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

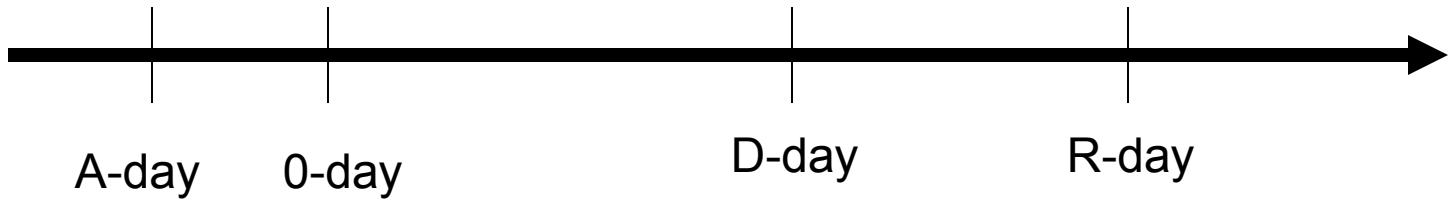
- Numerous private, semi-public malware collections
  - Need trust to join (for some value of “trust”)
  - “Too much sharing” often seen as competitive disadvantage
- Incomplete collections: reflect sensor bias
  - Darknet-based collection
  - IRC surveillance
  - Honeypot-based collection

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

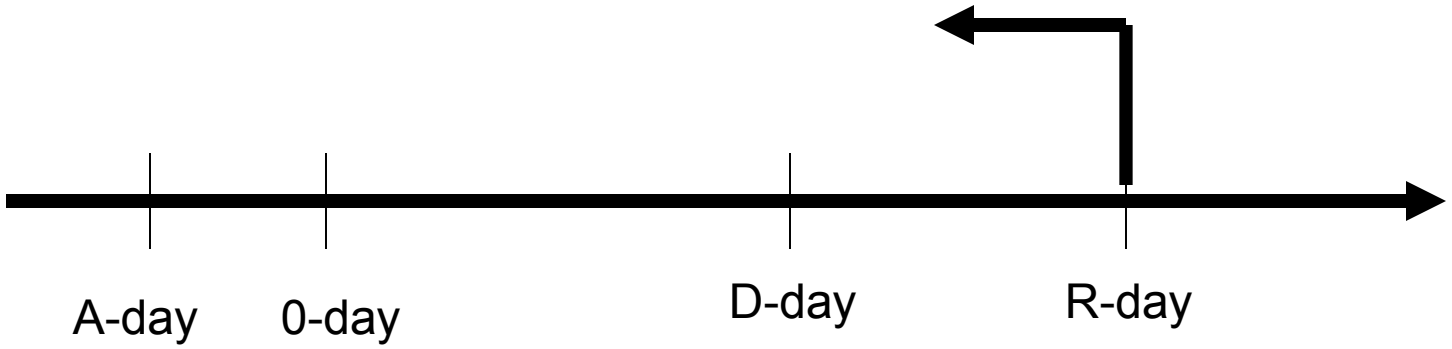
Four conceptual phases of malware life cycle:



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

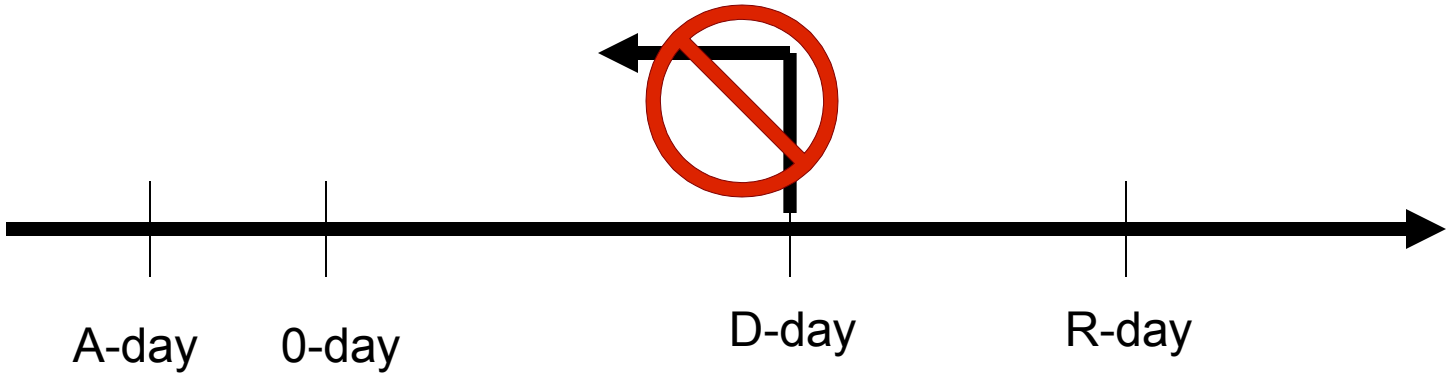


AV update cycles previously measured weeks/days

Now measured in hours/minutes (or should be)

Malware Life Cycle

How to improve detection time...

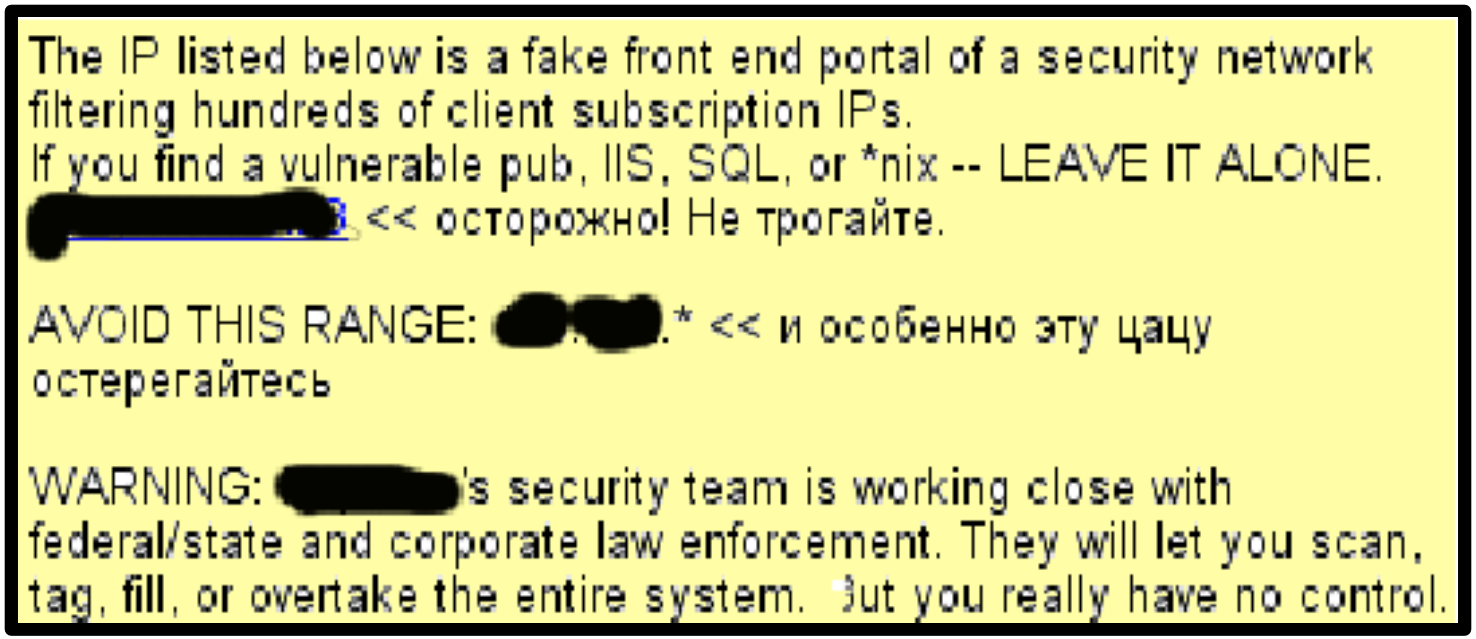


Given that...

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

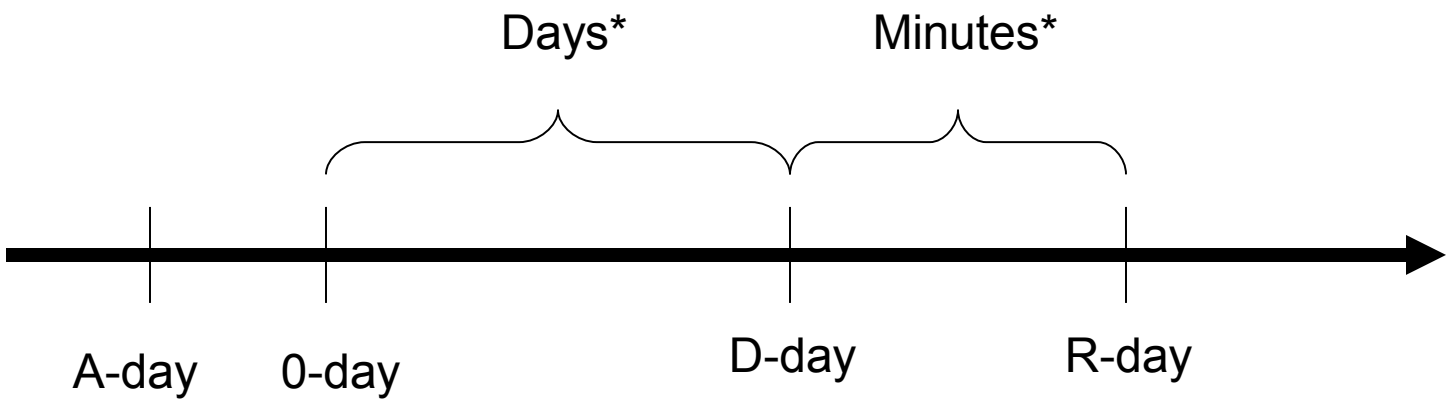
Sensor Illumination

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



Malware Life Cycle

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

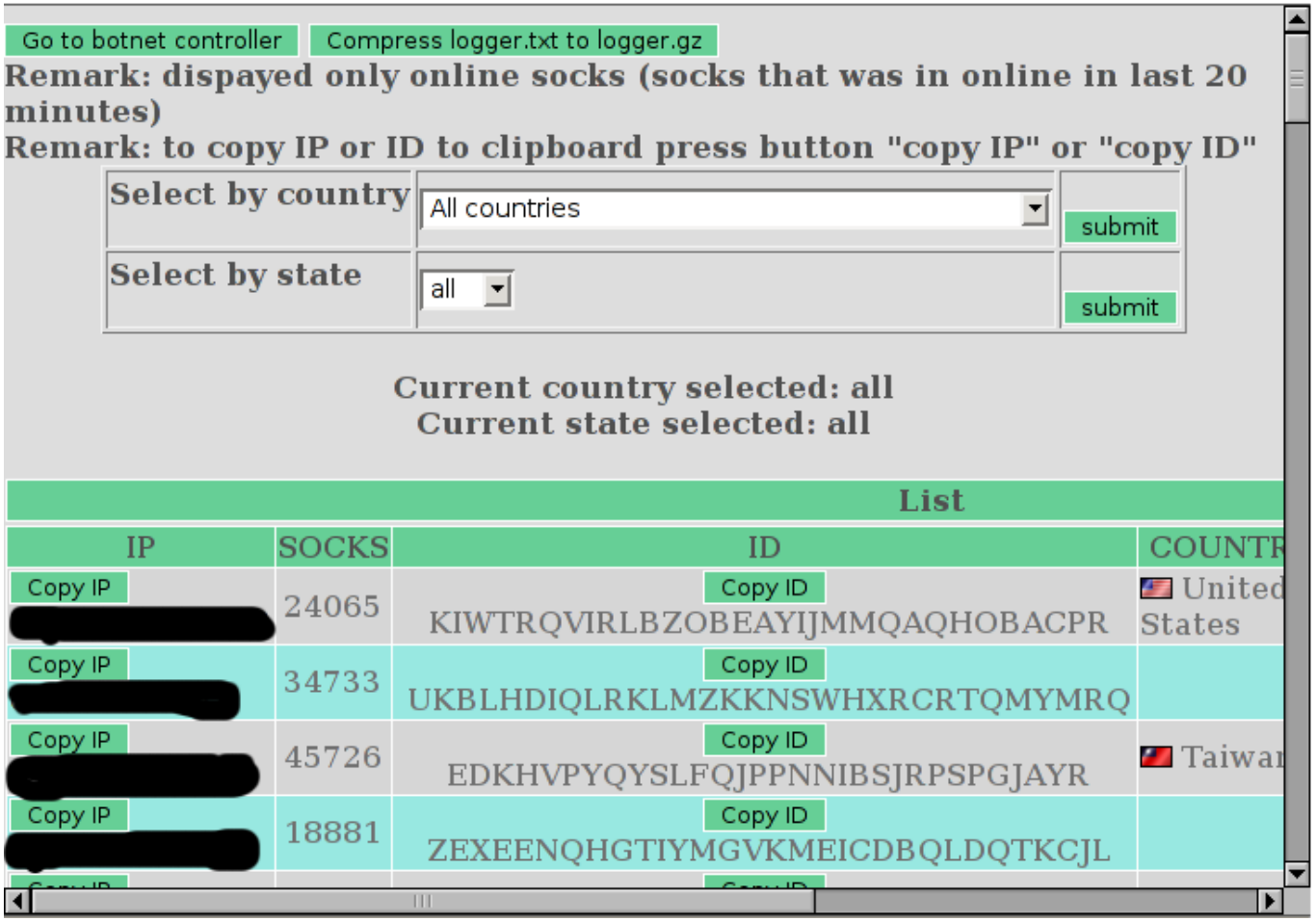


* Average order of time; anecdotes will vary

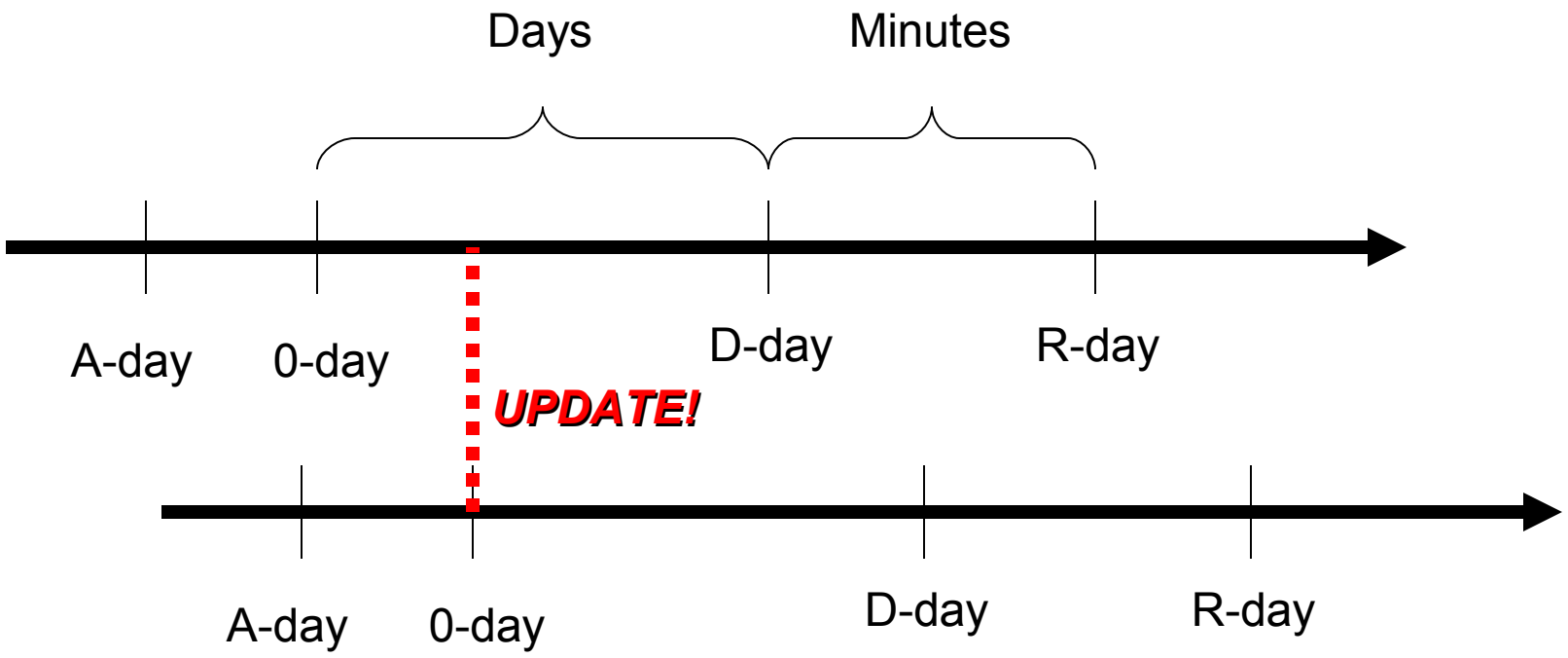
Queen Bot Programs

- 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

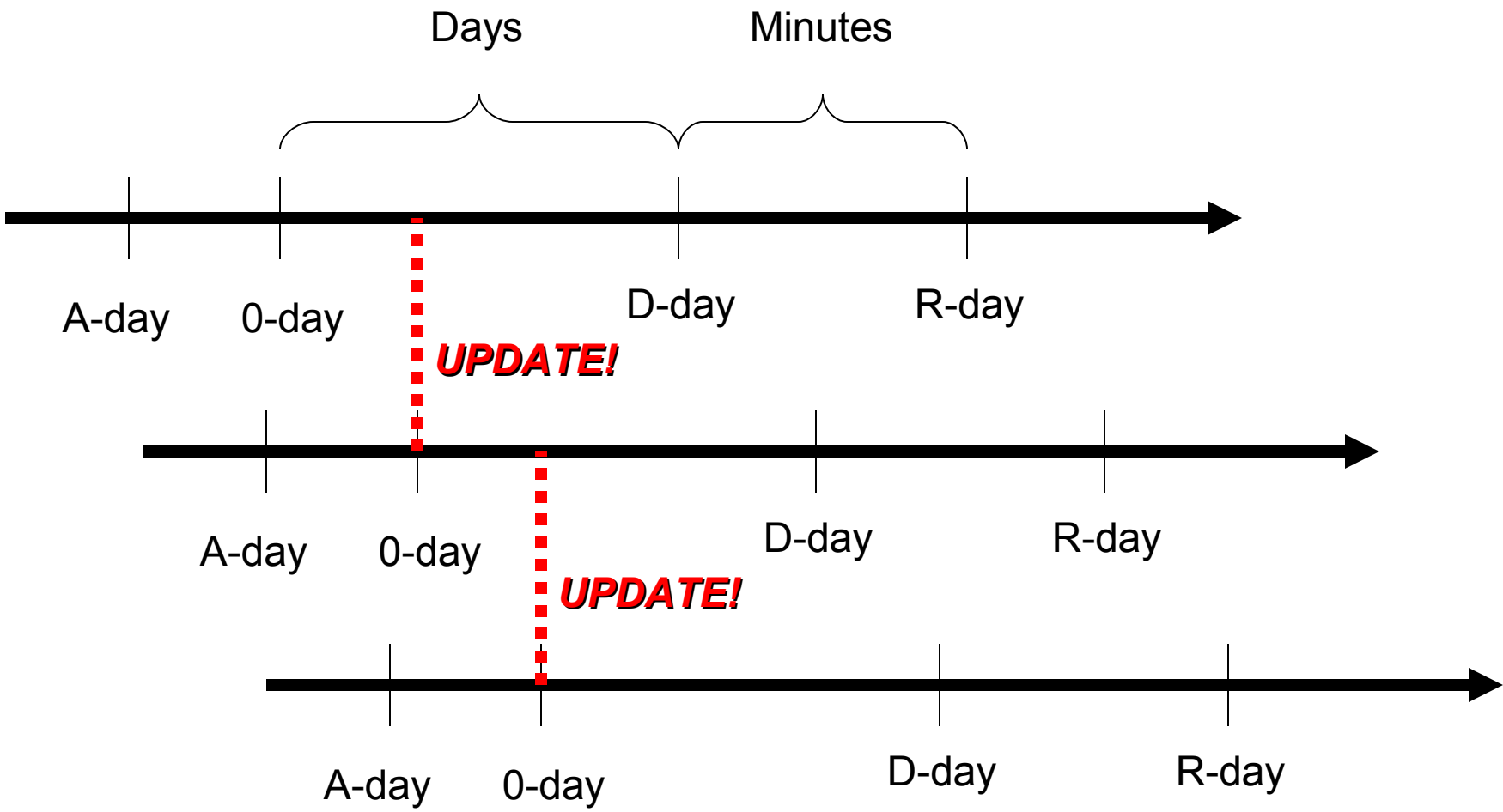


Malware Life Cycle

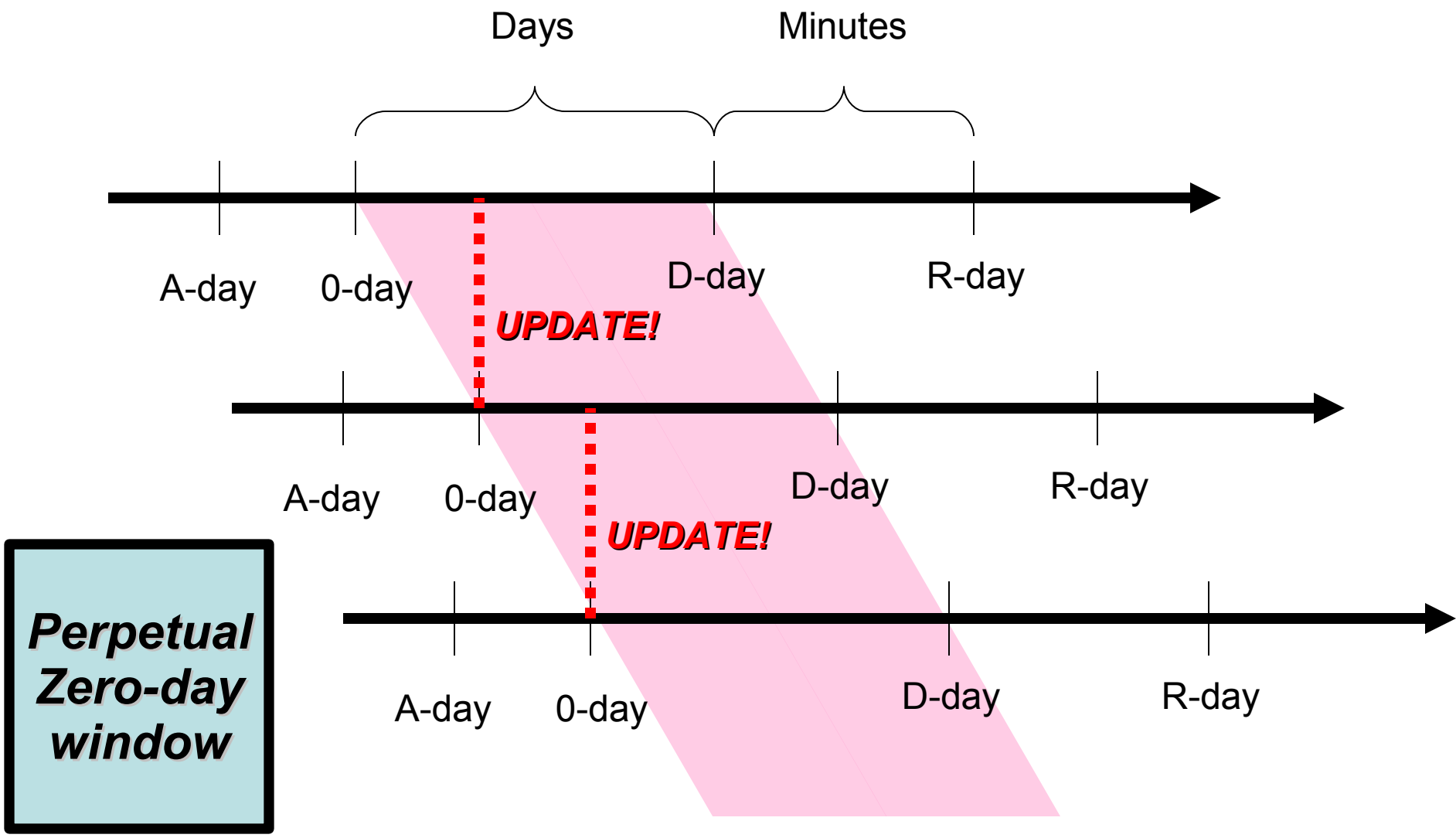


Bot runs for ~1/2 day, and updates to new, evasive binary

Malware Life Cycle

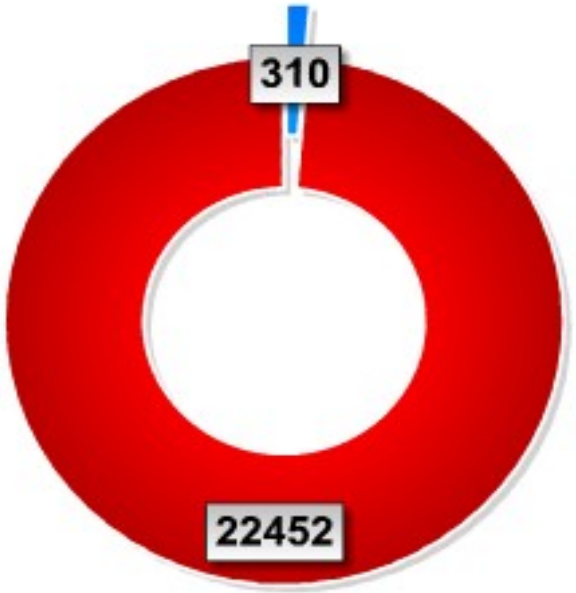


Malware Life Cycle



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”

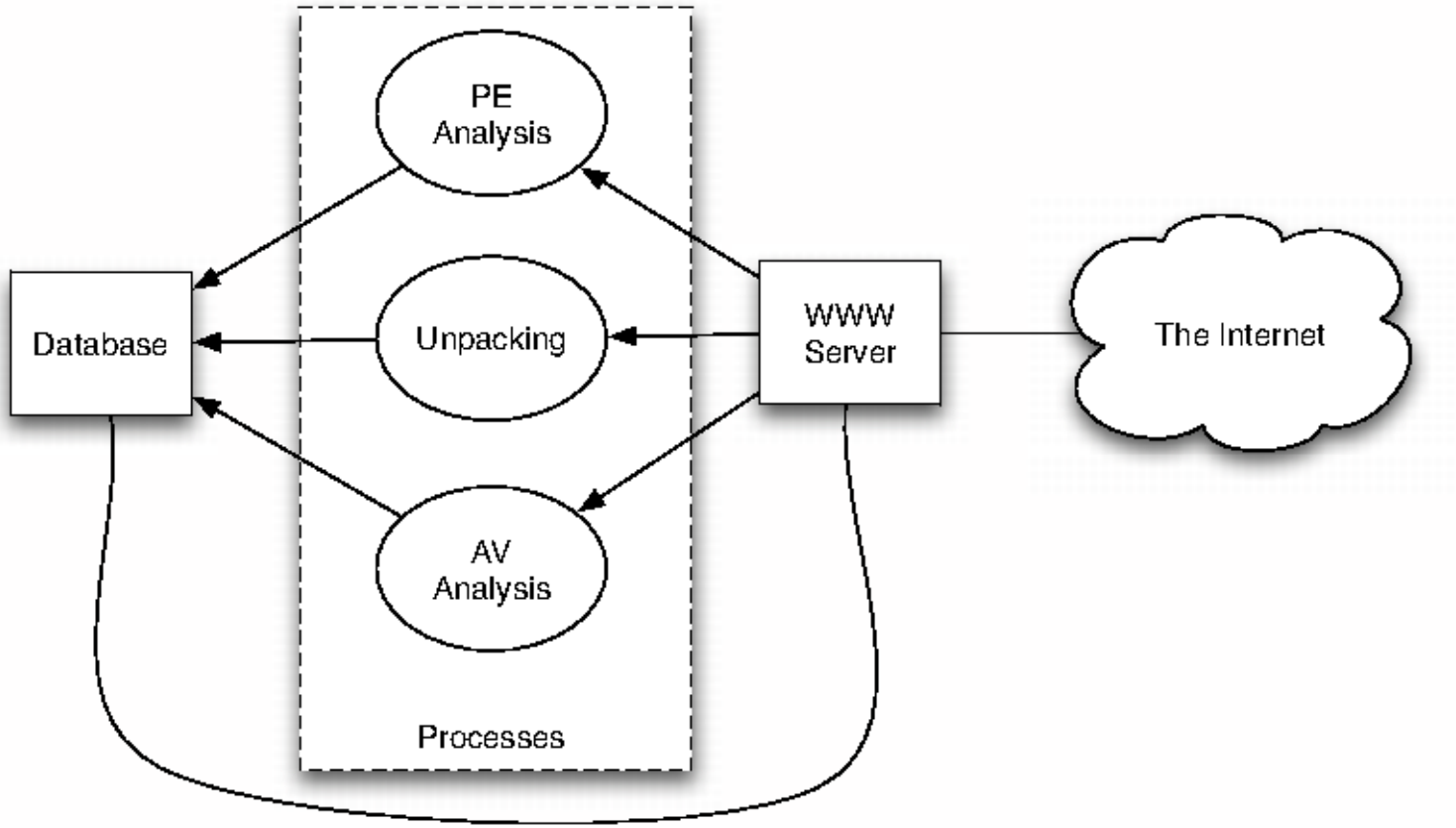
Malware Repo Requirements

- 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.)

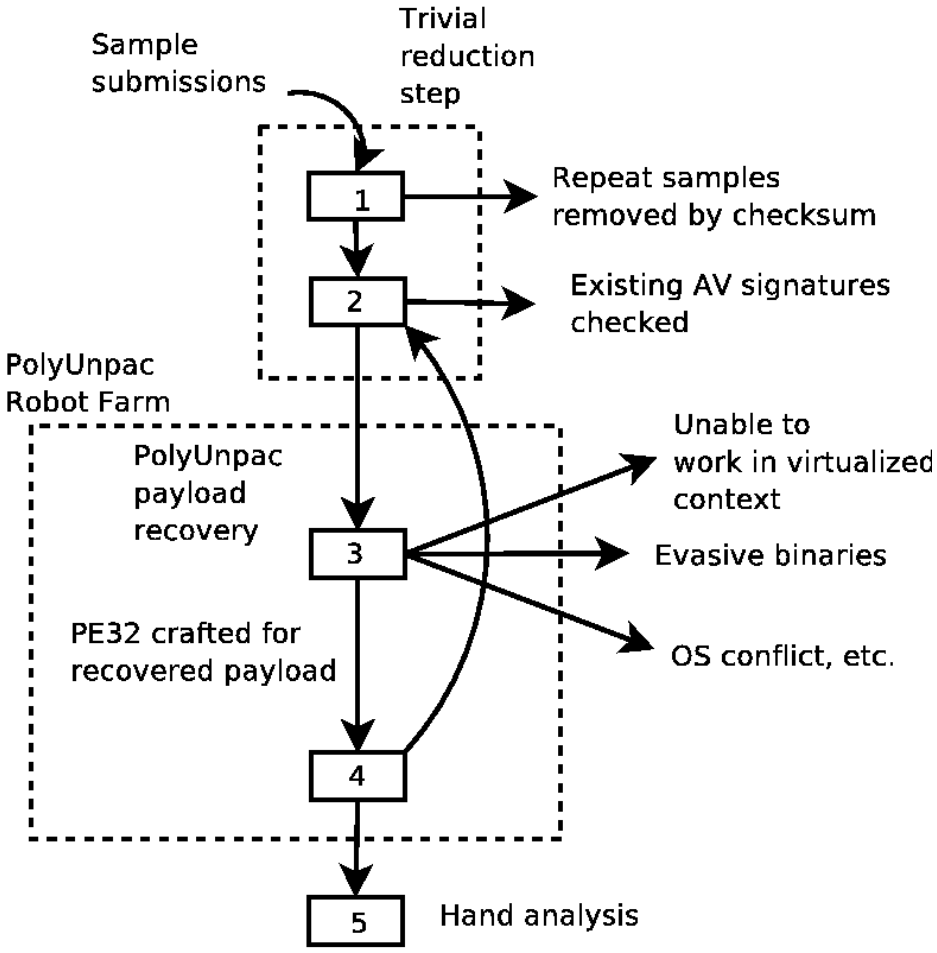
Approach: Service-Oriented Repository

- 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


PE
Analysis
Unpacking
AV
Analysis
Processes
Database
WWW
Server
The Internet

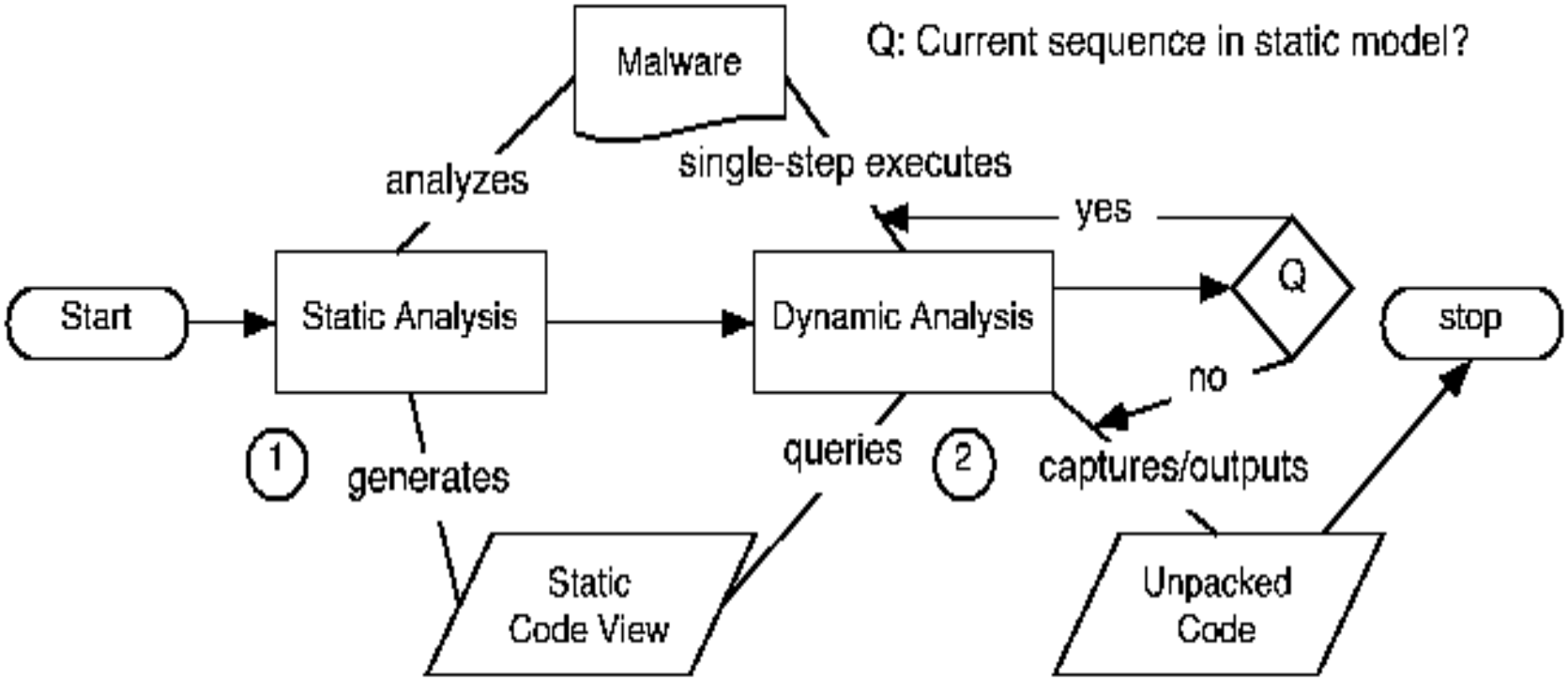
Work Flow


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

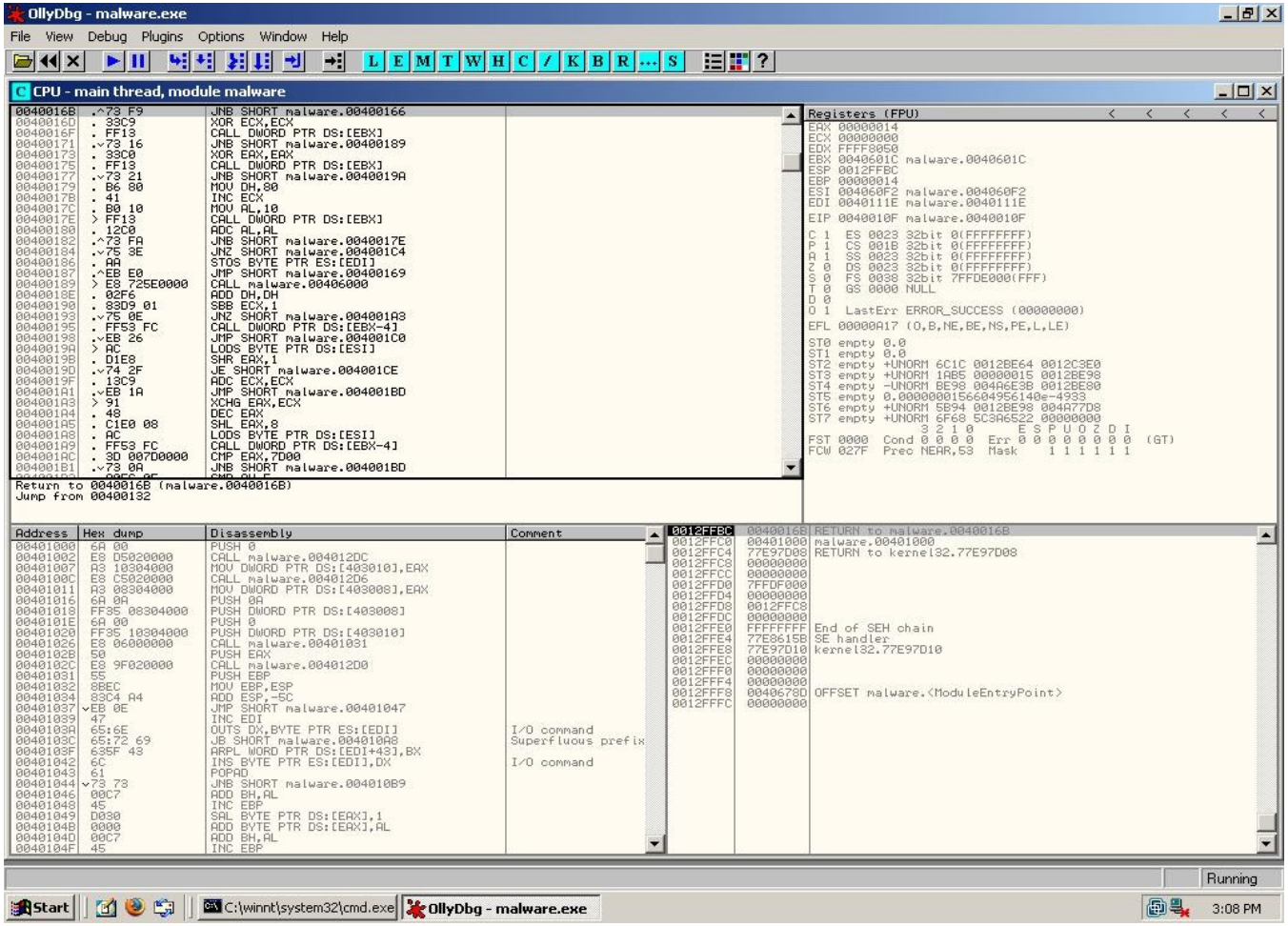
Unpacking

- 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

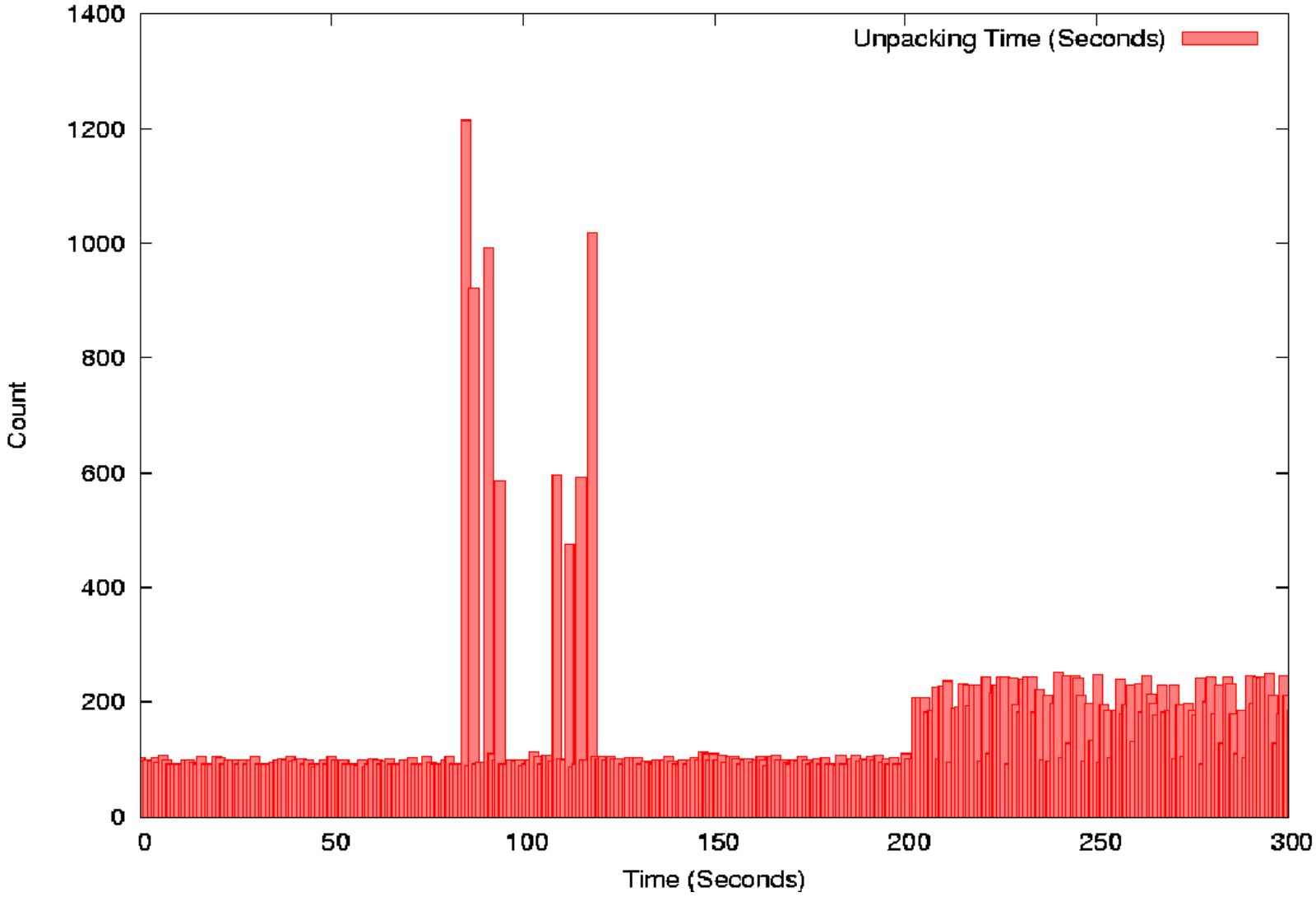
Unpacking Heuristic



Unpacking Example



Unpacking Performance



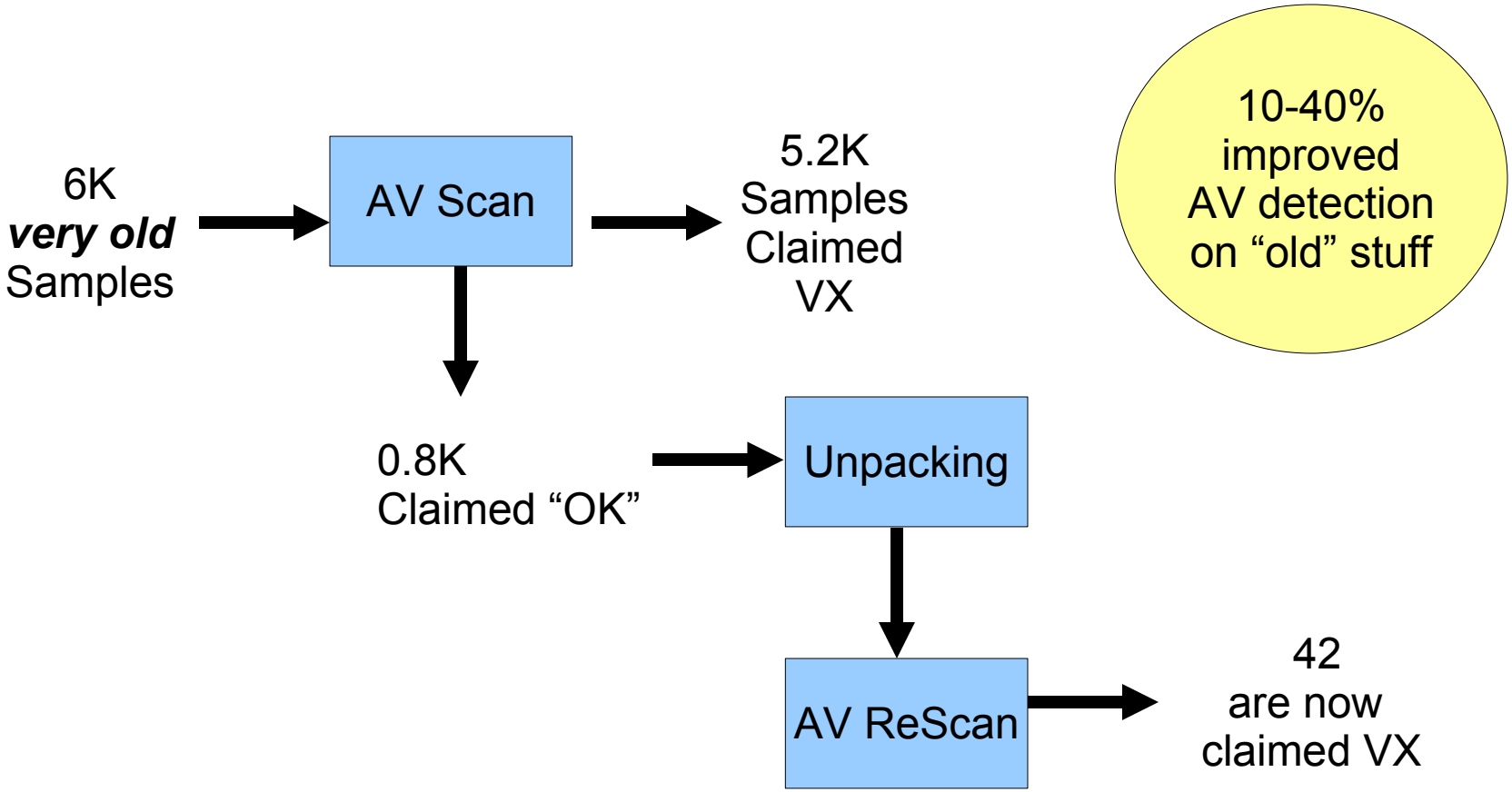
Results

- 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?