Limits of Static Analysis for Malware Detection

Roberto Fellin

Master of Science in Computer Science
University of Trento

14 Febbraio 2018

Contents

Introduzione

Obiettivi:

- fornire una tecnica di **offuscazione** su codice binario
- mostrare le limitazione della **static analysis**
- mostrare i vantaggi di detecotor basati sulla **semantica**

Tipi di detector:

- basati su syntactic signature: con database che classificano i malware
- basati su polymorphism e metamorphism: controllo sulla semantica

Offuscazione

idea generale:

- sostituire del codice con altro semanticamente uguale
- aggiungere istruzioni che non cambiano il comportamento

Opaque Constants

- sostituire il valore delle costatanti con delle istruzioni piú difficile da analizzare
- utilizzare queste costanti per rendere difficile il **control flow**, **indirizzo** delle variabili, **operazioni aritmetiche**

Simple Opaque Constants

Funzionamento:

- fare **xor** della costante con un valore arbitrario
- ripeterlo in un ciclo

Spiegazione:

- il valore della costatnte é diverso dopo ogni ciclo
- la static analysis ha difficoltá nel capire che path seguire

NP-Hard Opaque Constants

Funzionamento:

- utilizzare un problema NP-Hard (3SAT) per attribuire un valore alla costante
- nel nostro caso: se formula non soddisfacibile, costante a 1, altrimenti a 0

Spiegazione:

- il 3SAT problem é preparato affinché sia sempre non soddisfacibile(se la costante deve essere 1, é sufficiente invertire i valori)
- assumere che la formula sia sempre soddisfacibile é una grossa limitazione

NP-Hard Opaque Constants - Basic Block Chaining

- stessa procedura di 3SAT, solo che genero anche istanze soddisfacibili (in questo caso a runtime)
- la static analysis dovrebbe risolvere l'istanza del problema 3SAT

Control Flow Obfuscation

- offuscare il control flow, rimpiazzando le istruzioni di **jump** e **call** con istruzioni piú complesse
- per fare questo possiamo offuscare le costanti usate per il target adress
- possiamo anche offuscare nella stessa maniera il **return address**

Data Location and Usage Obfuscation

- possiamo utilizzare l'offuscazione sulla costante perché la **locazione** é specificata da una costante, da un indirizzo o da un offset
- utilizzare l'offuscazione sulla locazione per offuscare variabili locali e globali

Valutazione con virus scanner commericiali

- selezionare 3 malware: Klez.A, MyDoom.A, MyDoomAF
- testare i 3 worms sugli antivirus e verificare che tutti riescano a riconoscere il virus
- applicare l'offuscazione e ritestare i malware, seguono i risultati (X se non viene riconosciuto come malware):

	Klez.A	MyDoom.A	MyDoom.AF
McAfee		X	
Kaspersky	X	X	X
AntiVir			X
Ikarus	X	X	X

McAfee e AntiVir riescono a riconoscere il malware perché utilizzano signature sul codice e sono riusciti ad individuare una parte del codice come malware

Valutazione con malware detector avanzati (model checking)

- questi detector cercano code template che caratterizzano comportamente maligni
- seguendo la stessa procedura di prima, dopo l'offuscazione questo detecor non riconosceva i malware

Transformation Robustness

per verificare l'offuscazione, sono state offuscate varie applicazioni Linux e Windows binarie

- verifica stesso comportamento: a parte per qualche caso, il comportamento rimaneva lo stesso depo l'offuscazione
- dimensione: grosso aumento di dimensioni (es. per file di Lunux, incremento medio del 237%)
- performance: aumento del 50% circa

Conclusioni

- contromisure: é possibile riconoscere i malware dopo l'offuscazione perché contengono molti return istruction ma nessun call statements
- tuttavia si é dimostrato che toll che utilizzano soltanto tecniche statiche non riescono sempre ad individuare i malware
- sono necessarie procedure dinamiche che studiano il comportamento del malware