hash_value: 1 occurrences
10.Data Encoding/Decoding - hash_value: 1 occurrences
11.Vulnerability Identification and Analysis - vulnerability: 1
occurrences
12.Cryptographic Algorithm Identification and Analysis - hash_value: 1
occurrences
13.Cryptographic Exploitation Techniques - vulnerability: 1 occurrences
14.Cryptographic Exploitation Techniques - hash_value: 1 occurrences
15.Tool Configuration and Setup - code: 1 occurrences
16.Tool Configuration and Setup - tool: 1 occurrences
17.Script Development - cookie: 1 occurrences
18.Flag Data Leakage and Extraction - flag: 1 occurrences
```

Listing 5.3: Classifiche.

5.6 Rappresentazione completa

In questo capitolo è stato sviluppato un esempio concreto, partendo da un writeup, producendo tutti i file necessari per un'analisi completa dell'attacco. Vediamo dunque in Figura 5.2 una panoramica di tutti i file prodotti:

```
[~/Desktop/Tesi/cap8 > ls -l
0 - writeup.md
1 - AttackModel.json
10 - report.txt
11- rankings.txt
2 - summary.txt
3 - StepModel1.json
4 - StepModel2.json
5 - hidden_paths.txt
7 - node_model.pkl
8 - node_model.json
9 - graph.png
```

Figura 5.2: Rappresentazione completa dell'attacco.

5.7 Validazione empirica

La prima fase di validazione empirica si concentra sulla coerenza dei dati raccolti attraverso l'applicazione della metodologia proposta. L'analisi dei risultati ha rivelato che i dati estratti dai writeup tramite l'approccio di modellazione non presentano incoerenze significative. Ciò suggerisce che la metodologia sviluppata è efficace nel rappresentare accuratamente l'attività degli attaccanti senza compromettere l'integrità delle informazioni.

La valutazione si estende inoltre alla fiducia nel processo di modellazione, dall'elaborazione iniziale dell'AttackModel fino alla generazione del grafo. Gli strumenti sviluppati, dimostrano coerenza e affidabilità nella rappresentazione degli attacchi. La corretta mappatura delle informazioni all'interno dei modelli, insieme alla chiara visualizzazione grafica, suggerisce che il processo di modellazione segue un approccio coerente e riproducibile.

Poiché questo lavoro si inserisce in un progetto più ampio, la validazione completa dei risultati è rimandata ad ulteriori ricerche e applicazioni pratiche. Tuttavia, la mancanza di incoerenze nei dati suggerisce una base solida per lo sviluppo futuro.

Capitolo 6

Conclusioni e Prospettive di Sviluppo

Questa tesi ha proposto un approccio innovativo alla modellazione del comportamento degli attaccanti mediante l'impiego del modello di linguaggio avanzato GPT di OpenAI. L'applicazione della metodologia sviluppata ha dimostrato che è possibile generare con l'utilizzo di modelli IA avanzati, una rappresentazione completa degli attacchi utilizzando dati non strutturati.

I dati generati da questa ricerca trovano applicazioni in vari contesti. Le prospettive di sviluppo sono molteplici e possono contribuire al progresso in diversi settori. Di seguito vengono illustrate alcune possibili direzioni:

1. **Tool per il riconoscimento di pattern:** i dati prodotti da questo lavoro possono essere utilizzati per lo sviluppo di strumenti avanzati per il riconoscimento di pattern. Questi strumenti potrebbero essere impiegati per facilitare la rilevazione di comportamenti sospetti o attività malevole.
2. **Strumenti educativi:** i risultati ottenuti possono essere incorporati in strumenti educativi finalizzati alla formazione nel campo della sicurezza. Questi strumenti potrebbero offrire simulazioni realistiche di scenari di attacco e difesa, consentendo agli utenti di acquisire competenze pratiche nel gestire situazioni più o meno complesse.
3. **Approfondire la ricerca di Hidden path:** la ricerca di "Hidden path" potrebbe essere ulteriormente sviluppata per identificare correlazioni più complesse e perfezionare l'estrazione di informazioni tra substep non direttamente adiacenti. Questo potrebbe portare a una comprensione più approfondita delle connessioni tra diverse azioni.

4. **Sviluppo di metriche di sicurezza:** utilizzando i dati e le informazioni ottenute, potrebbero essere sviluppate metriche di sicurezza per valutare la robustezza di un sistema o di un'applicazione. Queste metriche potrebbero essere utilizzate per monitorare la sicurezza nel tempo e identificare potenziali aree di miglioramento.

In sintesi, questa ricerca offre un contributo significativo allo studio del comportamento degli attaccanti, presentando un approccio pratico e suggerendo possibili applicazioni in diversi contesti, sottolineando l'importanza delle metodologie e degli strumenti proposti.

Anticipando le prossime fasi della ricerca, si presentano le seguenti direzioni per gli sviluppi futuri:

- **Raffinamento della Metodologia e dei Modelli:** Esplorare la possibilità di sviluppare modelli più avanzati per rappresentare dettagli specifici degli attacchi.
- **Approfondimento della ricerca dei Path Nascosti:** la ricerca di "Hidden Paths" potrebbe essere estesa per identificare correlazioni più complesse e migliorare l'estrazione di informazioni tra passaggi non direttamente adiacenti.
- **Espansione del Framework AttackModeler:** aggiungere nuovi moduli e funzionalità al framework AttackModeler per aumentarne la versatilità e la capacità di adattamento a scenari di attacco più complessi. Esplorare l'integrazione con altri strumenti e framework di analisi.
- **Applicazioni Pratiche del Modello:** testare l'applicabilità del modello sviluppato in contesti reali, ad esempio utilizzando report di attacchi reali.

Appendice A

Writeup, Tassonomia e Tecniche

In questo capitolo è possibile osservare:

- Esempio di writeup
- Tassonomia Completa
- Tecniche

A.1 Esempio di writeup

```
# File Library
Author: [roerohan](https://github.com/roerohan)
## Description
Express query parsing vulnerability.
## Requirements
- Express.js
- qs package used by Express.js
## Sources
- [server.js](./server.js)
```

```

- Deployed website using server.js

'''
This is my file library. I don't have a lot of files, but I hope you
  like the ones I have!
'''

## Exploit
This exploit is owing to the 'qs' package used by 'express' to parse '
  req.query'. You can send an array instead of a string through the
  GET params in the following manner:
'''ja
/route?words[]=hello&words[]=world
'''

This makes the 'req.names' in the backend an array: '['hello', 'world
  ']'

[javascript code]
The '/getFile' route is meant to get you a file from the server. But
  there are certain restrictions. First, you can't have 's or '/'s
  in your file. This prevents path traversal to an extent.
<br />
Then, it checks that the 'file' parameter has a length of less than 5
  characters. If not, it takes just the first 5 characters. Then, it
  checks if the file extension is 'js', 'ts', 'cpp', or 'c', if yes,
  it allows you to read that file. If all these are satisfied, the
  path is resolved using 'path.resolve()'.
<br />
In the 'a.cpp' file, it says 'system("cat flag.txt")', indicating that
  the flag is present in the same directory in the 'flag.txt'. We can
  pass in an array which passes the first 2 checks. Then it has to
  check if the '.slice(file.indexOf('.') + 1)', so the last 2 elements
  have to be '['.', 'js']'. So, we can try passing '['flag.txt', '.',
  'js']', which makes the path '/home/user/flag.txt,..,js', which is
  invalid. However, we can make use of the 'path.resolve()', and the
  fact that only the first 5 elements are used. If the array becomes
  '['a', 'b', 'c', 'd', '/../flag.txt', '.', 'js']' ('js' because 'txt
  ' is not allowed). So, this passed the file check, because '.slice()
  ' returns '['js']' and '['js'] == 'js' is true (not used '===').
  Now, to get rid of this, we add 4 random elements before the element
  having 'flag.txt'.
<br />
Now, the string upon concatenation will give '${pwd}/a,b,c,d/../flag.
  txt' (now you see why we added '/../'), so that it resolves to '${
  pwd}/flag.txt' because of 'flag.txt'. The payload, therefore, is:
'''
/getFile?file[]=a&file[]=b&file[]=c&file[]=d&file[]=../flag.txt&file
  []=.&file[]=js
'''

The flag is:
'''
csictf{5h0uld_5tringify_th3_p4r4ms}
'''

```

Listing A.1: Esempio di writeup.

A.2 Tassonomia Completa

1. Web Interaction and Navigation
 - 1.1 Webpage Interaction
 - 1.1.1 Webpage Access
 - 1.1.2 Webpage Elements Interaction and Testing
 - 1.1.3 Webpage Content Observation and Understanding
 - 1.2 Browser and Client-Side Interaction
 - 1.2.1 Browser Access
 - 1.2.2 Client-Side Code Manipulation and Testing
 - 1.2.3 Browser Extension Utilization
 - 1.3 URL Interaction
 - 1.3.1 URL Access
 - 1.3.2 URL Manipulation and Testing
 - 1.3.3 URL Navigation Observation and Understanding
2. Network and Communication Analysis
 - 2.1 Network Analysis
 - 2.1.1 Network Traffic Monitoring and Analysis
 - 2.1.2 Protocol Examination
 - 2.1.3 Service and Infrastructure Analysis
 - 2.2 HTTP Communication Management
 - 2.2.1 HTTP Request-Response Crafting and Transmission
 - 2.2.2 HTTP Request-Response Inspection
 - 2.2.3 HTTP Request-Response Manipulation
 - 2.2.4 HTTP Flow Observation and Understanding
3. System Profiling and Analysis
 - 3.1 System Structure Analysis
 - 3.1.1 File and Directory Discovery
 - 3.1.2 Endpoint and Subdomain Discovery
 - 3.1.3 Cloud Infrastructure Discovery
 - 3.1.4 System Architecture Observation and Understanding
 - 3.2 System Composition Analysis
 - 3.2.1 System Fingerprinting and Identification
 - 3.2.2 System Configuration Analysis
 - 3.2.3 Service and API Profiling
4. Authentication and Authorization Management
 - 4.1 Authentication Process Handling
 - 4.1.1 Authentication Process Interaction and Testing
 - 4.1.2 Authentication Process Observation and Understanding
 - 4.1.3 Multi-Factor Authentication Analysis
 - 4.2 Access Control Security
 - 4.2.1 Access Control Mechanism Analysis
 - 4.2.2 Access Control Bypass
 - 4.3 Credential Management
 - 4.3.1 Token and Cookie Management
 - 4.3.2 Credential Vulnerability Assessment
5. Data Management
 - 5.1 Data Leakage and Extraction

- 5.1.1 Information Leakage and Extraction
- 5.1.2 Sensitive Data Leakage and Extraction
- 5.1.3 Flag Data Leakage and Extraction
- 5.2 Data Analysis
 - 5.2.1 Information Analysis
 - 5.2.2 Sensitive Data Analysis
 - 5.2.3 Flag Data Analysis
- 6. Cryptography and Encoding Management
 - 6.1 Cryptographic Data Transformation
 - 6.1.1 Data Encoding/Decoding
 - 6.1.2 Data Encryption/Decryption
 - 6.1.3 Cryptographic Technique Application
 - 6.2 Cryptographic Analysis and Exploitation
 - 6.2.1 Cryptographic Protocol Analysis
 - 6.2.2 Cryptographic Algorithm Identification and Analysis
 - 6.2.3 Cryptographic Key Identification and Analysis
 - 6.2.4 Cryptographic Exploitation Techniques
- 7. Vulnerability and Exploitation Management
 - 7.1 Vulnerability Assessment
 - 7.1.1 Security Policy Identification and Analysis
 - 7.1.2 Security Mechanism Identification and Analysis
 - 7.1.3 Vulnerability Identification and Analysis
 - 7.2 Exploitation Techniques and Payload Management
 - 7.2.1 Exploitation Payload Crafting and Distribution
 - 7.2.2 Exploitation Techniques Execution
 - 7.2.3 Security Policy Bypass
 - 7.2.4 Security Mechanism Bypass
 - 7.2.5 Advanced Post-Exploitation Techniques
- 8. Database and File System Interaction
 - 8.1 Database Interaction and Exploration
 - 8.1.1 Database Access and Inspection
 - 8.1.2 Database Exploration and Querying
 - 8.1.3 Database Structure Observation and Understanding
 - 8.2 File System Interaction and Exploration
 - 8.2.1 File System Access and Inspection
 - 8.2.2 File System Navigation and Management
 - 8.2.3 File Transferring
- 9. Tool Utilization and Scripting
 - 9.1 Tool Management
 - 9.1.1 Tool Configuration and Setup
 - 9.1.2 Tool Utilization
 - 9.2 Scripting and Automation Management
 - 9.2.1 Script Development
 - 9.2.2 Script Execution
 - 9.2.3 Command-Line Operations
- 10. Knowledge Management and Learning
 - 10.1 Online Research and Education
 - 10.1.1 Official Documentation Review

```
10.1.2 Community-Sourced Security Learning

11. Challenge and Strategy Management
  11.1 Challenge Management
    11.1.1 Challenge Environment Setup
    11.1.2 Challenge Details Observation and Understanding
  11.2 Gameplay Strategy Management
    11.2.1 Gameplay Strategy Formulation
    11.2.2 Gameplay Strategy Observation and Understanding
    11.2.3 Gameplay Strategy Documentation

12. Code Analysis and Debugging
  12.1 Debugging Techniques
    12.1.1 Code Review and Analysis
    12.1.2 Error Identification and Analysis
    12.1.3 Code Behavior and Execution Review
```

Listing A.2: Tassonomia Completa.

A.3 Tecniche

```
- Webpage Access
- Webpage Elements Interaction and Testing
- Webpage Content Observation and Understanding

- Browser Access
- Client-Side Code Manipulation and Testing
- Browser Extension Utilization

- URL Access
- URL Manipulation and Testing
- URL Navigation Observation and Understanding

- Network Traffic Monitoring and Analysis
- Protocol Examination
- Service and Infrastructure Analysis

- HTTP Request-Response Crafting and Transmission
- HTTP Request-Response Inspection
- HTTP Request-Response Manipulation
- HTTP Flow Observation and Understanding

- File and Directory Discovery
- Endpoint and Subdomain Discovery
- Cloud Infrastructure Discovery
- System Architecture Observation and Understanding

- System Fingerprinting and Identification
- System Configuration Analysis
- Service and API Profiling
```

- Authentication Process Interaction and Testing
- Authentication Process Observation and Understanding
- Multi-Factor Authentication Analysis

- Access Control Mechanism Analysis
- Access Control Bypass

- Token and Cookie Management
- Credential Vulnerability Assessment

- Information Leakage and Extraction
- Sensitive Data Leakage and Extraction
- Flag Data Leakage and Extraction

- Information Analysis
- Sensitive Data Analysis
- Flag Data Analysis

- Data Encoding/Decoding
- Data Encryption/Decryption
- Cryptographic Technique Application

- Cryptographic Protocol Analysis
- Cryptographic Algorithm Identification and Analysis
- Cryptographic Key Identification and Analysis
- Cryptographic Exploitation Techniques

- Security Policy Identification and Analysis
- Security Mechanism Identification and Analysis
- Vulnerability Identification and Analysis

- Exploitation Payload Crafting and Distribution
- Exploitation Techniques Execution
- Security Policy Bypass
- Security Mechanism Bypass
- Advanced Post-Exploitation Techniques

- Database Access and Inspection
- Database Exploration and Querying
- Database Structure Observation and Understanding

- File System Access and Inspection
- File System Navigation and Management
- File Transferring

- Tool Configuration and Setup
- Tool Utilization

- Script Development
- Script Execution
- Command-Line Operations

- Official Documentation Review
- Community-Sourced Security Learning

- Challenge Environment Setup
- Challenge Details Observation and Understanding

- Gameplay Strategy Formulation
- Gameplay Strategy Observation and Understanding
- Gameplay Strategy Documentation

- Code Review and Analysis
- Error Identification and Analysis
- Code Behavior and Execution Review

Listing A.3: Lista delle tecniche.

Appendice B

Rappresentazione completa di un attacco - Dati

In questo capitolo è possibile osservare alcuni dati prodotti come supporto al Capitolo 5:

- Writeup
- StepModel etichettato con la Tassonomia
- Report

B.1 Writeup

```
# CONFidence CTF Teaser 2019      My admin panel

* **Category:** web, warmup
* **Points:** 51

## Challenge

> I think I've found something interesting, but I'm not really a PHP
  expert. Do you think it's exploitable?
>
> https://gameserver.zajebistyc.tf/admin/

## Solution

The website has directory listing active, connecting to it two files are
listed:
```

```
* 'login.php'
* 'login.php.bak'

The first page prints the message: 'Not authenticated.'.

The [second file](login.php.bak) contains the PHP source code.

'''php
<?php

include '../func.php';
include '../config.php';

if (!$_COOKIE['otadmin']) {
    exit("Not authenticated.\n");
}

if (!preg_match('/^{"hash": [0-9A-Z\\"]+}$/', $_COOKIE['otadmin'])) {
    echo "COOKIE TAMPERING xD IM A SECURITY EXPERT\n";
    exit();
}

$session_data = json_decode($_COOKIE['otadmin'], true);

if ($session_data === NULL) { echo "COOKIE TAMPERING xD IM A SECURITY
    EXPERT\n"; exit(); }

if ($session_data['hash'] != strtoupper(MD5($cfg_pass))) {
    echo("I CAN EVEN GIVE YOU A HINT XD \n");

    for ($i = 0; i < strlen(MD5('xDdddddd')); i++) {
        echo(ord(MD5($cfg_pass)[$i]) & 0xC0);
    }

    exit("\n");
}

display_admin();
'''

Analyzing it, you can discover that a cookie named 'otadmin' must be
passed and its format must follow the following regex: '/^{"hash":
[0-9A-Z\\"]+}$/'.
```

A correct value for that cookie can be the following: '{"hash": "T35T
"}'.

If the passed value for that cookie is different from the MD5 hash of
the '\$cfg_pass' variable, a hint is given. You can use the correct
value crafted before to print it.

```
'''
```

```
I CAN EVEN GIVE YOU A HINT XD
0006464640640064000646464640006400640640646400
'''
```

The hint is obtained by the following snippet.

```
'''php
    for ($i = 0; i < strlen(MD5('xDdddddd')); i++) {
        echo(ord(MD5($cfg_pass)[$i]) & 0xC0);
    }
'''
```

Where 'strlen(MD5('xDdddddd'))' is equals to 32 (i.e. the length of MD5 hashes is 32).

In that snippet, an AND operation is performed between the ASCII value of each char and the constant '0xC0' (i.e. '11000000'). Considering that the first three chars of the hint are 0, it means that the first original chars of the hash were numbers. That is due to the fact in the ASCII representation of numbers, the first most significant bits are '00' and the AND operation for that bits is performed with '11'.

So, the problem could be located in a *type juggling* in this comparison :

```
'''php
$session_data['hash'] != strtoupper(MD5($cfg_pass))
'''
```

To exploit it, you have to guess the numbers at the beginning of the MD5 string.

A [Python script](my-admin-panel.py) can be written to exploit this task .

```
'''python
import time
import random
import os
import urllib2

target_url = "https://gameserver.zajebistyc.tf/admin/login.php"
cookie_name = "otadmin"
cookie_value = "{{\"hash\": {}}}"

# Check cookie method.
def check_cookie(value_to_check):

    # Trying cookie.
    cookie = "{}={}".format(cookie_name, cookie_value.format(
        value_to_check))
    print "[*] Trying cookie {}".format(cookie)
```

```
req = urllib2.Request(target_url)
req.add_header("Cookie", cookie)
req.add_header("User-Agent", "Mozilla/5.0 (Windows; U; MSIE 9.0;
Windows NT 9.0; en-US);")
page = urllib2.urlopen(req)
content = page.read()

# Flag found.
if "p4{" in content:
    print content
    return True
else:
    return False

# Main method.
for i in range(1, 1000):

    try:

        if check_cookie(i):
            break

    except urllib2.HTTPError as err:
        print "[*] Response > {}".format(err.code)
        break

'''
With the cookie 'otadmin={"hash": 389}' you will discover the flag.
'''
p4{wtf_php_comparisons_how_do_they_work}
'''
```

Listing B.1: Writeup.

B.2 StepModel etichettato con la Tassonomia

```
{
  "StepModel": {
    "Steps": [
      {
        "StepNumber": 1,
        "StepString": "The website has directory listing active,
connecting to it two files are listed: login.php and login.php.bak
.",
        "Substeps": [
          {
            "SubstepNumber": 1.1,
```



```

        "SubstepString": "Connect to the website with
active directory listing.",
        "Category": "Web Interaction and Navigation",
        "Subcategory": "Webpage Interaction",
        "Techniques": "Webpage Access"
    },
    {
        "SubstepNumber": 1.2,
        "SubstepString": "Observe the presence of two
files: login.php and login.php.bak.",
        "Category": "Web Interaction and Navigation",
        "Subcategory": "Webpage Interaction",
        "Techniques": "Webpage Elements Interaction and
Testing"
    }
]
},
{
    "StepNumber": 2,
    "StepString": "The login.php.bak file contains the PHP
source code.",
    "Substeps": [
        {
            "SubstepNumber": 2.1,
            "SubstepString": "Access the content of the
login.php.bak file.",
            "Category": "System Profiling and Analysis",
            "Subcategory": "System Structure Analysis",
            "Techniques": "File and Directory Discovery"
        },
        {
            "SubstepNumber": 2.2,
            "SubstepString": "Identify and analyze the PHP
source code within login.php.bak.",
            "Category": "Code Analysis and Debugging",
            "Subcategory": "Debugging Techniques",
            "Techniques": "Code Review and Analysis"
        }
    ]
},
{
    "StepNumber": 3,
    "StepString": "Analyzing the PHP source code, it is
revealed that a cookie named otadmin must be passed, and its format
must follow the regex: /~{\\"hash\\": [0-9A-Z\\"]+}$/.",
    "Substeps": [
        {
            "SubstepNumber": 3.1,
            "SubstepString": "Examine the PHP source code
for the specified cookie requirement.",
            "Category": "Code Analysis and Debugging",
            "Subcategory": "Debugging Techniques",
            "Techniques": "Code Review and Analysis"
        }
    ]
}

```

```

        },
        {
            "SubstepNumber": 3.2,
            "SubstepString": "Identify the regex pattern
/~{\"hash\": [0-9A-Z\\\"]+}$/ for the otadmin cookie.",
            "Category": "Cryptography and Encoding
Management",
            "Subcategory": "Cryptographic Data
Transformation",
            "Techniques": "Data Encoding/Decoding"
        }
    ]
},
{
    "StepNumber": 4,
    "StepString": "A correct value for the otadmin cookie is
identified: {\"hash\": \"T35T\"}.",
    "Substeps": [
        {
            "SubstepNumber": 4.1,
            "SubstepString": "Determine a valid value for
the otadmin cookie: {\"hash\": \"T35T\"}.",
            "Category": "Authentication and Authorization
Management",
            "Subcategory": "Credential Management",
            "Techniques": "Token and Cookie Management"
        }
    ]
},
{
    "StepNumber": 5,
    "StepString": "If the passed value for the otadmin
cookie is different from the MD5 hash of the $cfg_pass variable, a
hint is given.",
    "Substeps": [
        {
            "SubstepNumber": 5.1,
            "SubstepString": "Check if the provided otadmin
cookie value differs from the MD5 hash of $cfg_pass.",
            "Category": "Authentication and Authorization
Management",
            "Subcategory": "Credential Management",
            "Techniques": "Token and Cookie Management"
        },
        {
            "SubstepNumber": 5.2,
            "SubstepString": "Provide a hint if the values
do not match.",
            "Category": "Challenge and Strategy Management",
            "Subcategory": "Gameplay Strategy Management",
            "Techniques": "Gameplay Strategy Observation and
Understanding"
        }
    ]
}

```

```

    ]
  },
  {
    "StepNumber": 6,
    "StepString": "The hint is obtained by a PHP snippet
that performs an AND operation between the ASCII value of each
character in the MD5 hash and the constant 0xC0 (i.e., 11000000).",
    "Substeps": [
      {
        "SubstepNumber": 6.1,
        "SubstepString": "Execute a PHP snippet to
perform AND operation on MD5 hash characters and 0xC0.",
        "Category": "Cryptography and Encoding
Management",
        "Subcategory": "Cryptographic Data
Transformation",
        "Techniques": "Cryptographic Technique
Application"
      }
    ]
  },
  {
    "StepNumber": 7,
    "StepString": "The script identifies that the first
three characters of the hash are 0, indicating that the original
characters were numbers.",
    "Substeps": [
      {
        "SubstepNumber": 7.1,
        "SubstepString": "Analyze the script output to
determine the characteristics of the first three characters of the
hash.",
        "Category": "Cryptography and Encoding
Management",
        "Subcategory": "Cryptographic Data
Transformation",
        "Techniques": "Data Encoding/Decoding"
      }
    ]
  },
  {
    "StepNumber": 8,
    "StepString": "The potential vulnerability is located in
a type juggling in the comparison: $session_data['hash'] !=
strtoupper(MD5($cfg_pass)).",
    "Substeps": [
      {
        "SubstepNumber": 8.1,
        "SubstepString": "Identify the potential
vulnerability in the type juggling comparison.",
        "Category": "Vulnerability and Exploitation
Management",
        "Subcategory": "Vulnerability Assessment",

```

```

        "Techniques": "Vulnerability Identification and
Analysis"
    },
    {
        "SubstepNumber": 8.2,
        "SubstepString": "Examine the comparison:
$session_data['hash'] != strtoupper(MD5($cfg_pass)).",
        "Category": "Cryptography and Encoding
Management",
        "Subcategory": "Cryptographic Analysis and
Exploitation",
        "Techniques": "Cryptographic Algorithm
Identification and Analysis"
    }
]
},
{
    "StepNumber": 9,
    "StepString": "To exploit the vulnerability, the
attacker must guess the numbers at the beginning of the MD5 string
.",
    "Substeps": [
        {
            "SubstepNumber": 9.1,
            "SubstepString": "Understand the requirement for
exploiting the vulnerability: guessing the initial numbers of the
MD5 string.",
            "Category": "Cryptography and Encoding
Management",
            "Subcategory": "Cryptographic Analysis and
Exploitation",
            "Techniques": "Cryptographic Exploitation
Techniques"
        }
    ]
},
{
    "StepNumber": 10,
    "StepString": "A Python script (my-admin-panel.py) is
provided to automate the process of checking different cookie values
and finding the correct one.",
    "Substeps": [
        {
            "SubstepNumber": 10.1,
            "SubstepString": "Review the provided Python
script: my-admin-panel.py.",
            "Category": "Tool Utilization and Scripting",
            "Subcategory": "Tool Management",
            "Techniques": "Tool Configuration and Setup"
        },
        {
            "SubstepNumber": 10.2,

```

```

        "SubstepString": "Understand the purpose of the
script: automating the search for the correct otadmin cookie value
.",
        "Category": "Tool Utilization and Scripting",
        "Subcategory": "Scripting and Automation
Management",
        "Techniques": "Script Development"
    }
    ]
    },
    {
        "StepNumber": 11,
        "StepString": "With the cookie otadmin={\"hash\": 389},
the flag is obtained: p4{wtf_php_comparisons_how_do_they_work}.",
        "Substeps": [
            {
                "SubstepNumber": 11.1,
                "SubstepString": "Utilize the identified correct
otadmin cookie value: {\"hash\": 389}.",
                "Category": "Authentication and Authorization
Management",
                "Subcategory": "Credential Management",
                "Techniques": "Token and Cookie Management"
            },
            {
                "SubstepNumber": 11.2,
                "SubstepString": "Retrieve the flag: p4{
wtf_php_comparisons_how_do_they_work}.",
                "Category": "Data Management",
                "Subcategory": "Data Leakage and Extraction",
                "Techniques": "Flag Data Leakage and Extraction"
            }
        ]
    }
    ]
}
}
}
}

```

Listing B.2: StepModel etichettato.

B.3 Report

```

-----
Node: 1.1
    * String: Connect to the website with active directory listing.
    * Technique: Webpage Access
    * Digital Artefacts:
      - No digital artefacts
    * Hidden Neighbors:
      - Node: 2.1

```

```

        * String: Access the content of the login.php.
bak file.
        * Technique: File and Directory Discovery
        * Digital Artefacts:
          - file | Attributes: {'filename': 'login
.php.bak', 'extension': '.bak', 'description': ''}
-----
Node: 1.2
  * String: Observe the presence of two files: login.php and login
.php.bak.
  * Technique: Webpage Elements Interaction and Testing
  * Digital Artefacts:
    - file | Attributes: {'filename': 'login', 'extension':
'.php', 'description': ''}
  * Hidden Neighbors:
    - Node: 2.2
      * String: Identify and analyze the PHP source
code within login.php.bak.
      * Technique: Code Review and Analysis
      * Digital Artefacts:
        - code | Attributes: {'language': 'PHP',
'description': '', 'context': 'login.php.bak'}
-----
Node: 2.1
  * String: Access the content of the login.php.bak file.
  * Technique: File and Directory Discovery
  * Digital Artefacts:
    - file | Attributes: {'filename': 'login.php.bak', '
extension': '.bak', 'description': ''}
  * Hidden Neighbors:
    - Node: 3.1
      * String: Examine the PHP source code for the
specified cookie requirement.
      * Technique: Code Review and Analysis
      * Digital Artefacts:
        - cookie | Attributes: {'name': '', '
value': '', 'exipration_time': '', 'description': 'Examine the PHP
source code for the specified cookie requirement.'}
        - code | Attributes: {'language': 'PHP',
'description': '', 'context': 'specified cookie requirement'}
-----
Node: 2.2
  * String: Identify and analyze the PHP source code within login.
.php.bak.
  * Technique: Code Review and Analysis
  * Digital Artefacts:
    - code | Attributes: {'language': 'PHP', 'description':
'', 'context': 'login.php.bak'}
  * Hidden Neighbors:
    - Node: 3.2
      * String: Identify the regex pattern /~{"hash":
[0-9A-Z"]+}$ / for the otadmin cookie.
      * Technique: Data Encoding/Decoding
```

```

        * Digital Artefacts:
          - cookie | Attributes: {'name': '', 'value': '', 'expiration_time': '', 'description': 'Identify the regex pattern />{"hash": [0-9A-Z"]+}$/' for the otadmin cookie.}
          - encoded_data | Attributes: {'data': 'the regex pattern />{"hash": [0-9A-Z"]+}$/', 'encoding': '', 'description': 'Identify the regex pattern />{"hash": [0-9A-Z"]+}$/' for the otadmin cookie.}
-----
Node: 3.1
  * String: Examine the PHP source code for the specified cookie requirement.
  * Technique: Code Review and Analysis
  * Digital Artefacts:
    - cookie | Attributes: {'name': '', 'value': '', 'expiration_time': '', 'description': 'Examine the PHP source code for the specified cookie requirement.}
    - code | Attributes: {'language': 'PHP', 'description': '', 'context': 'specified cookie requirement'}
  * Hidden Neighbors:
    - Node: 4.1
      * String: Determine a valid value for the otadmin cookie: {"hash": "T35T"}.
      * Technique: Token and Cookie Management
      * Digital Artefacts:
        - cookie | Attributes: {'name': 'otadmin', 'value': '', 'expiration_time': '', 'description': ''}
-----
Node: 3.2
  * String: Identify the regex pattern />{"hash": [0-9A-Z"]+}$/' for the otadmin cookie.
  * Technique: Data Encoding/Decoding
  * Digital Artefacts:
    - cookie | Attributes: {'name': '', 'value': '', 'expiration_time': '', 'description': 'Identify the regex pattern />{"hash": [0-9A-Z"]+}$/' for the otadmin cookie.}
    - encoded_data | Attributes: {'data': 'the regex pattern />{"hash": [0-9A-Z"]+}$/', 'encoding': '', 'description': 'Identify the regex pattern />{"hash": [0-9A-Z"]+}$/' for the otadmin cookie.}
  * Hidden Neighbors:
    - Node: 5.1
      * String: Check if the provided otadmin cookie value differs from the MD5 hash of $cfg_pass.
      * Technique: Token and Cookie Management
      * Digital Artefacts:
        - cookie | Attributes: {'name': 'otadmin cookie', 'value': '', 'expiration_time': '', 'description': ''}
        - hash_value | Attributes: {'hash': 'MD5', 'algorithm': 'MD5', 'description': ''}
-----
Node: 4.1

```

```
* String: Determine a valid value for the otadmin cookie: {"hash": "T35T"}.
* Technique: Token and Cookie Management
* Digital Artefacts:
  - cookie | Attributes: {'name': 'otadmin', 'value': '', 'expiration_time': '', 'description': ''}
* Hidden Neighbors:
  - Node: 5.2
    * String: Provide a hint if the values do not match.
    * Technique: Gameplay Strategy Observation and Understanding
    * Digital Artefacts:
      - No digital artefacts
```

Node: 5.1

```
* String: Check if the provided otadmin cookie value differs from the MD5 hash of $cfg_pass.
* Technique: Token and Cookie Management
* Digital Artefacts:
  - cookie | Attributes: {'name': 'otadmin cookie', 'value': '', 'expiration_time': '', 'description': ''}
  - hash_value | Attributes: {'hash': 'MD5', 'algorithm': 'MD5', 'description': ''}
* No Hidden Neighbors
```

Node: 5.2

```
* String: Provide a hint if the values do not match.
* Technique: Gameplay Strategy Observation and Understanding
* Digital Artefacts:
  - No digital artefacts
* No Hidden Neighbors
```

Node: 6.1

```
* String: Execute a PHP snippet to perform AND operation on MD5 hash characters and 0xC0.
* Technique: Cryptographic Technique Application
* Digital Artefacts:
  - hash_value | Attributes: {'hash': 'MD5', 'algorithm': '', 'description': 'Execute a PHP snippet to perform AND operation on MD5 hash characters and 0xC0.'}
* No Hidden Neighbors
```

Node: 7.1

```
* String: Analyze the script output to determine the characteristics of the first three characters of the hash.
* Technique: Data Encoding/Decoding
* Digital Artefacts:
  - encoded_data | Attributes: {'data': '', 'encoding': '', 'description': 'Analyze the script output to determine the characteristics of the first three characters of the hash.'}
```



```
        - hash_value | Attributes: {'hash': '', 'algorithm': '',
'description': 'Analyze the script output to determine the
characteristics of the first three characters of the hash.')}
        * No Hidden Neighbors
-----
Node: 8.1
    * String: Identify the potential vulnerability in the type
juggling comparison.
    * Technique: Vulnerability Identification and Analysis
    * Digital Artefacts:
        - vulnerability | Attributes: {'type': 'type juggling
comparison', 'cve_number': '', 'description': ''}
        * No Hidden Neighbors
-----
Node: 8.2
    * String: Examine the comparison: $session_data['hash'] !=
strtoupper(MD5($cfg_pass)).
    * Technique: Cryptographic Algorithm Identification and Analysis
    * Digital Artefacts:
        - hash_value | Attributes: {'hash': "$session_data['hash
']", 'algorithm': 'MD5', 'description': ''}
        * No Hidden Neighbors
-----
Node: 9.1
    * String: Understand the requirement for exploiting the
vulnerability: guessing the initial numbers of the MD5 string.
    * Technique: Cryptographic Exploitation Techniques
    * Digital Artefacts:
        - vulnerability | Attributes: {'type': '', 'cve_number':
'', 'description': 'guessing the initial numbers of the MD5 string
.'}
        - hash_value | Attributes: {'hash': 'MD5', 'algorithm':
'MD5', 'description': 'guessing the initial numbers of the MD5
string'}
    * Hidden Neighbors:
        - Node: 10.2
            * String: Understand the purpose of the script:
automating the search for the correct otadmin cookie value.
            * Technique: Script Development
            * Digital Artefacts:
                - cookie | Attributes: {'name': '', '
value': '', 'expiration_time': '', 'description': ''}
-----
Node: 10.1
    * String: Review the provided Python script: my-admin-panel.py.
    * Technique: Tool Configuration and Setup
    * Digital Artefacts:
        - code | Attributes: {'language': 'Python', 'description
': '', 'context': ''}
        - tool | Attributes: {'name': 'my-admin-panel.py', '
description': '', 'purpose': ''}
    * Hidden Neighbors:
        - Node: 11.1
```

```

        * String: Utilize the identified correct otadmin
cookie value: {"hash": 389}.
        * Technique: Token and Cookie Management
        * Digital Artefacts:
            - cookie | Attributes: {'name': '', '
value': '389', 'exipration_time': '', 'description': ''}
            - hash_value | Attributes: {'hash':
'389', 'algorithm': '', 'description': ''}
-----
Node: 10.2
    * String: Understand the purpose of the script: automating the
search for the correct otadmin cookie value.
    * Technique: Script Development
    * Digital Artefacts:
        - cookie | Attributes: {'name': '', 'value': '', '
expiration_time': '', 'description': ''}
    * No Hidden Neighbors
-----
Node: 11.1
    * String: Utilize the identified correct otadmin cookie value:
{"hash": 389}.
    * Technique: Token and Cookie Management
    * Digital Artefacts:
        - cookie | Attributes: {'name': '', 'value': '389', '
exipration_time': '', 'description': ''}
        - hash_value | Attributes: {'hash': '389', 'algorithm':
'', 'description': ''}
    * No Hidden Neighbors
-----
Node: 11.2
    * String: Retrieve the flag: p4{
wtf_php_comparisons_how_do_they_work}.
    * Technique: Flag Data Leakage and Extraction
    * Digital Artefacts:
        - flag | Attributes: {'flag': 'p4{
wtf_php_comparisons_how_do_they_work}', 'description': '', 'context
': ''}
    * No Hidden Neighbors

```

Listing B.3: Report.

Bibliografia

- [1] Sergio Caltagirone, Andrew Pendergast e Christopher Betz. «The diamond model of intrusion analysis». In: *Threat Connect* 298.0704 (2013), pp. 1–61.
- [2] Tarun Yadav e Arvind Mallari Rao. «Technical aspects of cyber kill chain». In: *Security in Computing and Communications: Third International Symposium, SSCC 2015, Kochi, India, August 10-13, 2015. Proceedings 3*. Springer. 2015, pp. 438–452.
- [3] Cynthia Phillips e Laura Painton Swiler. «A Graph-Based System for Network-Vulnerability Analysis». In: *Proceedings of the 1998 Workshop on New Security Paradigms*. NSPW '98. Charlottesville, Virginia, USA: Association for Computing Machinery, 1998, pp. 71–79. ISBN: 1581131682. DOI: 10.1145/310889.310919. URL: <https://doi.org/10.1145/310889.310919>.
- [4] Hamad Al-Mohannadi, Qublai Mirza, Anitta Namanya, Irfan Awan, Andrea Cullen e Jules Disso. «Cyber-Attack Modeling Analysis Techniques: An Overview». In: *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. 2016, pp. 69–76. DOI: 10.1109/W-FiCloud.2016.29.
- [5] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever et al. «Improving language understanding by generative pre-training». In: (2018).
- [6] Francesco Lonardo. «Tassonomia dei comportamenti degli attaccanti nelle sfide di Capture The Flag (CTF): un approccio di categorizzazione attraverso l'analisi dei writeups». Tesi Magistrale. Politecnico di Torino, 2023.