Malware Repository Update

David Dagon
Georgia Institute of Technology
dagon@cc.gatech.edu

Context

- OARC is contemplating the operation of a malware repository
- I report on the implementation of this repository
  - Design rationale
  - Demo
  - Other developments that I trust may be received as good news
- These slides expand on a previous talk w/ Paul Vixie at Defcon
  - Errors in both are my own

Overview

- How malware is collected and shared now
- Malfease’s service-oriented repository
  - Automated unpacking
  - Header analysis
- Demonstration
- Policy considerations for OARCs operation

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
  - Quotas often used
- Incomplete collections: reflect sensor bias
  - Darknet-based collection
  - IRC surveillance
  - Honeypot-based collection

Shortcomings

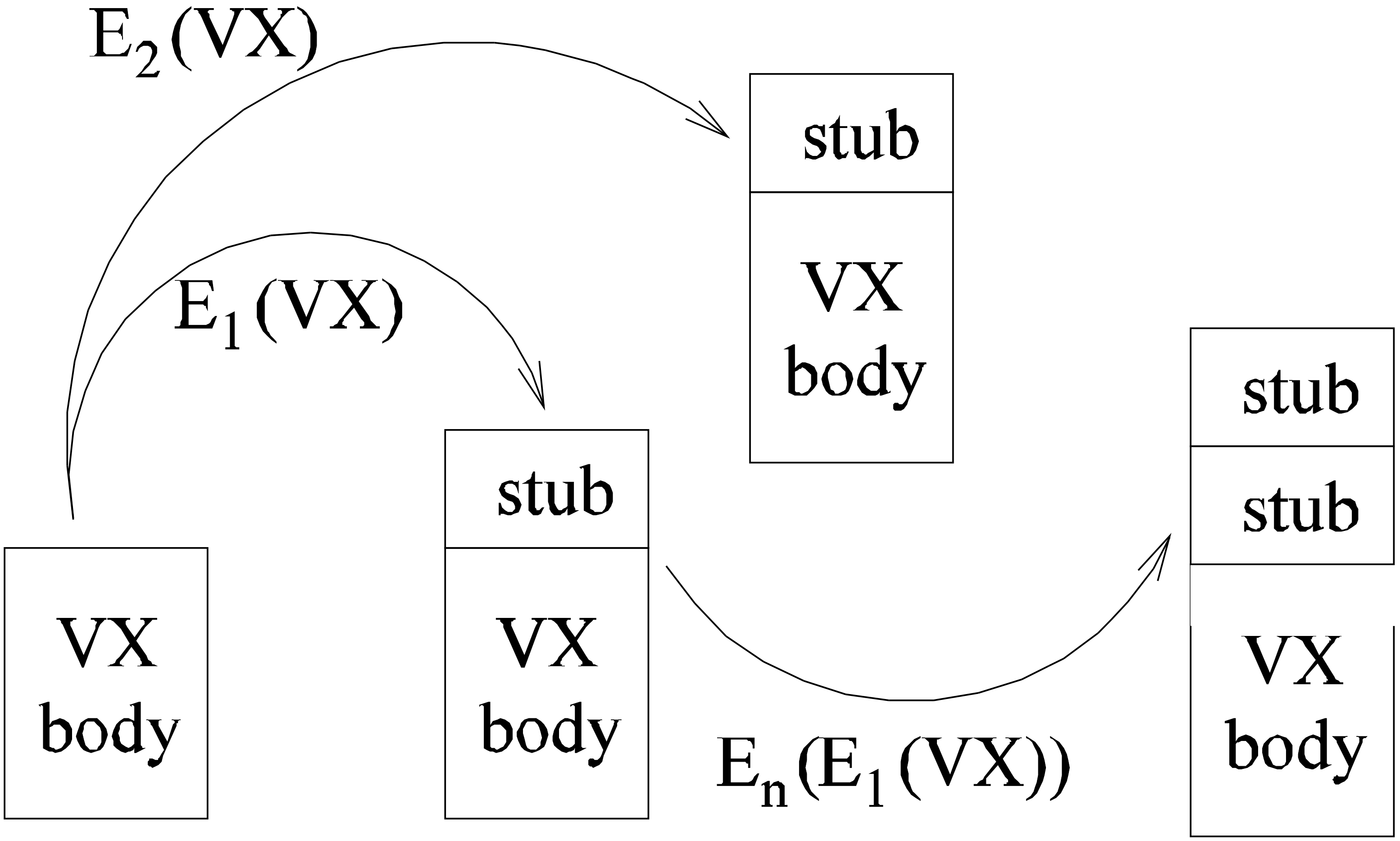
- Malware authors know and exploit weaknesses in data collection
- Illuminating sensors
  - “Mapping Internet Sensors with Probe Response Attacks”, Bethencourt, et al., Usenix 2005
- Automated victims updates
  - “Queen-bot” programs keep drones in 0-day window

Queen-Bot Programs

- Malware authors use packers
  - Encrypted/obfuscated payloads
  - Small stub programs to inflate the payload

- Queen bots
  - Automate the creation of new keys, binaries
  - Each new packed program is different
    - But the same semantic program
  - Compiler tricks used
    - Dead code injected, idempotent statements introduced, register shuffling, etc.

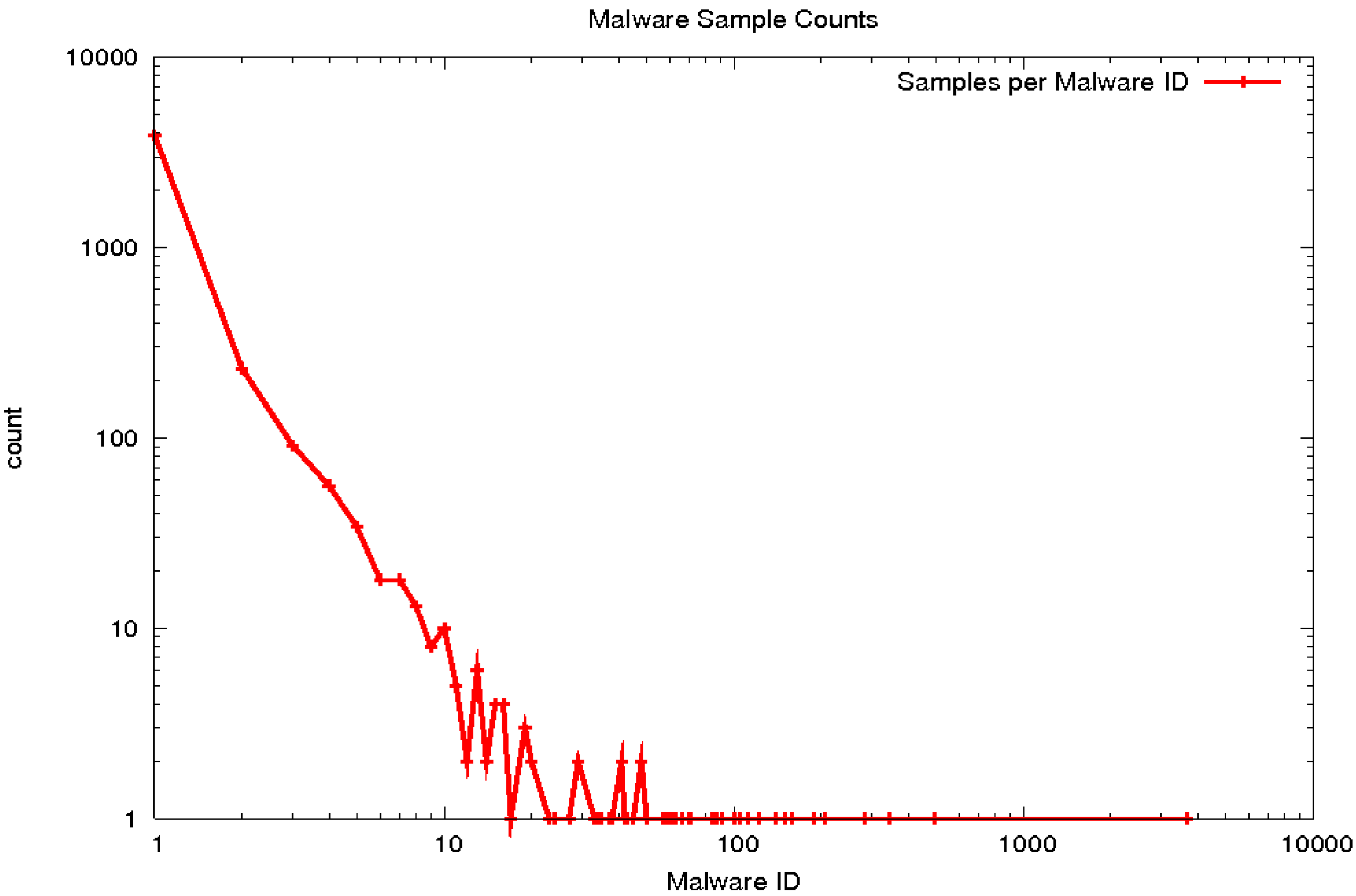
Quee


$E_2(VX)$
stub
VX
body
$E_1(VX)$
stub
stub
stub
VX
body
VX
body
VX
body
$E_n(E_1(VX))$

Queen-Bot Programs

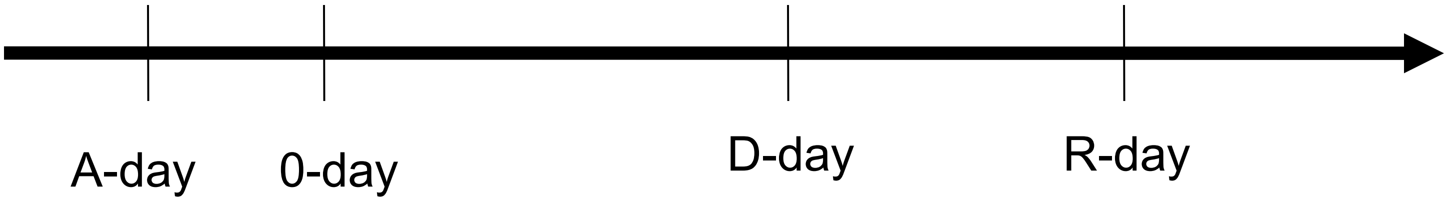
- Queen bots therefore an instance of generative programming
- What are their uses?
  - Automated updating
  - Evasion of AV signatures
- How do they evade AV?
  - We need a rough conceptual model of malware lifecycle …

Queen-Bot Programs: Indirect Evidence



Malware Life Cycle

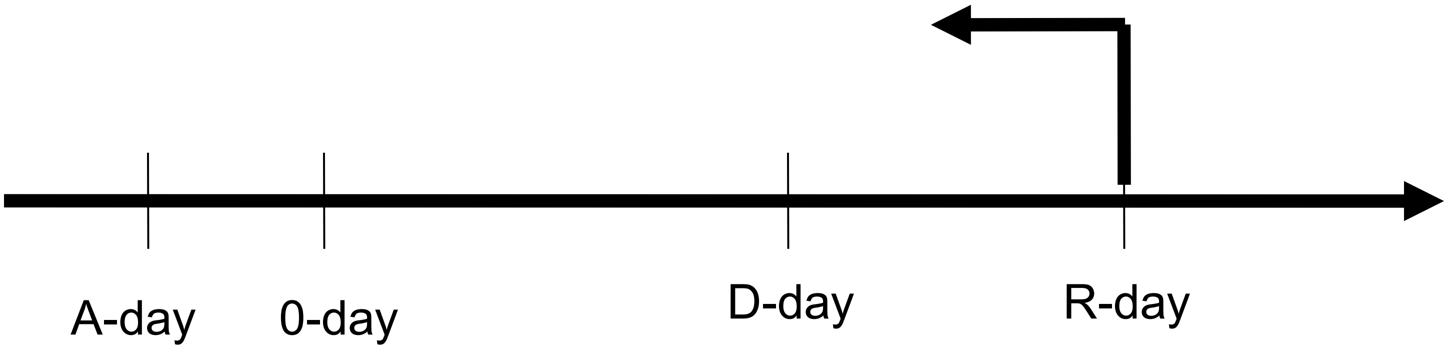
Four conceptual phases of malware life cycle:



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

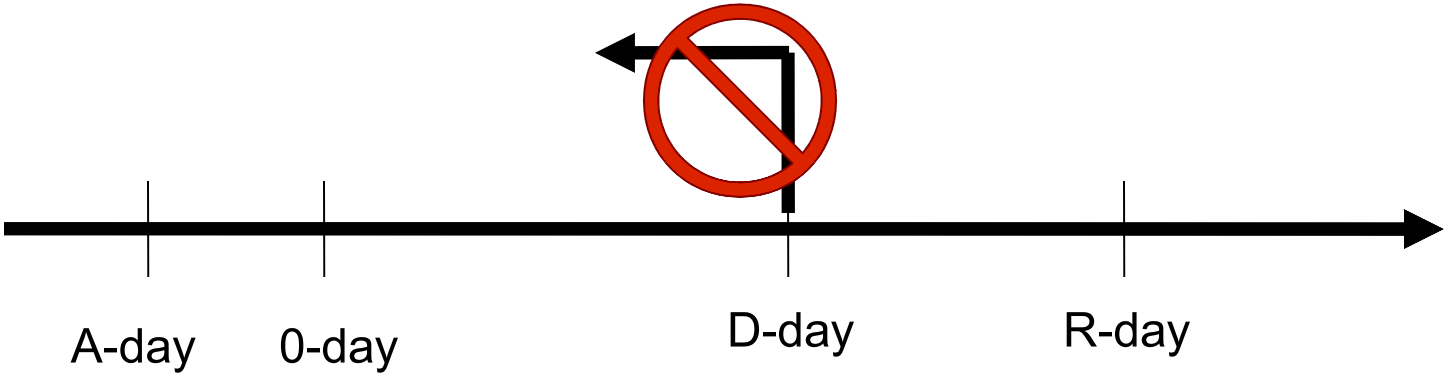


AV update cycles previously measured weeks/days

Now measured in hours/minutes (or should be)

Malware Life Cycle

How to improve detection time...

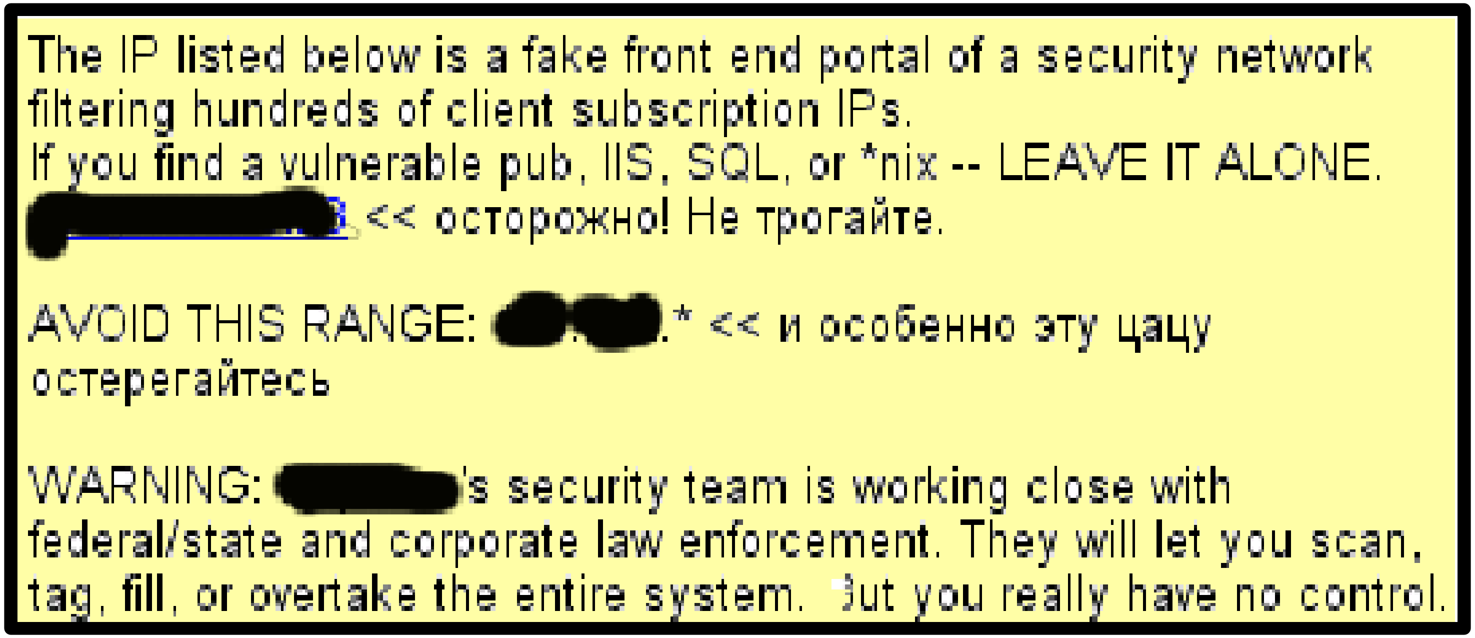


Given that...

- 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

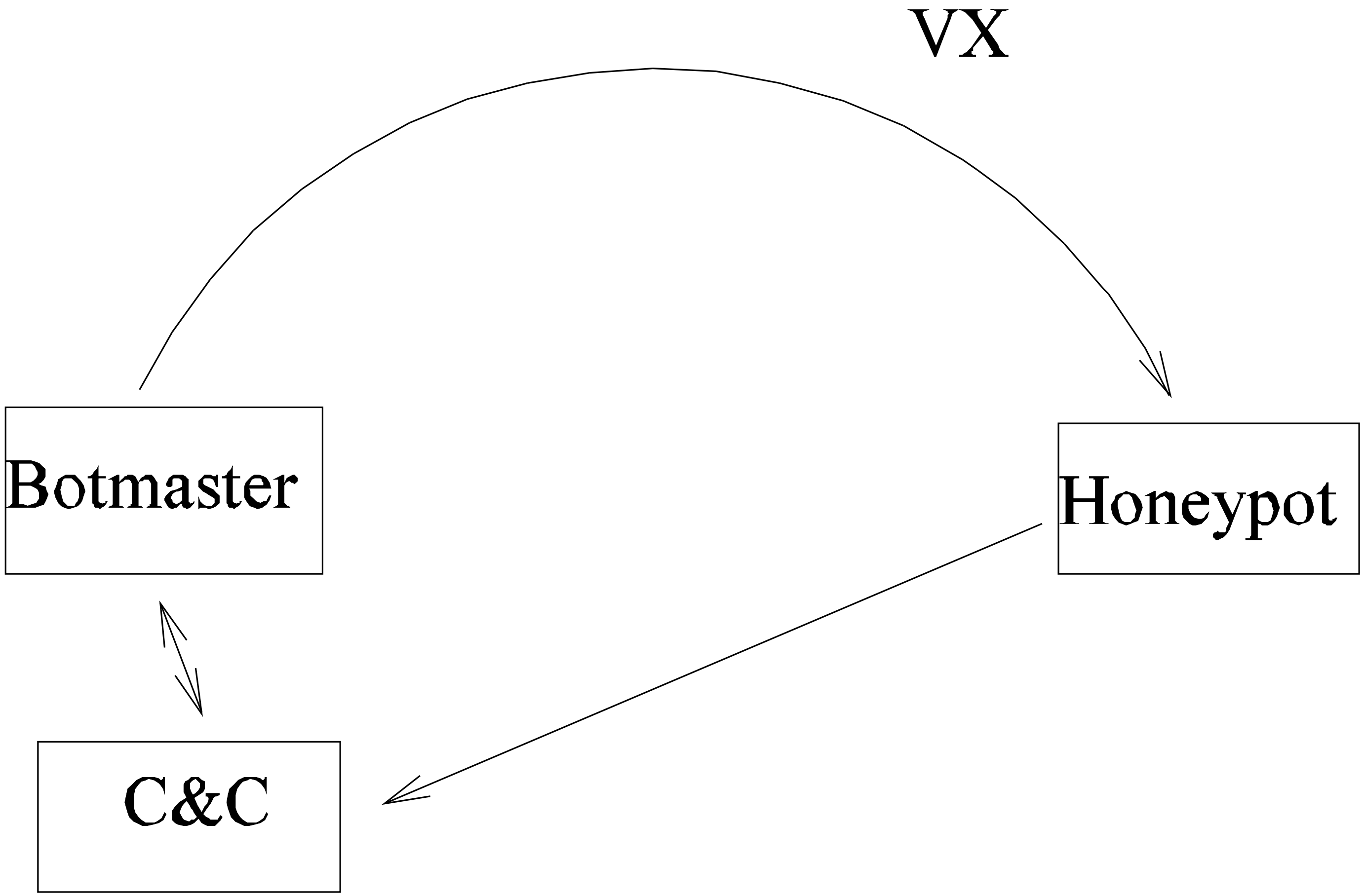


The IP listed below is a fake front end portal of a security network filtering hundreds of client subscription IPs.
If you find a vulnerable pub, IIS, SQL, or *nix -- LEAVE IT ALONE.
[redacted] << осторожно! Не трогайте.

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

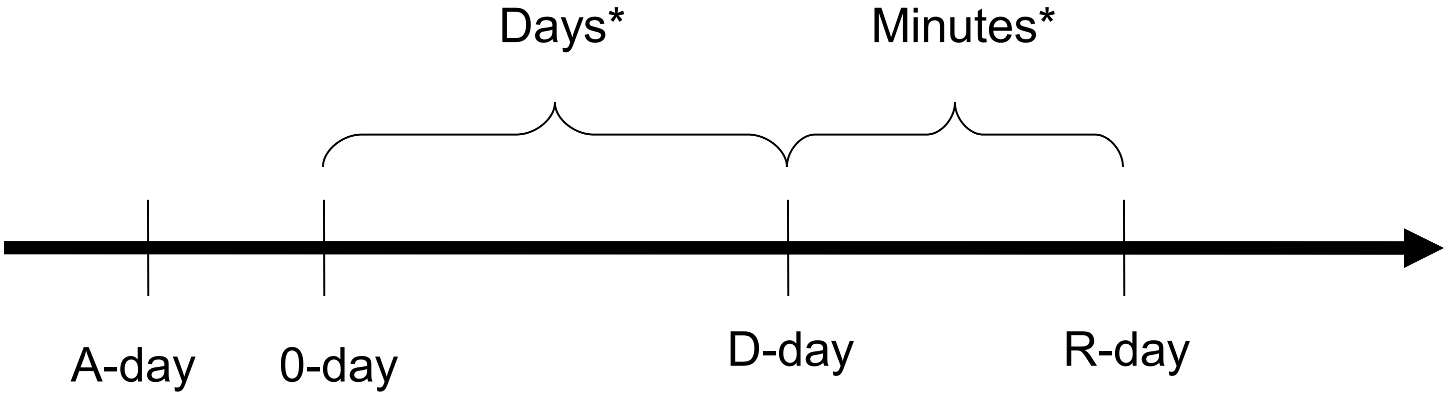
WARNING: [redacted]'s security team is working close with federal/state and corporate law enforcement. They will let you scan, tag, fill, or overtake the entire system. But you really have no control.

Sensor Illumination



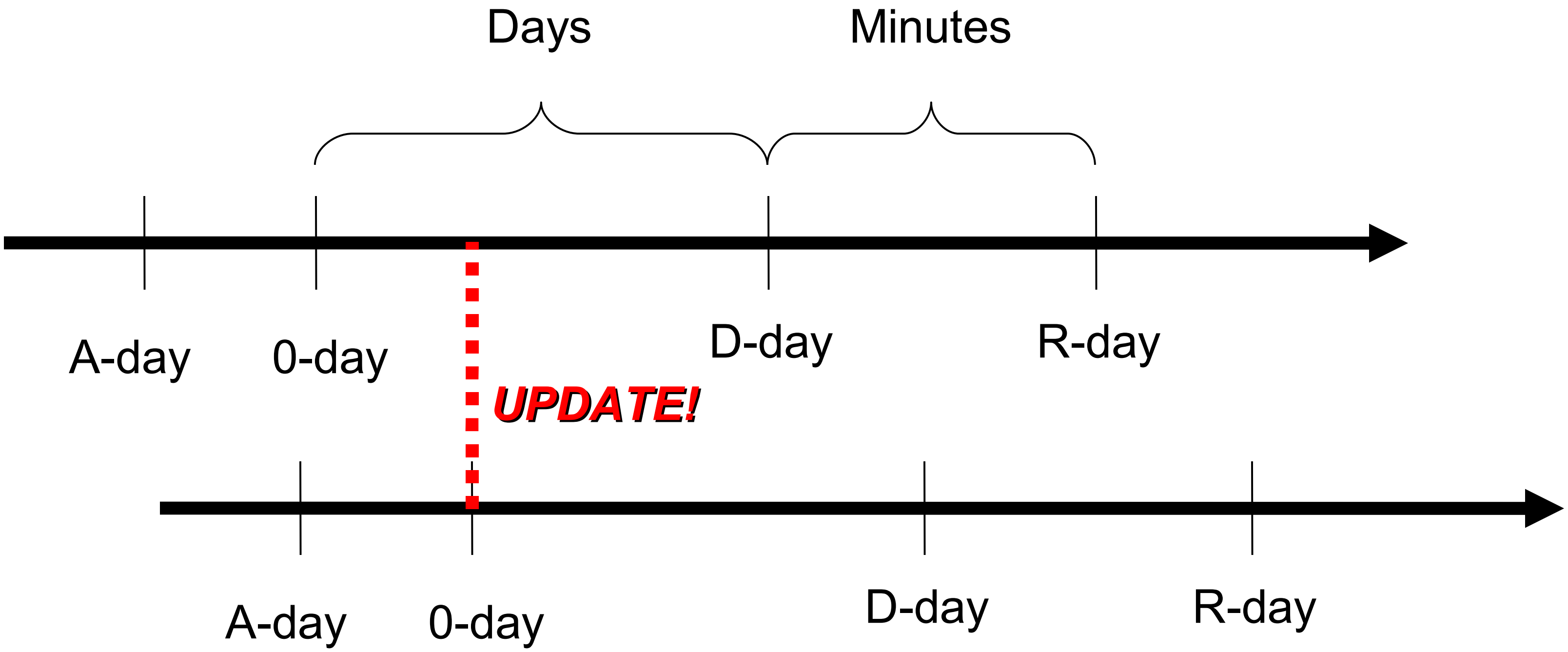
Malware Life Cycle

Because of illumination and limited sharing, distance (0day, detection) is days, while distance (detection, response) is (ideally) hours.



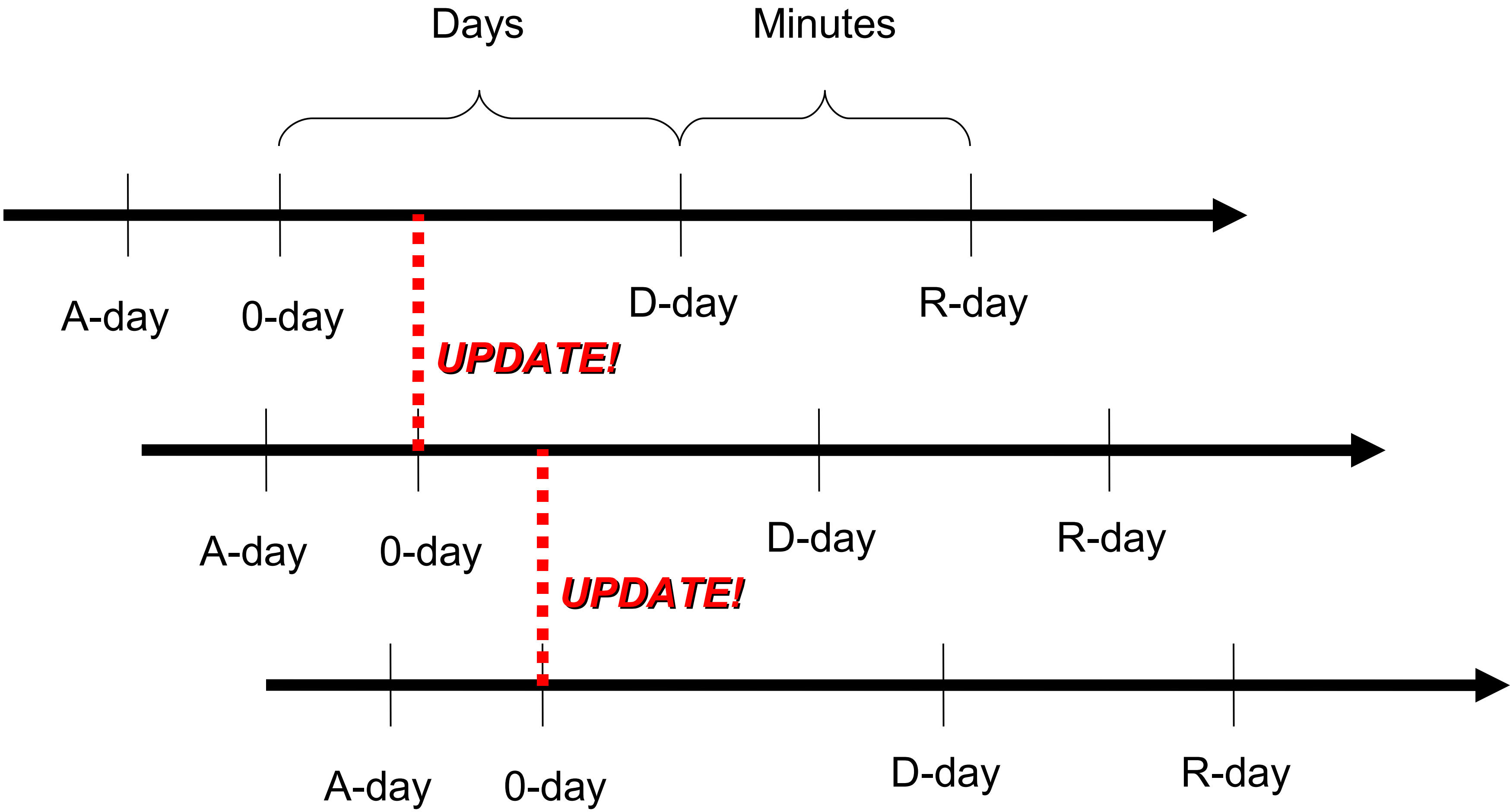
* Average order of time; anecdotes will vary

Malware Life Cycle

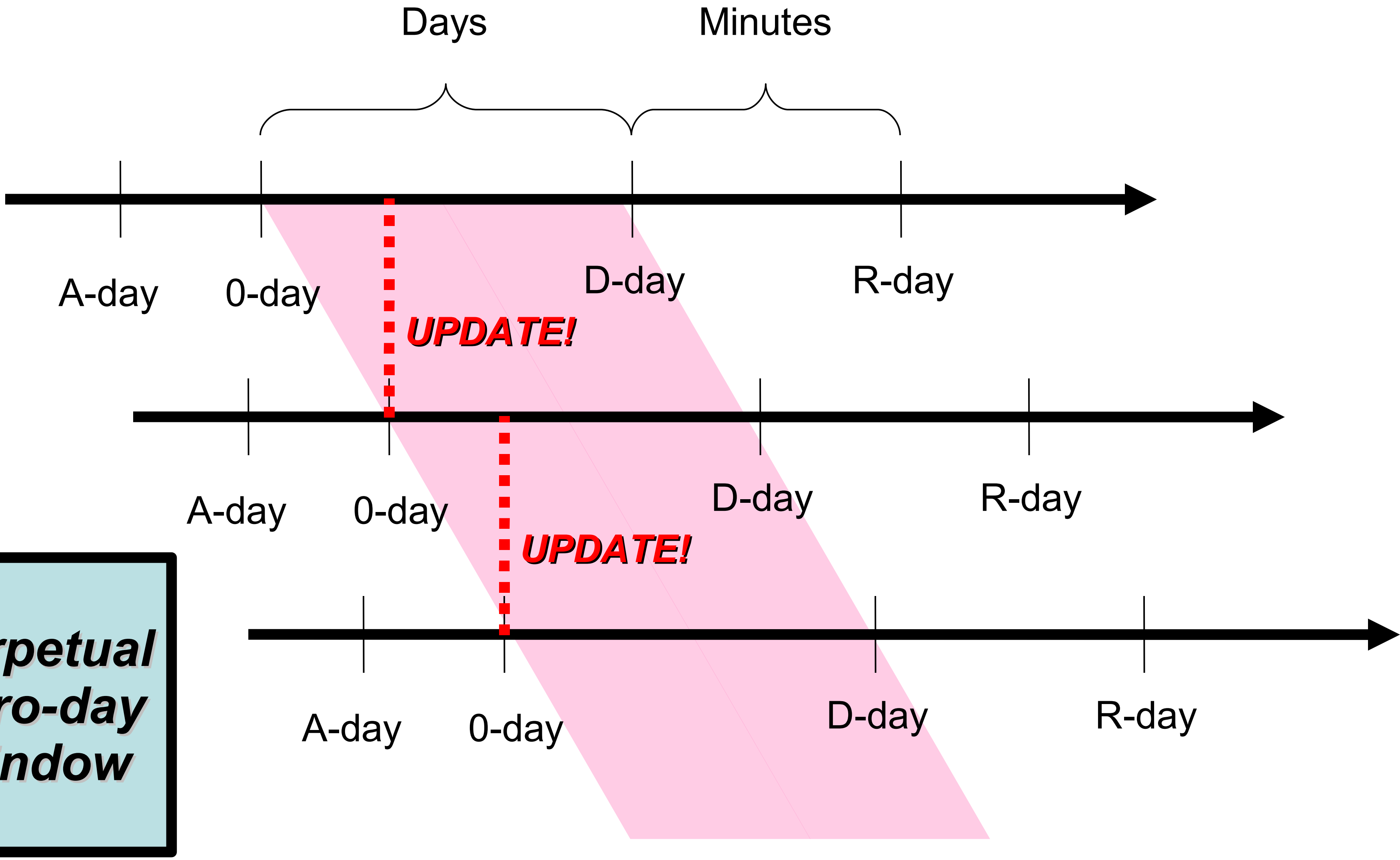


Bot runs for ~1/2 day, and updates to new, evasive binary

Malware Life Cycle

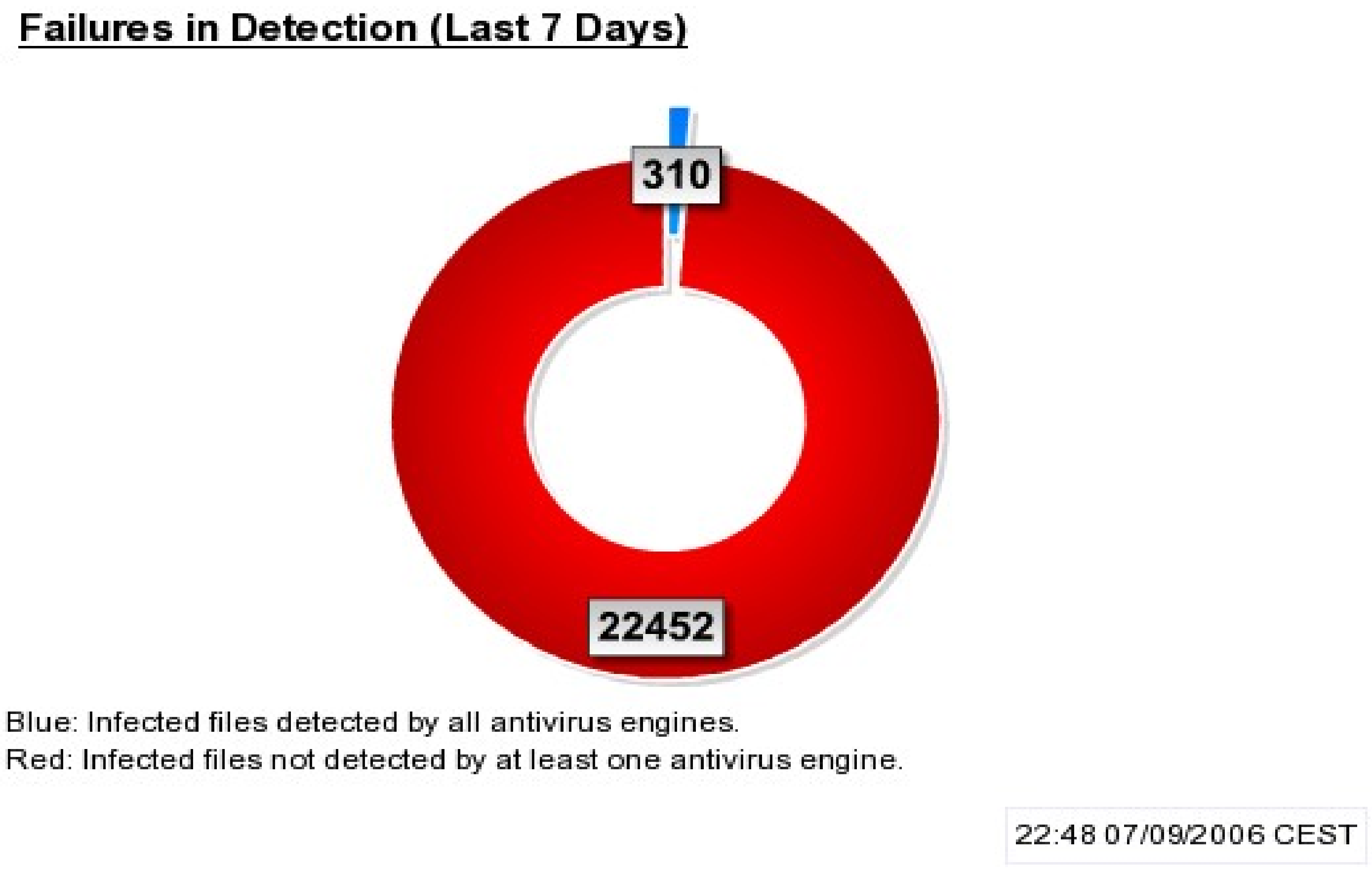


Malware Life Cycle



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

Solution:
Service-Oriented Repository

- Malfease uses hub-and-spoke model
  - Hub is central collection of malware
  - Spokes are analysis partners
- Hub:
  - Malware, indexing, search
  - Static analysis: header extraction, icons, libraries
  - Metainfo: longitudinal AV scan results
- Spoke:
  - E.g., dynamic analysis, unpacking

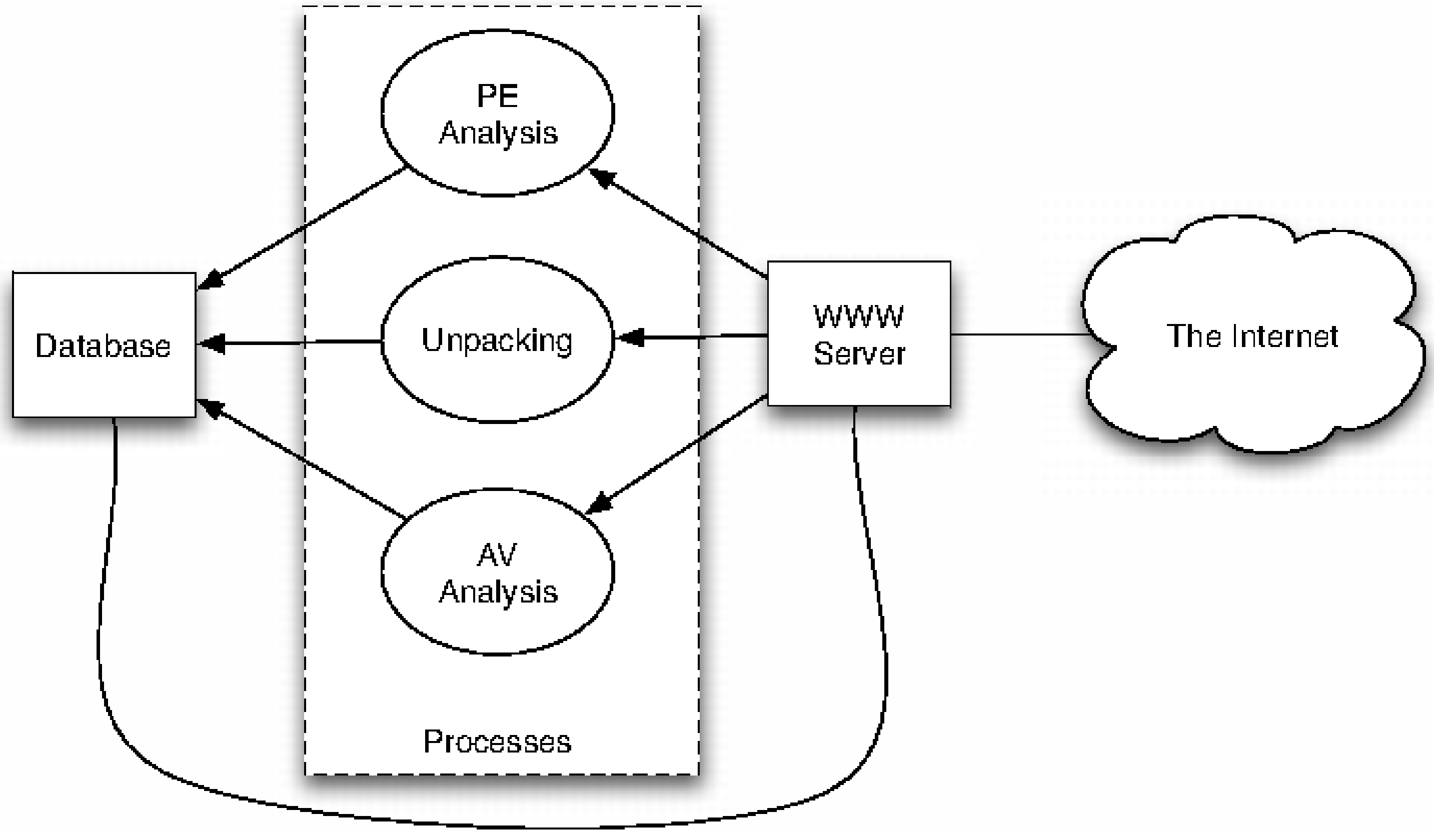
Malware Repo Requirements

- 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
Database
Unpacking
WWW
Server
The Internet
AV
Analysis
Processes

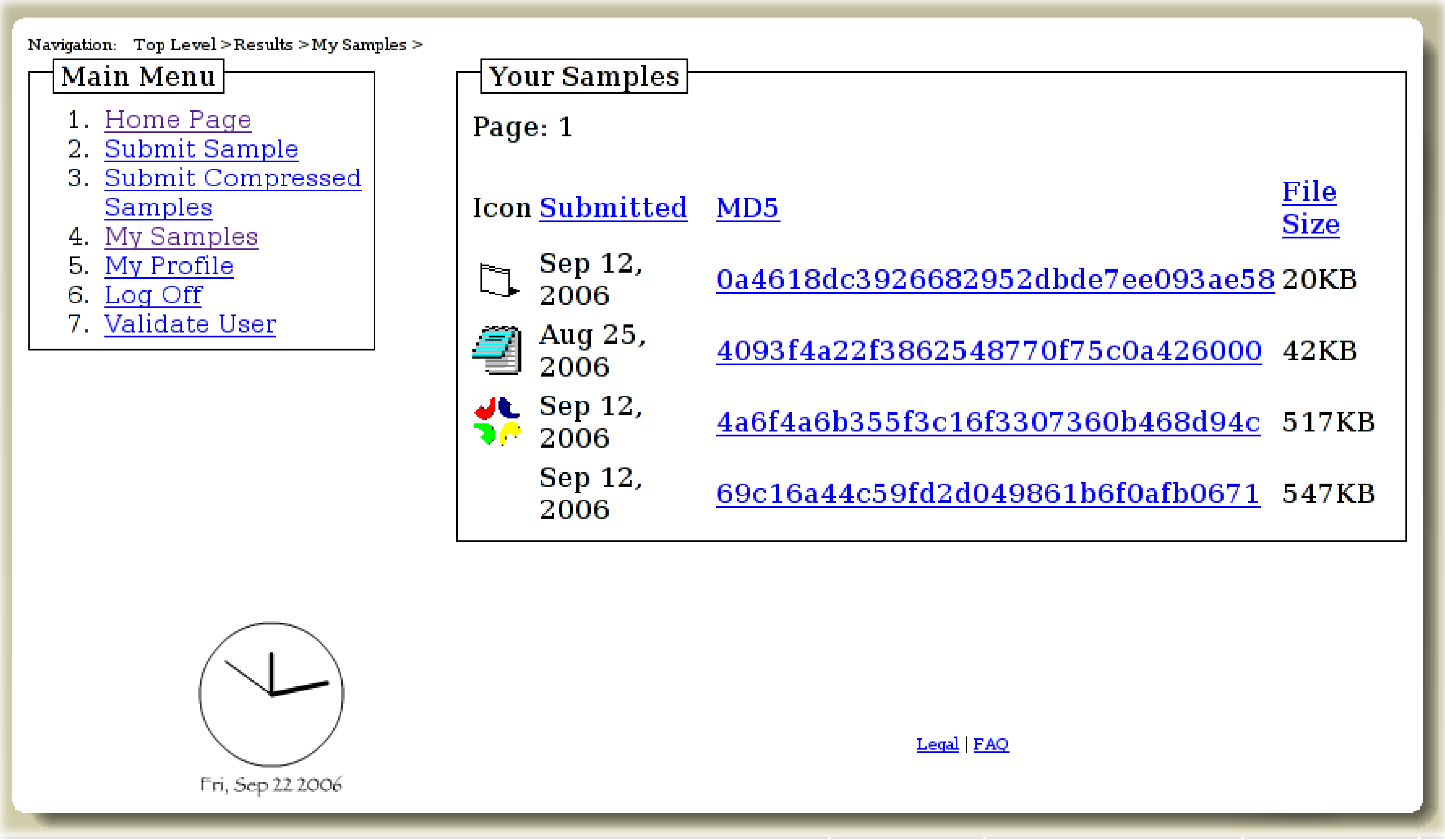
Repository User Classes

- 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

Basic User View



Navigation: Top Level > Results > My Samples >

Main Menu

1. Home Page
2. Submit Sample
3. Submit Compressed Samples
4. My Samples
5. My Profile
6. Log Off
7. Validate User

Your Samples

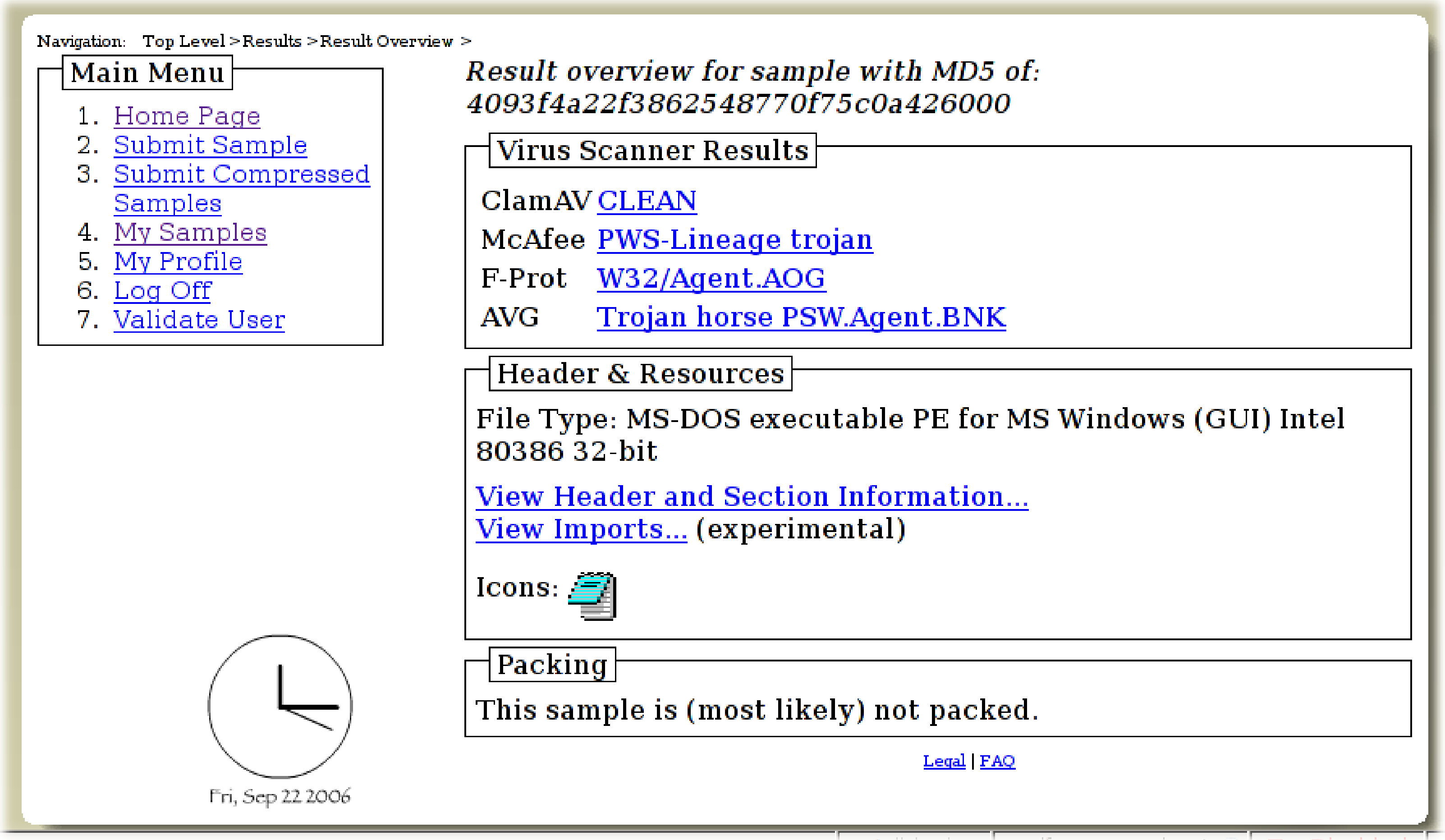
Page: 1

Icon	Submitted	MD5	File Size
	Sep 12, 2006	0a4618dc3926682952dbde7ee093ae58	20KB
	Aug 25, 2006	4093f4a22f3862548770f75c0a426000	42KB
	Sep 12, 2006	4a6f4a6b355f3c16f3307360b468d94c	517KB
	Sep 12, 2006	69c16a44c59fd2d049861b6f0afb0671	547KB

Fri, Sep 22 2006

Legal | FAQ

Analysis Page for Sample



Result overview for sample with MD5 of: 4093f4a22f3862548770f75c0a426000

Virus Scanner Results

ClamAV	CLEAN
McAfee	PWS-Lineage trojan
F-Prot	W32/Agent.AOG
AVG	Trojan horse PSW.Agent.BNK

Header & Resources

File Type: MS-DOS executable PE for MS Windows (GUI) Intel 80386 32-bit

View Header and Section Information...
View Imports... (experimental)

Icons: 🗒

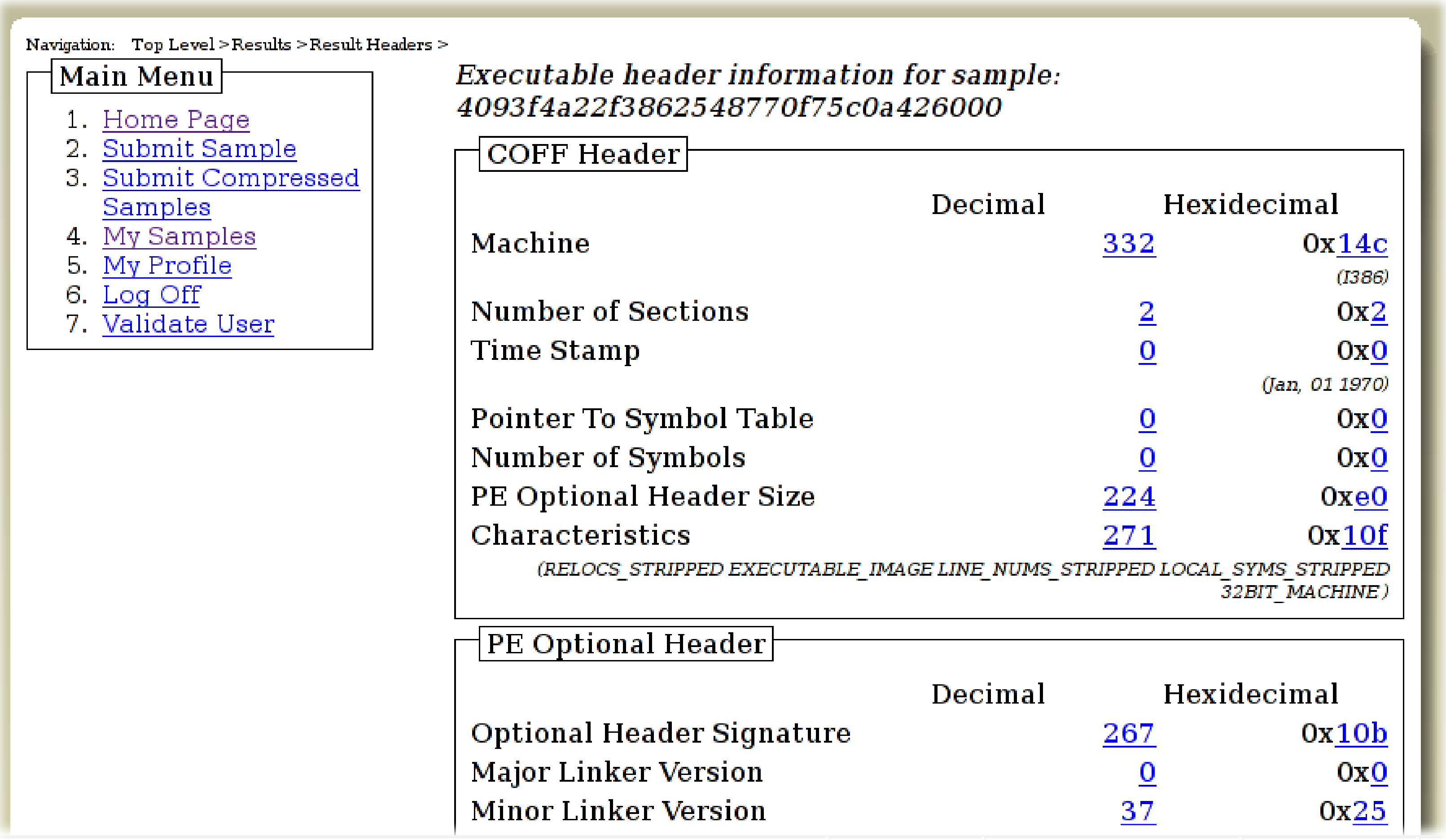
Packing

This sample is (most likely) not packed.

Legal | FAQ

Fri, Sep 22 2006

Static Analysis Example



Navigation: Top Level > Results > Result Headers >

Main Menu

1. Home Page
2. Submit Sample
3. Submit Compressed Samples
4. My Samples
5. My Profile
6. Log Off
7. Validate User

Executable header information for sample:
4093f4a22f3862548770f75c0a426000

COFF Header

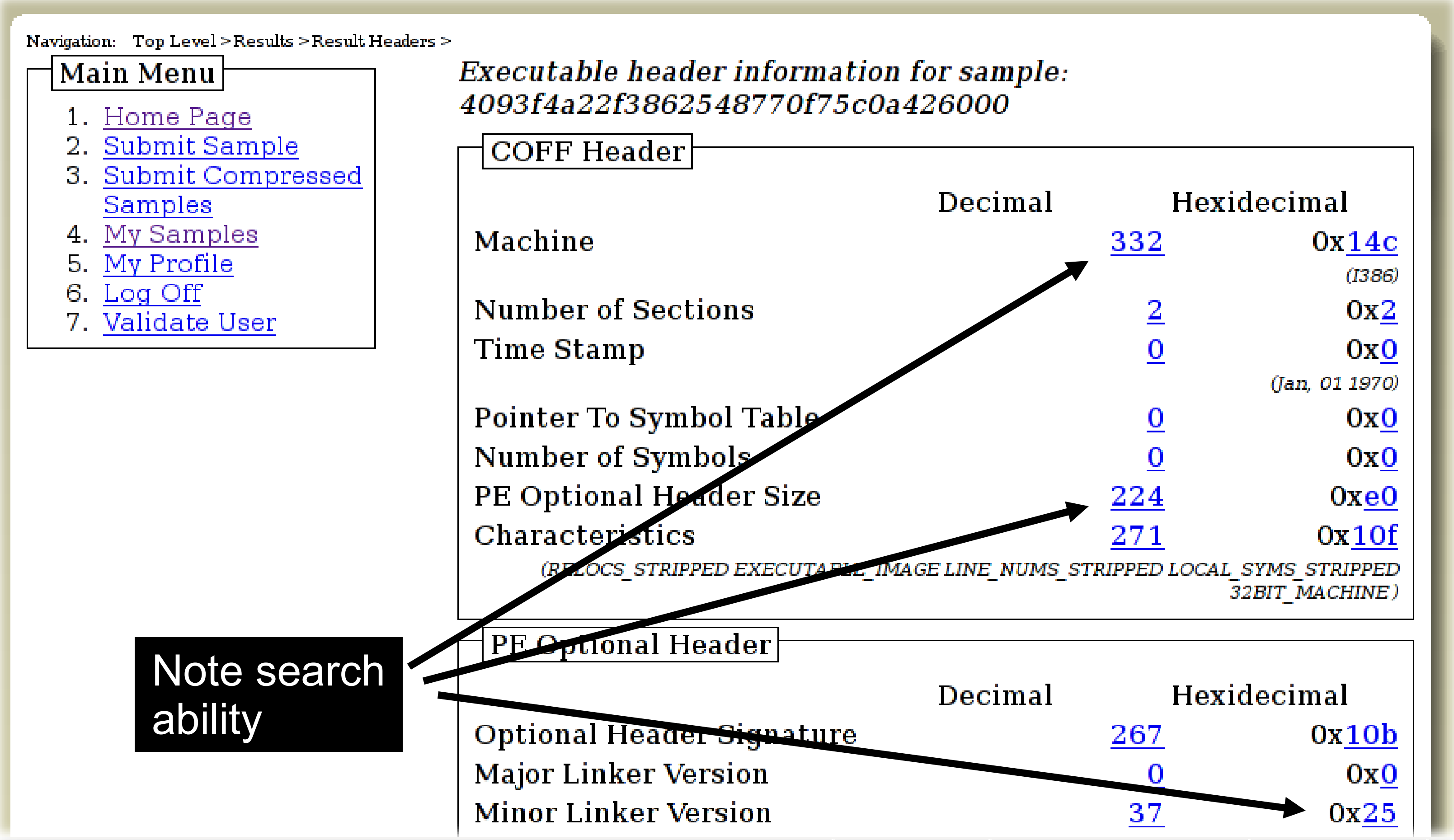
	Decimal	Hexidecimal
Machine	332	0x14c *(I386)*
Number of Sections	2	0x2
Time Stamp	0	0x0 *(Jan, 01 1970)*
Pointer To Symbol Table	0	0x0
Number of Symbols	0	0x0
PE Optional Header Size	224	0xe0
Characteristics	271	0x10f

(RELOCS_STRIPPED EXECUTABLE_IMAGE LINE_NUMS_STRIPPED LOCAL_SYMS_STRIPPED 32BIT_MACHINE)

PE Optional Header

	Decimal	Hexidecimal
Optional Header Signature	267	0x10b
Major Linker Version	0	0x0
Minor Linker Version	37	0x25

Static Analysis Example



Navigation: Top Level > Results > Result Headers >

Main Menu

1. Home Page
2. Submit Sample
3. Submit Compressed Samples
4. My Samples
5. My Profile
6. Log Off
7. Validate User

Executable header information for sample: 4093f4a22f3862548770f75c0a426000

COFF Header

	Decimal	Hexidecimal
Machine	332	0x14c (I386)
Number of Sections	2	0x2
Time Stamp	0	0x0 (Jan, 01 1970)
Pointer To Symbol Table	0	0x0
Number of Symbols	0	0x0
PE Optional Header Size	224	0xe0
Characteristics	271	0x10f

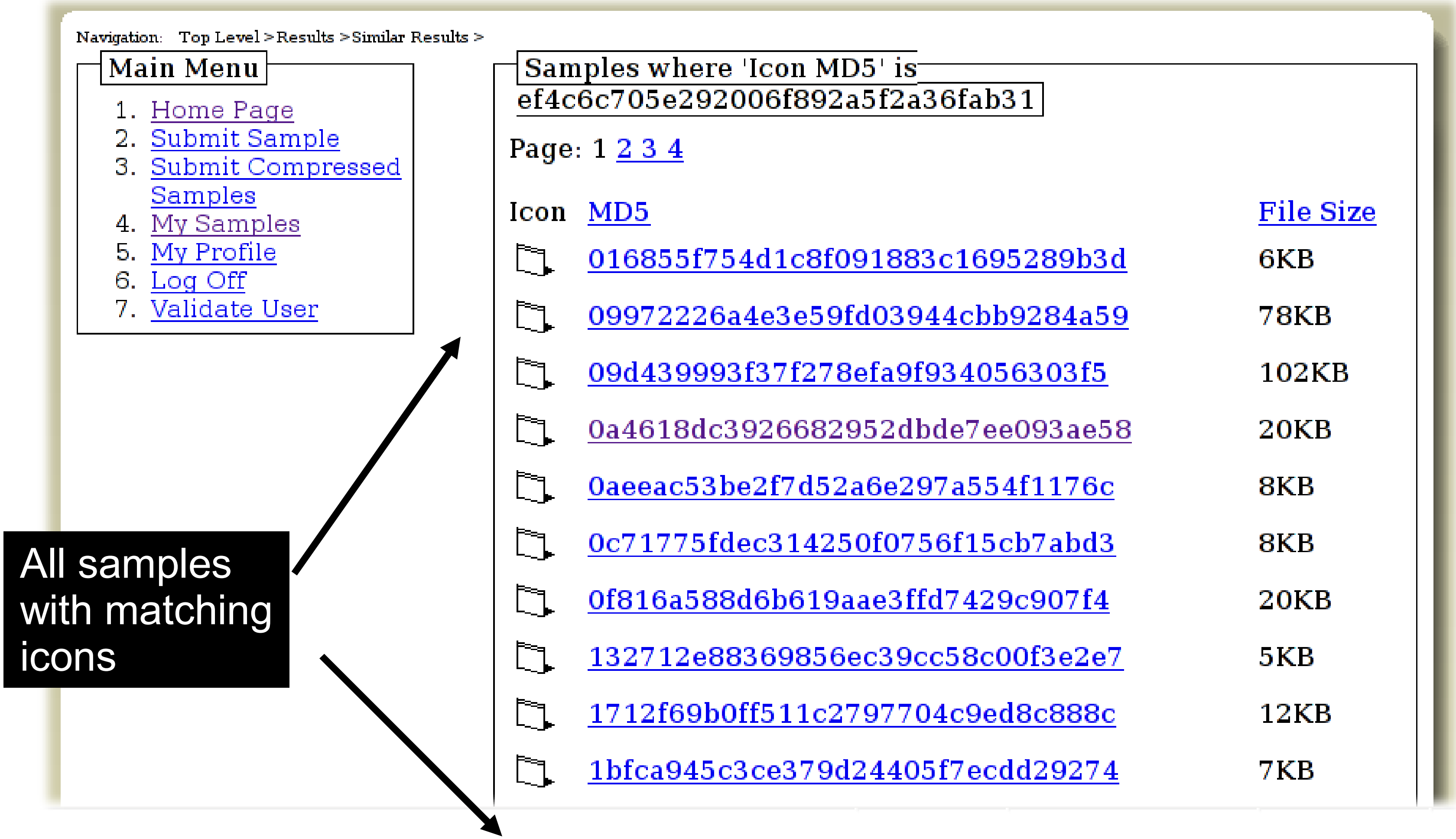
(RELOCS_STRIPPED EXECUTABLE_IMAGE LINE_NUMS_STRIPPED LOCAL_SYMS_STRIPPED 32BIT_MACHINE)

PE Optional Header

	Decimal	Hexidecimal
Optional Header Signature	267	0x10b
Major Linker Version	0	0x0
Minor Linker Version	37	0x25

Note search ability

Example: Search on icons



Navigation: Top Level > Results > Similar Results >

Main Menu

1. Home Page
2. Submit Sample
3. Submit Compressed Samples
4. My Samples
5. My Profile
6. Log Off
7. Validate User

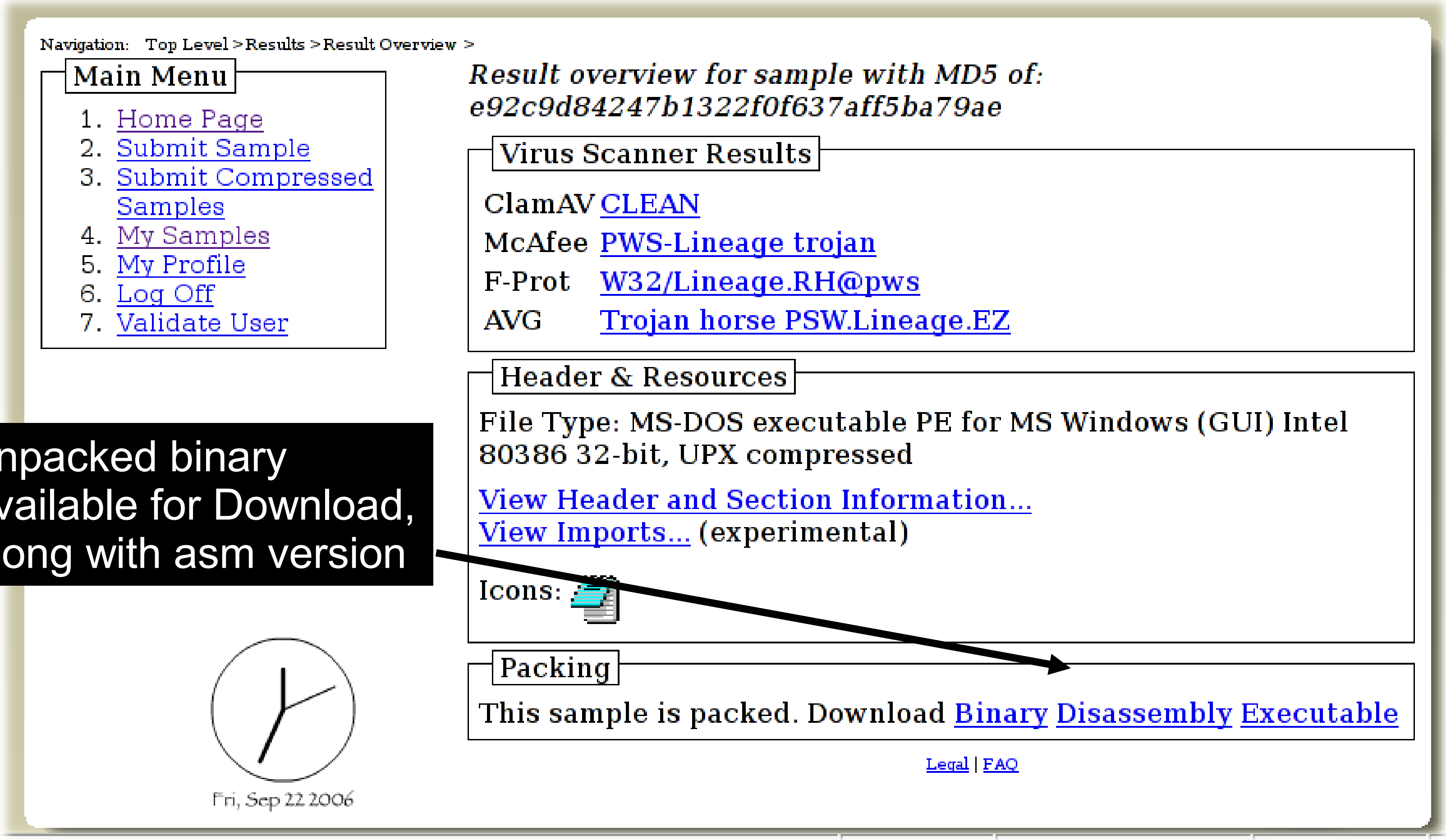
Samples where 'Icon MD5' is ef4c6c705e292006f892a5f2a36fab31

Page: 1 2 3 4

Icon	MD5	File Size
□	016855f754d1c8f091883c1695289b3d	6KB
□	09972226a4e3e59fd03944cbb9284a59	78KB
□	09d439993f37f278efa9f934056303f5	102KB
□	0a4618dc3926682952dbde7ee093ae58	20KB
□	0aeeac53be2f7d52a6e297a554f1176c	8KB
□	0c71775fdec314250f0756f15cb7abd3	8KB
□	0f816a588d6b619aae3ffd7429c907f4	20KB
□	132712e88369856ec39cc58c00f3e2e7	5KB
□	1712f69b0ff511c2797704c9ed8c888c	12KB
□	1bfca945c3ce379d24405f7ecdd29274	7KB

All samples with matching icons

Dynamic Analysis



Navigation: Top Level > Results > Result Overview >

Result overview for sample with MD5 of: e92c9d84247b1322f0f637aff5ba79ae

Virus Scanner Results

ClamAV	CLEAN
McAfee	PWS-Lineage trojan
F-Prot	W32/Lineage.RH@pws
AVG	Trojan horse PSW.Lineage.EZ

Header & Resources

File Type: MS-DOS executable PE for MS Windows (GUI) Intel 80386 32-bit, UPX compressed

View Header and Section Information...
View Imports... (experimental)

Icons:

Packing

This sample is packed. Download Binary Disassembly Executable

Legal | FAQ

Fri, Sep 22 2006

Unpacked binary Available for Download, Along with asm version

Binary Analysis (Spoke) Example

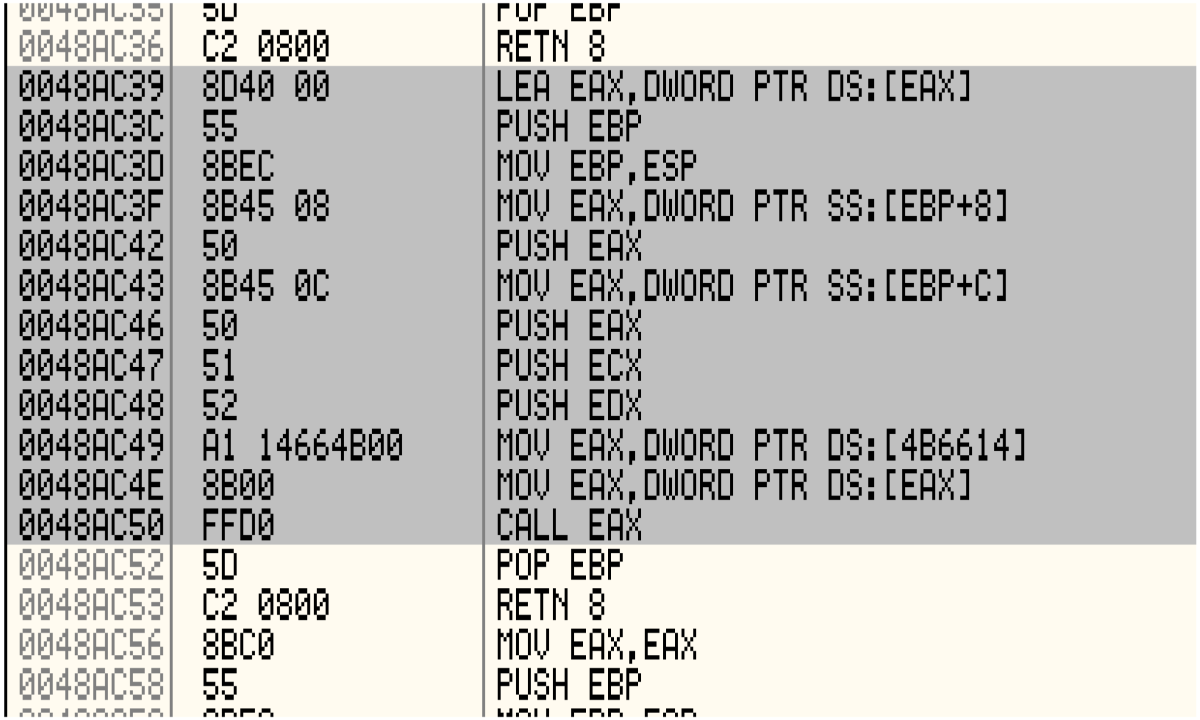
- Motivation: find “key” information in malware
- Previously, binaries trivially yielded relevant information:

```
strings samples/*.exe | grep -i \
  gmail
   0edcxzse @ gmail.com
   d4rkhdeflood @ gmail.com
     ...
```

Binary Analysis (Spoke) Example

- Now, however, malware is packed
  - E.g., of 409 samples, 11% were trivially unpackable.
    - Indicates high degree of packing
    - For 81 non-packed samples, only 7 contained strings recognizable as mail addrs.
- Why such a low result for all samples?
  - Implies runtime data transformations

Binary Analysis (Spoke) Example



```
0048AC35  5D            POP EBP
0048AC36  C2 0800       RETN 8
0048AC39  8D40 00       LEA EAX,DWORD PTR DS:[EAX]
0048AC3C  55            PUSH EBP
0048AC3D  8BEC          MOV EBP,ESP
0048AC3F  8B45 08       MOV EAX,DWORD PTR SS:[EBP+8]
0048AC42  50            PUSH EAX
0048AC43  8B45 0C       MOV EAX,DWORD PTR SS:[EBP+C]
0048AC46  50            PUSH EAX
0048AC47  51            PUSH ECX
0048AC48  52            PUSH EDX
0048AC49  A1 14664B00   MOV EAX,DWORD PTR DS:[4B6614]
0048AC4E  8B00          MOV EAX,DWORD PTR DS:[EAX]
0048AC50  FFD0          CALL EAX
0048AC52  5D            POP EBP
0048AC53  C2 0800       RETN 8
0048AC56  8BC0          MOV EAX,EAX
0048AC58  55            PUSH EBP
0048AC59  8BEC          MOV EBP,ESP
```

Address for `WS2_32.dll:Send` (and data for email address) are constructed dynamically

Spoke Example

```
trace_irc=> select distinct email
from abusive_email where email ilike
'%gmail.com';

          email
-----------------------
 0edcxzse@gmail.com
 0paparazzo@gmail.com
 100money@gmail.com
 1977.24@gmail.com
 1r4d3x@gmail.com
 2006.infos@gmail.com
```

...

etc. etc. etc.

Thus, malfease's collection is transformed to operationally relelvant feeds

Policy Considerations

- Who gets access?
  - Anonymous upload: limited analysis
  - Registered upload: collection management
  - Trusted researcher: full search/full analysis
  - Does this approach meet OARC's approval?

- Branding (Spoke) opportunities
  - Analysis partners may offer/demo analysis services

Policy Consideration

- Resources
  - All front-end code BSD licensed
    - Spoke analysis tools may sport any license
  - Hardware and development courtesy of Damballa
- Coordination with other malware repos?
  - MIRT/PIRT
  - APWG

OARC Resources

- So far, no cost to OARC
    - Hardware, dev work courtesy of Damballa
        - We have until January 2007 to finish major work
- Needed OARC resources:
    - Blessing/acceptance
        - A review/edit of policies
    - Mailing lists (one for dev, one for users)
    - Possible mirror
    - Feedback from members
    - Malware (send samples!)

Conclusion

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