Valutazione di Flawfinder su 3 progetti popolari su GitHub: Indagine sulle vulnerabilità in codice C

Approfondimento per il corso di Sicurezza Informatica

7 giugno 2023

Rocco Gianni Rapisarda, 845197

Indice

1 Introduzione

La presente relazione si propone di fornire un'analisi dettagliata sull'utilizzo di Flawfinder, uno strumento di analisi statica ampiamente adottato nel campo della sicurezza informatica, per valutare la presenza potenziale di vulnerabilità in tre progetti scritti in linguaggio C, disponibili su GitHub e caratterizzati da un significativo numero di stelle (stars).
Flawfinder è un tool che si concentra sull'individuazione di difetti di programmazione nel codice sorgente C e C++, al fine di rilevare vulnerabilità di sicurezza note. Questo strumento si basa su un database di pattern di codice che corrispondono a vulnerabilità comuni, come ad esempio buffer overflow, race condition e mancanza di controlli sugli input.
L'accuratezza di Flawfinder non è infallibile e possono verificarsi situazioni in cui vengono segnalati potenziali problemi di sicurezza che in realtà non sono presenti nel codice. Questi falsi positivi possono essere il risultato di complessità nell'analisi statica, dell'incapacità di riconoscere determinati modelli di codice o di altre limitazioni dell'algoritmo utilizzato.
Nel corso di questa relazione, saranno presentati i risultati ottenuti dall'applicazione di Flawfinder sui tre progetti prescelti.
La lista delle CWE che Flawfinder rileva è disponibile a pagina 10 della documentazione del tool e di seguito viene riportata una sotto lista.

- CWE-20: validazione impropria dell'input.
- CWE-120: copia del buffer senza verifica della dimensione dell'input.
- CWE-126: Buffer Over-read.
- CWE-250: esecuzione senza privilegi.

Si sottolinea che, nonostante la presenza di falsi positivi, l'utilizzo di Flawfinder rimane uno strumento valido per avviare un processo di analisi delle vulnerabilità di sicurezza nel codice sorgente scritto in C e per una valutazione completa e accurata può essere necessario eseguire dell'analisi statica con altri strumenti oppure dell'analisi dinamica.

2 Metodologia d'analisi

Per condurre l'analisi statica dei progetti presi in considerazione, è stato sviluppato uno script in *Bash* che automatizza il processo di recupero delle repository GitHub dei progetti e l'esecuzione di Flawfinder. L'intero flusso di lavoro è stato strutturato come segue:

1. **Creazione delle cartelle di output:** lo script verifica se le cartelle "projects" e "output" esistono. In caso contrario, le crea per consentire l'organizzazione dei file generati durante l'analisi.

2. **Recupero delle repository GitHub:** una volta verificata la presenza delle cartelle, lo script procede al recupero delle repository dei progetti specificati. Le repository vengono clonate all'interno della cartella "projects" per consentire un'analisi locale.

3. **Esecuzione di Flawfinder:** dopo aver clonato le repository, lo script esegue Flawfinder sui file sorgente dei progetti. Flawfinder viene configurato per considerare solo le vulnerabilità con un livello pari o superiore a 4 (su una scala da 0 a 5), al fine di focalizzare l'analisi sulle potenziali vulnerabilità più critiche.

4. **Salvataggio dei risultati in formato CSV:** i risultati ottenuti dall'analisi di Flawfinder vengono salvati in un file CSV all'interno della cartella "output". Questo file contiene informazioni dettagliate sulle vulnerabilità individuate, inclusi i nomi dei file, i livelli di rischio associati e i tipi di vulnerabilità corrispondenti.

Successivamente, per la generazione dei grafici, sono stati sviluppati due script in Python che elaborano i dati presenti nei file CSV e crea due tipi di grafici:

- Grafico a barre: vengono utilizzati i valori della colonna "CWEs" (Common Weakness Enumerations) del file CSV, che rappresentano i tipi di vulnerabilità identificate da Flawfinder. Utilizzando tali valori, viene generato un grafico a barre che mostra la distribuzione delle vulnerabilità per ogni progetto analizzato.

- Grafico a torta: i dati presenti nel file CSV vengono analizzati per generare un grafico a torta che visualizza la percentuale di ogni categoria di vulnerabilità rispetto al totale delle vulnerabilità rilevate. Le vulnerabilità con una percentuale inferiore al 3% vengono raggruppate nella categoria *Others*.

Nella sezione successiva della relazione, saranno presentati i risultati ottenuti per ciascun progetto. Per ogni progetto, verrà fornita una breve descrizione del suo contesto e degli obiettivi principali.
Inoltre, saranno inclusi commenti relativi ai risultati rappresentati nei grafici, al fine di fornire un'analisi più approfondita delle vulnerabilità individuate e delle relative distribuzioni.

3 Risultati

I risultati dell'analisi condotta utilizzando Flawfinder presi in esame presentano un'importante fonte di informazioni riguardo alle vulnerabilità potenziali rilevate nel codice sorgente scritto in linguaggio C.
La valutazione è stata effettuata considerando i livelli di rischio pari o superiori a 4, come definito all'interno dello strumento Flawfinder.
Di seguito, verranno presentati i risultati ottenuti per ciascun progetto analizzato, fornendo una breve descrizione del contesto e degli obiettivi principali del progetto stesso.

Project name	**GitHub stars**	**Commit hash**
OBS-Studio	47,800	adb702929
Scrcpy	84,000	958f2249
VLC	10,900	f7bb59d9f5

Tabella 1: GitHub Statistics

3.1 OBS-Studio

Nel progetto OBS-Studio, come è possibile notare dalla figura 1, vi è una predominanza della vulnerabilità *CWE-134* (48.9%), seguita dalla *CWE-120* (33.3%).
La prima vulnerabilità si verifica se è utilizzata una funzione che accetta una format string come argomento, ma questa proviene da un'origine esterna.
Un utente malintenzionato può modificare una format string controllata dall'esterno e può causare Buffer overflow oppure DoS oppure problemi relativi alla rappresentazione dei dati.
In alcune circostanze, come l'internazionalizzazione, l'insieme di format string è controllato esternamente dalla progettazione.
Se l'origine di una format string è attendibile (ad esempio, contenuta solo in file di libreria modificabili solo dall'amministratore di sistema), allora il controllo esterno potrebbe non rappresentare di per sé una vulnerabilità.
La *CWE-120*, invece è presente nei programmi in cui viene eseguita una copia di un buffer di input in un buffer di output senza verificare che la dimensione del buffer di input sia inferiore alla dimensione del buffer di output portando ad avere una possibile generazione di Buffer overflow.

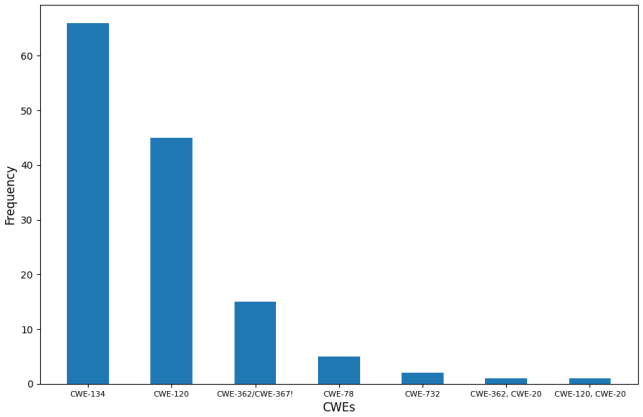

Figura 1: Istogramma delle vulnerabilità CWE individuate nel progetto OBS-Studio tramite Flawfinder.

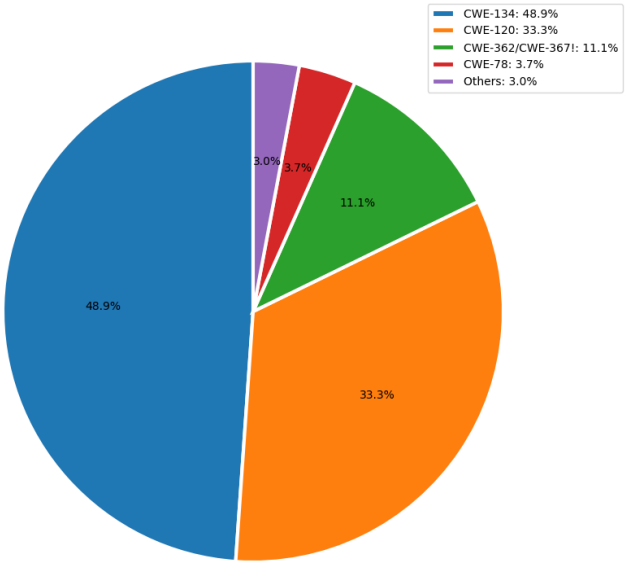

Figura 2: Grafico a torta contenente la percentuale di presenza per ciascuna vulnerabilità CWE individuata nel progetto OBS-Studio tramite Flawfinder.

3.2 Scrcpy

Scrcpy è uno strumento open-source che consente di visualizzare e controllare utilizzando il mouse e la tastiera del computer un dispositivo Android tramite una connessione USB o una connessione Wi-Fi.
Rispetto agli altri progetti che sono stati analizzati, Scrcpy è quello che contiene

meno vulnerabilità con livello di rischio pari o superiore a 4.e quella più presente è la *CWE-78*, che permette ad un attaccante di eseguire comandi imprevisti e pericolosi direttamente sul sistema operativo.
Questa debolezza può portare a una vulnerabilità in ambienti in cui l'attaccante non ha accesso diretto al sistema operativo, come ad esempio nelle applicazioni Web. Se al programma vengono dati i permessi d'amministratore allora l'attaccante potrebbe eseguire eseguire comandi che normalmente non sarebbero accessibili o di richiamare comandi alternativi con privilegi che non possiede.

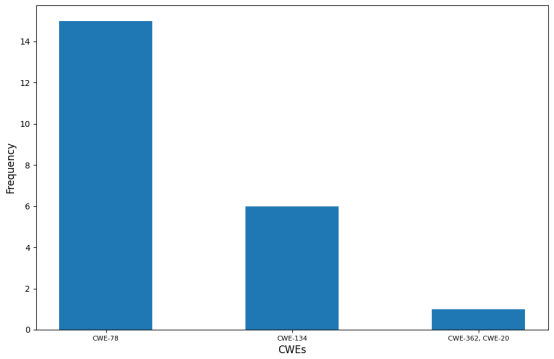

Figura 3: Istogramma delle vulnerabilità CWE individuate nel progetto OBS-Studio tramite Flawfinder.

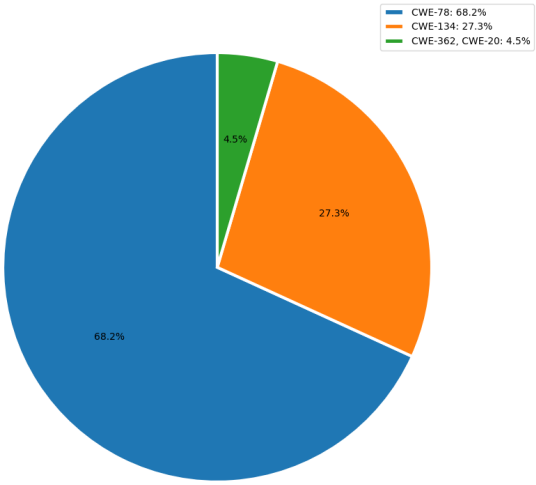

Figura 4: Grafico a torta contenente la percentuale di presenza per ciascuna vulnerabilità CWE individuata nel progetto Scrcpy tramite Flawfinder.

3.3 VLC

Nel progetto VLC, come è possibile notare dalla figura 5, vi è una predominanza della vulnerabilità *CWE-362/CWE-367* (53.9%).
La vulnerabilità CWE-362/CWE-367 è conosciuta come "Race Condition". Una race condition si verifica quando il comportamento di un programma dipende dall'ordine in cui vengono eseguite diverse operazioni concorrenti o interagenti. In particolare, nel contesto del progetto VLC, questa vulnerabilità può emergere se un aggressore riesce a modificare qualcosa lungo il percorso tra la chiamata ad access() e l'utilizzo effettivo del file, ad esempio spostando i file.
Quando un'applicazione utilizza la chiamata di sistema access() per verificare l'accesso a un file, potrebbe esserci un intervallo di tempo tra la verifica e l'utilizzo effettivo del file. Durante questo intervallo di tempo, se un attaccante riuscisse a modificare il percorso del file o i suoi permessi, potrebbe sfruttare la race condition per ottenere un accesso non autorizzato al file.
Una possibile soluzione per affrontare questa vulnerabilità potrebbe essere l'impostazione dei permessi corretti sul file.
Ad esempio, utilizzando la funzione di sistema setuid(), si potrebbe assegnare al file i permessi adeguati in modo che solo gli utenti autorizzati possano accedervi. Inoltre, potrebbe essere utile aprire direttamente il file dopo la chiamata ad access(), senza lasciare un intervallo di tempo in cui la race condition potrebbe essere sfruttata.
Altre due vulnerabilità rilevate da Flawfinder sono la CWE-120 (buffer overflow) e la CWE-134 (format string exploit). La CWE-120 si verifica quando si copia un buffer di input in un buffer di output senza verificarne la dimensione, causando un overflow del buffer. La CWE-134 si verifica quando un programma utilizza una stringa di formato proveniente da un'origine esterna senza una corretta validazione, consentendo attacchi di manipolazione o esecuzione di codice dannoso.

```
1     static int Control(stream_t *access, int query, va_list args)
```

Codice 1: Istanza di CWE-362/CWE-367 rilevata da Flawfinder.

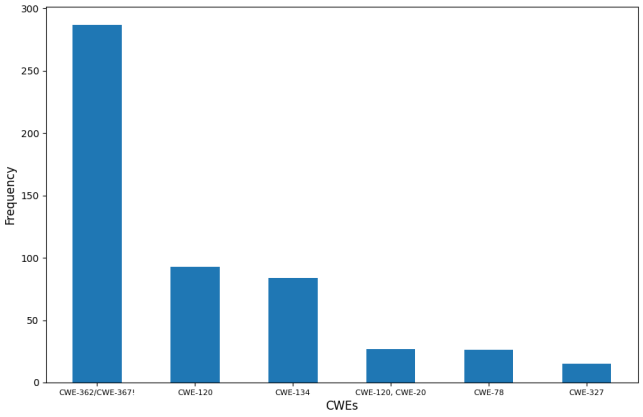

Figura 5: Istogramma delle vulnerabilità CWE individuate nel progetto OBS-Studio tramite Flawfinder.

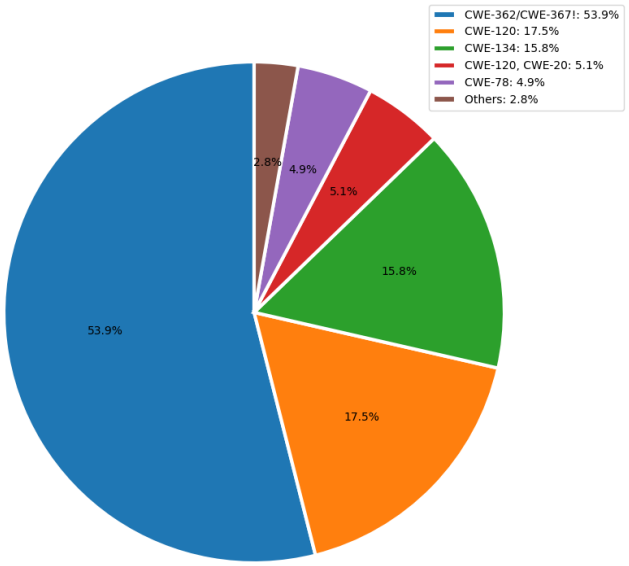

Figura 6: Grafico a torta contenente la percentuale di presenza per ciascuna vulnerabilità CWE individuata nel progetto VLC tramite Flawfinder.

4 Conclusioni

Dall'analisi effettuata è emerso che i tre progetti sono affetti da varie tipologie di vulnerabilità ed in alcuni casi si riscontrano le stesse vulnerabilità in tutti e tre i progetti, come ad esempio la *CWE-134* e la *CWE-78*. Questa informazione è rappresentata in modo sintetico nel grafico riassuntivo mostrato nella figura 7.

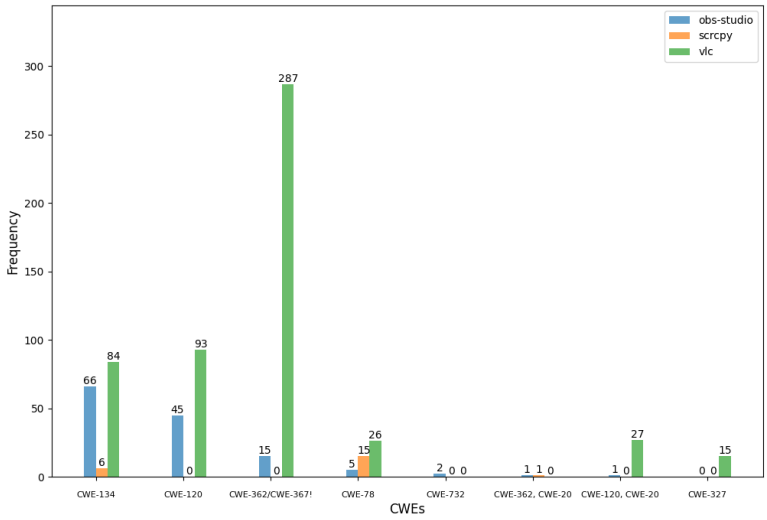

Figura 7: Istogramma delle vulnerabilità CWE individuate in tutti i progetti tramite Flawfinder.

Come detto nell'introduzione Flawfinder esegue un'analisi statica del codice, il che significa che potrebbe produrre falsi positivi. Per gestire correttamente i falsi positivi, è consigliabile eseguire una verifica manuale delle linee di codice segnalate da Flawfinder e nel caso in cui non si tratta di un'istruzione pericolosa indicare al tool, utilizzando uno dei due commenti presenti nel listato 2 nel codice, di ignorare le linee che non rappresentano una reale minaccia per la sicurezza. In questo modo, si ottiene un'analisi più accurata e si riduce il rischio di segnalazioni errate.

```
1 // Flawfinder: ignore
2 /* Flawfinder: ignore */
```

Codice 2: Commento per ignorare falso positivo.