

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

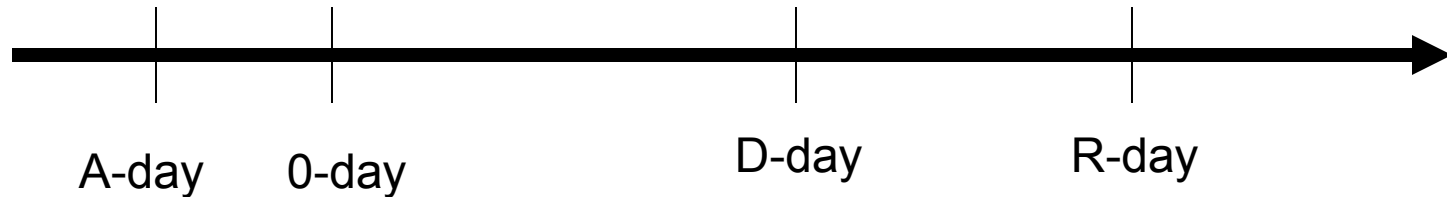
- Numerous private, semi-public malware collections
 - Need trust to join (for some value of “trust”)
 - “Too much sharing” often seen as competitive disadvantage
- Incomplete collections: reflect sensor bias
 - Darknet-based collection
 - IRC surveillance
 - Honeypot-based collection

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

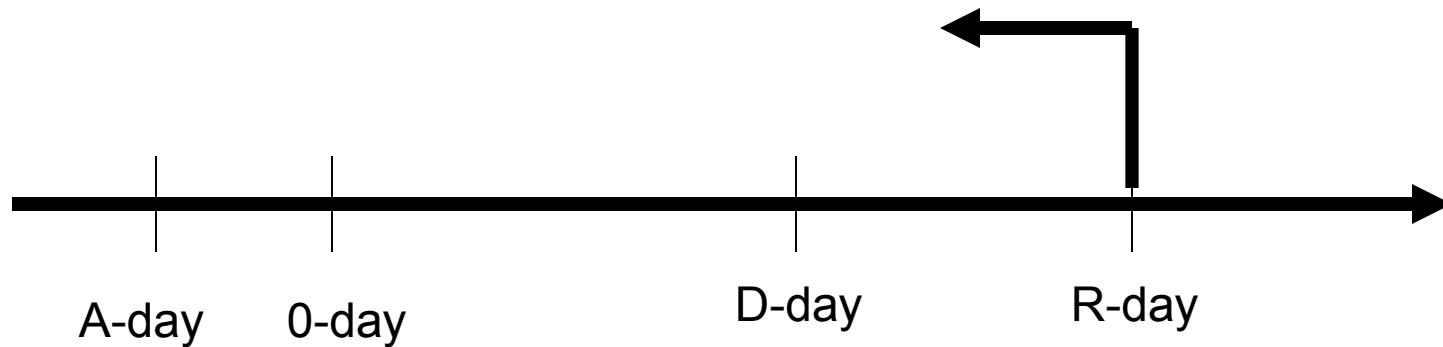
0-day: release

D-day: first opportunity for detection

R-day: response (e.g., virus signature update)

Malware Life Cycle

Recent AV goal: reduce response time

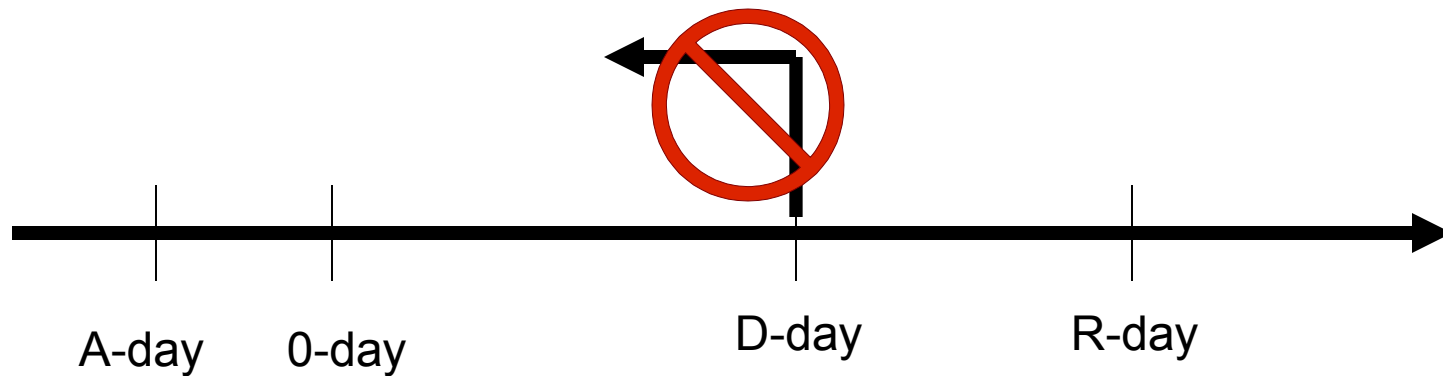


AV update cycles previously measured weeks/days

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

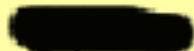
- Technique
 - Malware authors compile *single*, unique virus;
 - Send to suspected sensor
 - Wait and watch for updates

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.

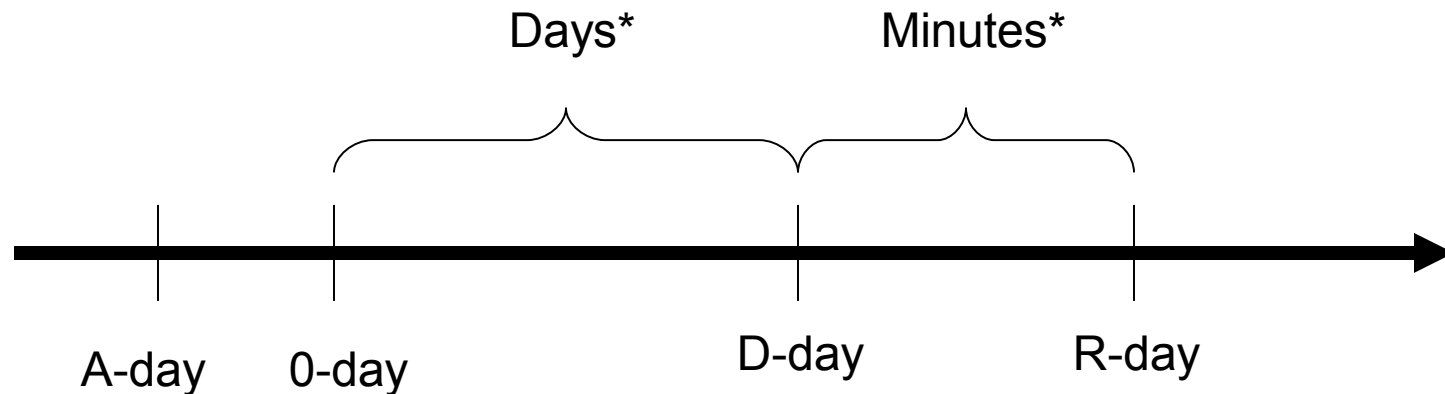
 << осторожно! Не трогайте.

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

WARNING: '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++



* 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

[Go to botnet controller](#) [Compress logger.txt to logger.gz](#)

Remark: displayed 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"

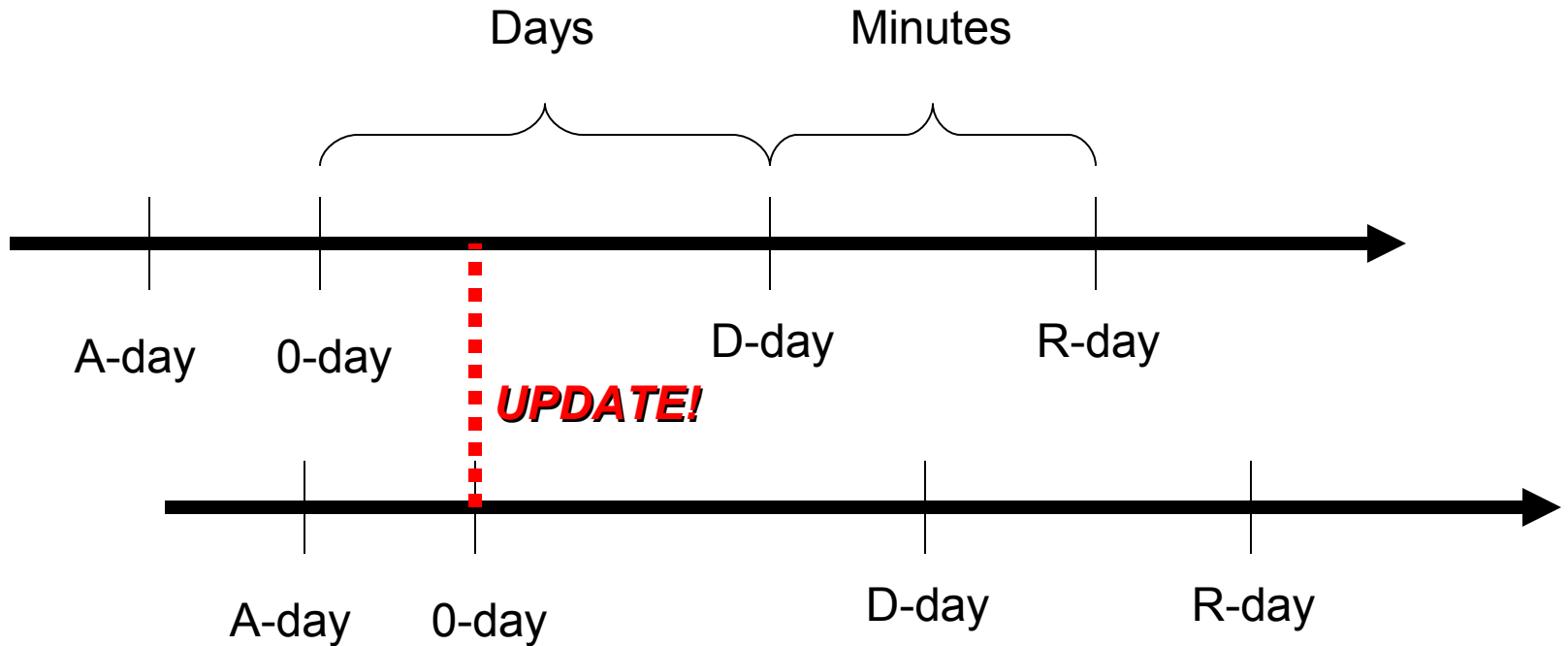
Select by country:

Select by state:

Current country selected: all
Current state selected: all

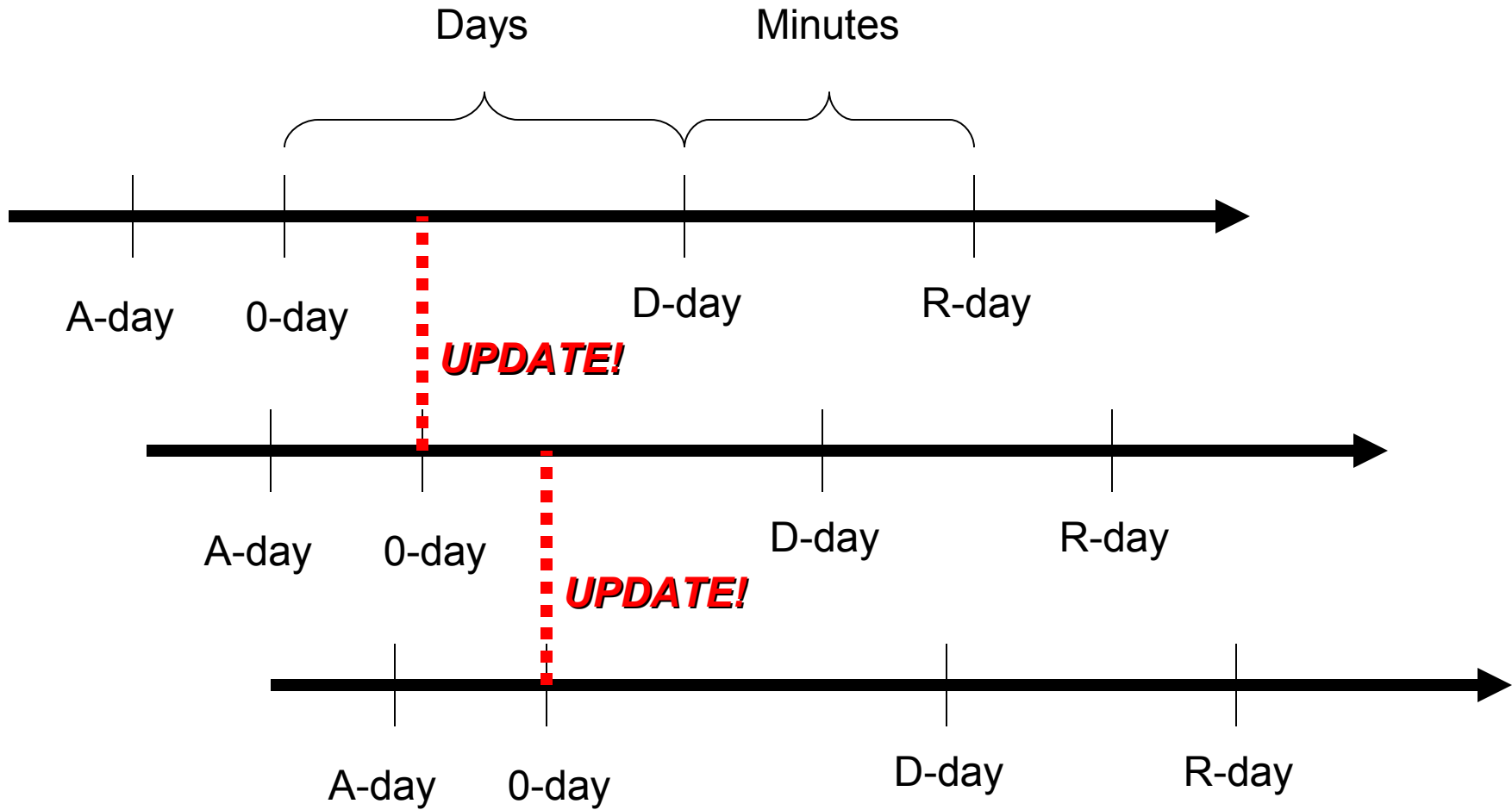
| List | | | |
|---------------------------------------|-------|--|---------------|
| IP | SOCKS | ID | COUNTRY |
| Copy IP [REDACTED] | 24065 | Copy ID KIWTRQVIRLBZOBEAYIJMMQAQHOBACPR | United States |
| Copy IP [REDACTED] | 34733 | Copy ID UKBLHDIQLRKLKLMZKKNSWHXRCRTQMVMRQ | |
| Copy IP [REDACTED] | 45726 | Copy ID EDKHVPYQYSLFQJPPNNIBSJRPSPGJAYR | Taiwan |
| Copy IP [REDACTED] | 18881 | Copy ID ZEXEENQHGTIYMGVKMEICDBQLDQTKCJL | |

Malware Life Cycle

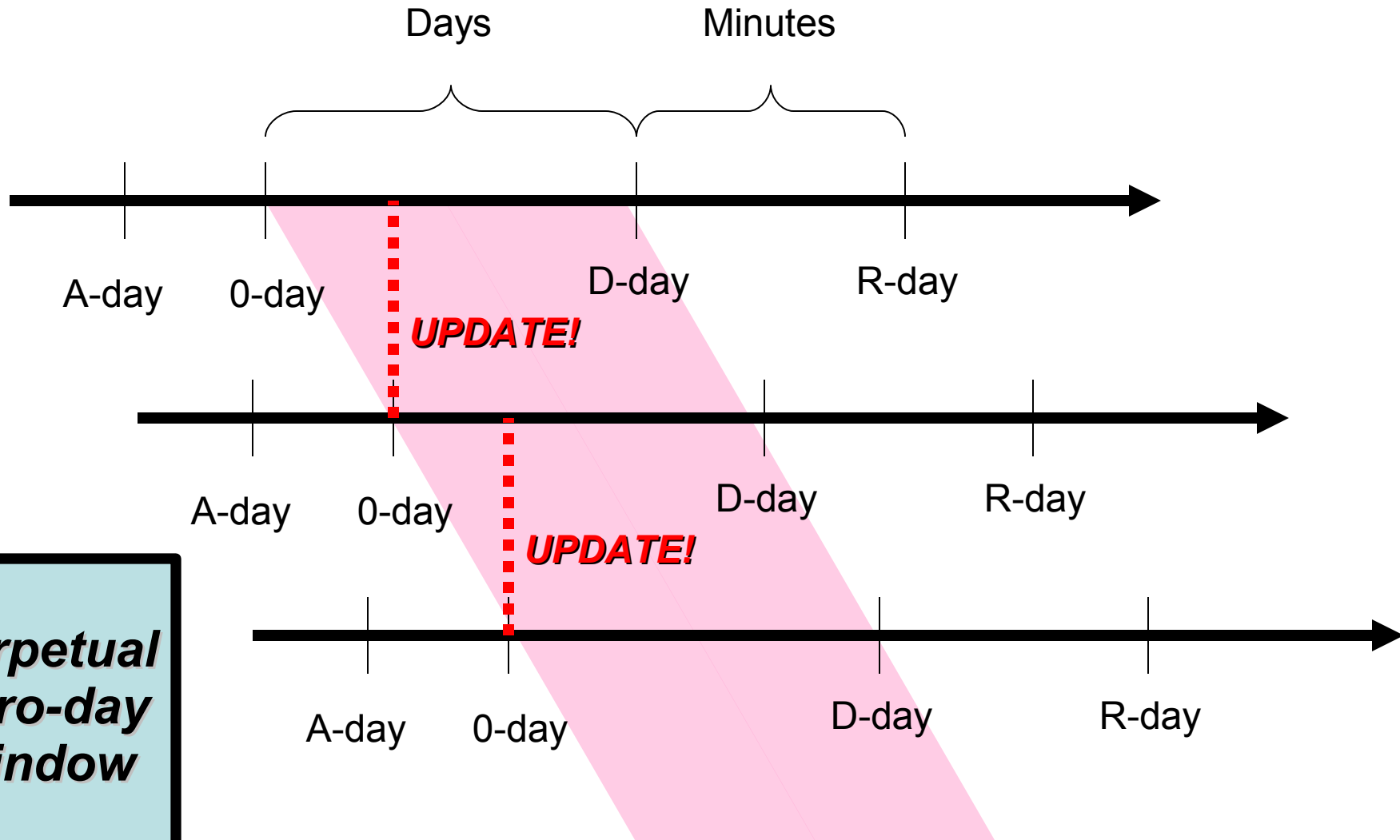


Bot runs for $\sim 1/2$ day, and updates to new, evasive binary

Malware Life Cycle

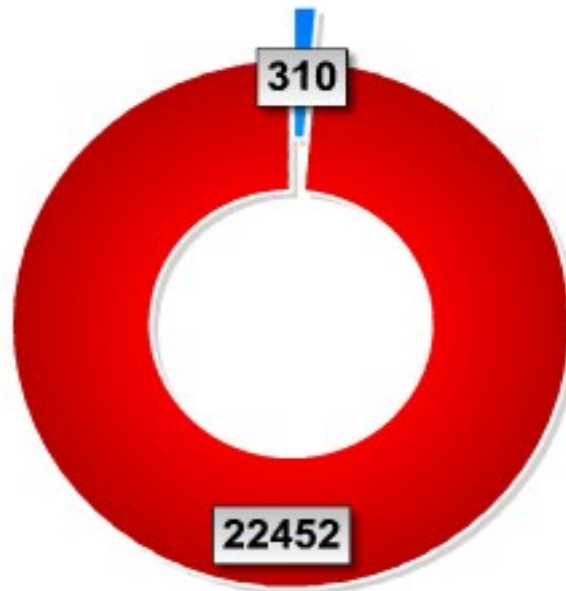


Malware Life Cycle



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”

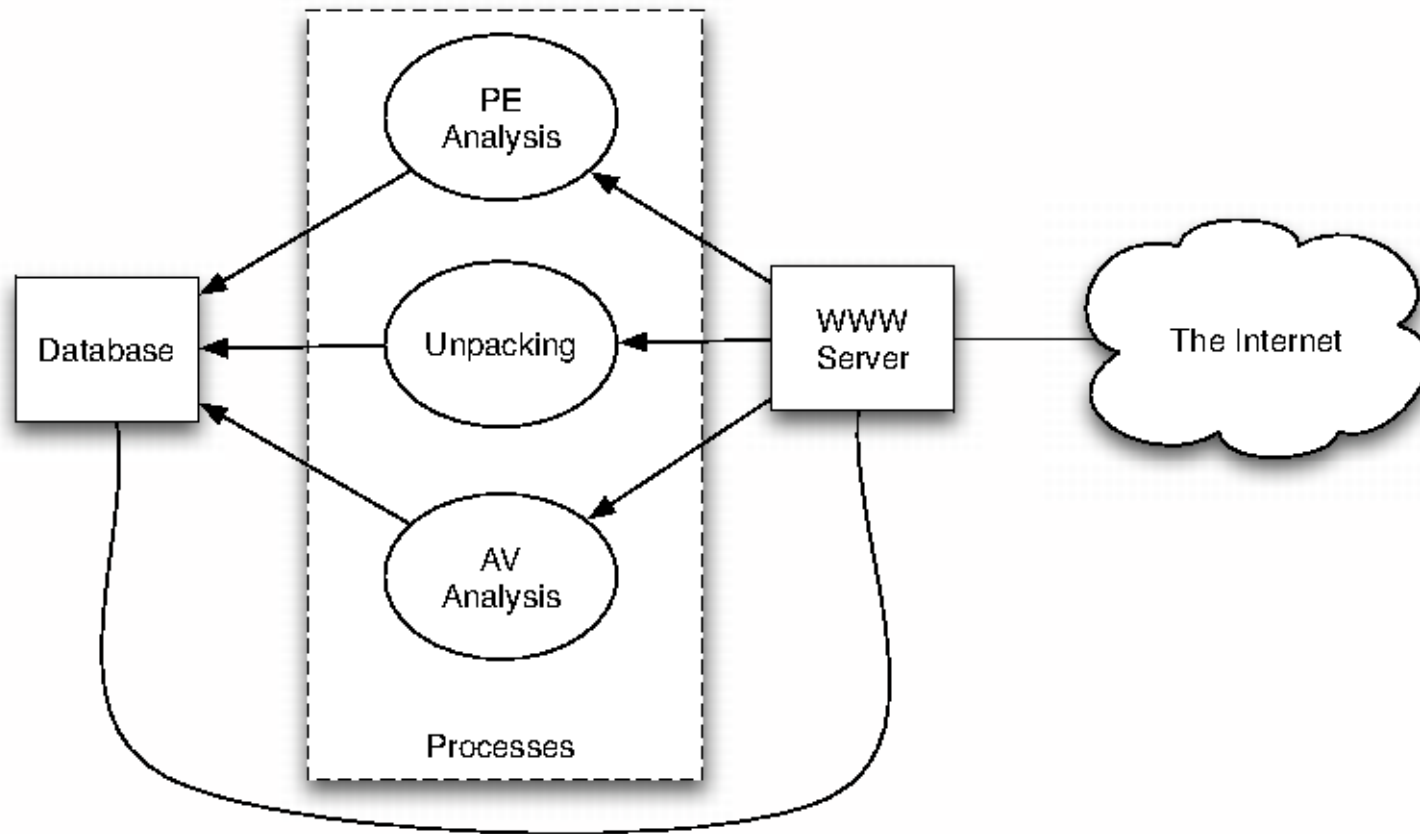
Malware Repo Requirements

- 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.)

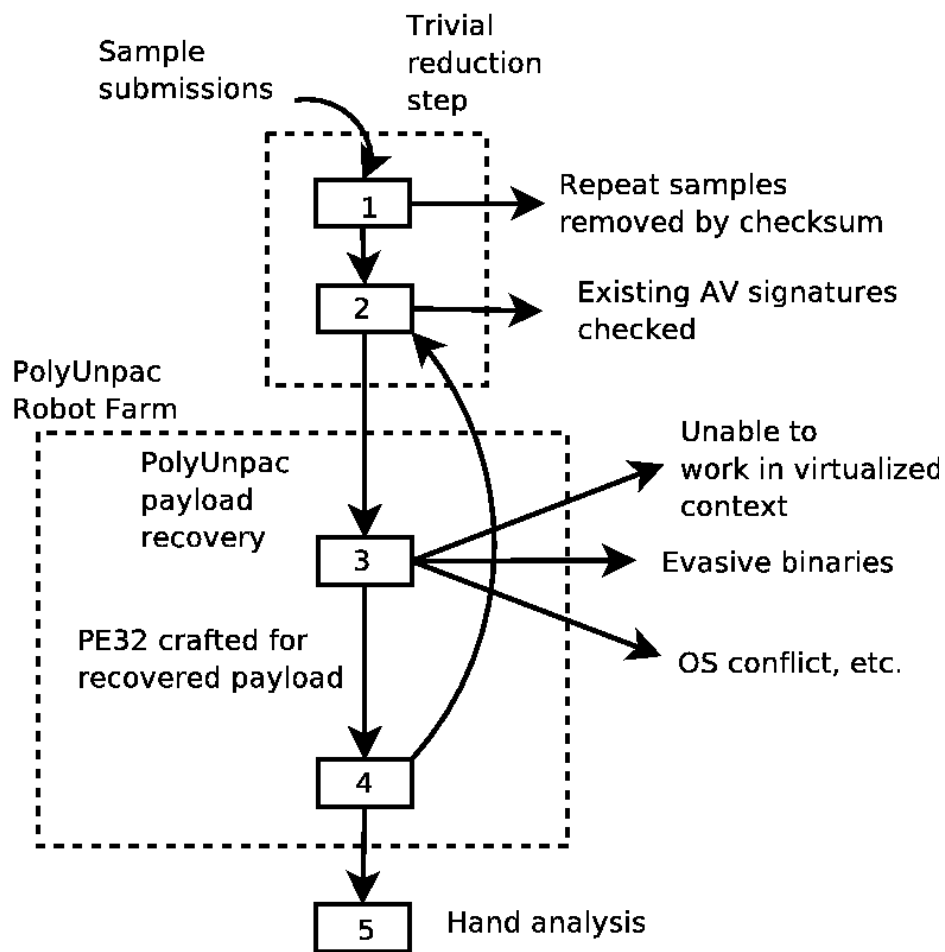
Approach: Service-Oriented Repository

- 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



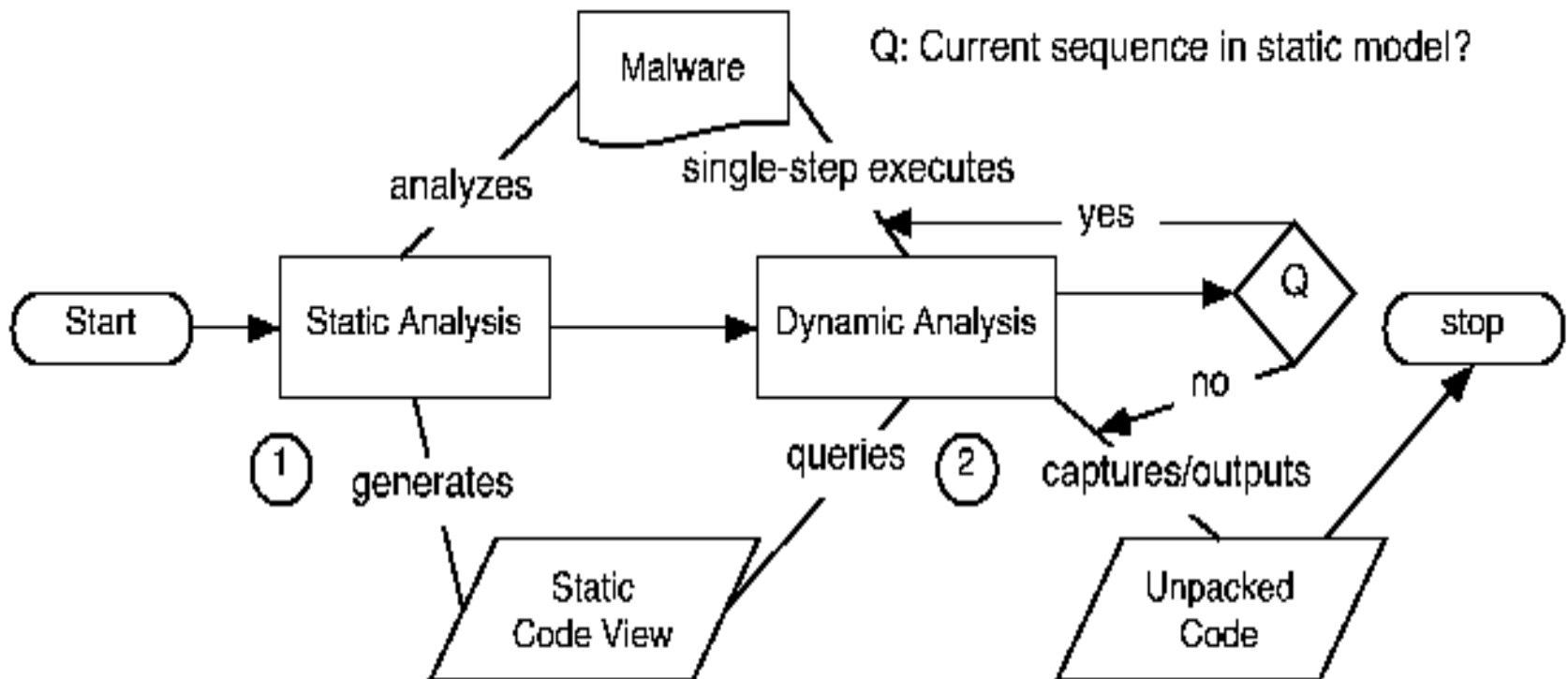
Work Flow



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



Unpacking Example

The screenshot displays the OllyDbg interface for a malware executable. The main window shows assembly code for the CPU - main thread, module malware. The registers window shows the state of the FPU registers. The disassembly window shows the assembly code with hex dump and comments.

Assembly Code (CPU - main thread, module malware):

```
00400168 ^73 F9 JNB SHORT malware.00400166
00400169 . 3D09 XOR ECX,ECX
0040016F . FF13 CALL DWORD PTR DS:[EBX]
00400171 ^73 16 JNB SHORT malware.00400189
00400173 . 3D09 XOR EAX,EAX
00400175 . FF13 CALL DWORD PTR DS:[EBX]
00400177 ^73 21 JNB SHORT malware.0040019A
00400179 . 86 80 MOV DH,80
0040017B . 41 INC ECX
0040017C . B0 10 MOV AL,10
0040017E > FF13 CALL DWORD PTR DS:[EBX]
00400180 . 1209 ADC AL,AL
00400182 ^73 FA JNB SHORT malware.0040017E
00400184 . 75 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 . 82F6 ADD DH,DH
00400190 . SDB9 01 SDB 01
00400193 ^75 0E JNZ SHORT malware.004001A3
00400195 . FF53 FC CALL DWORD PTR DS:[EBX-4]
00400198 . 75 26 JMP SHORT malware.004001C0
0040019A > AC LODS BYTE PTR DS:[ESI]
0040019B . 01E8 SHR EAX,1
0040019D ^74 2F JE SHORT malware.004001CE
0040019F . 13C9 ADC ECX,ECX
004001A1 ^EB 1A JMP SHORT malware.004001BD
004001A3 > 91 DEX ECX
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 ^73 0A JNB SHORT malware.004001BD
```

Registers (FPU):

```
ERX 00000014
ECX 00000000
EDX FFFFFFFF
EBX 0040601C malware.0040601C
ESP 0012FFDC
EBP 00000014
ESI 004060F2 malware.004060F2
EDI 0040111E malware.0040111E
EIP 0040010F malware.0040010F
C 1 ES 0023 32bit 0FFFFFFFFF
F 1 CS 0010 32bit 0FFFFFFFFF
A 1 SS 0023 32bit 0FFFFFFFFF
Z 0 DS 0023 32bit 0FFFFFFFFF
S 0 FS 0030 32bit 7FDE0000(FFF)
T 0 GS 0000 NULL
D 0
O 1 LastErr ERROR_SUCCESS (00000000)
EPL 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 00406E38 0012BE98
ST5 empty 0.000000015664956140e-4933
ST6 empty +UNORM 5E94 0012BE98 004A77D8
ST7 empty +UNORM 6F68 5C3A6E22 00000000
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
```

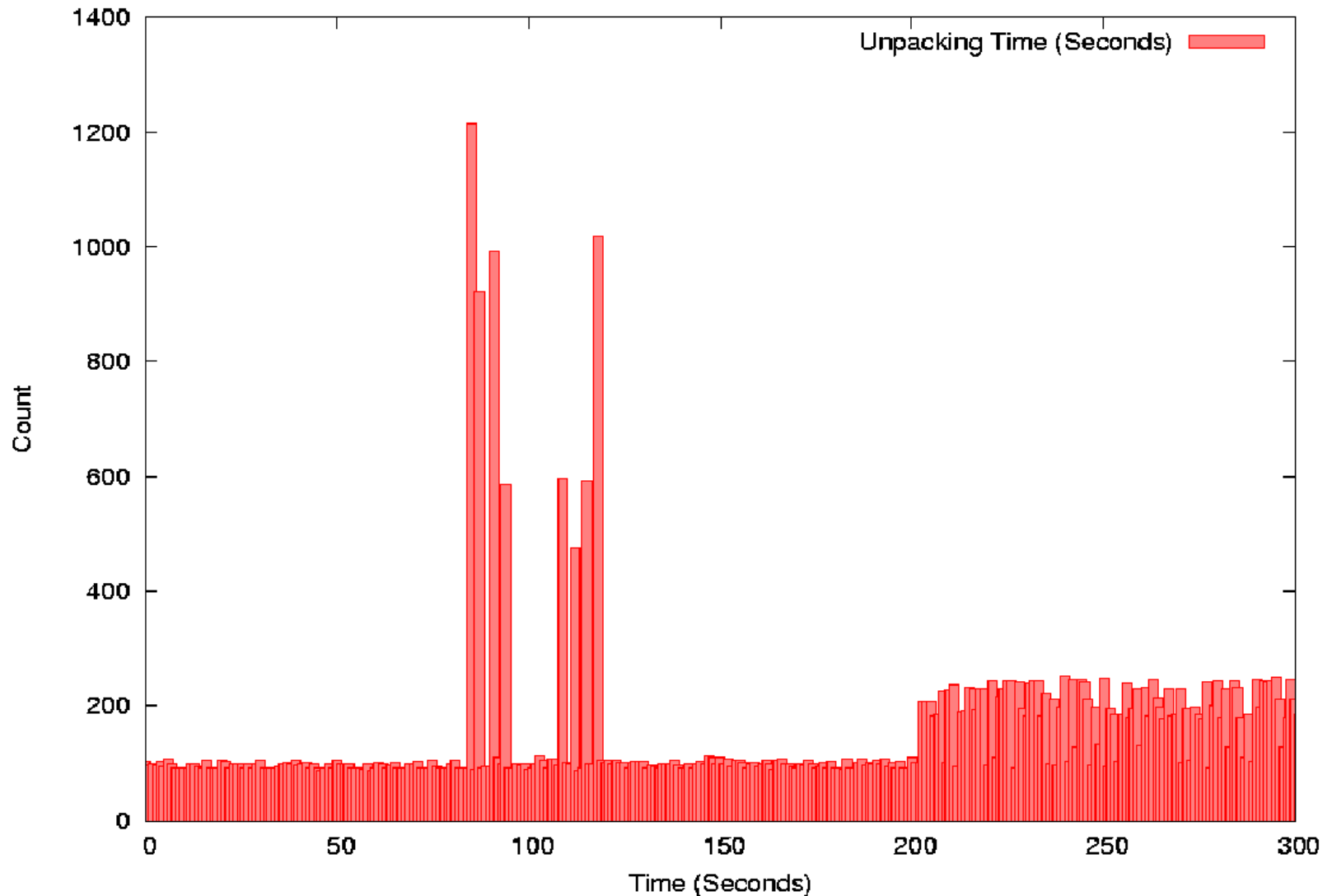
Disassembly:

| 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 | 83 C5020000 | CALL malware.004012DC | |
| 00401011 | A3 00304000 | MOV DWORD PTR DS:[403000],EAX | |
| 00401015 | 6A 00 | PUSH 0A | |
| 00401019 | FF35 00304000 | PUSH DWORD PTR DS:[403000] | |
| 0040101E | 6A 00 | PUSH 0 | |
| 00401020 | FF35 10304000 | PUSH DWORD PTR DS:[403010] | |
| 00401025 | E3 06000000 | CALL malware.00401031 | |
| 0040102B | 50 | PUSH EAX | |
| 0040102C | E8 9F020000 | CALL malware.004012D0 | |
| 00401031 | 55 | PUSH EBP | |
| 00401032 | 88EC | MOV ESP,ESP | |
| 00401034 | 83C4 A4 | ADD ESP,-A4 | |
| 00401037 | ^EB 0E | JMP SHORT malware.00401047 | |
| 00401039 | 47 | INC EDI | |
| 0040103A | 65:16 | OUTS DX, BYTE PTR ES:[EDI] | I/O command |
| 0040103C | 65:72 69 | JB SHORT malware.004010A8 | Superfluous prefix |
| 0040103F | 635F 43 | RRR WORD PTR DS:[EDI+43],BX | |
| 00401042 | 6C | INS BYTE PTR ES:[EDI],DX | I/O command |
| 00401043 | 61 | POPAD | |
| 00401044 | ^73 73 | JNB SHORT malware.004010B9 | |
| 00401045 | 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 | |

Disassembly (continued):

```
0012FF8C 00400168 RETURN to malware.00400168
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 77E9615B SE handler:
0012FFE8 77E97D10 kernel32.77E97D10
0012FFEC 00000000
0012FFF0 00000000
0012FFF4 00000000
0012FFF8 0040678D OFFSET malware.<ModuleEntryPoint>
0012FFFC 00000000
```

Unpacking Performance



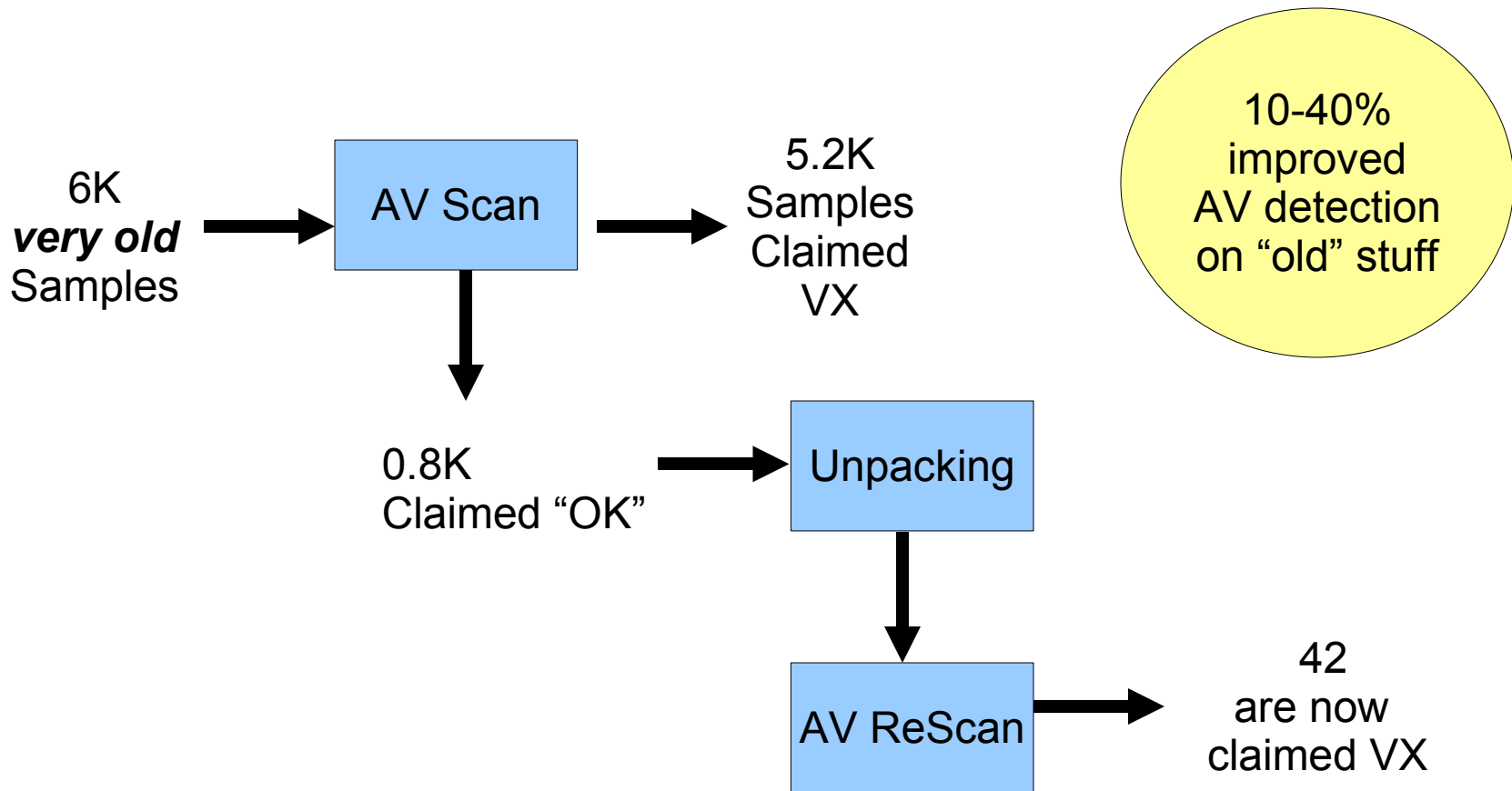
Results

- 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



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?