

Technische Universität München



Lehrstuhl für Angewandte Softwaretechnik

Applied Software Engineering



Experimental Methods in Software Engineering
Intrusion Detection

Andreas Löhr

Virus Detection (Pattern Matching)

Seminar in überfachlichen Grundlagen WS1999/2000

Motivation

- viruses pose an increasing risk to computer data integrity
- viruses can cause loss of valuable data
- enormous costs for recreation or restoration
- number of viruses increases threat increases

- viruses are a possible consequence of an intrusion

www.jessen.informatik.tu-muenchen.de/~fischerv/skripten/

Sicherheit in Netzen (Internetsicherheit)

... even if you don't lose your data, you lose your trust ...

Overview

- Discussion of Kumar's and Spafford's paper
 - what is a computer virus
 - several detection methods
 - short description of the generic virus scanner

- Related work
 - paper: „The Behavioral Solution“ (Sung Moo Yang)
 - „Integrity Master“, a commercially available tool
 - Reviews of Integrity Master

A Generic Virus Scanner in C++

Technical Report CSD-TR-92-062

Sandeep Kumar

Eugene H. Spafford

The COAST Project
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907-1398
{kumar, spaf}@cs.purdue.edu

17 September 1992

What is a computer virus ?

- common definition: *„[...] is a segment of machine code (DNA) that installs itself into one or more larger host programs (cells) [...] and enables further spread of itself and/or destruction of data [...] or does other damage to a system [...] when the host program is executed [...]“*
- **code must be executed for being destructive**
- different kinds of viruses:
 - system sector virus
 - file virus
 - nowadays: macro virus
 - (- virus hoax)

Virus detection - by behavioral abnormality

- program monitors the activities of normal day-to-day use
- program monitors known methods of virus activity
- most effective: big difference betw. day-to-day use and virus activity

- fails, if virus stays within the „normal“ behavior
- high sensitivity creates many „false positives“

- works for all viruses (already known and future ones)
- can avoid infections before they occur

Virus detection - by emulation

- program under test is emulated with certain inputs, not executed
- program is suspicious, when doing some virus-like activities
- difficult to determine suitable sample inputs
- never „precise“
- works for all viruses (already known and future ones)
- can avoid infections before they occur
- example: VProt (MS-DOS) has this as an option

Virus detection - by static analysis / policy adherence

- construction of a security policy (regular expression)
- construction of a *minispec* of a program (reg. exp., mirrors the behavior)
- decision, whether program fits a policy or not

- undecidable, whether program contains virus or not
- minispec must be generated from the source code

- could work very fast (test if one reg.exp. is subset of another)

- no program like this existent (1992)

Virus detection - by checksumming

- content based (encrypted) checksum stored within/outside the program
- before executing: recomputation and comparison of checksum
- system support (hardware) for automatic check-and-execution required
- no detection in already infected files
- failure with „ambiguity viruses“ and „stealth viruses“
- works for all kinds of unwanted program modifications
- example: Integrity Master uses checksums for change detection

Virus detection - by runtime program integrity checking

- computing checksum for predefined granules of a program
- recomputing and matching of checksum when control flow enters granule
- system support (hardware) for automatic check-and-execution required
- no detection in already infected files
- detection of infections between integrity check and execution
- ensuring integrity during runtime
- no program working like this known (1992)

Virus detection - by timestamp modification

- time of last modification serves as a checksum (is kept external)
- matching timestamps in regular intervals
- virus could get access to the timestamps
- timestamp operation must be irreversible by **any** process

Virus detection - by signature scanning

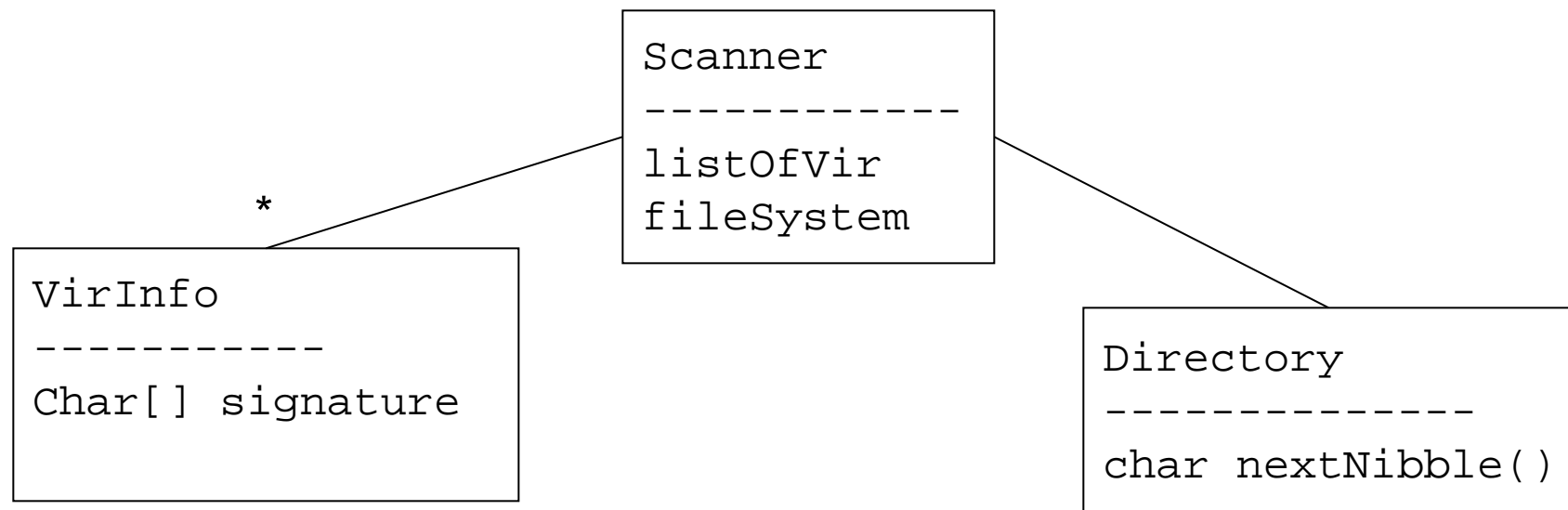
- executable code of virus is source of a signature
- files containing this signature are flagged as being infected

- signature extraction is difficult and time-consuming
- assumption: virus does not alter its code arbitrarily, like polymorphic viruses
- works **only** for already known viruses
- signatures to test increase with the number of known viruses
chance of matching some legal code increases

- most cost-/time-effective approach
- can detect trojan horses, logic bombs etc.
- only technique, where virus is still inactive on the disk (!)

Short description of the generic virus scanner

- object orientated programming (C++)
- major classes: `VirInfo`, `Directory`



- a prototype implementation was done in 1992

Prototype implementation results

- test machine: Sun SPARC running SunOS 4.1.1
- test conditions: - 1197 infected MS-DOS files, 12.3 MB of disk space
- two different signature files

No. of sigs	CPU time (usr + sys)	rate
1	17s	740.9 KB/s
(TBSCAN) 347	10m 52s	19.3 KB/s
(VB) 317	4m 35s	45.8 KB/s

The Behavioral Solution

Copyright 1991, 92, 96 Sung Moo Yang

First published 1991

Second Edition published 1992

Third Edition (incomplete, last updated on Jan 31, 97)

Problem with conventional virus scanners

- number of unknown viruses increases in proportion of time
knowledge of scanner decreases
- reason: growing by modification and creation
- solution: set of definitions and rules of virus-behavior on
imaginary storage

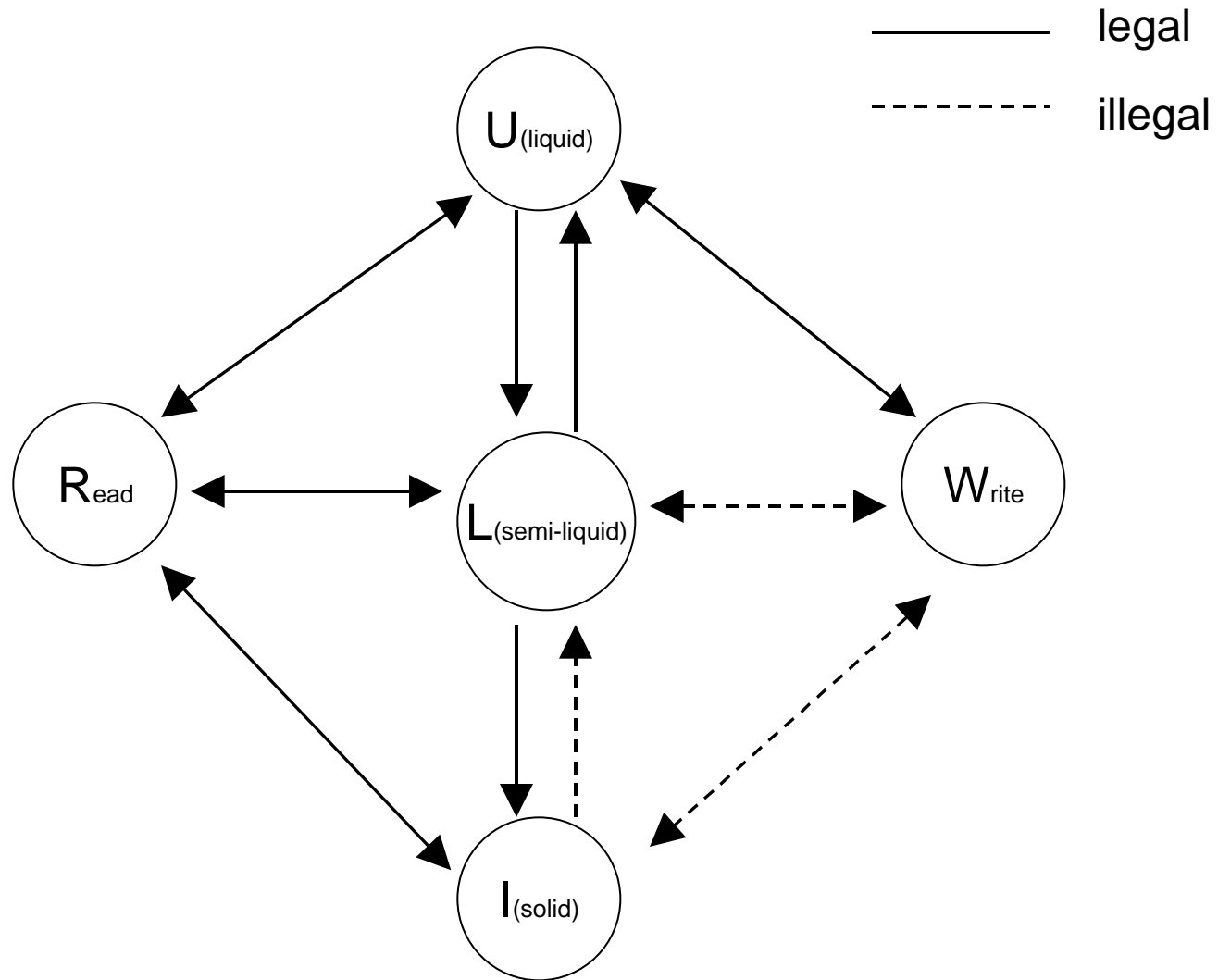
Imaginary storage model

- a file consists of several elements (name, size, mode, timestamp, block, etc.)
- contents of elements (data) can have different states:
 - solid: data *and* state is *unchangeable*
 - semi-liquid: data is *unchangeable*, while state is *changeable*
 - liquid: data *and* state are *changeable*
- all operations to a file (write, read, delete, append, etc.) are considered as a transition of elements within an imaginary storage medium

Imaginary storage model (ctd.)

- different regions within an imaginary storage medium, in which elements can reside:
 - region U: all elements that are *liquid*
 - region L: all elements that are *semi-liquid*
 - region I: all elements that are *solid*
 - region W: element is being *write*-accessed
 - region R: element is being *read*-accessed
- elements move, when force is applied to them
- elements within W, R tend to go back to previous region

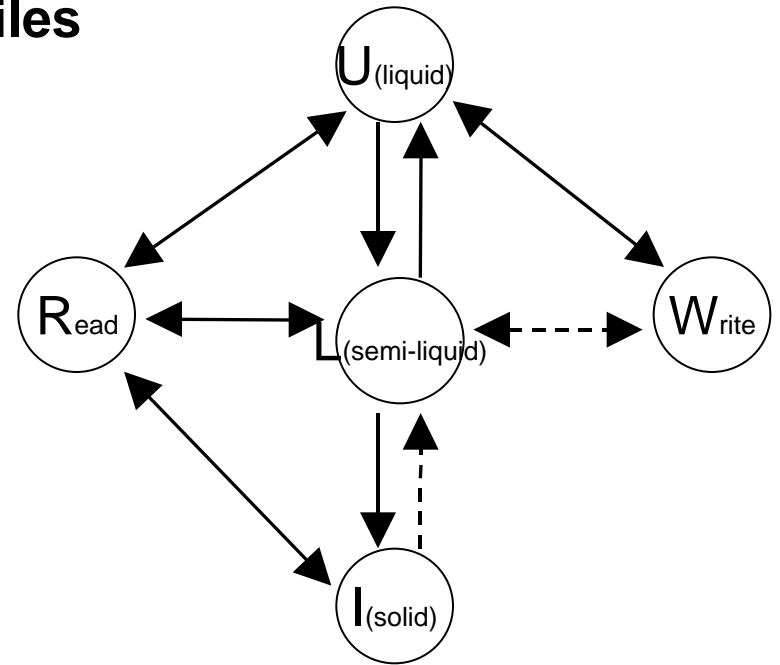
Imaginary storage model - graph



Imaginary storage model - example for files

	name, size, mode, block	

a read-write file



	name, size, mode	
	block	

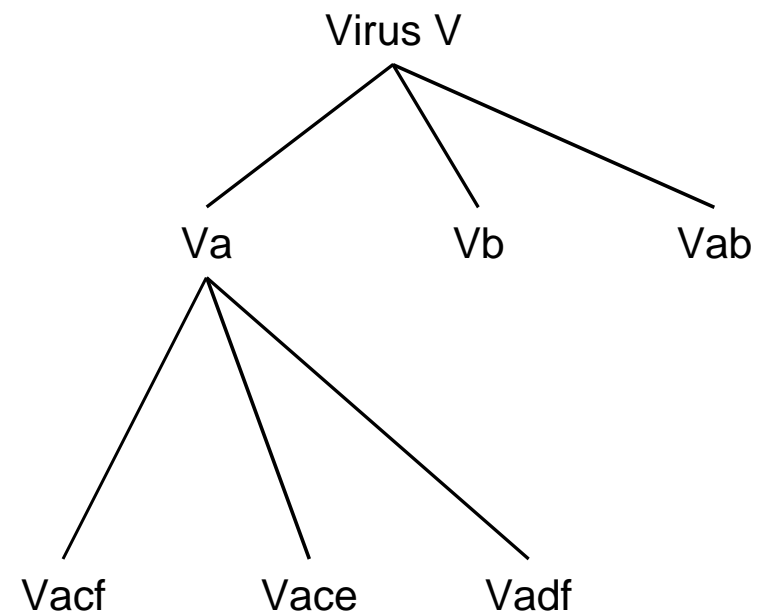
a read-only file

	name, size, mode, block	

a read-only file on a CD-ROM

Classes of virus behavior

Category	Behavior	Symbol
Growing	GBM	a
	GBC	b
Modifying	mod. of contents	c
	mod. of links	d
Residing	on non file elemt.	e
	on file element	f



Example: Virus “FORM”, modifies bootstrap-loader and resides on non-file element (virus code in last cluster of HDD) Vace

Detection of viruses

- by matching certain patterns of element transitions
- attempt of using illegal paths

Results, conclusions and criticism

- classes V_a and V_{ab} are detectable
- a non-activated virus is **not** detected
- comparison behavioral solution (A) fixed knowledge system (B)

13 months runtime: 24 unknown viruses for A

330 unknown viruses for B

- paper has too much formalism and definitions
- “japanese / chinese” English



Don't settle for just virus protection Integrity Master protects you against all threats!

a commercial tool from Stiller Research, www.stiller.com

- signature scanning for known viruses
- data integrity scans (by checksumming)
- system sector and CMOS integrity scans
- detection of file corruption (e.g. date in future)
- ...

signature scanning and change detection

Review of IM by Robert M. Slade and by Virus Bulletin

Robert M. Slade

- “It certainly has the best explanations of the antiviral process and the options for security of any installation program.”
- “Both the change detection and scanner components of the product are mature and stable.”
- “A minimum of 260K memory and DOS 2.x or higher is required. Refreshingly, a hard disk is not.”
- “Some new viri were detected on the basis of similarity to known code. “

Virus Bulletin, DOS Scanner Comp. 1996

IM Detection Results

ItW Boot 93.5%
Standard 97.3%
ItW File 96.8%
Polymorphic 46.9%
ItW Overall 95.4%
False positives: 40

Cybec VET v9.0 Results

ItW Boot 100.0%
Standard 97.3%
ItW File 98.0%
Polymorphic 81.6%
ItW Overall 98.9%

Outcome

- never depend on just one signature scanner system (Maltese Amoeba, 01.11.1991)
- never depend just on a behavioral method (no offline scanning)

References

- www.virusbtn.com
- www.stiller.com
- Sung Moo Yang, *The Behavioral Solution*, 31. Jan 1997,
- Sandeep Kumar and Eugene H. Spafford, *A Generic Virus Scanner in C++*,
The COAST Project, Dep. of Computer Sciences, Purdue University, West Lafayette, 17.9.1992
- Robert M. Slade, *Antiviral Protection Comparison Reviews*, 1995