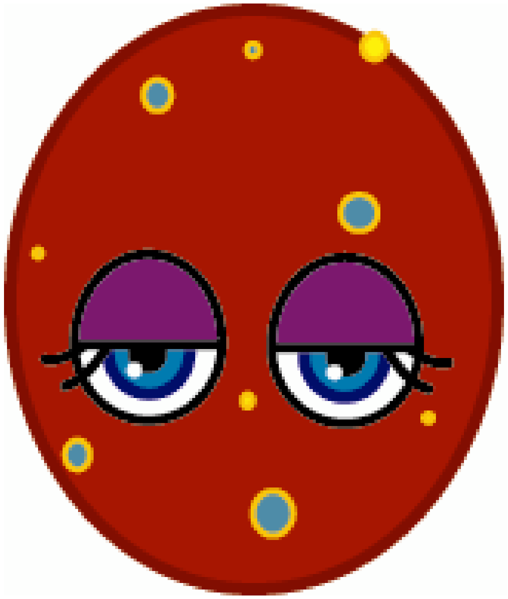
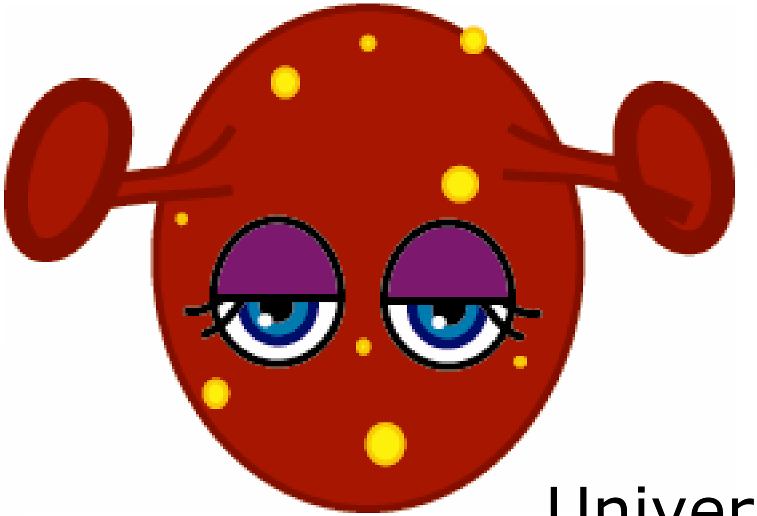
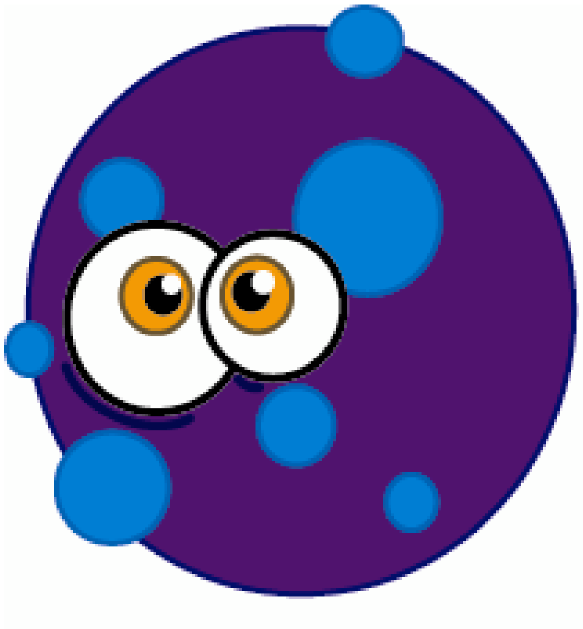
Disk-Level Behavioral Virus Detection

5 March 2007

North Carolina
State University







David Evans
University of Virginia

work with Nathanael Paul,
Adrienne Felt,
and Sudhanva Gurumurthi
http://www.cs.virginia.edu/malware

David Smith
“Melissa” 1999

Michael Buen

Onel de Guzman

“ILoveYou” Worm, 2000

Stereotypical Malwarist, circa 2000

“ILoveYou” Worm Code

```
rem  barok -loveletter(vbe) <i hate go to school>
rem by: spyder  /  ispyder@mail.com  /
        @GRAMMERSoft Group  / Manila,Philippines
…
x=1
for ctrentries=1 to a.AddressEntries.Count
   set male=out.CreateItem(0)
   male.Recipients.Add(a.AddressEntries(x))
   male.Body = “kindly check the attached LOVELETTER …”
   male.Attachments.Add(dirsystem
                    &“\LOVE-LETTER-FOR-YOU.TXT.vbs”)
   male.Send
   x=x+1
next
```

Thoughtful message

Hid location

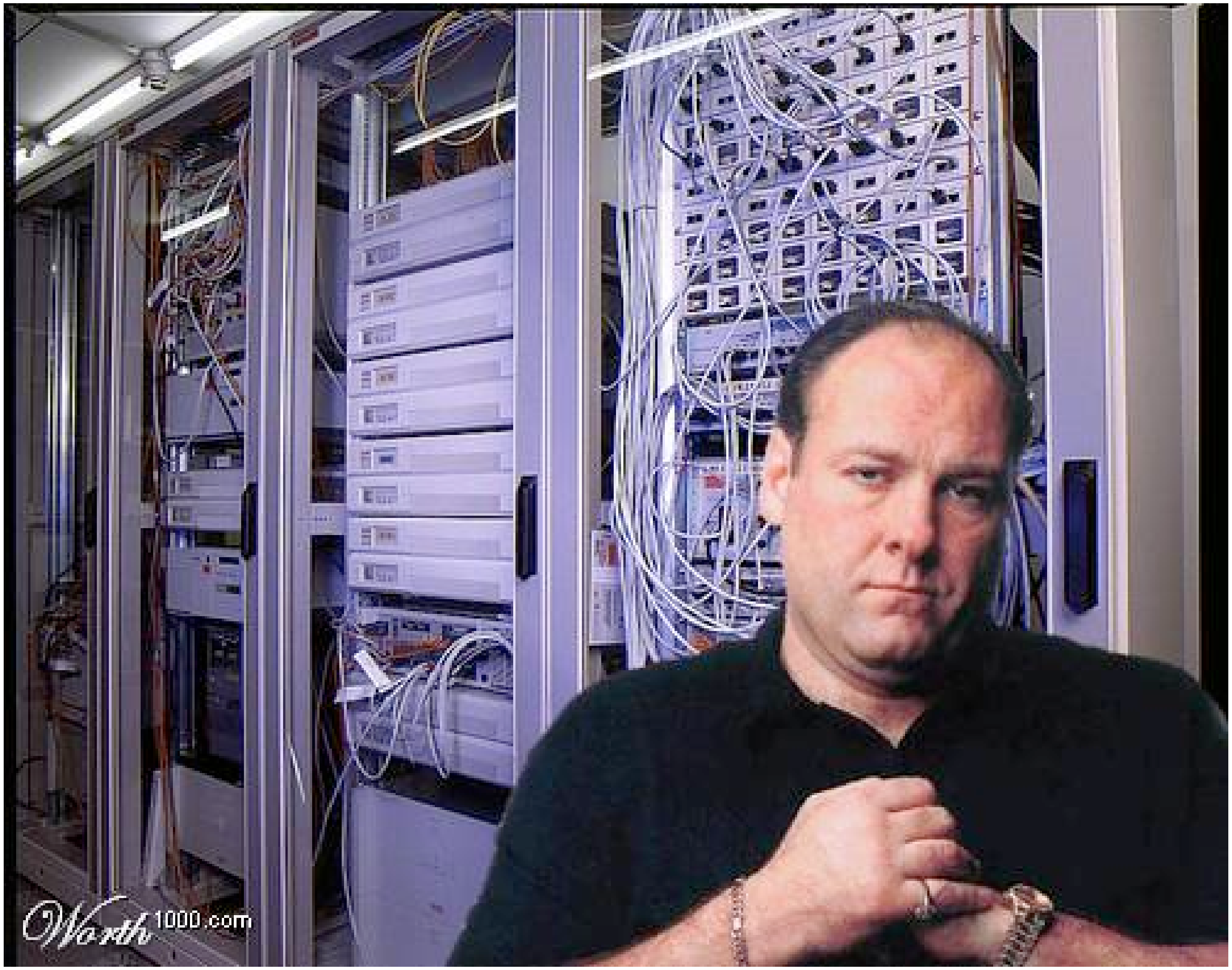
Creative speller

Good understanding of for loops

Detecting "ILoveYou"

file.contains("@GRAMMERSoft Group")

- Signature Scanning
  - Database of strings that are found in known viruses
  - A/V scanner examines opened files (on-access) or stored files (on-demand) for that string



Picture by Tobic, http://www.worth1000.com/emailthis.asp?entry=31033

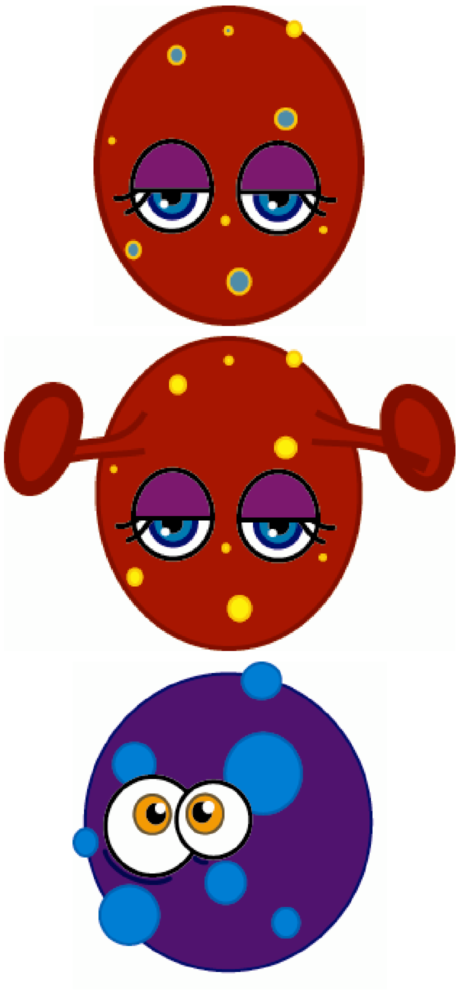
Stereotypical Malwarist, 2007

The Organized Malware Industry

- Multi-million dollar industry
- Vulnerability black market
  - Zero-day exploits sell for ~$4000
- Virus "professionals"
  - Sell viruses, or use them to build botnets and rent spamming/phishing service
- See Peter Guttman's talk

Bad news for society, but great news for security researchers!

W32/Efish.A



- Multi-threaded, stealthy, parasitic
- Self-encrypted: each infection is encrypted with a new key
  - No static strings to match except decryption code
- Slow polymorphic: the decryption code is modified with each infection
  - Slow changes make it harder to develop and test signatures

De-Polymorphers

[Kaspersky's "Skeleton Detection"]
[Christodorescu, Jha, + 2005, 2007]

- Reverse polymorphic transformations
- In theory, obfuscation is impossible (for some functions) [Barak+ 2001], so "con-fuscators" must be
- In practice:
  - Con-fuscation is much harder than obfuscation
  - Con-fuscators are too slow
  - Virus obfuscators don't need to be general or semantics-preserving

Emulators

- Emulate virus until it decrypts itself
- In theory, it should be possible to build a perfect emulator
- In practice, emulators are imperfect:
  - Programs can determine if they are running in an emulator
  - Several viruses exhibit anti-emulation techniques [Stepan06, Ciubotariu06]
  - Performance concerns mean emulator can only run for beginning of execution

Circumvention

- A/V software runs on the host OS
- Malware can get below host: avoid or tamper with detection
- SubVirt [Samuel King & Peter Chen, Oakland 2006]
- BluePill [Joanna Rutkowska, Black Hat 2006]

Summary: Traditional Detection is Doomed

Its not an arms race, it’s a bludgeoning: current approach will always be playing catch-up in the arms race between virus authors and detectors

- **Reactive:** signatures only detect known viruses
- **Static:** code is easy to change and hard to analyze
- **Circumventable:** malware can get below the detector

Our Target: File-Infecting Viruses

- Spread by infecting executable files
- Includes complex, stealthy, polymorphic viruses
- Does not include all malware:
  - Memory-Resident (spread by infecting processes in memory)
  - Network Worms (spread without infecting executable files)
  - Rootkits, spyware, etc. (don’t spread)

Ideal Solution

Today's Talk

- Detect viruses:
  - At a level malware can't compromise
  - Without disrupting non-malicious applications
  - Without (overly) impacting performance
- Recognize the **fundamental behavior** of viruses, instead of relying on blacklists of known viruses

- Recover from infections seamlessly

Semi-Obvious Riddle

What is:

- Available on almost every computer
- Able to see all disk activity
- And has processing power and memory comparable to ~2000 Apple II's?

The disk processor.

200MHz ARM Processor, 16-32MB Cache

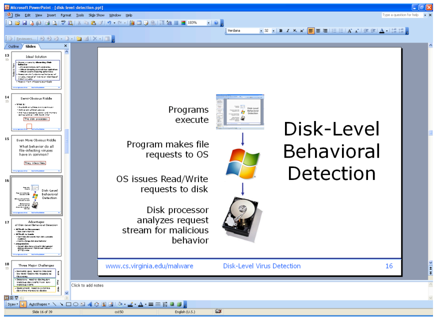
Even More Obvious Riddle

What behavior do all file-infecting viruses have in common?

They infect files.

Disk-Level Behavioral Detection

Executing Program



Programs execute

Program makes file requests to OS

OS issues Read/Write requests to disk

Disk processor analyzes request stream for malicious behavior

Disk-Level Behavioral Detection

www.cs.virginia.edu/malware Disk-Level Virus Detection 16

Program makes file requests to OS

Operating System

OS issues Read/Write requests to disk

Disk processor analyzes request stream for malicious behavior

Advantages of Disk-Level Behavioral Detection

- **Difficult to Circumvent**
  - Runs *below* host OS
- **Difficult to Evade**
  - Can’t hide disk events from disk: complete mediation
  - Hard to change disk-level behavior
- **Inexpensive**
  - Current disks have a (mostly idle) general purpose processor
  - Typical seek request ~ 700,000 cycles

Three Major Challenges

- Semantic gap: need to interpret low-level read/write requests as file events

Next

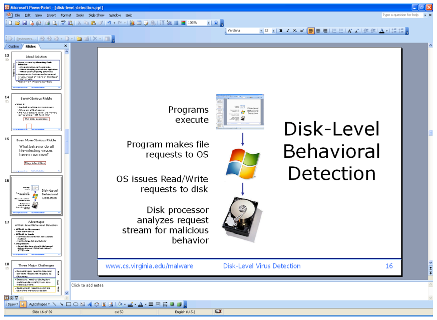
- Detectors: need to distinguish malicious disk traffic from non-malicious traffic

Most

- Deployment: need to convince disk drive makers to deploy

Help!

The Semantic Gap


Microsoft PowerPoint - [disk-level-detection.ppt]

Programs execute

Program makes file requests to OS

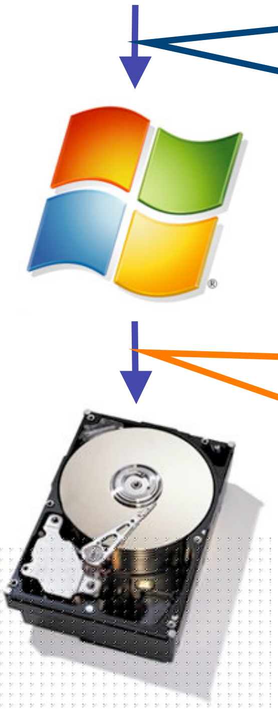
OS issues Read/Write requests to disk

Disk processor analyzes request stream for malicious behavior

Disk-Level Behavioral Detection

www.cs.virginia.edu/malware Disk-Level Virus Detection 16

Slide 16 of 39



READ file="\system32\system.ini"
offset=0
WRITE file="\system32\system.ini"
offset=0 data="ïA]¤."

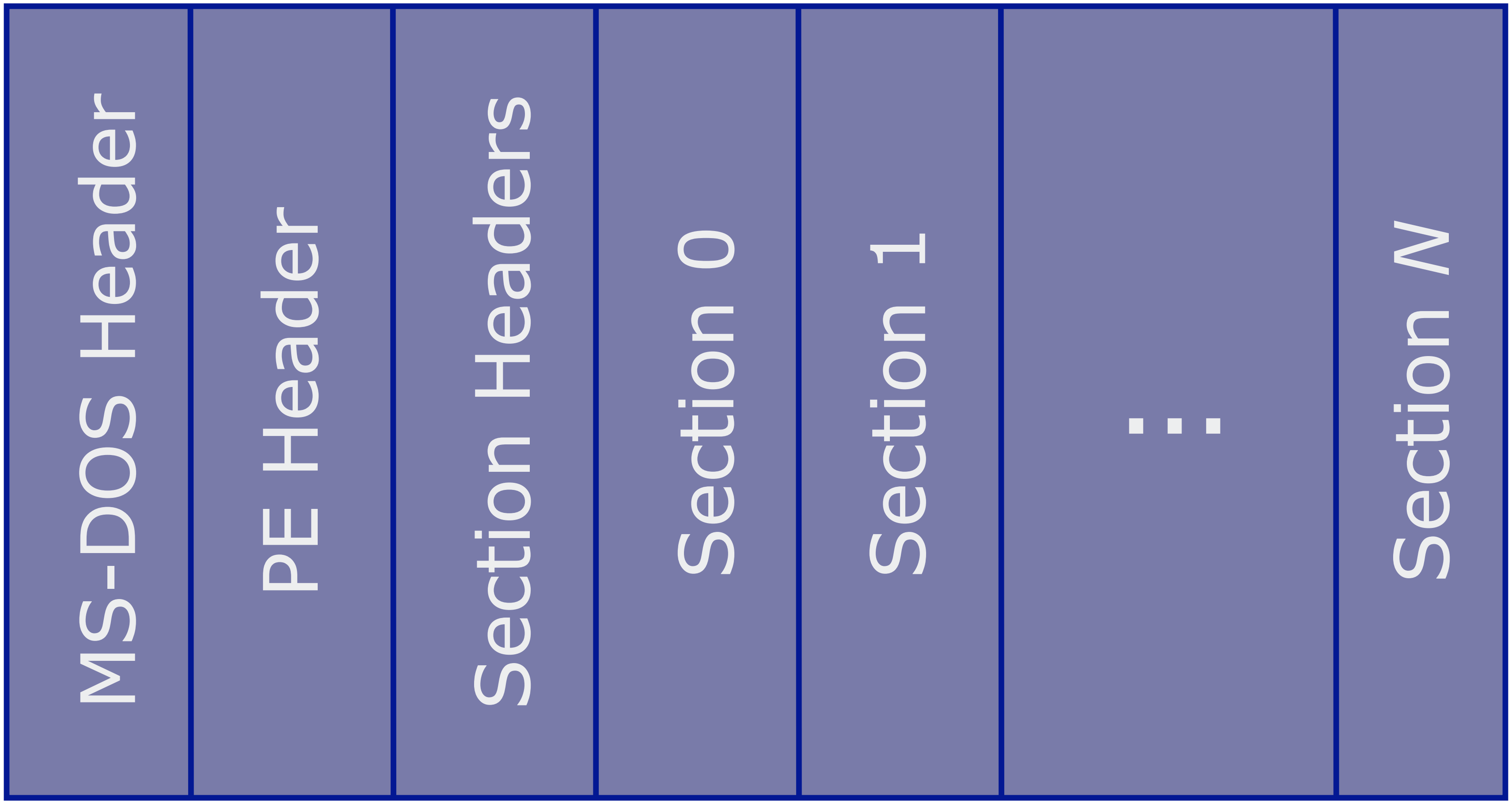
READ block=2995263
len=4096
WRITE block=2995263
len=4096 data="ïA]¤. "

Bridging the Gap

- Object-based Storage (OSD)
- Semantic Disks [Sivathanu+ 2003, Arpaci-Dusseau+ 2006, Sivanthanu+ 2006]
- Our Solution (for now):
  - Prototype collects traces at OS level
  - Detector sees only what would be visible to a semantically-smart disk
  - In progress: implementing at lower level

Developing Detectors

Next: a generic file-infection detector
After: virus-specific signatures



MS-DOS Header

PE Header

Section Headers

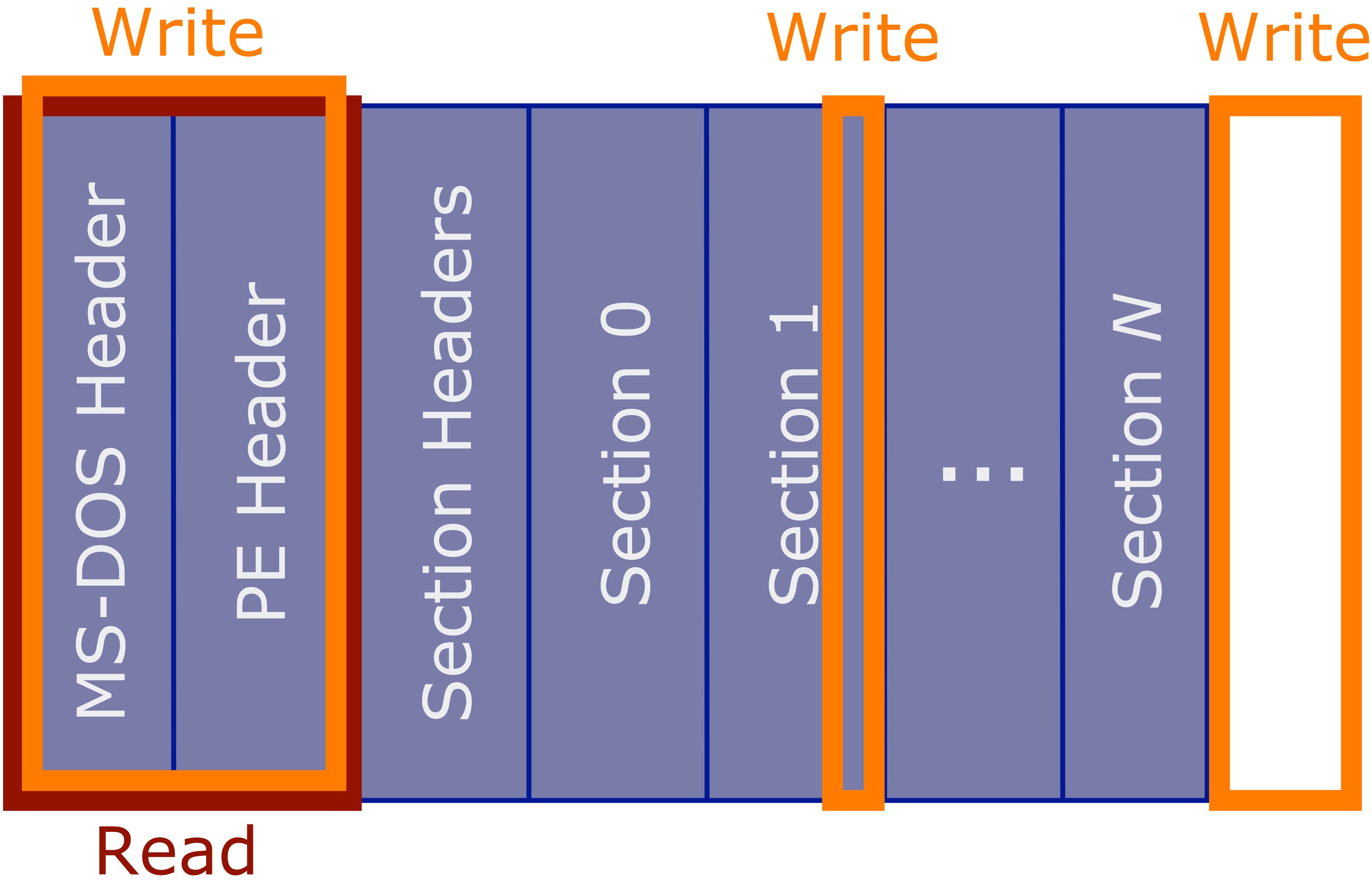
Section 0

Section 1

…

Section *N*

Windows PE File



Infecting a Windows PE File

First Generic Infection Rule

read [*name*@offset:0,
read [*name*@offset:∗]+;
write [*name*@offset:0],
write [*name*@offset:∗]+

Multi-Read/Write Rule

,-separated events in any order

;-separated groups are ordered

name is an executable file (starts with MZ or ZM)

Additional Infection Rules

Single-Read/Write Rule:

```
    read [name@0];
    write [name@0]
```

Reading and writing the file header.

Single-Write Rule:

```
    create [name];
    write [name@0]
```

Any write to an existing executable file.

Evaluation: Detection

- Five selected viruses
  - Detnat, Efish, Ganda, Simile, Tuareg
- Randomly selected 70 samples from http://www.offensivecomputing.net
  - Classified as “virus” by at least one A/V vendor
- Eliminated those that didn’t run
  - Depended on Windows version, crashed, etc.
- 28 samples remained
- Executed viruses, collected disk traces, checked against rules

Virus	Multi R/W	Single R/W	Single Write		Virus	Multi R/W	Single R/W	Single Write
Alcaul.o	✓	✓	✓		Magic.1590	✓	✓	✓
Aliser.7825	◉	✓	✓		Matrix.750	✓	✓	✓
Aula.a	*— Not a virus —*				Maya.4108	✓	✓	✓
Billrus.a	*— Not a virus —*				NWU	*— Not a virus —*		
Chiton.b	✓	✓	✓		Oblion.a	*— Not a virus —*		
Detnat	✓	✓	✓		Oroch.5420	✓	✓	✓
Efish	◉	◆	◆		**Parite.b**	◆	◆	◆
Eletiamo	*— Not a virus —*				Resur.f	✓	✓	✓
Enerlam.b	✓	✓	✓		**Sality.l**	◆	◆	◆
Evyl	◉	✓	✓		Savior.1832	✓	✓	✓
Ganda	✓	✓	✓		Seppuku.2764	✓	✓	✓
Harrier	✓	✓	✓		Simile	✓	✓	✓
Jetto	**C**	**C**	✓		Stupid.b	*— Not a virus —*		
Kriz	*— Not a virus —*				Tuareg	✓	✓	✓

✓ Matched all infections before any damage

◆ Matches, but after malicious activity

◉ Matches most infections of virus

C Not matched because of caching

Evaluation: Non-Disruption

- Disk tracer implemented as a mini-filter file system driver: collects a sample of disk traffic every 30 minutes
- Eight brave and noble volunteers: 6 geeky users, Nate’s dad, Nate’s fiancée*
- Running for up to 3 months
- Collected >200 Million total disk requests (only ~36 Million of them had enough information to test single-write rule)

*Despite crashing her machine and filling up her disk, they are still engaged.

False Positives

	Multi R/W	***Single R/W***	***Single Write***
Viruses detected out of 21 (previous table)	15; 3; 2	17; 3	18; 3
False positives (total in all traces)	5 in 201 M	28 in 201 M	19 in 36.5 M
False positives per million events	0.025	0.139	0.520

Seems most promising

“Virus-Like” Programs

- Program Updates
  - Signed updates using public key embedded in original executable
  - Legacy solution: “trusted” button
- System Restores
  - Restore from disk directly
- DRM Software, Virus Scanners
- Only to single-write rule: program installs, compilers

Virus Detection Results

- A simple, generic, behavioral, disk-level rule detects all file-infecting viruses in our sample
- A generic rule cannot detect malicious pre-infection behavior
- False positives seem solvable
  - Requires either some reengineering of systems or annoyance to user

Virus-Specific Signatures

- Examine collected traces of virus execution
  - Many generations, file infections
- Develop a disk-level signature that characterizes all executions
  - Precise enough to avoid false positives
- Requires mechanisms for updating signatures on disk

W32/Parite

read [*file*.exe@0|data:“MZ” or “ZM”];
create [*name*.tmp];
write [*name*.tmp@0|data:“MZ”];
write∗3 [*name*.tmp];
read∗7 [*name*.tmp@336,274,2,66,130,194,258];
write [ntuser.dat.LOG|data:“PINF”]

Robust: detects 5 tested generations

Very specific: no false positives (in all 201M events)

W32/Sality.L

read [*orig*.exe@0|data:"MZ" or "ZM"];

write [*drop*.dll@0|data:"MZ"];

read∗4 [*drop*.dll];

read [\system32\system.ini@0];

write [\system32\system.ini@0|data:"TFTempCache"]

- Sample (from vx.netlux.org repository) infected with both Sality and Linkbot.M
- Signature developed for Sality.L also matched Sality.M, O, and Q (but not K or earlier)

Summary: Virus-Specific Signatures

- Developed signatures for Efish, Ganda, Parite, Sality.L
- Perfect detection results: no missed executions, no false positives
- Still blacklisting (but much better than static blacklisting)
- After experience, ~1 day/signature
- Working on automating signature generation

Recap

- Virus writing pays
- Traditional virus detection is doomed
  - Wrong level, too static, too reactive
- Disk processor can detect viruses:
  - Sees all requests, powerful processor
- Simple rule can detect all file-infecting viruses with few false positives
- Specific, precise rules can detect malicious behavior exactly

Remaining Problems

- Bridging the semantic gap
  - Working on a disk-level implementation
- Security against determined attacker
  - Circumventing our rule is easy!
  - Behavioral-morphing viruses?
  - Resource exhaustion attacks
- Response and recovery
  - Need secure channel to user
- Deployment

Mixed-Metaphor Mantra

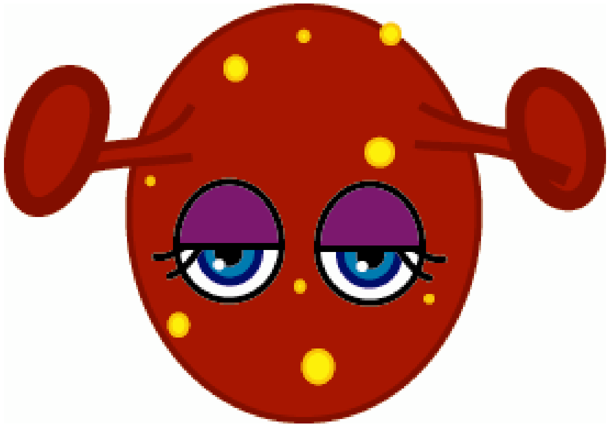
Traditional techniques will always be a step behind the malwarists.

Disk-level behavioral detection can give the “good” side a leg up in the virus detection arms race.

Students

Nate "Don't worry, I'm just going to install a harmless program on your PC" **Paul** (N-0.3th year PhD student)

Adrienne "Can I borrow your USB key to copy hundreds of viruses?" **Felt** (3rd year undergraduate)



For more information:

evans@cs.virginia.edu

http://www.cs.virginia.edu/malware

Nathanael Paul, Adrienne Felt, Sudhanva Gurumurthi, David Evans. *Disk-Level Behavioral Virus Detection*. (In submission, request by email)

Thanks:

Funding: NSF Cyber Trust

Running the disk tracer: Jamie Burnham, Wei Le, Jie Li, Ronny Paul, Shahrukh Tarapore, Chris Taylor, Dan Williams

Ideas, insights, comments: Shaun Hutton, Yan Huang, Anh Nguyen-Tuong, Mark Reis, Erik Riedel, Peter Szor, Shahrukh Tarapore, Chris Taylor