# Software Reverse Engineering

# Cosa?

Il *software reverse engineering* è il processo di analizzare un programma di cui non abbiamo il codice sorgente per capirne il funzionamento.

# Perchè?

La maggior parte del software che utilizziamo è closed source, perciò l'unico modo per sapere cosa sta facendo under the hood è analizzarlo.

Ci sono diverse motivazioni che potrebbero portarvi a voler capire ciò, provo ad elencarvene qualcuna.

# Applicazioni

Malware Analysis

Exploit Analysis - Deep dive into an NSO zero click iMessage Exploit

Software Preservation

Compatibility

Security assessment

Vulnerability Research

~~Game Cheating~~

# Come?

Esistono due approcci diversi per analizzare un programma, ognuno dei quali ha i suoi pro e i suoi contro.

- Analisi **Statica**: Il programma non viene eseguito, si analizza solamente il contenuto.
- Analisi **Dinamica**: Il programma viene eseguito in un ambiente controllato, che permette di ispezionare tutto ciò che accade a runtime.

# Analisi Statica: I Pro

Non c'è il rischio di danneggiare la macchina, se il programma sotto analisi è malevolo

Non serve avere l'hardware o l'emulatore per lanciare il programma

È l'unico modo per analizzare tutte le possibili *execution paths*

Bypassa di default qualsiasi controllo di anti debug

# Analisi Statica: I Contro

È molto difficile tenere traccia di quali dati il codice stia utilizzando

Se il codice è offuscato pesantemente, diventa difficile analizzarlo in modo statico

È un processo lento

# Analisi Statica: Come?

Per una analisi preliminare:

strings

ldd

nm

readelf

# Analisi Statica: Come?

Per una analisi approfondita:

- Ghidra
- Binary Ninja
- IDA (Free)
- Radare / Rizin
- Hopper

# Analisi Dinamica: I pro

Permette di vedere chiaramente quali dati vengono utilizzati dal codice

È un processo veloce

# Analisi Dinamica: I Contro

Potrebbero esserci dei meccanismi di antidebug, in questo caso l'unico modo per bypassarli è capirne prima il funzionamento attraverso l'analisi statica.

Potrebbe non essere possibile eseguire il codice, ad esempio perché manca una libreria, o perché non avete il processore adatto e non esiste un emulatore per quella architettura.

# Analisi Dinamica: Come?

Per una analisi preliminare:

ltrace

strace

# Analisi Dinamica: Come?

Per una analisi approfondita:

- gdb
- rr
- frida
- hooks con LD_PRELOAD

# Analisi Statica vs Analisi Dinamica

Per capire perché il codice che avete scritto per l'esame di programmazione non funziona, preferite leggervi ogni riga del codice o mettere delle printf a caso in giro per il programma?

# Tips and tricks

Giocate col programma per farvi una idea di cosa fa velocemente

Durante l'analisi statica, fate delle ipotesi e provate a confermarle o negarle attraverso l'analisi dinamica

Se state reversando un programma grande, non perdetevi subito nei dettagli, prima di analizzare attentamente qualcosa assicuratevi che sia importante, o vi ritroverete a perdere 30 minuti analizzando una strcmp ottimizzata

# Tips and tricks

Mettete il focus sul come vengono utilizzati i dati dal codice, la maggior parte delle volte potrete permettervi di ignorare il contorno e capire ugualmente cosa sta succedendo.