Universitatea
Politehnica
București

Facultatea de
Automatică și
Calculatoare

Catedra de
Calculatoare

Malspec: Malicious Application Analysis

Bachelor Thesis, July 2013

Autor(i)

Cristian Condurache

Conducător științific

As.dr.ing. Laura Gheorghe

Motivation & Goals

- Why?
  - Popularity of Linux based OS
  - Use in embedded systems
- How?
  - Malicious behavior pattern mining

Malware Detection

- Signature-based
  - Problem: fails to detect new malware, obfuscation
- Behavior-based
  - Problem: behavior patterns require manual identification

Malspec Mining Algorithm

- Input: a malware sample and a set of benign programs
- Output: a malicious behavior pattern
- Creates a graph for each program
  - A node represents a system call
  - An edge is an argument dependency
- Computes malware specifications as “difference” between graphs
  - Maximal common subgraph algorithm
  - Complement graph
  - Minimal transversal

Dependency Graph

- Initial nodes

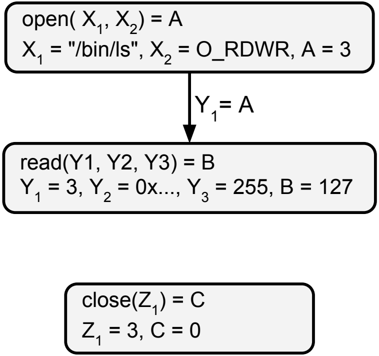
open(X_1, X_2) = A
X_1 = "/bin/ls", X_2 = O_RDWR, A = 3

read(Y1, Y2, Y3) = B
Y_1 = 3, Y_2 = 0x..., Y_3 = 255, B = 127

close(Z_1) = C
Z_1 = 3, C = 0

Dependency Graph

- Adding dependency edge between open and read



open(X_1, X_2) = A
X_1 = "/bin/ls", X_2 = O_RDWR, A = 3

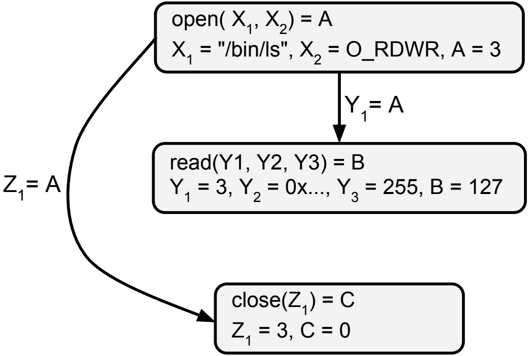
Y_1= A

read(Y1, Y2, Y3) = B
Y_1 = 3, Y_2 = 0x..., Y_3 = 255, B = 127

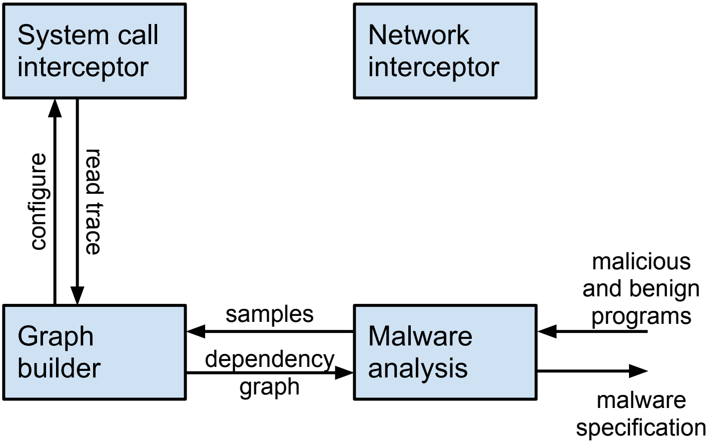
close(Z_1) = C
Z_1 = 3, C = 0

Dependency Graph

- Adding dependency edge between open and close



Malsharp Architecture



Implementation (1)

- System Call Interceptor Driver (SCID)
  - Logs execution trace for a process
  - Kernel module, registers by using miscdevice
  - Controlled via the ioctl system call
- Network Interceptor (NI)
  - Uses netfilter hooks to monitor traffic
  - Can be configured to monitor specific protocols
  - Statistics can be read from /proc/interceptor

Implementation (2)

- Graph Builder
  - Runs each program
  - Reads execution traces from SCID
  - Finds argument dependencies
- Malware Analysis
  - Uses the graph builder for each program
  - Applies the malspec mining algorithm

Evaluation (1)

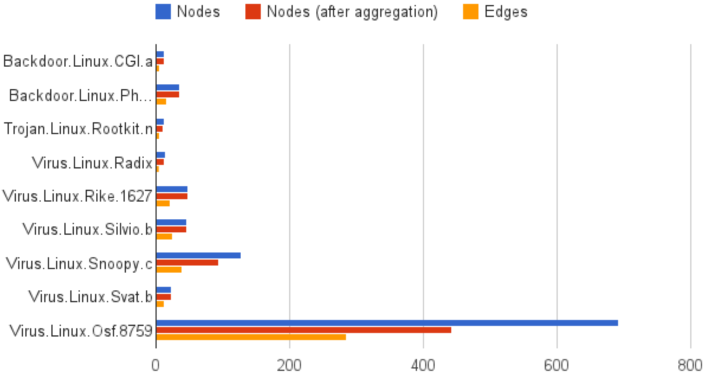
- Virtual machine, snapshots
- Revert to snapshot before each test
- Bridged network access
- A set of known malware samples
  - Viruses: Virus.Linux.Rike.1627, Virus.Linux.Osf.8759
  - Backdoor: Backdoor.Linux.CGI, Backdoor.Linux.Phobi.1

Evaluation (2)

- Execution traces and graphs successfully built
- Small malware patterns identified, 3-5 nodes
- Node aggregation reduced total number of nodes in large graphs by 25-30%

Evaluation (3)

- Node aggregation results



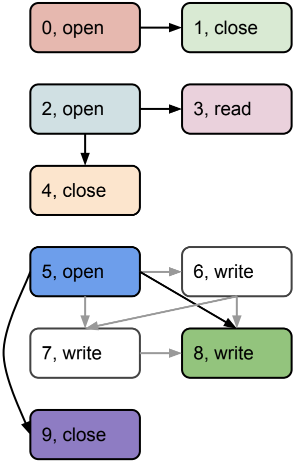
Conclusions

- Proof of concept for a Linux malware behavior miner
- Node aggregation successfully reduced total number of nodes
- Possible future improvements:
  - Additional pruning: node ordering strategies
  - Adding other types of dependency edges

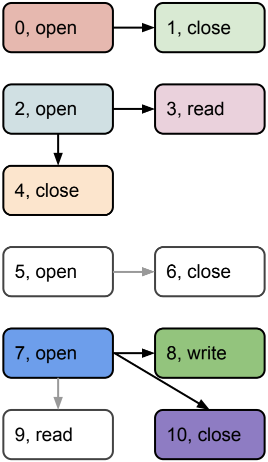
Test programs

	program_test	diff_test
1	open(...) = fd1	open(...) = fd1
2	close(fd1)	close(fd1)
3	open(...) = fd2	open(...) = fd2
4	read(fd2, ...)	read(fd2, ...)
5	close(fd2)	close(fd2)
6	open(...) = fd3	open(...) = fd3
7	write(fd3, ...)	close(fd3)
8	write(fd3, ...)	open(...) = fd4
9	write(fd3, ...)	write(fd4, ...)
10	close(fd3)	read(fd4, ...)
11	–	close(fd4)

Maximal common edge set

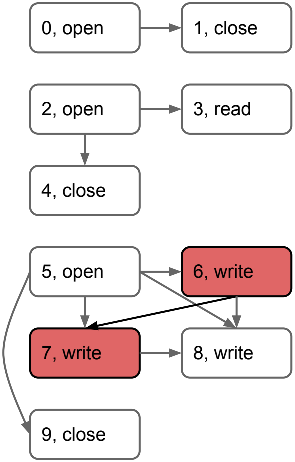


(a)

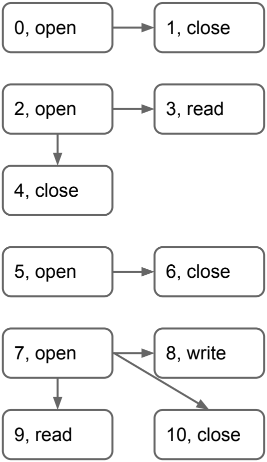


(b)

Complement and minimal transversal



(a)



(b)