3/10/2023

Lezione 2

Introduzione alla RCE

Analisi del Malware

3 ottobre 2023

Marco Cesati

Dipartimento di Ingegneria Civile e Ingegneria Informatica
Università degli Studi di Roma Tor Vergata

Di cosa parliamo in questa lezione?

Introduzione alla RCE

1. Definizione di RCE
2. Motivazioni
3. Principi generali
4. Metodologia

Introduzione alla RCE
Marco Cesati

Cosa si intende per Reverse Engineering

Reverse Engineering

Il processo di estrazione della conoscenza e degli schemi di progetto di qualsiasi oggetto costruito dall'uomo

è l'operazione inversa alla creazione, poichè sto estrapolando informazioni dal prodotto finale

- Nasce con la rivoluzione industriale
- Molto prima dell'avvento dei calcolatori elettronici

Non è un processo meccanico, può fallire, come nel caso della ricerca scientifica. La differenza è che, mentre nella scienza quello che studiamo potrebbe andare oltre le nostre conoscenze;
nel caso del malware sappiamo che è stato costruito
da un essere umano. Devo "riscoprire" tale conoscenza. Ma esiste!

Ricerca Scientifica e Reverse Engineering

Introduzione alla RCE
Marco Cesati

Schema della lezione
Definizione di RCE
Motivazioni
Principi generali
Metodologia

La ricerca scientifica e il Reverse Engineering sono per certi aspetti molto simili tra loro:

- estraggono da un insieme di dati quantitativi (osservazioni, esperimenti, . . .)
- un modello formale che rappresenti in modo adeguato la "realtà"
- realtà che non è direttamente desumibile dalla base di dati raccolti
- utilizzando l'intuizione personale
- e utilizzando opportune metodologie che consentono la continua verifica dei risultati ottenuti

Ricerca Scientifica e Reverse Engineering (2)

La differenza principale:

- la ricerca scientifica investiga i fenomeni naturali
- il Reverse Engineering investiga gli artefatti umani

Chi pratica il Reverse Engineering ha un vantaggio rispetto al ricercatore scientifico:

- la conoscenza cercata è stata nascosta, persa o dimenticata
- tuttavia tale conoscenza è stata, od è tuttora, patrimonio di qualche essere umano

Reverse Code Engineering

Reverse Code Engineering

Il processo di ricostruzione delle finalità, delle strutture di dati, dell'algoritmo e dei dettagli implementativi di un programma per calcolatore elettronico

Nomi equivalenti:

- *Reverse Code Engineering (RCE)*
- *software reverse engineering*
- *ingegneria inversa dei programmi*
- *decompilazione*

Introduzione alla RCE
Marco Cesati

Schema della lezione
Definizione di RCE
Motivazioni
Principi generali
Metodologia

Reverse Code Engineering (2)

- L'*input* del RCE è una rappresentazione "a basso livello" di un programma per calcolatore, ad esempio:
  - il file eseguibile contenente il codice macchina
- L'*output* del RCE è una rappresentazione "ad alto livello" delle conoscenze apprese durante l'attività di reversing, ad esempio:
  - La descrizione degli effetti del programma
  - Una porzione dell'algoritmo del programma espresso in uno pseudo-linguaggio di programmazione
  - La descrizione formale di un protocollo di comunicazione o autenticazione

"Alto livello" e "basso livello" possono essere concetti relativi!

Questi due livelli sono legati da una scatola (nera, bianca o grigia) che vedremo più avanti.

Le motivazioni del RCE

I motivi per i quali si intraprende l'attività di RCE possono essere svariati e molteplici

- Produrre adeguata documentazione del funzionamento di un proprio programma sviluppato in passato
- Sviluppare un programma che interagisca o utilizzi un altro programma di terze parti
- Sviluppare un programma che si ponga come concorrente di un altro programma di terze parti
- Verificare che il funzionamento di un programma di terze parti non costituisca un rischio per la propria organizzazione
- Superare i meccanismi di protezione di materiale digitale (film, musica, libri, giochi elettronici, software)

Esiste una sottile linea di confine tra quello che è lecito e illecito.

Le motivazioni del RCE (2)

- Analizzare il comportamento del malware al fine di
  - determinare i suoi effetti
  - sviluppare un sistema di protezione
  - derivarne una nuova versione
- Acquisire conoscenze sulle vulnerabilità dei sistemi informatici per
  - attaccarli
  - renderli più robusti
- Determinare possibili debolezze teoriche o implementative di una applicazione di crittografia

RCE viene eseguita su codice di tipo closed source, su quello open, avendo il codice in chiaro, non ha senso.

Aspetti legali del RCE

- Alcune attività di RCE sono illegali in quasi tutte le nazioni del mondo
- Altre attività sono legali o tollerate
- La maggior parte delle applicazioni commerciali vengono utilizzate con *licenze d'uso* che esplicitamente proibiscono l'attività di RCE
- Spesso non è semplice stabilire se una attività di RCE è legale o meno
  - Ad esempio, in Italia sembra essere consentito fare RCE del software di una stampante allo scopo di vendere cartucce d'inchiostro riciclate (rifornite)
  - mentre sembra essere illegale fare RCE del software di una stampante allo scopo di produrre cartucce d'inchiostro non originali (compatibili)

Hacker e cracker

Ok per lo scopo educativo (anche se il contratto d'uso lo vieta), NON OK per casi come cartucce riciclate.

- L'attività di RCE non è in generale illecita *di per se*
  - Lo diventa certamente quando ha una finalità illegale
  - oppure lede i diritti di qualche persona fisica o giuridica

Hacker (buono)

Persona che studia e sfrutta in modo creativo il codice dei programmi o le debolezze dei sistemi informatici per interesse professionale oppure per curiosità personale

Cracker (cattivo)

Persona che persegue l'obiettivo di superare i meccanismi di protezione di un sistema informatico per fini di lucro e senza il consenso dei legittimi proprietari o aventi diritto

Principi generali del RCE

RCE : programmazione = integrale : derivata
la derivata e la programmazione sono più facili rispetto all'operazione inversa, ovvero integrali ed RCE.

- Il RCE può essere considerato come l'operazione inversa alla programmazione
- Il buon programmatore adotta una serie di buoni principi di programmazione che facilitano lo sviluppo di software di qualità
- Analogamente, il bravo hacker cerca di ottemperare ad una serie di principi di RCE che facilitano ed accelerano il suo lavoro

Principio di RCE #1

Maggiore è la comprensione dell'intero sistema, tanto più rapida, efficiente e produttiva è l'attività di reversing

L'hacker deve essere una persona versata e competente in molti svariati settori dell'informatica:

- L'architettura dei calcolatori elettronici
  - come funzionano i microprocessori
- i sistemi operativi
  - interfacce con le applicazioni
  - protocolli di comunicazione
  - sistema specifico del programma sotto esame

Principio di RCE #1 (2)

- Il processo di compilazione dei programmi
  - istruzioni assembly corrispondenti ai vari costrutti del linguaggio ad alto livello
  - compilatore e linguaggio specifici del programma in esame
- Il formato in cui è codificato il file eseguibile
- I principali metodi di offuscamento, protezione, rilevazione dei debugger e delle macchine virtuali utilizzate dagli hacker per l'attività di RCE
  - la "guerra" tra programmatori e hacker è sempre aperta
  - le innovazioni in entrambi i campi sono continue
  - richiede un continuo sforzo di apprendimento e aggiornamento

Principio di RCE #2

Per capire il codice scritto da qualcun altro, è necessario capire come funziona il proprio codice

- Il programma in esame è il risultato di un processo di compilazione di un linguaggio ad alto livello scritto da uno o più programmatori
- L'obiettivo del RCE non è la comprensione di ogni singola istruzione macchina
- Al contrario, è la comprensione del funzionamento del programma cosìcome i programmatori lo hanno concepito
- È essenziale saper programmare bene per essere in grado di riconoscere gli algoritmi che i programmatori hanno inteso implementare nel programma in esame

ad esempio, se un programma usa una tabella Hash, come posso capire la sua presenza se non so che cosa sia?

Principio di RCE #3

La qualità e la conoscenza degli strumenti di RCE determina la qualità dell'intero processo di reversing

- Le applicazioni informatiche moderne sono costituite da ingenti quantità di codice macchina
- Possedere e padroneggiare appropriati programmi di analisi semi-automatica del codice e del comportamento dei programmi è essenziale per la rapidità del RCE
- Gli strumenti di analisi sono inoltre importanti per facilitare il lavoro di decodifica delle informazioni presenti nei file eseguibili
- D'altra parte questi strumenti di analisi sono molto sofisticati ed il loro uso richiede molta pratica ed esperienza da parte dell'hacker

Principio di RCE #4

La chiave del reversing è l'abilità di identificare e comprendere gli schemi ricorrenti nel codice; di conseguenza, non esiste sostituto dell'esperienza!

- I compilatori traducono i costrutti ad alto livello dei linguaggi di programmazione in sequenze di istruzioni macchina più o meno riconoscibili
- D'altra parte i compilatori tendono ad ottimizzare il codice macchina in modi così diversi che risulta difficile scrivere programmi di decompilazione automatica che producono un sorgente ad alto livello facile da comprendere
- L'abilità principale dell'hacker esperto è quella di saper riconoscere gli schemi nascosti nel codice macchina
  - Riesce a riconoscere senza sforzo conscio le strutture a basso livello e si concentra sulle strutture ad alto livello
  - Tanto maggiore è la sua esperienza, tanto più rapido ed efficiente è il suo lavoro

Lavoro o forma d'arte

- Si afferma talvolta che la programmazione sia una *forma d'arte*, apprezzabile però soltanto dagli altri programmatori
- In effetti anche la disciplina del RCE può essere considerata una forma d'arte che richiede agli hacker buone capacità di
  - intuizione
  - ragionamento logico
  - problem solving
- Metodo, costanza, impegno e tempo permettono a chiunque di ottenere risultati nel campo del RCE
- D'altra parte, passione e doti personali permettono ai migliori hacker di ottenere notevoli risultati in tempi brevi
- Gli aspetti legati al pensiero creativo possono rendere l'attività di RCE gratificante e appassionante

Metodologia

- Quando si intraprende qualunque attività di RCE è necessario
  - avere ben chiari gli obiettivi che ci prefigge
  - stabilire in che modo operare per ottenere gli obiettivi
  - verificare continuamente la correttezza delle informazioni ottenute
- È importante perciò adottare una precisa metodologia nell'attività di reversing
  - meglio se formalizzata con la redazione di documentazione scritta

Gli approcci fondamentali del RCE

- Analisi "black-box", o Live analysis
  - Osserva il funzionamento del programma "dal vivo", senza esaminare la sua struttura interna
  - Utilizza strumenti di monitoraggio specifici del sistema operativo del programma target
  - Talvolta rischiosa o impossibile da effettuare
  - Non può fornire informazioni su porzioni di codice non attivate dall'esecuzione

    alcuni malware, se capisco di essere in run su una VM, possono comportarsi in modo diverso
- Analisi "white-box", o Dead listing analysis
  - Analizza il codice macchina codificato nel file eseguibile del programma in esame
  - Utilizza tipicamente strumenti di decodifica quali i disassemblatori e i decompilatori
  - Dispendiosa in termini di impegno e tempo richiesto

dispendiosa perchè disassemblo!

Gli approcci fondamentali del RCE (2)

- Analisi “gray-box”, o Mixed analysis
  - È costituita in linea di principio dall’utilizzo combinato dei metodi “black-box” e “white-box”
  - Osserva il funzionamento del programma eseguendo il suo codice macchina in modo controllato
  - Utilizza tipicamente strumenti quali i debugger applicativi e di sistema‘
  - In genere molto efficace, anche se esistono tecniche ad-hoc per ostacolarla

eseguo in maniera controllata porzioni di codice, mediante uso del debugger.

5/10/2023

Esempio di metodologia in nove passi

1. Descrizione preliminare
2. Formalizzazione dell’obiettivo
3. Ottenimento del codice macchina
4. Osservazione del funzionamento
5. Disassemblaggio del codice macchina
6. Localizzazione di un frammento assembly
7. Analisi del frammento assembly
8. Verifica dei risultati
9. Riepilogo delle informazioni ottenute

Metodologia: 1 — Descrizione preliminare

Il primo passo consiste nel descrivere ciò che si conosce del programma prima di iniziare il reversing

- provenienza
- ipotesi sulle finalità
- circostanze in cui è attivato
- file prodotti o modificati
- ecc.

L'obiettivo non può mai essere "dimmi vita, morte e miracoli di questo malware", bensì deve essere un qualcosa di più specifico.

Metodologia: 2 — Formalizzazione dell'obiettivo

Il secondo passo consiste nel formalizzare la finalità dell'attività di reversing

- identificare il meccanismo di protezione del programma
- ricostruire un algoritmo utilizzato per una certa funzionalità
- decodificare il formato di salvataggio dei dati su file
- ricostruire il funzionamento dell'intero programma
- ecc.

Questo passo è cruciale, in quanto:

- Aiuta a focalizzare l'attività di reversing, riducendo i rischi di perdere tempo per ricostruire meccanismi che non sono di interesse
- Permette di definire l'ambito del lavoro di reversing, e quindi anche i tempi ed i costi attesi per l'attività

Metodologia: 3 — Ottenimento del codice macchina

A volte è banale, ma bisogna essere sicuri che ciò che otteniamo sia il malware vero e proprio. Può infatti capitare che l'esemplare di malware non sia effettivamente quello, potrebbe anche essere cifrato, compresso, potrebbe autodistruggersi etc.

- Questo passo può essere immediato e banale
- Se però il file eseguibile è compresso e/o cifrato, come protezione contro l'attività di RCE, è necessario ottenere il codice macchina "in chiaro"
- Spesso l'offuscamento del codice macchina è realizzato da programmi commerciali appositi
- Talvolta sono disponibili anche strumenti che "de-offuscano" automaticamente il codice
- In altri casi è necessario fare il reversing del codice offuscato, ricostruire il funzionamento dell'offuscatore, ed ottenere poi il codice macchina in chiaro del programma di interesse

Metodologia: 4 — Osservazione del funzionamento

- Se possibile, si dovrebbe cercare di ottenere ogni possibile informazione osservando il funzionamento del programma con tecniche di analisi "black-box"
  - Quali librerie dinamiche utilizza
  - Quali chiamate di sistema vengono utilizzate
  - Quali file vengono acceduti
  - Quali chiavi di registro vengono accedute
  - ecc.
- In alcuni casi questo passo non potrà essere applicato:
  - Codice malware che potrebbe sfuggire al controllo
  - Mancanza della piattaforma hardware o del sistema operativo di supporto
  - Mancanza del codice completo dell'applicazione

Vogliamo "osservazioni in maniera controllata" tramite black box.
Ciò è possibile solo col programma completo.

Metodologia: 5 — Disassemblaggio del codice macchina

- Per cominciare l'analisi "white-box" è necessario disassemblare il codice macchina
- Questo lavoro è svolto da opportuni disassemblatori
- È in genere conveniente salvare il programma disassemblato in un documento che consenta di modificare liberamente il testo e aggiungere commenti
- È possibile che il disassemblatore non riesca a decodificare correttamente alcune porzioni di codice
  - Come stabilire se un certo byte è parte di una struttura di dati o di una istruzione macchina?
  - Nelle architetture CISC con istruzioni macchina a lunghezza variabile, come stabilire dove comincia una istruzione macchina?
  - I programmatori possono aggiungere sequenze di byte al solo scopo di trarre in inganno i disassemblatori

Disassemblo, ma comunque ci vuole l'aiuto umano, in quanto i disassemblatori possono essere tratti in inganno. Come? Un malware potrebbe capire di essere sotto un disassemblatore, e comportarsi diversamente.

Metodologia: 6 — Localizzazione di un frammento assembly

- Se il codice macchina è lungo, è necessario identificare una porzione di codice che sia interessante per perseguire l'obiettivo del reversing
- Esistono molte tecniche per facilitare questo compito
  - Con analisi di tipo "white-box", cercando occorrenze di funzioni di libreria o chiamate di sistema che potrebbero essere legate alla funzionalità cercata
  - Con analisi di tipo "gray-box", utilizzando un debugger per inserire opportuni breakpoint nel codice macchina del programma commenti
- Se l'obiettivo è la ricostruzione integrale del funzionamento del programma, si inizia con l'identificare l'entry point, ossia l'istruzione macchina dalla quale il programma inizia l'esecuzione

Metodologia: 7 — Analisi del frammento assembly

- Si analizza un frammento di programma in assembly per arrivare a comprendere
  - le strutture di dati utilizzate
  - le operazioni effettuate
  - le funzioni invocate
  - ecc.
- È essenziale annotare ogni informazione determinata in questa fase, anche se incompleta o ipotetica
- Se durante questo passo si scoprono altri frammenti assembly di potenziale interesse, si ricomincia la metodologia dai passi 4, 5 o 6, a seconda dei casi

Metodologia: 8 — Verifica dei risultati

- Si arriva a questo passo quando si ritiene di aver raggiunto la finalità predefinita dell'attività di reversing
- È essenziale cercare di verificare la correttezza delle conclusioni raggiunte con una procedura possibilmente differente da quella impiegata per il reversing, ad esempio:
  - Creare una patch che disattivi il meccanismo di protezione
  - Scrivere un programma esterno che intrepreti il formato dei dati su disco
  - Realizzare una applicazione che implementi il protocollo di comunicazione e che dialoghi con il programma in esame

Metodologia: 9 — Riepilogo delle informazioni ottenute

- Alla fine è necessario riepilogare in un documento scritto
  - tutto il processo di reversing
  - i risultati ottenuti
  - le informazioni collaterali apprese durante il processo
- È facile dimenticare i dettagli del lavoro di reversing, quindi questo passo deve essere completato immediatamente dopo la conclusione della attività