Host-Based Detection of Worms through Peer-to-Peer Cooperation

WORM 2005

David J. Malan and Michael D. Smith
Division of Engineering and Applied Sciences
Harvard University

Contact
malan@eecs.harvard.edu

Source Code
http://www.eecs.harvard.edu/~malan/

Motivation

The fastest of worms do not
allow time for human intervention.

Automated Responses Are Necessary

Today's responses are commonly...

- Signature-Based
  - Fast; false positives unlikely
  - But **takes time** to craft signatures;
    can be defeated by metamorphic/polymorphic worms
- Behavior-Based
  - Less susceptible to defeat by transformations
  - Faced with some anomalous action, tend to block that action (which risks **false positives**) or await user's judgement (which **takes time**)

Automated Responses Are Necessary

Our response is…

- Peer-Based
  - Time isn't something we have
    - We spend space instead of time—space in the form of peers
  - False positives aren't something we want
    - We define **anomalous behavior** as correlation among otherwise independent peers' behavior

Hypotheses

Worms stand out among other processes
because of their simplicity and periodicity:
their design is to spread, their execution thus cyclical.

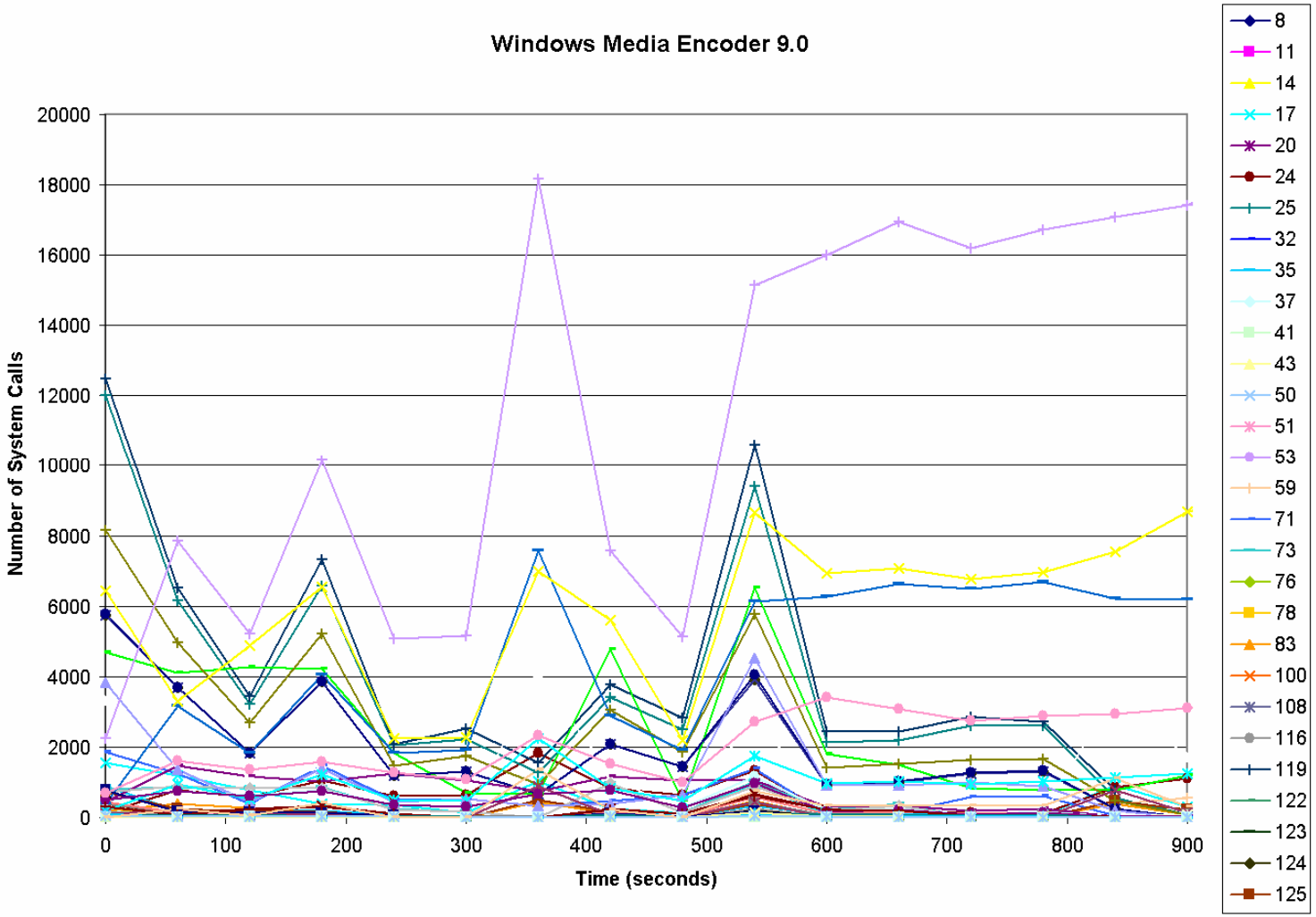
We are unlikely to witness such behavior in near lockstep on multiple hosts unless triggered by some external threat (distributed applications aside).

We can leverage peer-to-peer (P2P) cooperation to detect fast-spreading worms quickly.

We can leverage combinatorics to lower our risk of false positives.

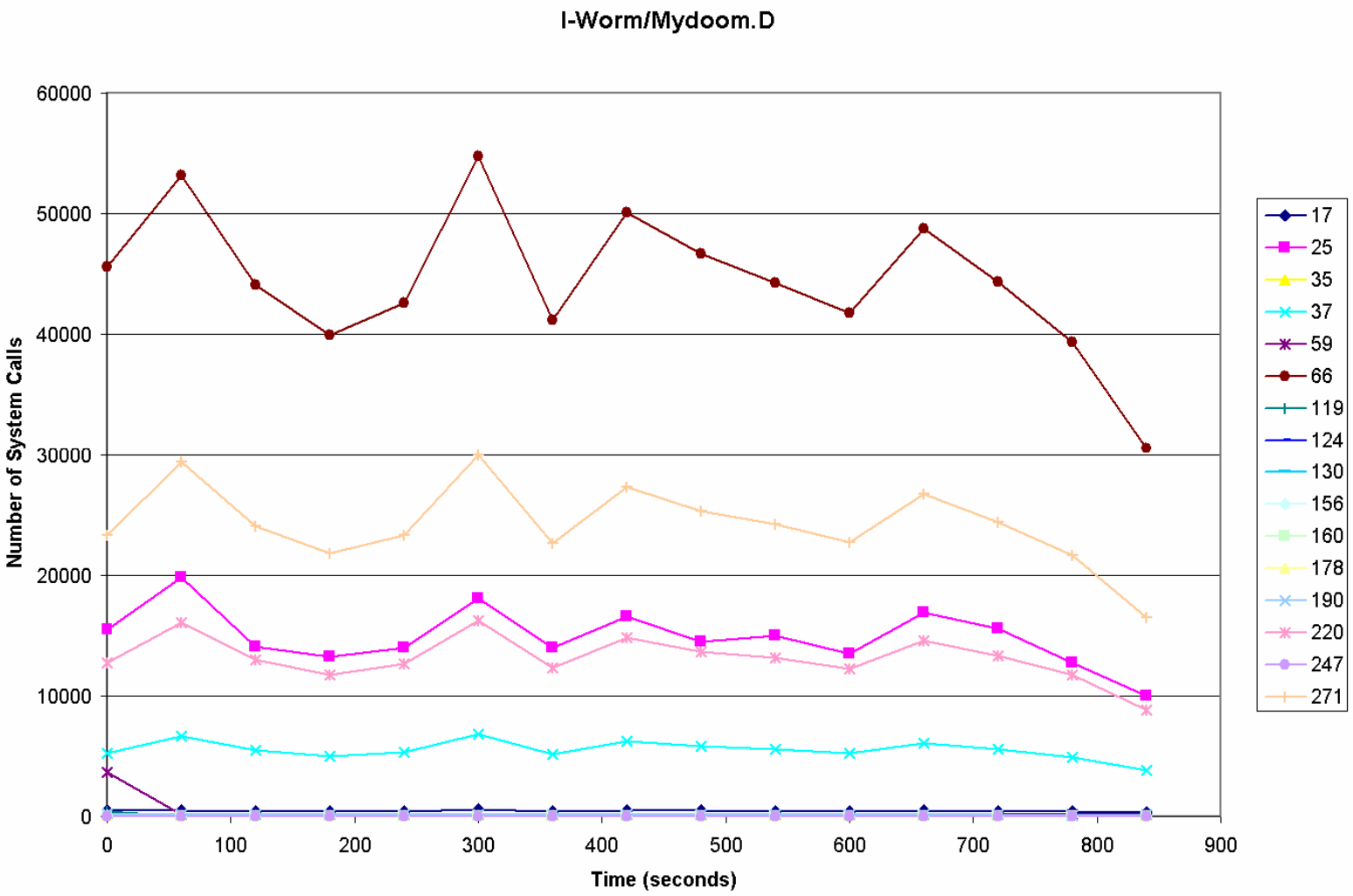
Motivation for Hypotheses

Trace of a Non-Worm's System Calls



Motivation for Hypotheses

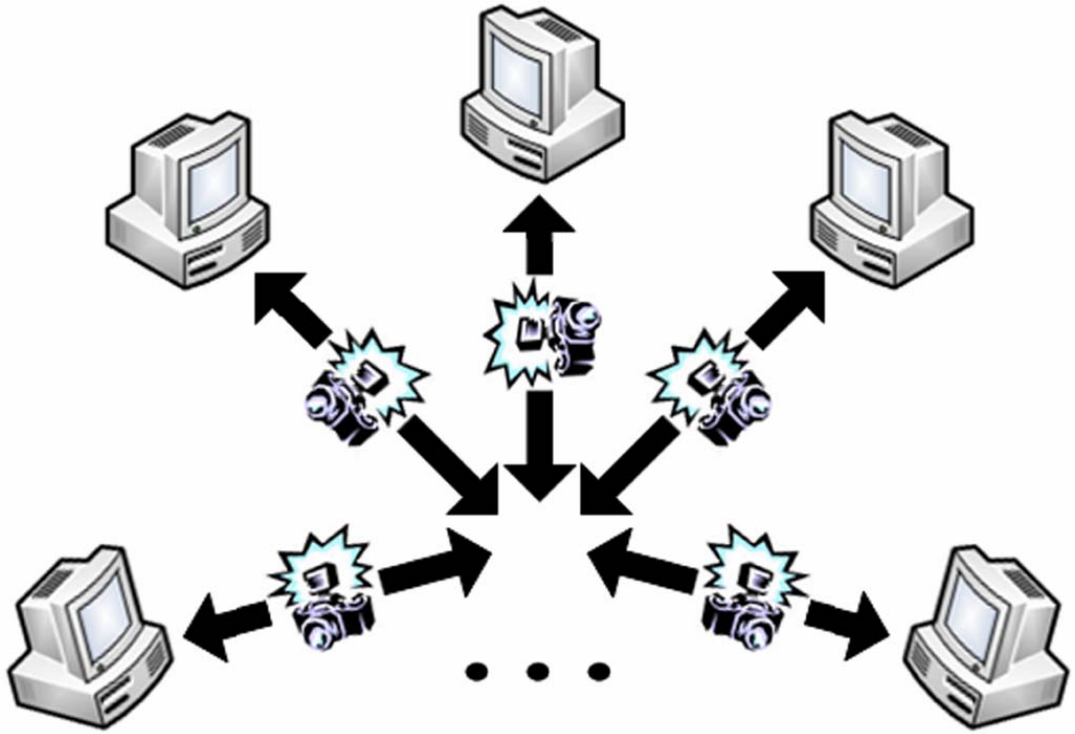
Trace of a Worm's System Calls



Our Vision

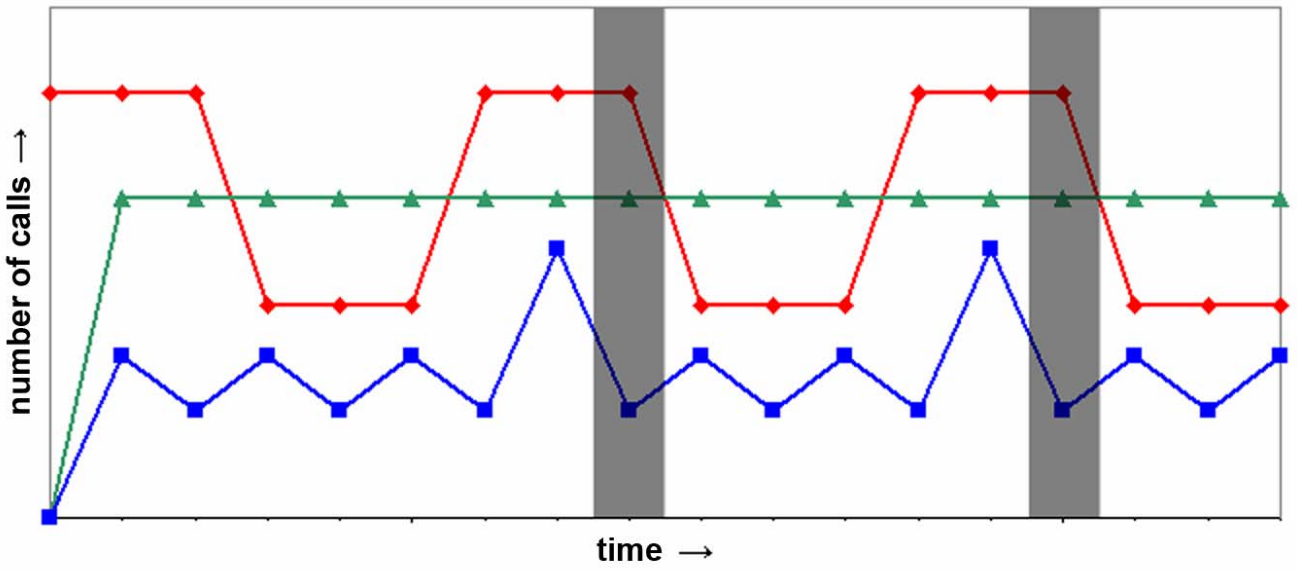
On some schedule, peers exchange snapshots of their internal behavior; too many similarities suggest **anomalous behavior**, a worm's presence.

We assume, for now, instantaneous communication, infinite bandwidth, and centralized detection.



What's a snapshot?

Hypothetical trace of a process's invocation over time of three system calls (our proxy for behavior). Shaded are two snapshots.



Our Focus

Distinguishing Worms from Non-Worms

If our vision is to be feasible,
we first must demonstrate that worms do, in fact,
tend to stand out in traces of behavior based on system calls.

We focus in this work on the problem of detection.

Given two or more snapshots of behavior, can we distinguish an attacking worm from an otherwise benevolent application?

Research Questions

Distinguishing Worms from Non-Worms

1) How likely is a worm to look like itself?
   - Are worms **temporally consistent**?
2) How likely is a non-worm to look like itself?
   - Are non-worms **temporally consistent**?
3) How likely is a non-worm to look like a worm?

Measuring Similarity

Temporal Consistency

- Measuring Similarity with Edit Distance (*d*)
  - Treat each snapshot as a set, *S*, of system calls, **ordered** by frequency
  - A process is said to be temporally consistent if

    $$1 - \frac{d}{\max(|S_1|,|S_2|)} \geq 0.5$$

    for a majority of pairs of snapshots over time

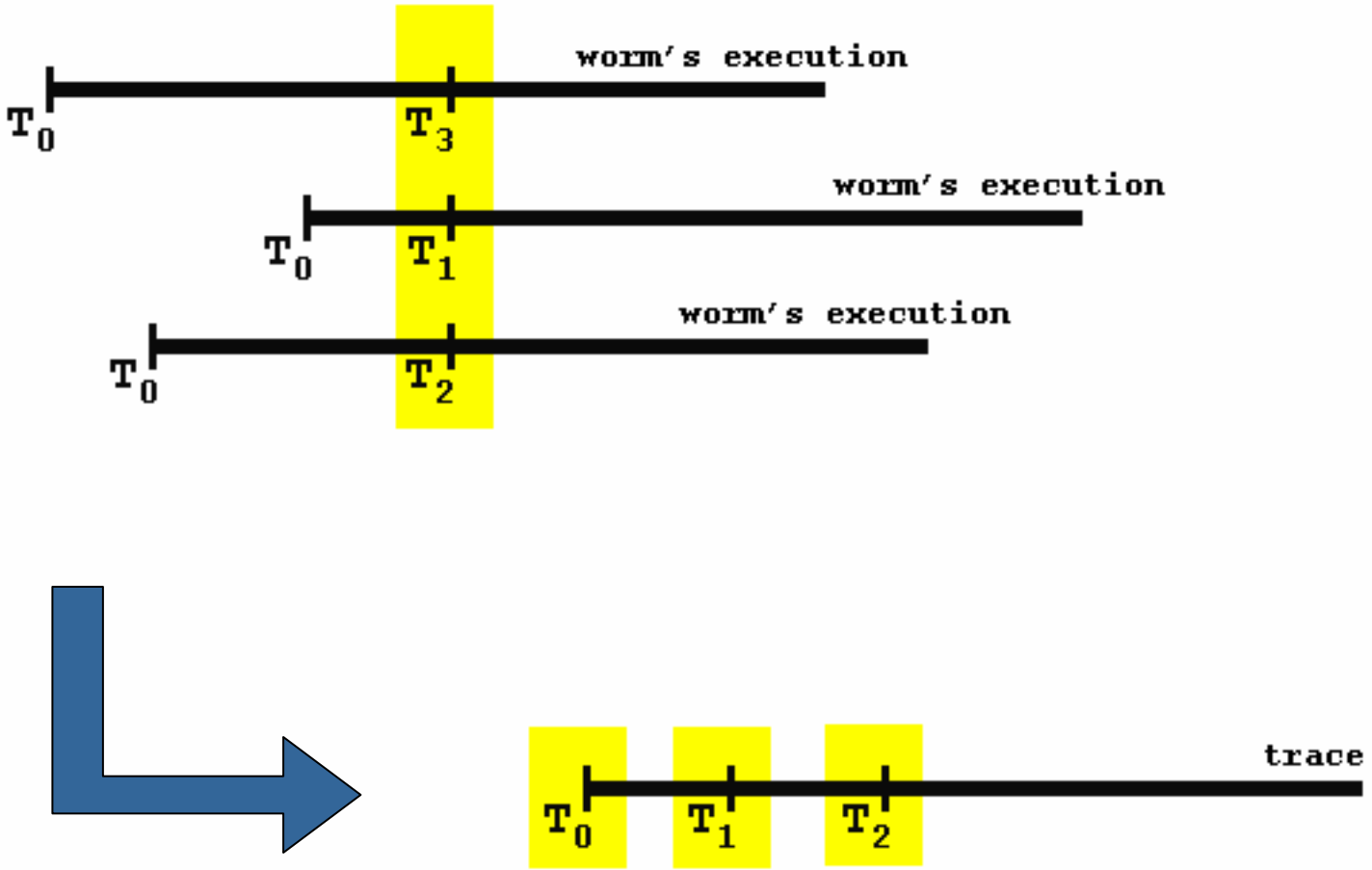
- Measuring Similarity with Intersection
  - Treat snapshots as **unordered** sets of system calls
  - A process is said to be temporally consistent if

    $$\frac{|S_1 \cap S_2|}{\max(|S_1|,|S_2|)} \geq 0.5$$

    for a majority of snapshots over time

Our Methodology

Using Traces to Simulate Peers for Now



Our Methodology

Windows XP with Service Pack 2

Worms	Non-Worms	
	Commercial Applications	WinXP SP2 Processes
I-Worm/Bagle.Q	Adobe Photoshop 7.0.1	alg.exe
I-Worm/Bagle.S	Microsoft Access XP SP2	csrss.exe
I-Worm/Jobaka.A	Microsoft Excel XP SP2	defrag.exe
I-Worm/Mydoom.D	Microsoft Outlook XP SP2	dfrgntfs.exe
I-Worm/Mydoom.F	Microsoft Powerpoint XP SP2	explorer.exe
I-Worm/Sasser.B	Microsoft Word XP SP2	helpsvc.exe
I-Worm/Sasser.D	Network Benchmark Client 1.0.3	lsass.exe
Worm/Lovesan.A	Nullsoft Winamp 5.094	msmsgs.exe
Worm/Lovesan.H	Windows Media Encoder 9.0	services.exe
	WinZip 8.1	spoolsv.exe
		svchost.exe
		wmiprvse.exe
		winlogon.exe
		wscntfy.exe
		wuauclt.exe

Our Methodology

Wormboy 1.0

```
> 1 631547  1544  1000  k n NtOpenFile  116 (1, 2, 4, 5, )=(0x80100000, {24, 0, 0x240, 0,
< 1 631557  1544  1000  k n NtOpenFile  116 ()  STATUS_OBJECT_NAME_NOT_FOUND
> 2 631557  1544  1000  u n NtOpenKey 119 (1, 2, )=(0x80000000, {24, 0, 0x40, 0, 0, "\Reg
< 2 631557  1544  1000  u n NtOpenKey 119 ()  STATUS_OBJECT_NAME_NOT_FOUND
> 3 631557  1544  1000  u n NtOpenKeyedEvent  280 (1, 2, )=(0x2000000, {24, 0, 0x0, 0, 0,
< 3 631557  1544  1000  u n NtOpenKeyedEvent  280 (0, )=(4, ) 0x0
> 4 631557  1544  1000  u n NtQuerySystemInformation  173 (0, 2, )=(Basic, 44, )  n/a
< 4 631557  1544  1000  u n NtQuerySystemInformation  173 (1, 3, )=({Unknown=0,MaximumInc
> 5 631557  1544  1000  u n NtAllocateVirtualMemory 17  (0, 1, 2, 3, 4, 5, )=(-1, 0, 0, 1
< 5 631557  1544  1000  u n NtAllocateVirtualMemory 17  (1, 3, )=(1310720, 1048576, ) 0x0
> 6 631557  1544  1000  u n NtAllocateVirtualMemory 17  (0, 1, 2, 3, 4, 5, )=(-1, 1310720
< 6 631557  1544  1000  u n NtAllocateVirtualMemory 17  (1, 3, )=(1310720, 4096, )  0x0
> 7 631557  1544  1000  u n NtAllocateVirtualMemory 17  (0, 1, 2, 3, 4, 5, )=(-1, 1314816
< 7 631557  1544  1000  u n NtAllocateVirtualMemory 17  (1, 3, )=(1314816, 8192, )  0x0
> 8 631557  1544  1000  u n NtQuerySystemInformation  173 (0, 2, )=(Basic, 44, )  n/a
< 8 631557  1544  1000  u n NtQuerySystemInformation  173 (1, 3, )=({Unknown=0,MaximumInc
> 9 631557  1544  1000  u n NtAllocateVirtualMemory 17  (0, 1, 2, 3, 4, 5, )=(-1, 0, 0, 6
< 9 631557  1544  1000  u n NtAllocateVirtualMemory 17  (1, 3, )=(2359296, 65536, ) 0x0
```

Source code available at:
http://www.eecs.harvard.edu/~malan/

Results

Temporal Consistency of Worms

	5	15	30
I-Worm/Bagle.Q	14%	11%	10%
I-Worm/Bagle.S	14%	11%	11%
I-Worm/Jobaka.A	59%	50%	69%
I-Worm/Mydoom.D	92%	81%	73%
I-Worm/Mydoom.F	17%	31%	41%
I-Worm/Sasser.B	60%	54%	72%
I-Worm/Sasser.D	95%	97%	93%
Worm/Lovesan.A	99%	98%	93%
Worm/Lovesan.H	47%	95%	93%

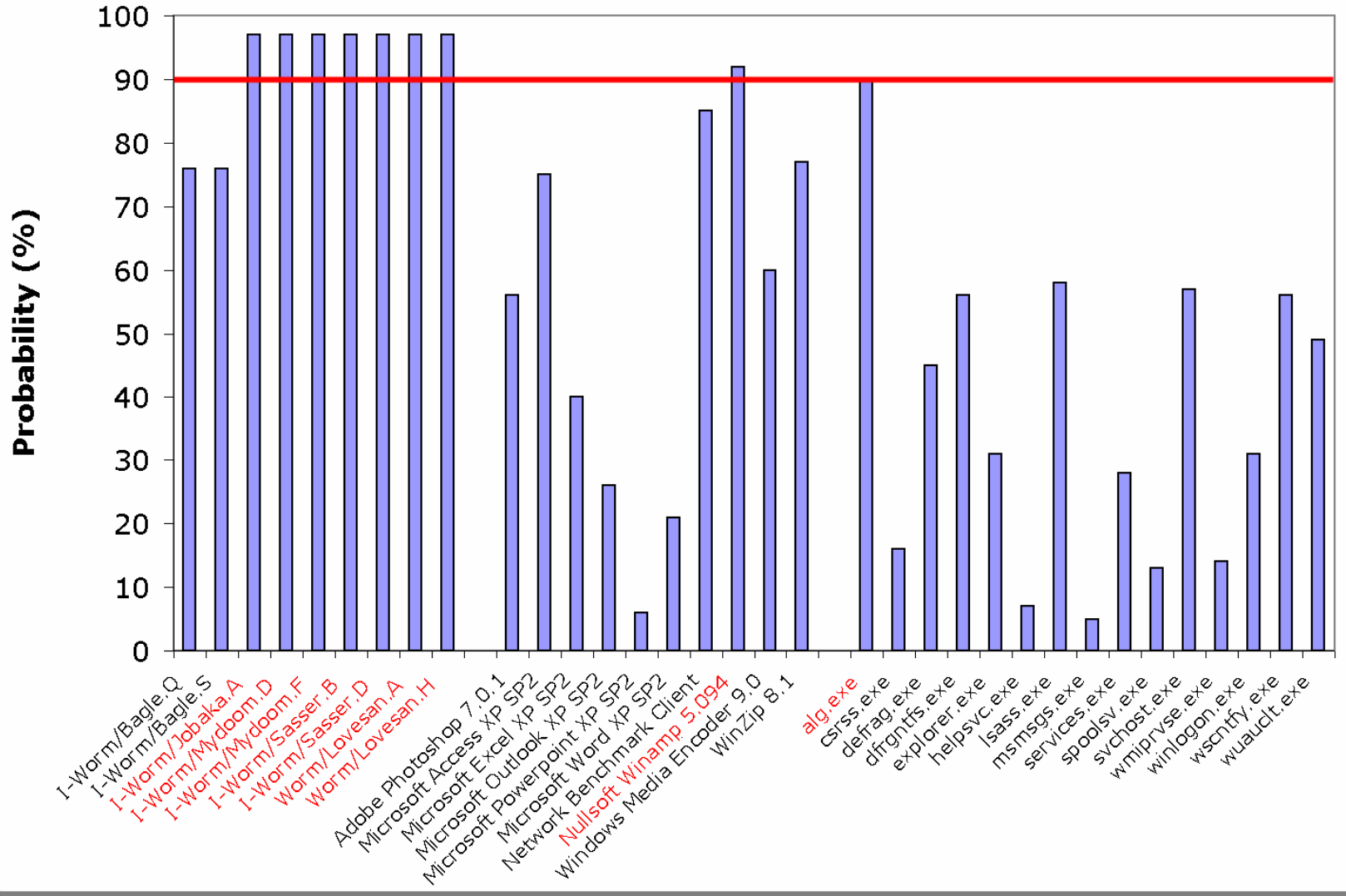
	5	15	30
I-Worm/Bagle.Q	80%	76%	81%
I-Worm/Bagle.S	82%	76%	73%
I-Worm/Jobaka.A	99%	97%	93%
I-Worm/Mydoom.D	99%	97%	93%
I-Worm/Mydoom.F	99%	97%	93%
I-Worm/Sasser.B	99%	97%	93%
I-Worm/Sasser.D	99%	97%	93%
Worm/Lovesan.A	99%	97%	93%
Worm/Lovesan.H	49%	97%	93%

Probabilities with which two peers, upon exchanging snapshots of their internal behavior, can decide using edit distance or intersection that they are, more likely than not, both executing the same worm during some window of time, for window sizes of 5, 15, and 30 seconds.

Edit distance (table at left) did not appear to detect temporal consistency as well as intersection (table at right) did.

Results

Temporal Consistency of Worms v. Non-Worms (15-sec windows)



Results

Distinguishing Worms from Non-Worms

1) How likely is a worm to look like itself?
   - Two peers can decide that they are, more likely than not, executing a worm between 76% and 97% of the time.
2) How likely is a non-worm to look like itself?
   - Only `alg.exe`'s and Nullsoft Winamp's snapshots resembled each other more than 90% of the time.
3) How likely is a non-worm to look like a worm?
   - Only Network Benchmark Client's behavior resembled, more often than not, that of a worm.

Conclusion and Future Work

Host-Based Detection of Worms through P2P Cooperation

- We define **anomalous behavior** as correlation among otherwise independent peers' behavior.
- This work focuses entirely on the problem of collaborative detection.
- Future work will relax our assumptions that communication is instantaneous, bandwidth infinite, and detection centralized.

Host-Based Detection of Worms through Peer-to-Peer Cooperation

WORM 2005

David J. Malan and Michael D. Smith
Division of Engineering and Applied Sciences
Harvard University

Contact
malan@eecs.harvard.edu

Source Code
http://www.eecs.harvard.edu/~malan/

Why Window Size Matters

Worm/Lovesan.H: 5- v. 15-Second Windows