Malware Repository Requirements

Policy Analysis

Paul Vixie && David Dagon

Outline

- How malware is collected and shared now
- Proposed service-oriented repository
- Automated unpacking
- Header analysis

Current Practices

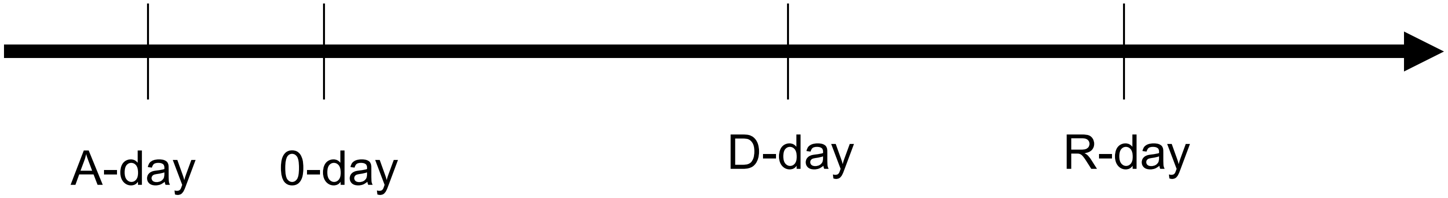
- Numerous private, semi-public malware collections
  - Need trust to join (for some value of “trust”)
  - “Too much sharing” often seen as competitive disadvantage
- Incomplete collections: reflect sensor bias
  - Darknet-based collection
  - IRC surveillance
  - Honeypot-based collection

Shortcomings

- Malware authors know and exploit weaknesses in data collection
- Illuminating sensors
  - “Mapping Internet Sensors with Probe Response Attacks”, Bethencourt, et al., Usenix 2005
- Automated victims updates
  - “Queen-bot” programs keep drones in 0-day window

Malware Life Cycle

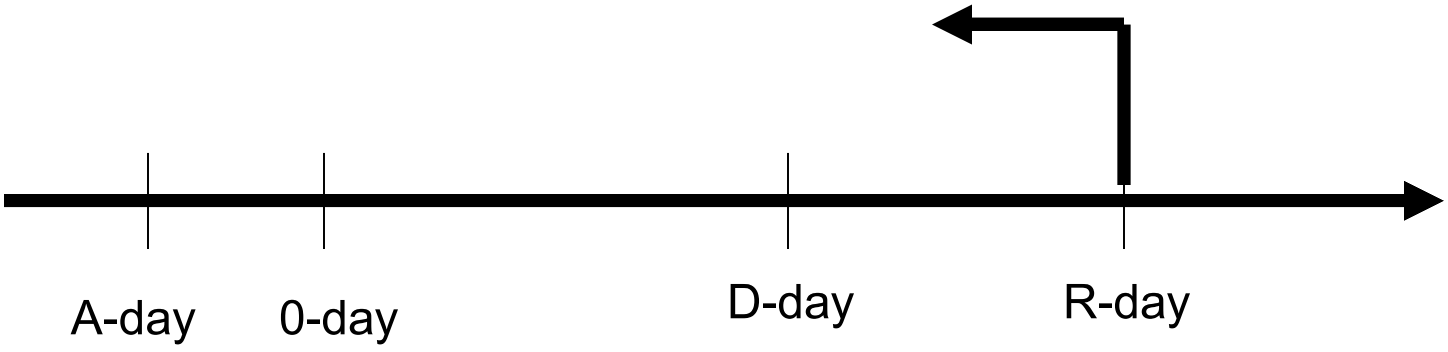
Four conceptual phases of malware life cycle:



A-day: malware authored
0-day: release
D-day: first opportunity for detection
R-day: response (e.g., virus signature update)

Malware Life Cycle

Recent AV goal: reduce response time



A-day

0-day

D-day

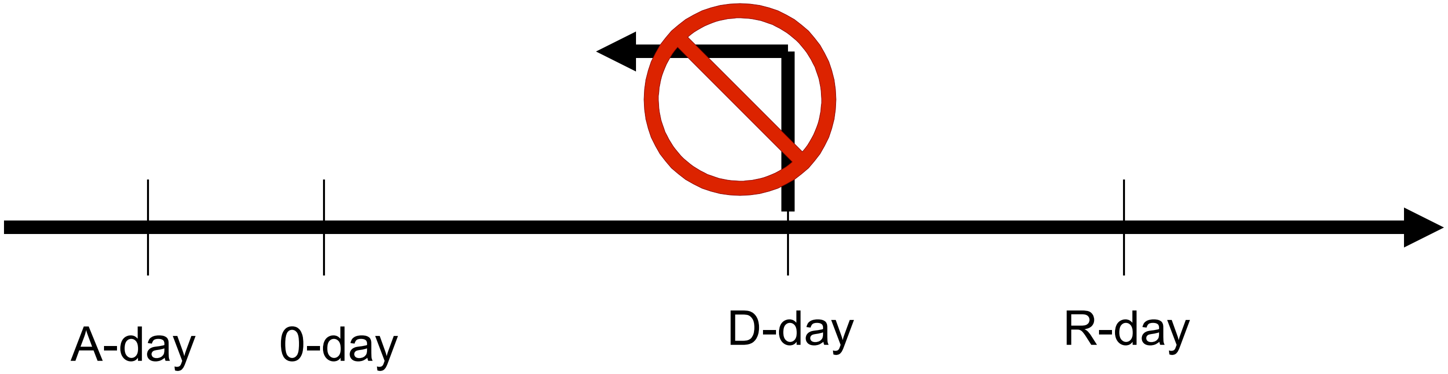
R-day

AV update cycles previously measured weeks/days

Now measured in hours/minutes (or should be)

Malware Life Cycle

How to improve detection time...

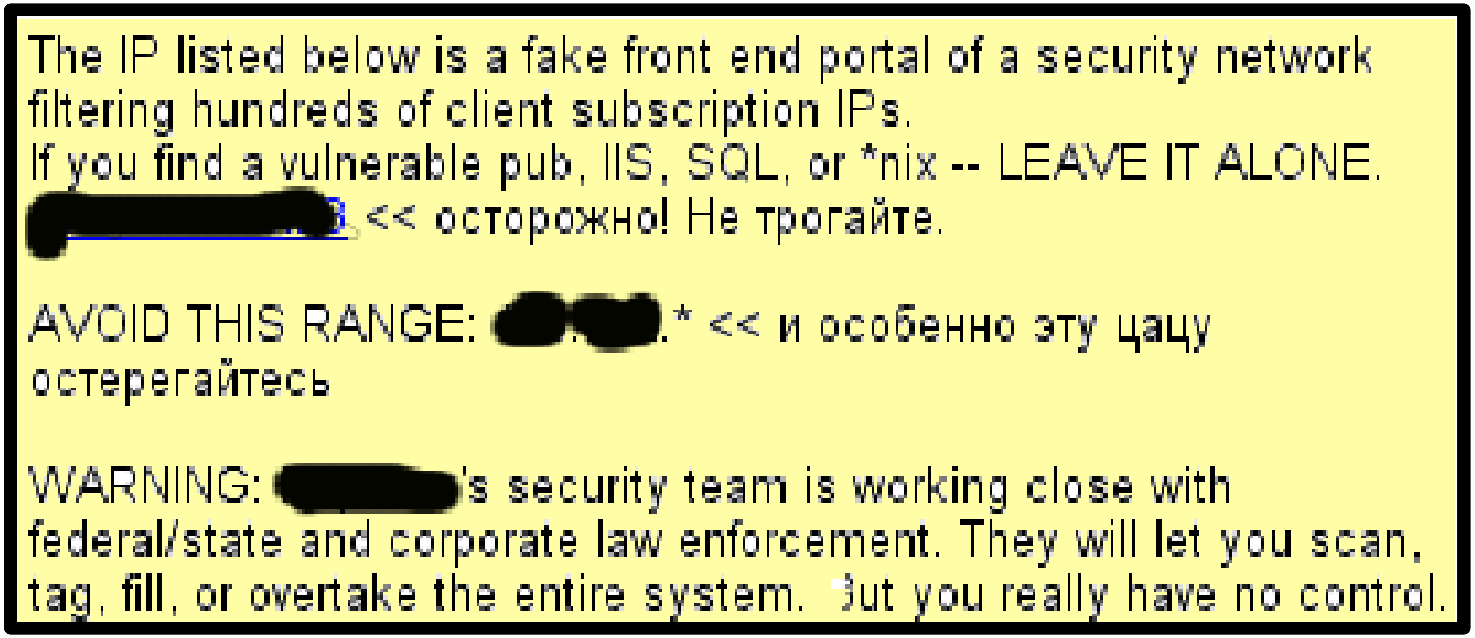


Given that...

- Malware authors avoid known sensors
- Repositories don’t share

Sensor Illumination

- Technique
  - Malware authors compile *single*, unique virus;
  - Send to suspected sensor
  - Wait and watch for updates



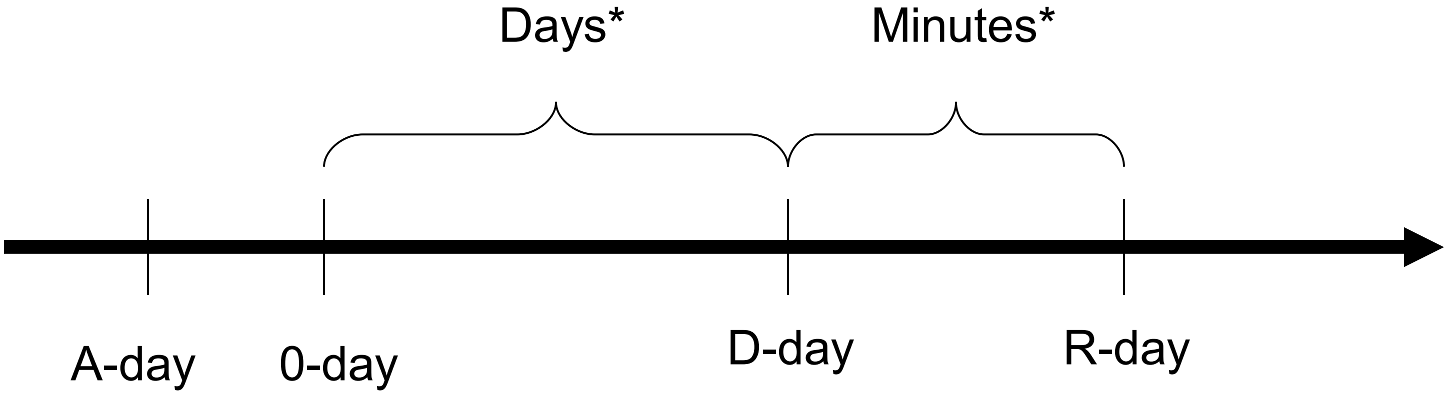
The IP listed below is a fake front end portal of a security network filtering hundreds of client subscription IPs.
If you find a vulnerable pub, IIS, SQL, or *nix -- LEAVE IT ALONE.
[redacted] << осторожно! Не трогайте.

AVOID THIS RANGE: [redacted].[redacted].* << и особенно эту цацу остерегайтесь

WARNING: [redacted]'s security team is working close with federal/state and corporate law enforcement. They will let you scan, tag, fill, or overtake the entire system. But you really have no control.

Malware Life Cycle

Thus, response is hours/days; detection is days++



Days*

Minutes*

A-day

0-day

D-day

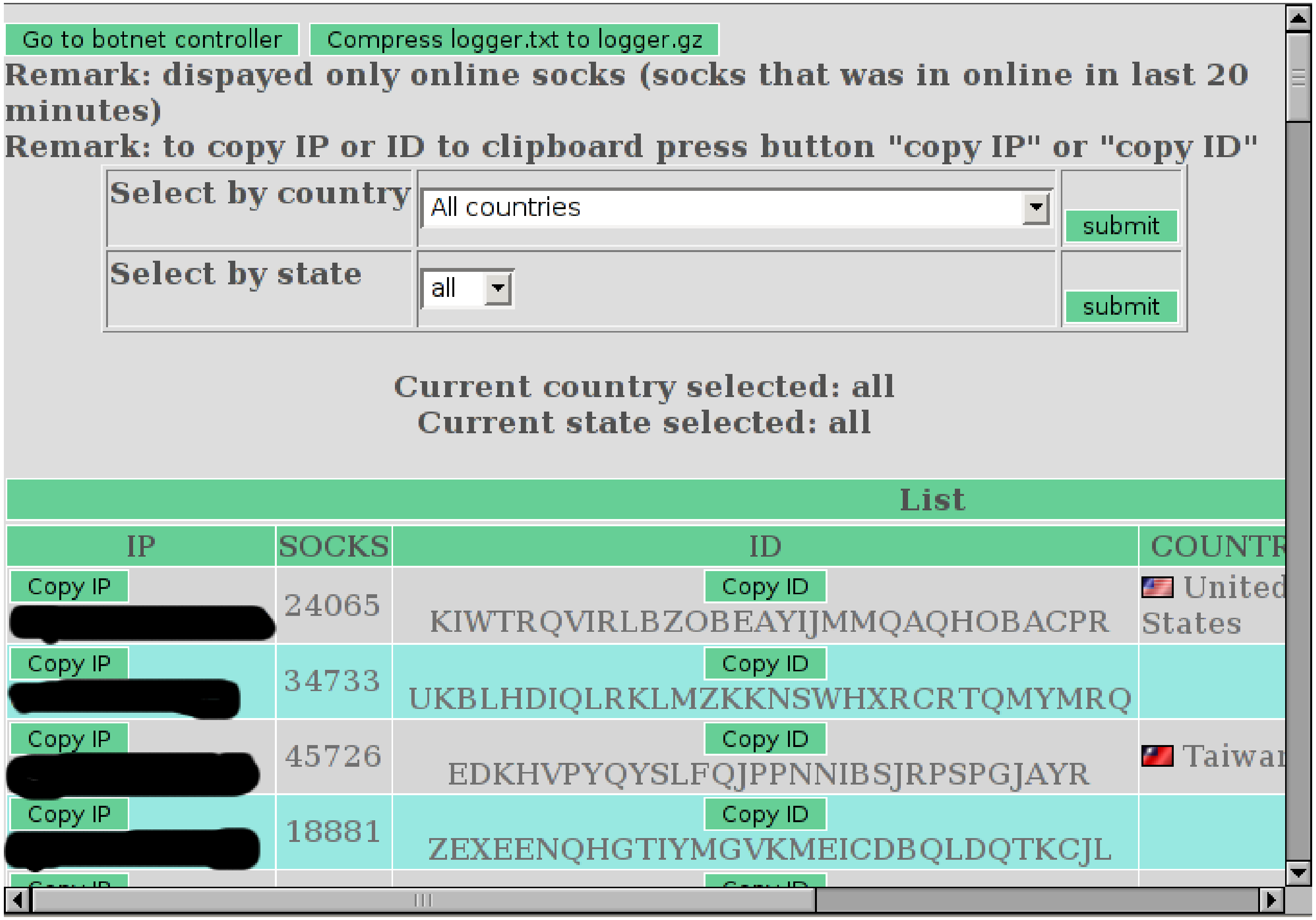
R-day

* Average order of time; anecdotes will vary

Queen Bot Programs

- Automated update of existing bot
  - Repacking with new key; multiple packers
  - Dead code injection
  - Variable renaming, functional decomposition
- New bot will
  - Evade prior AV signatures
  - Have same behaviour, goal, feature set

Malware Updating



Remark: dispayed only online socks (socks that was in online in last 20 minutes)
Remark: to copy IP or ID to clipboard press button "copy IP" or "copy ID"

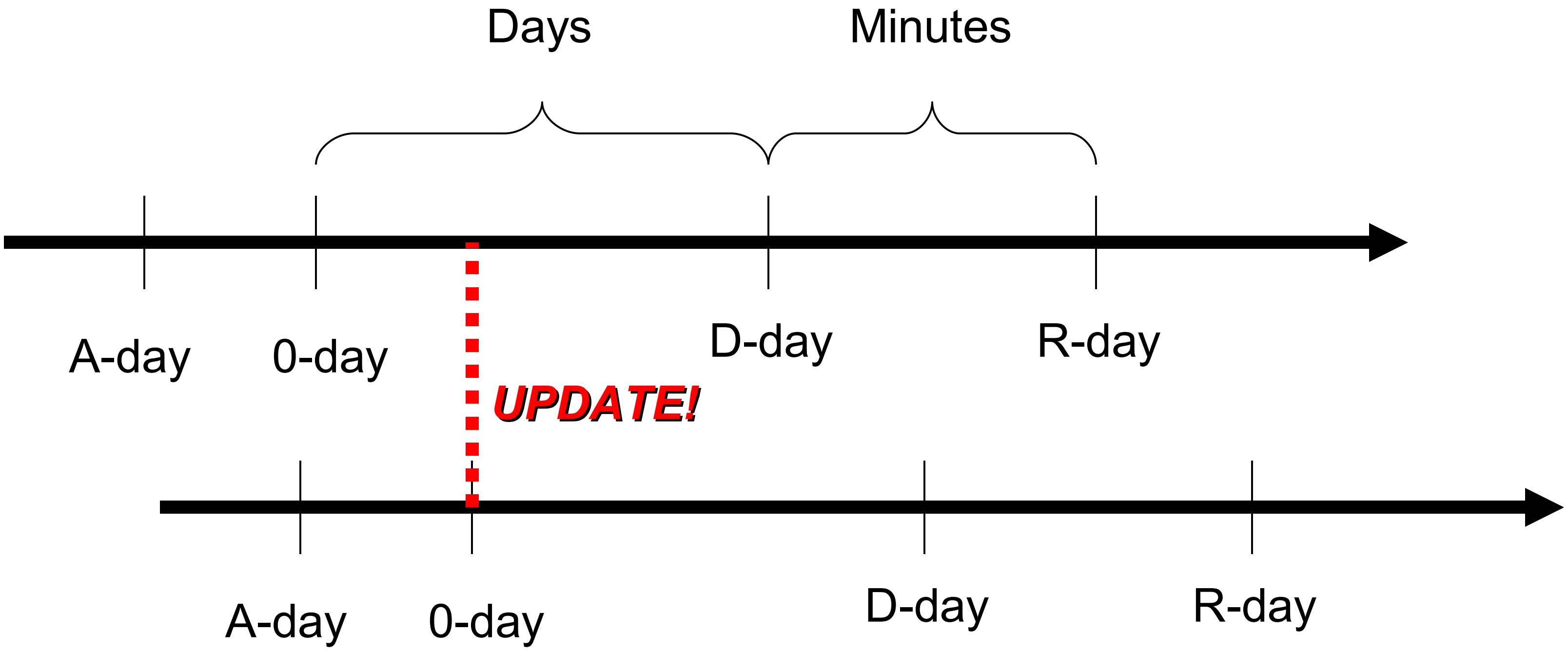
Select by country	All countries	submit
Select by state	all	submit

Current country selected: all
Current state selected: all

List

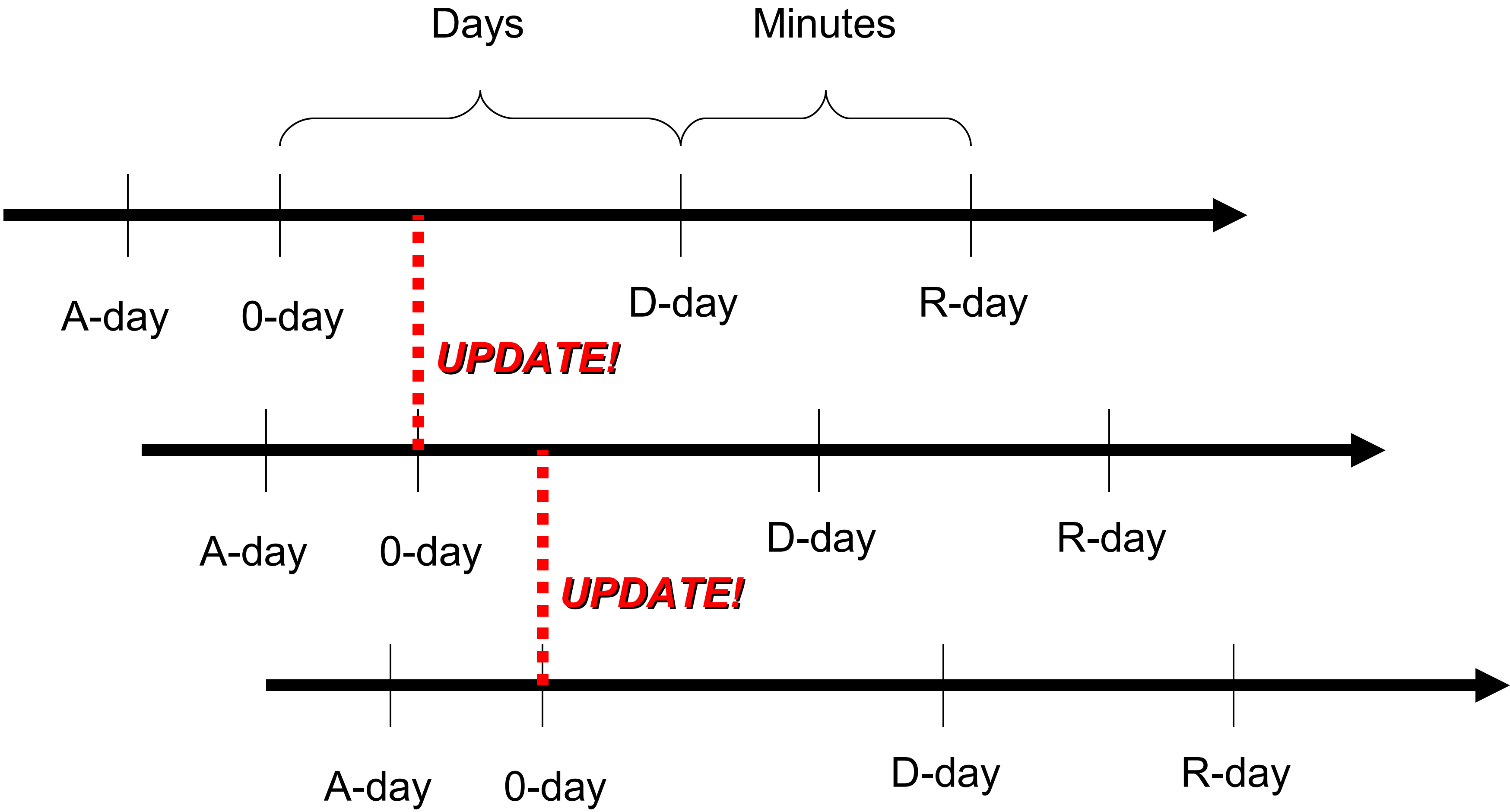
IP	SOCKS	ID	COUNTR
Copy IP [redacted]	24065	Copy ID KIWTRQVIRLBZOBEAYIJMMQAQHOBACPR	United States
Copy IP [redacted]	34733	Copy ID UKBLHDIQLRKLMZKKNSWHXRCRTQMYMRQ	
Copy IP [redacted]	45726	Copy ID EDKHVPYQYSLFQJPPNNIBSJRPSPGJAYR	Taiwa
Copy IP [redacted]	18881	Copy ID ZEXEENQHGTIYMGVKMEICDBQLDQTKCJL	

Malware Life Cycle

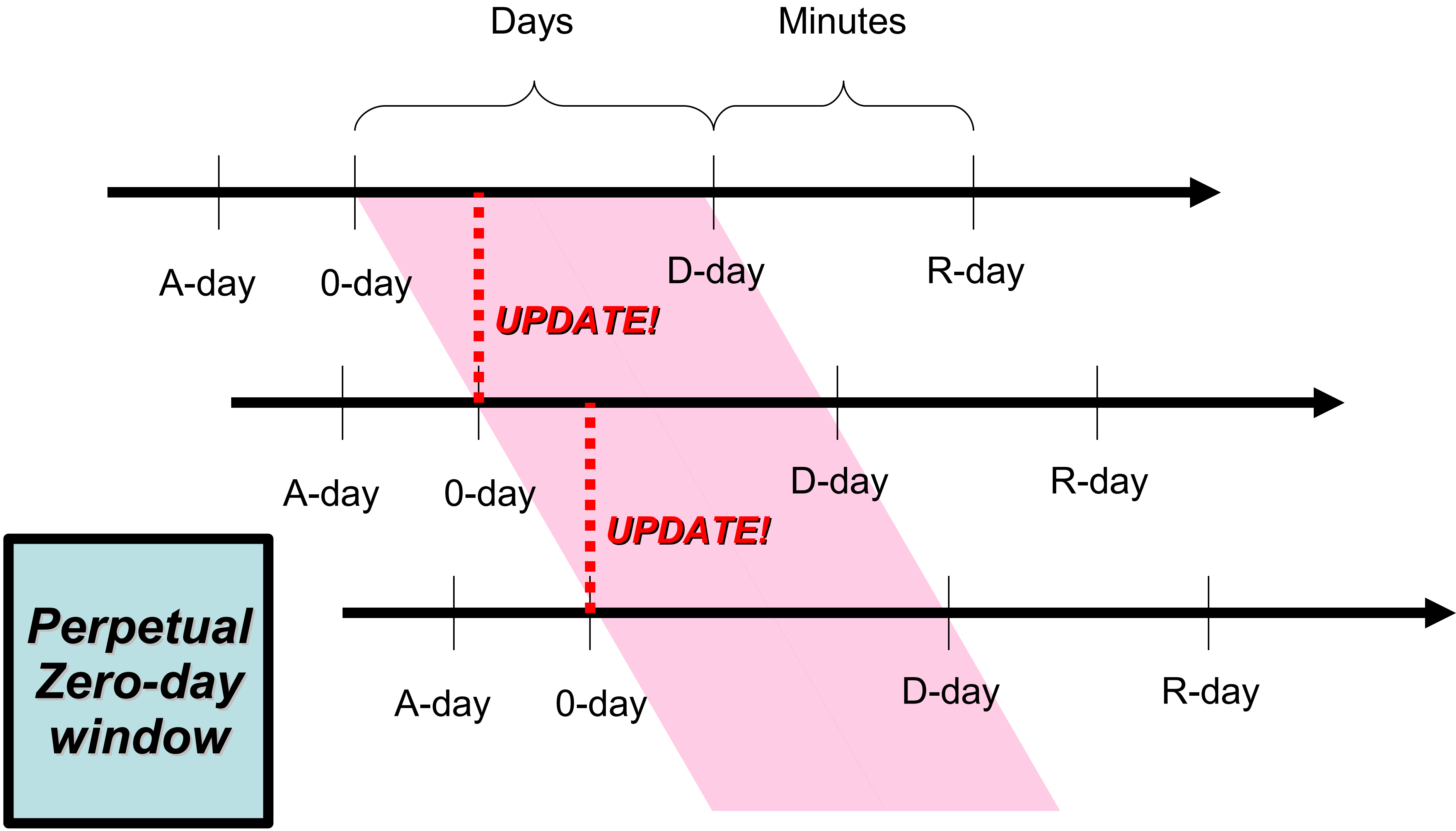


Bot runs for ~1/2 day, and updates to new, evasive binary

Malware Life Cycle



Malware Life Cycle



Days

Minutes

A-day

0-day

D-day

R-day

UPDATE!

A-day

0-day

D-day

R-day

UPDATE!

A-day

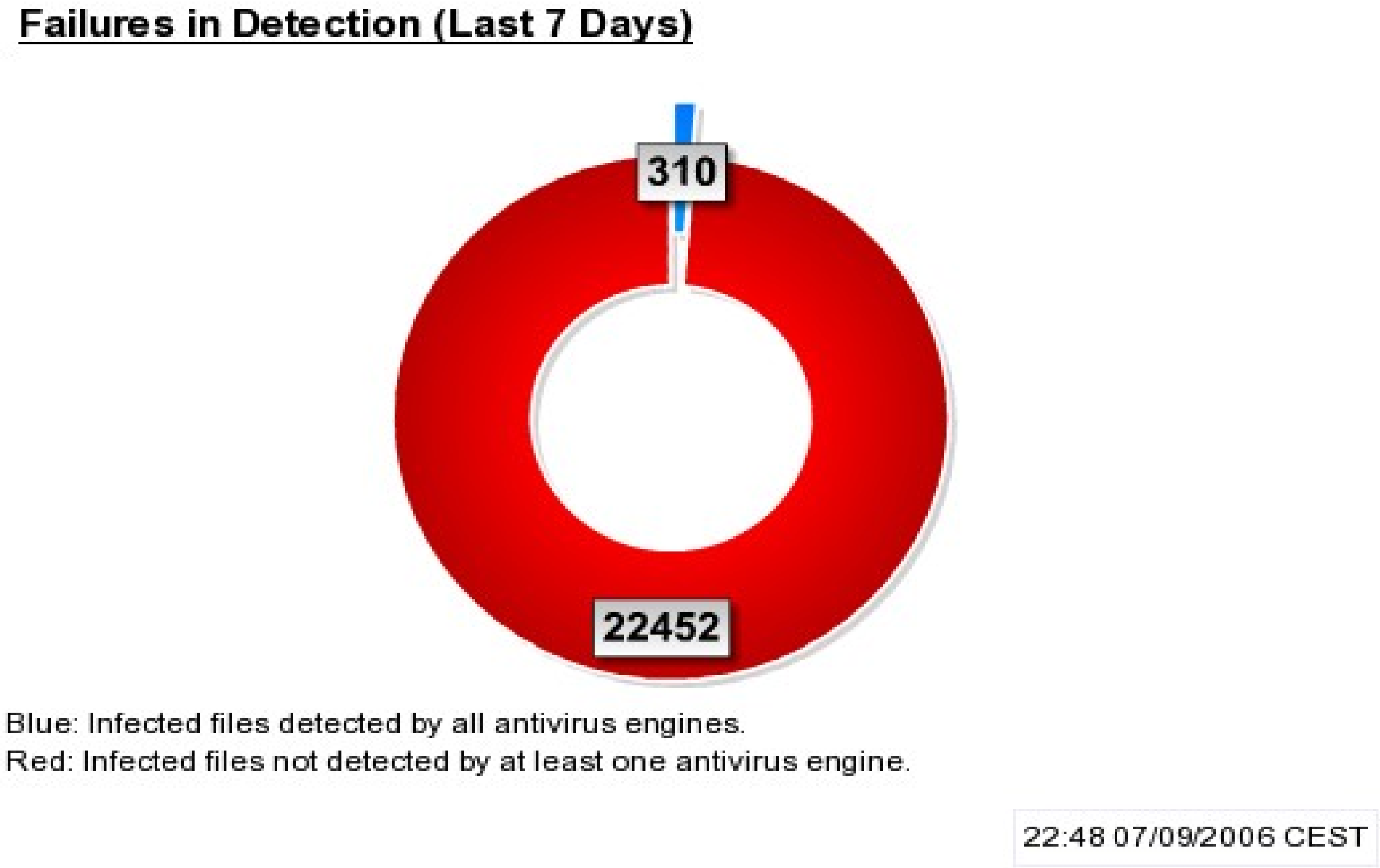
0-day

D-day

R-day

Perpetual Zero-day window

Example from virustotal.com



Failures in Detection (Last 7 Days)

Blue: Infected files detected by all antivirus engines.
Red: Infected files not detected by at least one antivirus engine.

22:48 07/09/2006 CEST

Why Pack?

- Reduced malware size
- Obfuscation transformation
  - Opaque binaries prevent pattern analysis
  - Invalid PE32 headers complicate RE
- Increases response time
  - Unpacking often requires specialized skill sets

Who Shares Now, and Why?

- Current AV industry practice is basically “hostage exchange”
- Time-value of shared material has to approach zero first
- “Competitive advantage” means advantage to an AV's shareholders – or does it?

Is More Sharing Better?

- In F/OSS, the value of sharing is known to outweigh the value of hoarding
- We propose a similar model for malware
- Cost:Benefit ratio (to AV shareholders and customers) of specialization was obvious to Adam Smith (see *Wealth of Nations*)
- As with SALT-II, the interesting part of the problem is “compliance monitoring”

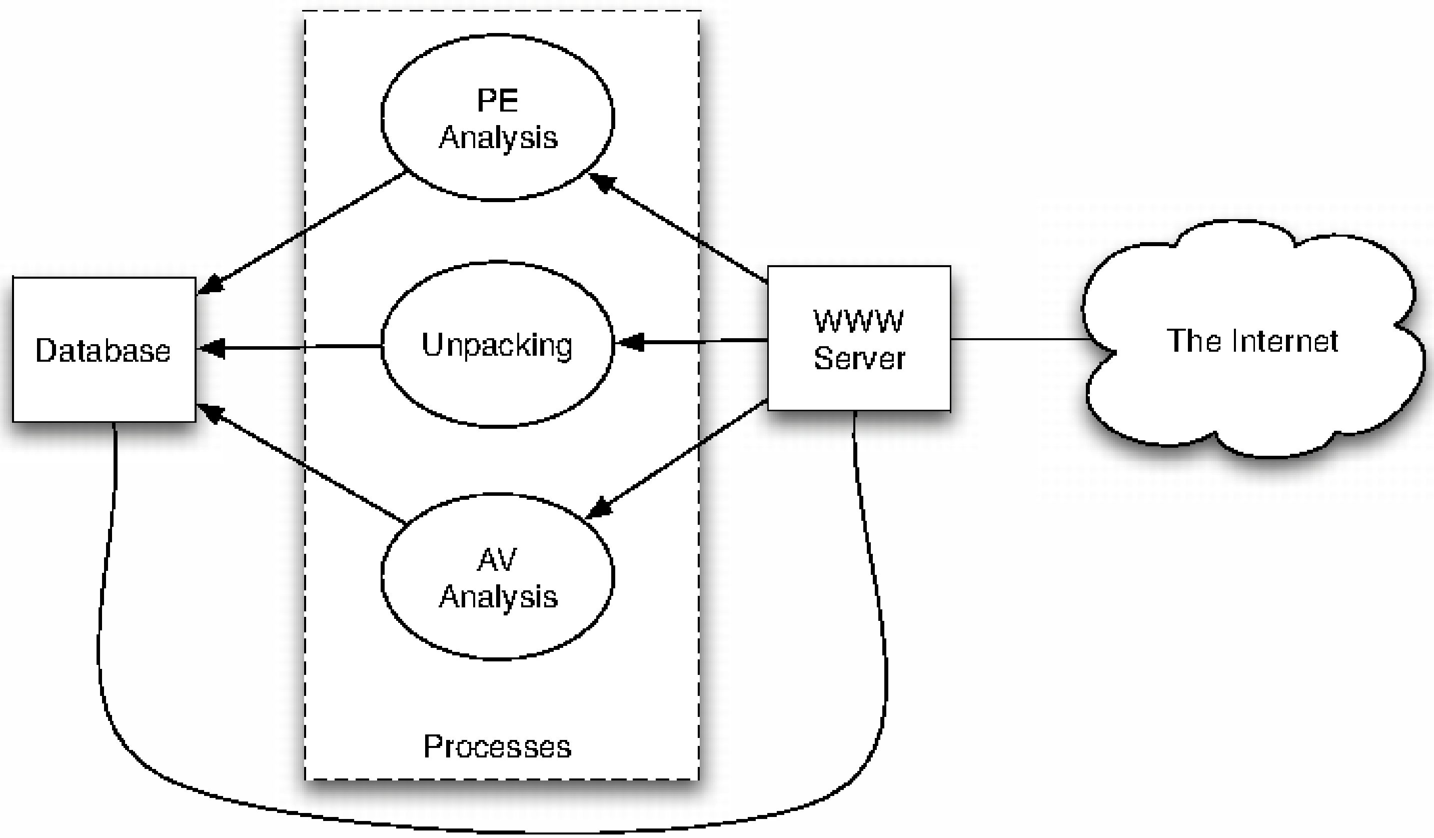
Malware Repo Requirements

- Malware repos *should not*:
  - Help illuminate sensors
  - Serve as a malware distribution site
- Malware repo *should*:
  - Help automate analysis of malware flood
  - Coordinate different analysts (RE gurus, MX gurus, Snort rule writers, etc.)

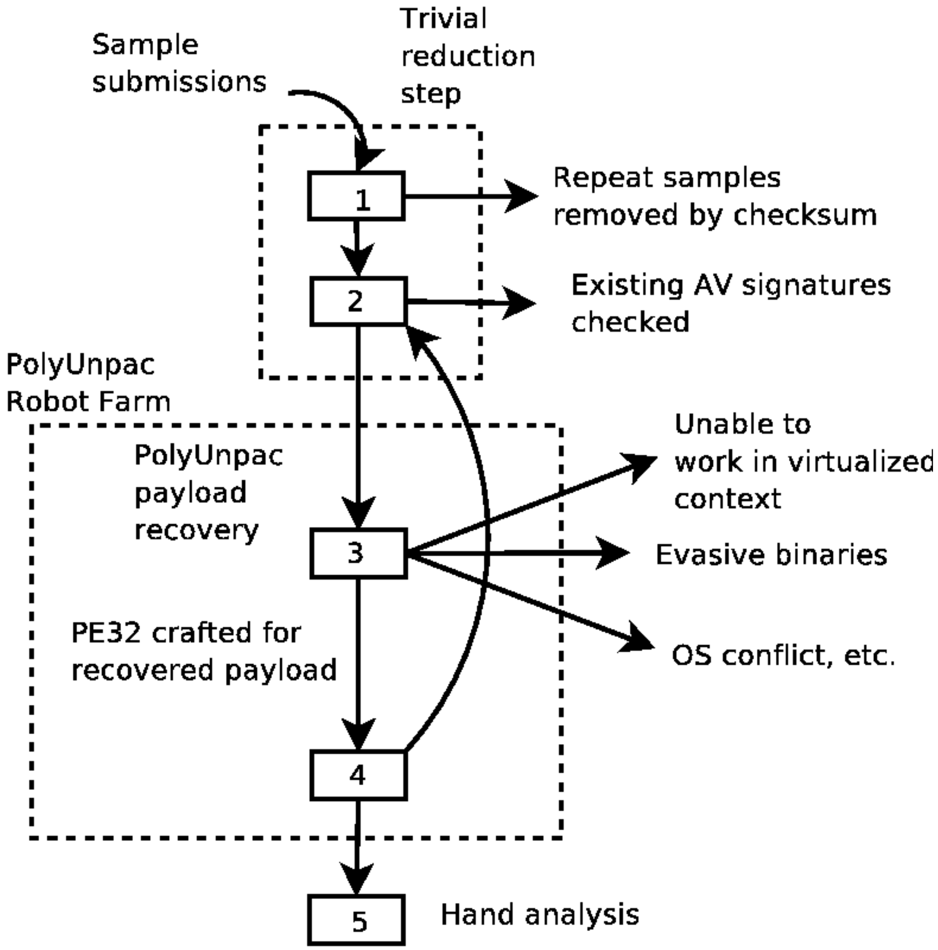
Approach: Service-Oriented Repository

- Repository allows upload of samples
  - Downloads restricted to classes of users
- Repository provides binaries *and* analysis
  - Automated unpacking
  - Win32 PE Header analysis
  - Longitudinal detection data
    - What did the AV tool know, and when did it know it?
  - Soon: Malware similarity analysis, family tree

Overview


PE Analysis
Unpacking
AV Analysis
Processes
Database
WWW Server
The Internet

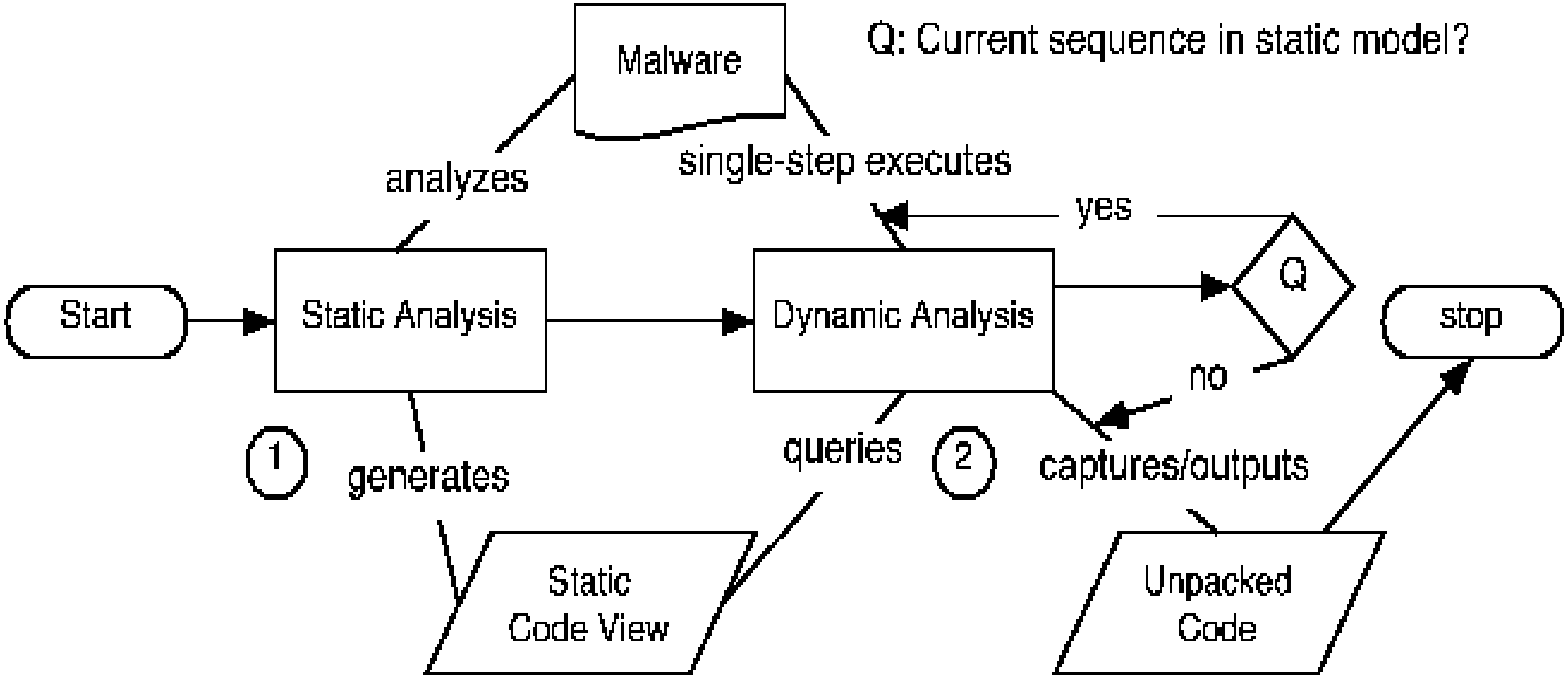
Work Flow


Sample submissions
Trivial reduction step
1
Repeat samples removed by checksum
2
Existing AV signatures checked
PolyUnpac Robot Farm
PolyUnpac payload recovery
3
Unable to work in virtualized context
Evasive binaries
OS conflict, etc.
PE32 crafted for recovered payload
4
5
Hand analysis

Unpacking

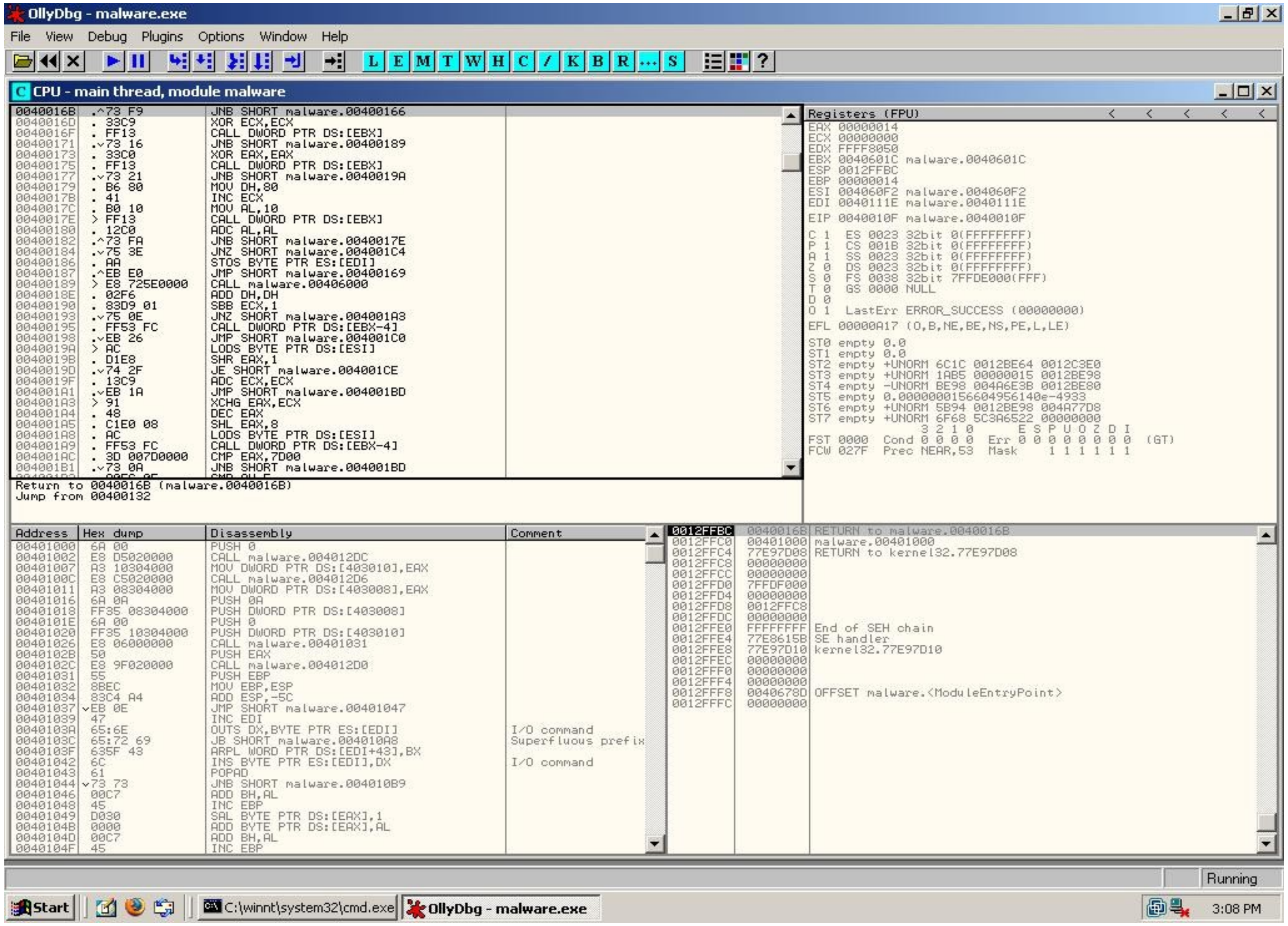
- Dynamic analysis permits unpacking
  - Analogous to halting problem
- Heuristic approximation
  - White list jumps to: (static) basic block entry points, and DLL functions
  - If known, continue; else assume halting
  - Rinse, lather, repeat for recursive packing

Unpacking Heuristic



Q: Current sequence in static model?

Unpacking Example



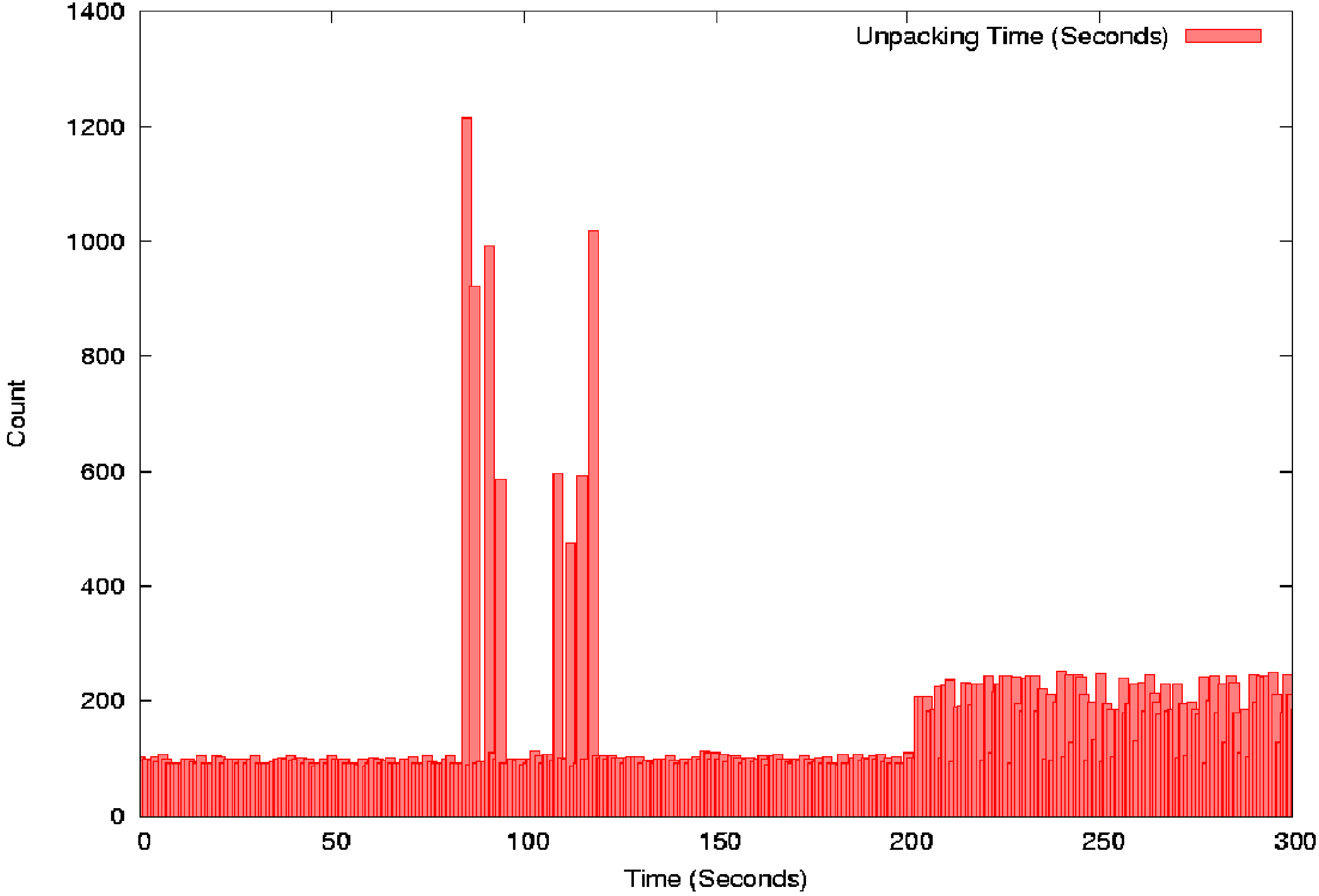
```
0040016B  .^73 F9         JNB SHORT malware.00400166
0040016D  . 33C9          XOR ECX,ECX
0040016F  . FF13          CALL DWORD PTR DS:[EBX]
00400171  .v73 16         JNB SHORT malware.00400189
00400173  . 33C0          XOR EAX,EAX
00400175  . FF13          CALL DWORD PTR DS:[EBX]
00400177  .v73 21         JNB SHORT malware.0040019A
00400179  . B6 80         MOV DH,80
0040017B  . 41            INC ECX
0040017C  . B0 10         MOV AL,10
0040017E  > FF13          CALL DWORD PTR DS:[EBX]
00400180  . 12C0          ADC AL,AL
00400182  .^73 FA         JNB SHORT malware.0040017E
00400184  .v75 3E         JNZ SHORT malware.004001C4
00400186  . AA            STOS BYTE PTR ES:[EDI]
00400187  .^EB E0         JMP SHORT malware.00400169
00400189  > E8 725E0000   CALL malware.00406000
0040018E  . 02F6          ADD DH,DH
00400190  . 83D9 01       SBB ECX,1
00400193  .v75 0E         JNZ SHORT malware.004001A3
00400195  . FF53 FC       CALL DWORD PTR DS:[EBX-4]
00400198  .vEB 26         JMP SHORT malware.004001C0
0040019A  > AC            LODS BYTE PTR DS:[ESI]
0040019B  . D1E8          SHR EAX,1
0040019D  .v74 2F         JE SHORT malware.004001CE
0040019F  . 13C9          ADC ECX,ECX
004001A1  .vEB 1A         JMP SHORT malware.004001BD
004001A3  > 91            XCHG EAX,ECX
004001A4  . 48            DEC EAX
004001A5  . C1E0 08       SHL EAX,8
004001A8  . AC            LODS BYTE PTR DS:[ESI]
004001A9  . FF53 FC       CALL DWORD PTR DS:[EBX-4]
004001AC  . 3D 007D0000   CMP EAX,7D00
004001B1  .v73 0A         JNB SHORT malware.004001BD
Return to 0040016B (malware.0040016B)
Jump from 00400132
```

```
Registers (FPU)
EAX 00000014
ECX 00000000
EDX FFFF8050
EBX 0040601C malware.0040601C
ESP 0012FFBC
EBP 00000014
ESI 004060F2 malware.004060F2
EDI 0040111E malware.0040111E

EIP 0040010F malware.0040010F

C 1  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 1  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 0038 32bit 7FFDE000(FFF)
T 0  GS 0000 NULL
D 0
O 1  LastErr ERROR_SUCCESS (00000000)

EFL 00000A17 (O,B,NE,BE,NS,PE,L,LE)

ST0 empty 0.0
ST1 empty 0.0
ST2 empty +UNORM 6C1C 0012BE64 0012C3E0
ST3 empty +UNORM 1AB5 00000015 0012BE98
ST4 empty -UNORM BE98 004A6E3B 0012BE80
ST5 empty 0.0000000156604956140e-4933
ST6 empty +UNORM 5B94 0012BE98 004A77D8
ST7 empty +UNORM 6F68 5C3A6522 00000000
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

```
Address   Hex dump       Disassembly                      Comment
00401000   6A 00         PUSH 0
00401002   E8 D5020000   CALL malware.004012DC
00401007   A3 10304000   MOV DWORD PTR DS:[403010],EAX
0040100C   E8 C5020000   CALL malware.004012D6
00401011   A3 08304000   MOV DWORD PTR DS:[403008],EAX
00401016   6A 0A         PUSH 0A
00401018   FF35 08304000 PUSH DWORD PTR DS:[403008]
0040101E   6A 00         PUSH 0
00401020   FF35 10304000 PUSH DWORD PTR DS:[403010]
00401026   E8 06000000   CALL malware.00401031
0040102B   50            PUSH EAX
0040102C   E8 9F020000   CALL malware.004012D0
00401031   55            PUSH EBP
00401032   8BEC          MOV EBP,ESP
00401034   83C4 A4       ADD ESP,-5C
00401037  vEB 0E         JMP SHORT malware.00401047
00401039   47            INC EDI
0040103A   65:6E         OUTS DX,BYTE PTR ES:[EDI]        I/O command
0040103C   65:72 69      JB SHORT malware.004010A8        Superfluous prefix
0040103F   635F 43       ARPL WORD PTR DS:[EDI+43],BX
00401042   6C            INS BYTE PTR ES:[EDI],DX         I/O command
00401043   61            POPAD
00401044  v73 73         JNB SHORT malware.004010B9
00401046   00C7          ADD BH,AL
00401048   45            INC EBP
00401049   D030          SAL BYTE PTR DS:[EAX],1
0040104B   0000          ADD BYTE PTR DS:[EAX],AL
0040104D   00C7          ADD BH,AL
0040104F   45            INC EBP
```

```
0012FFBC   0040016B  RETURN to malware.0040016B
0012FFC0   00401000  malware.00401000
0012FFC4   77E97D08  RETURN to kernel32.77E97D08
0012FFC8   00000000
0012FFCC   00000000
0012FFD0   7FFDF000
0012FFD4   00000000
0012FFD8   0012FFC8
0012FFDC   00000000
0012FFE0   FFFFFFFF  End of SEH chain
0012FFE4   77E8615B  SE handler
0012FFE8   77E97D10  kernel32.77E97D10
0012FFEC   00000000
0012FFF0   00000000
0012FFF4   00000000
0012FFF8   0040678D  OFFSET malware.<ModuleEntryPoint>
0012FFFC   00000000
```

Unpacking Performance



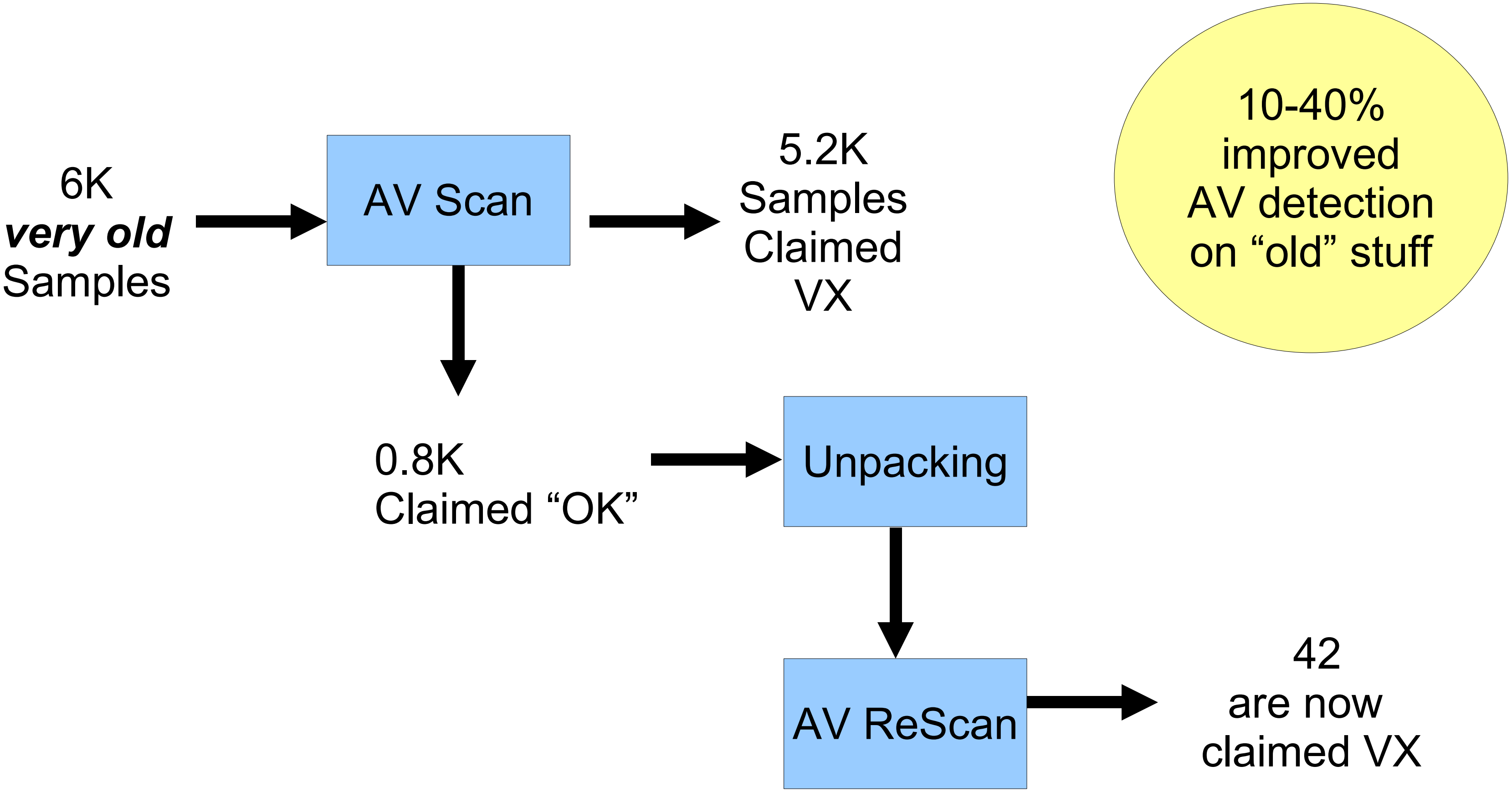
Results

- Detecting packing
  - 6K sample set
  - Compared with PEiD

Tool	Packed	Unpacked
PE iD	43.00%	53.00%
MalwareRepo	63.00%	37.00%

Results

- Improved AV detection



6K
very old
Samples

AV Scan

5.2K
Samples
Claimed
VX

10-40%
improved
AV detection
on “old” stuff

0.8K
Claimed “OK”

Unpacking

AV ReScan

42
are now
claimed VX

Repository User Classes

- Unknown users
  - Scripts, random users, even bots
- Humans
  - CAPTCHA-verified
- Authenticated Users
  - Known trusted contributors

Repository Access Goals

- Unknown users
  - Upload; view aggregate statistics
- Humans
  - Upload; download analysis of their samples
- Authenticated Users
  - Upload; download all; access analysis

Hub/Spoke Structure

- Hub: web server, file store, database, authentication system – mirrorable
- Spokes: unpacking and analysis partners
  - receive a feed showing new malware
  - can download any/all of it
  - can upload unpacked versions, output of their in-house (proprietary) analyzers
  - can advertise value-added in-house content

Economic Goals

- Economics is about *human action* not simply money
- So, what do we want people to do, or stop doing, or do differently?
- *Act in their own best interests*, of course!
- So, we intend to make the benefits of sharing more intuitive to an AV CEO

Social Goals

- In human (biological) viruses, disclosure is an obligation – hoarding is unthinkable
- Somehow when the virus is not biological, hoarding is thinkable
- Is it life-safety that makes the difference, or is it profitability?
- We see no necessary conflict between sharing and profitability

Conclusion

- *Service*-oriented repository
- See tisf.net for details
- Questions?