Università degli Studi di Milano - Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di Laurea in Informatica

Piattaforma web per la ricerca automatica di vulnerabilità in file binari

Relatore: Prof. Claudio Ferretti

Correlatore: Dott.sa Martina Saletta

Tesi di Laurea di:
Andrea Consonni
Matricola 900116

Anno Accademico 2024-2025

Vorrei esprimere i miei più sentiti ringraziamenti al ***prof. Claudio Ferretti****, alla* ***Dott.sa Martina Saletta*** *e al* ***prof. Giovanni Denaro*** *per il costante supporto, la disponibilità e i preziosi consigli offerti durante tutto il percorso di tesi.*

Un ringraziamento speciale va alla ***mia famiglia****, che con il suo supporto incondizionato mi ha permesso di affrontare con serenità questo percorso di studi.*

Sono profondamente grato anche ***a tutti i miei amici****, il quale costante sostegno mi ha accompagnato e incoraggiato lungo tutto il percorso universitario*

Abstract
Il compito di analisi

Indice

Capitolo 1

Introduzione

In un mondo sempre più digitalizzato ed interconnesso, la tematica della sicurezza informatica ha assunto sempre più un'importanza chiave in ogni processo di sviluppo software. La potenziale presenza e lo sfruttamento di una vulnerabilità all'interno di un'applicazione da parte di un'attaccante potrebbe avere conseguenze disastrose: dall'escalation di privilegi all'accesso non autorizzato a dati sensibili, compromettendo quindi l'integrità e la confidenzialità di quest'ultimi. È quindi fondamentale che i potenziali rischi per la sicurezza siano considerati sin dai primi momenti del processo di sviluppo. Effettuare un'analisi di sicurezza approfondita risulta quindi fondamentale nell'evitare che potenziali vulnerabilità persistano all'interno del programma; tuttavia, questo processo si complica notevolmente quando l'analista è in **solo possesso del file binario** e non ha accesso al codice sorgente dell'applicazione. In questo caso, l'analista non solo dovrà avere ampie competenze specifiche in ambito di reverse engineering, ma dovrà essere in grado di utilizzare tool e framework che potrebbero avere un'interfaccia a primo impatto ostica, richiedere conoscenze di scripting o di tematiche di sicurezza avanzate oppure avere un costo elevato, il quale potrebbe non rientrare nei limiti di budget prefissati. Questa tesi propone l'implementazione di una piattaforma web per l'analisi di file binari denominata **Binoculars**; la quale si prefigge l'obbiettivo di semplificare il processo di analisi di sicurezza su file binari ELF compilati per architettura x86 tramite un'interfaccia semplice ed intuitiva, permettendo anche ad analisti con competenze di sicurezza non specialistiche di effettuare una prima valutazione del programma, la quale potrà poi essere approfondita tramite analisi più specifiche. La piattaforma si basa su **angr**, un toolkit open-source multi-architettura per l'analisi binaria, per eseguire automaticamente diverse tipologie di analisi statiche e dinamiche, sul framework python **Flask** per l'implementazione di una REST API progettata per comunicare i risultati dell'analisi e sul framework javascript **SvelteKit**, il quale si occupa della strutturazione delle pagine web della piattaforma e della presentazione dei risultati dell'analisi all'utente.

1.1 Differenza tra debolezza e vulnerabilità

Spesso il termine "vulnerabilità" è utilizzato per riferirsi ad una qualsiasi problematica di sicurezza all'interno del software sotto analisi. Tuttavia, è fondamentale distinguere il concetto di **vulnerabilità** da quello di **debolezza**. Per delineare con precisione questa distinzione, adotteremo le definizioni fornite dal glossario compilato dal MITRE [1]:

- **Debolezza**: Una condizione nel software, firmware, hardware o in una componente di servizio che, sotto certe circostanze, potrebbe contribuire all'introduzione di vulnerabilità
- **Vulnerabilità**: Un errore nel software, firmware, hardware o componente di servizio **derivante dalla presenza di una debolezza** che può essere sfruttata da un'attaccante, causando un impatto negativo sull'integrità, la confidenzialità e la disponibilità dei componenti impattati

Una vulnerabilità è quindi **un'istanza sfruttabile di una debolezza**. Per riferirci alle categorie di difetti che le tecniche di analisi automatica offerte dalla piattaforma sono in grado di rivelare, questa tesi adotterà la tassonomia **Common Weakness Enumeration** (CWE), anch'essa compilata dal MITRE.

1.2 Struttura della relazione

La relazione è articolata nei seguenti capitoli:

- **Capitolo 2: Stato dell'arte**: Questo capitolo presenta una rassegna di alcune tecniche, metodologie e soluzioni esistenti per l'analisi di file binari. Verrà evidenziato l'approccio adottato per affrontare il problema della ricerca di vulnerabilità e i rispettivi limiti di ogni soluzione presentata.
- **Capitolo 3: Metodologie utilizzate**: Questo capitolo discute i fondamenti teorici che costituiscono la base delle analisi implementate dalla piattaforma. Saranno discussi in dettaglio sia i concetti di **disassembling** e **decompiling** sia le metodologie di analisi statica e dinamica utilizzate per effettuare la ricerca delle vulnerabilità. Verranno inoltre forniti degli esempi per illustrarne il funzionamento.
- **Capitolo 4: Tecnologie utilizzate**: Questo capitolo presenta in dettaglio le tecnologie e i framework scelti per l'implementazione della piattaforma. Saranno presentati sia i componenti del backend sia le tecnologie adottate per lo sviluppo del frontend.
- **Capitolo 5: Analisi implementate**: Questo capitolo illustra nel dettaglio le analisi implementate all'interno della piattaforma. Verrà descritto come ciascuna tecnica di analisi porti al rilevamento di una vulnerabilità e verrà fornita una lista comprensiva di tutte le debolezza software che ogni tecnica è capace di rilevare.
- **Capitolo 6: Architettura della soluzione**: Questo capitolo descrive l'architettura generale della piattaforma Binoculars. Verrà illustrato il modello architetturale della soluzione, illustrando le interazioni fra i vari componenti e come essi collaborano per presentare all'utente il risultato dell'analisi richiesta
- **Capitolo 7: Sperimentazione** Questo capitolo presenta le varie sperimentazioni effettuate sulla piattaforma al fine di validarne l'accuratezza. Per ogni tecnica di analisi implementata, verranno presentati i programmi che sono stati utilizzati al fine di validare l'efficacia e l'accuratezza dell'analisi e i risultati prodotti da quest'ultima

- **Capitolo 8: Conclusioni**: Questo capitolo presenterà le conclusioni finali del lavoro. Saranno inoltre esposte le limitazioni e le problematiche incontrate durante l'implementazione della piattaforma e i suoi possibili sviluppi futuri

Capitolo 2

Stato dell'arte

2.1 Metodologie basate su tecniche di analisi statica

L'analisi statica di un programma consiste in un'insieme di metodologie, tool e algoritmi che permettono l'analisi del codice sorgente o della sua rappresentazione binaria (per esempio, un file eseguibile) senza che il programma venga effettivamente eseguito [2]. Questa tecnica è ampiamente adottata nell'ambito della ricerca delle vulnerabilità, in quanto consente di inferire e determinare se certe proprietà sono soddisfatte (per esempio, le condizioni che possono portare ad una certa vulnerabilità) senza direttamente eseguire il programma. Tuttavia, l'analisi statica condotta direttamente su un file binario è intrinsecamente più complessa rispetto all'analisi statica del codice sorgente: la principale difficoltà risiede nella mancanza di informazioni riguardante i tipi e la struttura ad alto livello del codice [3]. Nonostante queste sfide, nel corso degli anni sono stati sviluppati diversi approcci e metodologie di analisi statica progettati per effettuare la ricerca di vulnerabilità all'interno di file binari. Le tecniche di analisi statica, tuttavia, possono produrre un elevato numero di falsi positivi e falsi negativi : poiché queste tecniche non effettuano un'esecuzione concreta del programma, esse devono effettuare diverse assunzioni sul suo stato a runtime. Ciò potrebbe quindi portare i tool basati su questa tipologia di analisi a segnalare vulnerabilità in porzioni di programma non vulnerabili.

2.1.1 Taint analysis statica: Bintaint

La *taint analysis* (o *taint checking*) è una tecnica di analisi che mira a tracciare e monitorare la propagazione di flussi di dati inaffidabili o potenzialmente dannosi all'interno del programma. La taint analysis si compone di tre elementi chiave:

1. **Sorgenti** (Sources): Sono i punti del programma dove si origina un flusso di dati inaffidabile. Una sorgente potrebbe per esempio essere l'input di un utente oppure i dati letti da un file.

2. **Propagazione**: Viene effettuato un monitoraggio continuo della propagazione nel programma dei dati provenienti da una sorgente

3. **Sink**: Sono i punti del programma che effettuano operazioni sensibili, come per esempio l'accesso al filesystem o la chiamata ad operazioni di libreria non sicure.

Una possibile vulnerabilità verrà quindi rilevata quando il programma permette ad un dato "tainted" di raggiungere un sink; ciò può avvenire quando, per esempio, il dato non

viene adeguatamente sanificato.
Bintaint è un tool di parsing capace di effettuare taint analysis statica su file binari [4]. Il taint analyzer proposto implementato dal tool è basato sul tool commerciale di reverse engineering *IDA*, il quale viene utilizzato per recuperare il codice assembly dal codice binario, ed è implementato utilizzando il linguaggio funzionale *OCaml*. Bintaint è composto da quattro moduli distinti:

- **Decoder module**: Questo modulo si occupa di tradurre il codice assembly recuperato da IDA in una rappresentazione in un linguaggio intermedio chiamato *REIL* (Reverse Engineering Intermediate Language), le quali espressioni verranno a loro volta convertite in espressioni simboliche.

- **Taint Processing Configuration Module**: Questo modulo gestisce la configurazione per l'inizializzazione della taint analysis, leggendo la configurazione fornita dall'utente in formato XML, la quale dovrà contenere tutte le informazioni necessarie per effettuare la taint analysis. Questo modulo si occupa inoltre di stabilire una relazione tra l'input esterno e le varie sorgenti definite

- **Expression Parsing Module**: Questo modulo si occupa di definire come avviene la propagazione dei flussi di dati tainted all'interno del programma

- **TCFG Generation Module**: Questo modulo si occupa di generare una struttura a grafo diretta chiamata *Taint Control Flow Graph*, la quale rappresenterà tutte le possibili aree del programma che un determinato flusso tainted può raggiungere. L'analisi del TCFG permetterà quindi di evincere se un determinato sink dipende dai dati generati da una determinata sorgente.

L'approccio proposto dal tool permette di ridurre il numero di falsi positivi e falsi negativi rilevati rispetto ad una tain analysis tradizionale; inoltre, l'utilizzo del linguaggio intermedio REIL permette al tool di essere facilmente integrabile in sistemi di analisi più complessi, a patto che anch'essi utilizzino lo stesso linguaggio di rappresentazione intermedia. Tuttavia il tool risulta comunque dipendente dall'input dell'analista; l'accuratezza dell'analisi dipenderà quindi dalla corretta definizione di sorgenti, sink e propagazione da parte di quest'ultimo. Infine, Bintaint si basa sul framework commerciale IDA, il quale non offre tutte le sue funzionalità nella sua versione gratuita.

Bibliografia

[1] MITRE, *Common Weakness Enumeration Glossary*, Accesso effettuato il 26 settembre 2025, 2024. indirizzo: https://cwe.mitre.org/documents/glossary/

[2] P. Thomson, «Static analysis,» *Commun. ACM*, vol. 65, n. 1, pp. 50–54, dic. 2021, ISSN: 0001-0782. DOI: 10.1145/3486592 indirizzo: https://doi.org/10.1145/3486592

[3] Y. Xu et al., «A Review of Code Vulnerability Detection Techniques Based on Static Analysis,» in *Computational and Experimental Simulations in Engineering*, S. Li, cur., Cham: Springer Nature Switzerland, 2024, pp. 251–272, ISBN: 978-3-031-44947-5.

[4] Z. Feng, Z. Wang, W. Dong e R. Chang, «Bintaint: A Static Taint Analysis Method for Binary Vulnerability Mining,» in *2018 International Conference on Cloud Computing, Big Data and Blockchain (ICCBB)*, 2018, pp. 1–8. DOI: 10.1109/ICCBB.2018.8756383