Design e sviluppo di un tool per il Vulnerability Assesment di protocolli avanzati di rete

Veronica Falgiani (Matricola 21191A)

Relatore: Claudio Agostino Ardagna **Correlatore**: Nicola Bena

Le reti di organizzazioni ed aziende sono costantemente minacciate da attacchi informatici provenienti da tutto il mondo. Uno dei principali metodi per potersi difendere è verificare continuamente che le versioni dei servizi utilizzati e le rispettive configurazioni non siano etichettate come vulnerabili. Il compito di controllare costantemente questi requisti è oneroso da dover compiere manualmente e di conseguenza nel corso degli anni sono stati sviluppati software per poter aiutare gli esperti di sicurezza in questo compito. I vulnerability scanner sono strumenti altamente avanzati che interagiscono con i servizi presenti sulla rete inviando pacchetti ed analizzandone le risposte per individuare criticità. Questi software sono in grado di riconoscere un gran numero di protocolli di rete e le rispettive vulnerabilità in breve tempo.

L'obiettivo di questo elaborato consiste nello sviluppo di un vulnerability scanner da zero, studiando i punti di forza e le criticità dei sistemi già in uso per proporre una nuova implementazione più leggera e modulare. L'idea è di creare un applicativo facilmente modificabile, per permettere l'aggiunta di nuove funzionalità e test più rapidamente. Questo può essere ottenuto grazie all'utilizzo di template prestabiliti che rendono il programma anche più facile e veloce da utilizzare. Nello sviluppo del seguente progetto è stato scelto un numero di protocolli ridotto, in modo tale da poterne dimostrare il funzionamento generale. Grazie alla modularità, questo progetto è aperto a nuove aggiunte e modifiche per tentare di stare al passo con le nuove tecnologie e vulnerabilità che vengono scoperte ogni giorno.

Il contenuto della testi si articola in diverse sezioni che percorrono le fasi dello sviluppo, partendo dallo studio delle tecnologie per arrivare all'applicativo funzionante.

- *Studio dello stato dell'arte.* Prima di poter sviluppare il progetto è stato necessario studiare i software già presenti in commercio. Questo è stato fatto per analizzare le funzionalità base da implementare, ma anche per capire cosa andare a modificare o migliorare. Il vulnerability scanner analizzato è Nessus, ma sono stati presi in considerazione anche le funzionalità di scanning e riconoscimento delle vulnerbailità di Metasploit e Nmap. Le criticità riscontrate riguardano la complessità e la dimensione eccessiva di queste soluzioni, sottolineando anche il livello di difficoltà nell'utilizzo e nella scrittura dei test da eseguire per individuare le vulnerabilità.

- *Progettazione di un vulnerability scanner.* A seguito dell'individuazione dei punti di debolezza dei programmi citati nello "stato dell'arte", vengono definiti in modo chiaro gli obiettivi e le migliorie che l'applicativo deve implementare, tra cui la modularità e la facilità di utilizzo. In questa fase viene individuato Python come linguaggio di programmazione, grazie alla sua semantica semplice e alle molteplici librerie, e il formato JSON per la sua grande versatilità. Il funzionamento di un vulnerability scanner viene suddiviso in cinque fasi ben distinte: scansione dell'host, scansione delle porte, scansione dei protocolli e dei servizi, esecuzione dei test e infine stampa dei risultati. Queste cinque vengono portate a termine grazie all'utilizzo e all'analisi dei messaggi ricevuti dai vari protocolli di rete utilizzati.

- *Implementazione dell'applicativo e dei test.* Il progetto è suddiviso in due moduli: Agent, che contiene tutte le funzioni principali per la comunicazione con gli host, e Utils, contenente tutte le funzionalità per interagire con l'utente e generare risultati. L'applicativo viene eseguito richiamando il file main.py che utilizza al suo interno tutte le funzioni contenute nelle varie librerie implementate. Il programma inizia la sua esecuzione verificando lo stato dell'host e, nel caso in cui sia attivo, procede con la scansione delle porte aperte. Terminata questa fase prova a riconoscere i protocolli e i servizi presenti sul dispositivo e successivamente testa le vulnerabilità presenti. L'ultima operazione che compie è quella di scrittura dei risultati.
  Per poter svolgere correttamente la fase di testing è necessario fornire all'applicativo dei file contenenti i test da svolgere. Questi sono suddivisi in test per i protocolli e test per i servizi ed entrambi devono seguire una struttura di template ben definita per poter essere utilizzati correttamente dal programma. All'interno di ogni file è necessario riportare le vulnerabilità che si vogliono testare, specificando i comandi per scatenarle e le risposte per valutarne la presenza. Può anche essere specificata una stringa contenente i comandi per svolgere l'autenticazione verso il protocollo/servizio che presenta dei campi placeholder in cui verranno riportate le credenziali chieste a runtine all'utente. Altri elementi vengono usati per specificare i servizi che possono essere utilizzati in combinazione con i protocolli e le versioni dei servizi che presentano vulnerabilità, accompagnati dalle rispettive CVE.

- *Analisi dei risultati.* L'applicativo sviluppato cerca di venire incontro al bisogno di più soggetti, implementando la creazione di risultati secondo tre tipologie di file ben differenti. In primis i risultati devono

poter essere letti e compresi dall'uomo, per questo motivo sono stati utilizzati i formati TXT, per una più rapida interpretazione, e HTML, per un riscontro grafico. Inoltre è stato tenuto conto della necessità di poter far elaborare questi dati a programmi di terze parti. Per questo motivo si è scelto di utilizzare il formato JSON, uno dei più supportati per il parsing di informazioni. All'interno di questi tre file sono presenti le medesime informazioni, servite in modalità differenti. Per ogni host vengono riportate le porte aperte e i servizi individuati e a loro volta sono presenti le vulnerabilità confermate per ogni servizio, con tanto di nome, descrizione e gravità.

L'applicativo sviluppato permette di svolgere un vulnerability scan su un limitato numero di servizi e vulnerabilità. Sviluppi futuri potrebbero espandere le funzionalità base già implementate, permettendo l'aumentare del numero di servizi riconosciuti dalle scansioni e aggiungendo nuove feature all'interno del software. Alcune di queste nuove funzionalità vertono sull'automazione, partendo da un semplice script in grado di creare automaticamente batterie di test, fino ad arrivare allo sviluppo di una versione dell'applicativo che viene eseguita in autonomia attraverso degli script.