Malware Behavior Analysis

Gérard Wagener[1][2] Radu State[1][3] Alexandre Dulaunoy[2]

[1]MADYNES - LORIA
Laboratoire Lorrain de Recherche en Informatique et ses Applications

[2]CSRRT-LU
Computer Security Research and Response Team - Luxembourg

[3]INRIA

Institut National de Recherche en Informatique et Automatique

May 10, 2007

Outline

- Introduction
- Malware behavior
  - Similarities
  - Phylogenetic tree
- Experimental validation
- Conclusion & future work

Introduction

- Malware collectors gather a large amount of malwares ± 1000 malwares per month[1]
- A lot of malwares are unknown, on average 42% are not detected by 5 pieces of up-to-date anti-virus software[2]
- Anti-reverse engineering techniques
  - 18% of the malwares cannot be disassembled with objdump, a disassembler
  - According to PEiD a tool that identifies packers, 47% use an unknown packer / encryption technique
  - 15,4% cannot be debugged and not be emulated by the Norman sandbox
- Need for automated analysis & identification, classification of malwares → *phylogenetic tree* of malwares

---

[1] ASTRAnet

[2] Sample of 104 random chosen malwares

Related Work

- Malware Normalization [Christodorescu, 2005]
  - Revert code obfuscation by analyzing a malware's assembler code
- S.A.V.E. [A.H. Sung, 2004]
  - Use static analysis to extract function calls
  - Represent function calls as vector & compute distances
  - Generate signatures from vectors
- Behavioral Classification [T.Lee, 2006]
  - Define malware events and classify them
  - Partition clustering based on a modified LevenShtein distance
- Recognizing Self-Mutating Malware By Code Normalization And Control-flow Graph Analysis [Danilo Bruschi, 2006]
  - Malware detection is reduced to the sub-graph isomorphism problem

Malware behavior

- A malware is a software that uses OS function calls
- A sequence of called functions is a malware behavior
- Identify a malware's function calls in a list of executed functions
- The function call sequence is mapped on a numerical sequence

Example

LoadLibraryA	GetProcAddress	GetProcAddress	CreateFile
⇕	⇕	⇕	⇕
1	2	2	10

Malware behavior

- Let $\mathcal{F}$ be the set of functions called during execution
- Let $\mathcal{A}$ be the set of function calls that a malware M can perform
- $\mathcal{A} \subset \mathcal{F}$
- A function call $a \in \mathcal{A}$ is mapped to a code $c \in \mathcal{C} \subset \mathbb{N}$
- A malware behavior is a word: $a_1 a_2 a_3 \dots a_n \in \mathcal{A}^*$

Malware behavior

Determination of malware actions

- Virtual OS shows the function calls done by the malware and those by the OS itself
- Let D be the set of memory addresses $D \subset \mathbb{N}$
- Let L be the set of loaded libraries
- A library with its functions is loaded in memory $(L, \mathcal{I}, D)$ $\mathcal{I} \subset L \times D$

Property

Let $m \in D$,let $f \in \mathcal{F}$ $(f, m) \notin \mathcal{I} \Leftrightarrow f \in \mathcal{A}$

Malware behavior similarities

- A W32 malware is technically limited!
- Gathers system information i.e. LoadLibrary, GetProcAddress
- Becomes durable on the machine i.e. CreateFile, RegCreateKey
- Communicate i.e. *connect*, *send*, *recv*

→ imagine a similarity / distance function between malware behaviors and create a phylogenetic tree

Malware behavior similarities

Idea: Define a similarity or distance between two malware behaviors, based on edit distance matrix

- Let $S_{M_1} = a_1 a_2 a_3 \ldots a_m \in \mathcal{C}^*$
- Let $S_{M_2} = b_1 b_2 b_3 \ldots b_n \in \mathcal{C}^*$
- Map the sequences of called functions on the matrix R

$$\begin{array}{c|ccccc} & b_1 & b_2 & b_3 & b_j & b_n \\ \hline a_1 & & & & & \\ a_2 & & & & & \\ a_3 & & & & & \\ ai & & & & & \\ a_m & & & & & \end{array}$$

- Assigning scores

Malware behavior similarities

Assigning scores

- Matrix R
- $M_{ij} = \begin{cases} 1 & \text{if } a_i = b_j \\ 0 & \text{otherwise} \end{cases}$
- $R_{1j} = M_{1j},\ R_{i1} = M_{i1}$
- $R_{ij} = M_{ij} + \max\left(\max_{1\leq k\leq i-1} R_{k,j-1}, \max_{1\leq k\leq j-1} R_{i-1,k}\right)$

Similarity function

$$\sigma'(S_{M_1}, S_{M_2}) = 1 - \frac{2\cdot\max R_{ij}}{m+n}$$

Malware behavior similarities

Property of the similarity σ'

- Order of function calls influence the similarity
  - Parallelism during execution
  - Order is influenced by multi-threading
  - A malware may create more processes, processes may communicate IPC

- Create a similarity where the order is not important
- Extend malware behavior to a set E of called functions sequences
- E.g. $E = \{\underbrace{a_1 a_2 a_3 \ldots a_n}_{\text{done by thread 1}}, \underbrace{b_1 b_2 b_3 \cdots b_m}_{\text{done by thread 2}}, \ldots\}$

Malware behavior similarities

Usage of the Hellinger distance

- Consider the frequencies of called functions
- Create a contingency table
- Apply smoothing technique
- Compute Hellinger distance

$$d(a,b) = \left[\Sigma_i^N \left(\sqrt{a_i} - \sqrt{b_i}\right)^2\right]^{\frac{1}{2}}$$

Malware behavior similarities

Usage of the Hellinger distance

Example

	GetProcAddress	LoadLibrary	connect	CreateFile
M_1	$\frac{10(1-\varepsilon)}{17}$	$\frac{5(1-\varepsilon)}{17}$	$\frac{2(1-\varepsilon)}{17}$	$\frac{\varepsilon}{1}$
M_2	$\frac{2(1-\varepsilon)}{3}$	$\frac{\varepsilon}{2}$	$\frac{1(1-\varepsilon)}{3}$	$\frac{\varepsilon}{2}$
M_3	$\frac{5(1-\varepsilon)}{11}$	$\frac{5(1-\varepsilon)}{11}$	$\frac{1(1-\varepsilon)}{11}$	$\frac{\varepsilon}{1}$

Distance between M_1 and M_2

$$d(M_1, M_2) = \sqrt{\left(\sqrt{\frac{10(1-\varepsilon)}{17}} - \sqrt{\frac{2(1-\varepsilon)}{3}}\right)^2 + \left(\sqrt{\frac{5(1-\varepsilon)}{17}} - \sqrt{\frac{\varepsilon}{2}}\right)^2 + \dots}$$

Malware behavior similarities

Classification

- A malware behavior is compared with all other collected malware behaviors
- Determine average distance / similarity average σ of a malware ← classification
- Usage examples
  - A mutated malware has a high average similarity with its friends
  - Similar malware is regrouped in a family
  - A new malware type or a sandbox weakness can be identified with a low average similarity

Phylogenetic tree of malware behaviors

Motivation

- A common history of species is visualized by a phylogenetic tree
- Such a tree groups various families
- Malware behaviors are leaves
- Similarity / distance between nodes is represented by parents
- A malware needs to execute functions. Therefore a root exists
- Mutated malware is put in a group

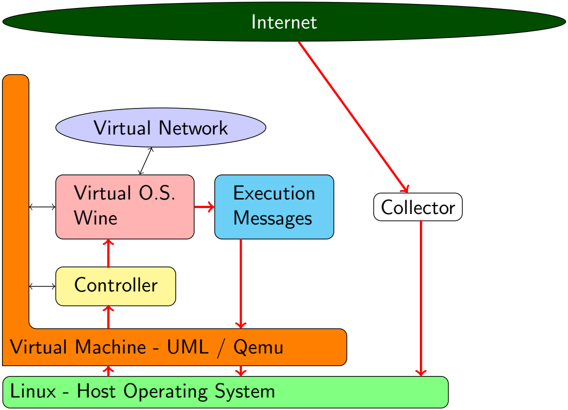
Phylogenetic tree of malware behaviors

How it works

- Hierarchical clustering
- Close malware behaviors are put in a group
- Groups are linked with their minimal distance
- The tree is continuously built until everything is linked

Experimental validation

Analyzing malware



There are vulnerabilities in Virtual Machines [Peter Ferrie]

Experimental validation

Results

Table: General information about the malware set

Number of malwares	104
Observation period	2005-2007
Malware from 2005	10
Malware from 2006	91
Malware from 2007	3
Average file size	135KB
Smallest file	8KB
Biggest file	665KB
Worms	34%
Not detected by anti-virus	42%[3]

---

[3] mean detection rate of Norman, Clamav, Antivir, Fprot & Bitdefender

Experimental validation

Classification results

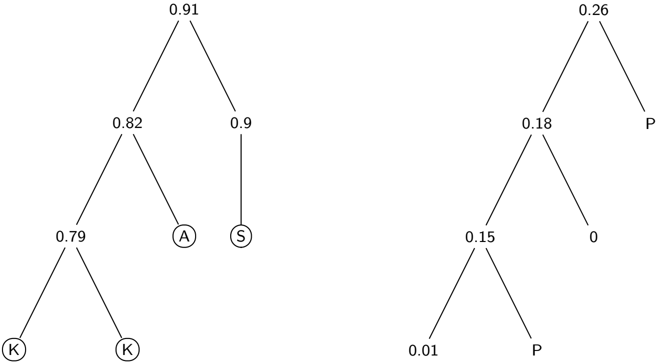
Table: Most Similar observed malwares *distance* $= 0$

WORM/Rbot.193536.29	WORM/Rbot.177664.5
Worm/Sdbot.1234944.1	Backdoor-Server/Agent.aew
Worm/Sdbot.1234944.1	unknown
Worm/IRCBot.AZ.393	Worm/Rbot.140288.8
Backdoor-Server/Agent.N.1	Worm/Win32.Doomber
Trojan.Gobot-4	Trojan.Gobot.R
Trojan/Dldr.Agent.CY.3	W32/Virut.A virus
Trojan.Gobot-4	Trojan.Downloader.Delf-35
Trojan.Mybot-5011	Trojan.IRCBot-121
Trojan.Mybot-5079	Trojan.EggDrop-5

Experimental validation

Phylogenetic tree results

`http://nepenthes.csrrt.org:10080/malware_behaviour/cache/`



K	Kernel Family
A	API Family
S	Sandbox weaknesses

P W32/Pinfi.A (Norman name)

Future work

- Evaluate other potential distances between malware behaviors
- Weighting function calls
- Analyze a set of malwares of several giga bytes
- Build other trees, evaluate other potential distances
- Recover the control flow graph of a malware with fault injection techniques
- Improve execution heuristics

Conclusion

- Function calls of unknown W32 binaries are extracted with free available tools by executing the binary
- It was proposed a simple malware behavior model with its implementation
    - Sequence of function calls = malware behavior
- Such sequences can be compared and a classification is done
- Unknown malware behaviors can be detected
- A phylogenetic tree of malwares was proposed
- A lot of things remain to do . . .

Questions & Answers

Thank you for listening!

- http://madynes.loria.fr
- http://www.csrrt.org
- gerard <dot> wagener <at> gmail <dot> com

Bibliography

Danilo Bruschi,Lorenzo Martignoni,Mattia Monga
Recognizing self-mutating malware by code normalization and control-flow graph analysis. Proceedings of the International Symposium of Secure Software Engineering, Arlington, VA, U.S.A., 2006.

Mihai Christodorescu and Johannes Kinder and Somesh Jha and Stefan Katzenbeisser and Helmut Veith
Malware Normalization, Technical Report 1539 University of Wisconsin, Madison,WI,USA Nov 2005.

A.H. Sung and J. Xu and P. Chavez and S. Mukkamala
Static Analyzer of Vicious Executables (SAVE), In Proceedings of ACSAC, 2004.

T. Lee,J.Mody
Behavioral Classification, EICAR,May 2006.

Peter Ferrie
Attacks on Virtual Machine Emulators,
http://www.symantec.com/avcenter/reference/Virtual_Machine_Threats.pdf