VxClass
Clustering Malware, Generating Signatures

zynamics GmbH
info@zynamics.com

Overview

zynamics VxClass

- Introduction
- What is VxClass? How does it work?
  - The Malware Pipeline
- Generating signatures from clusters of malware

Malware

zynamics VxClass

- Keeping up with the flood of malware is hard:
  - Steady increase in number of new variants
    measured by “unique hash” (MD5, SHA1)
  - Off-the-shelf tools to produce malware-variants
    think Swizzor
  - AV-signature databases growing fast
    problems: duplicates, junk, false-positives, …
  - Time of human malware analysts is scarce
    don’t let them do repetitive and error-prone work

→Automated methods needed

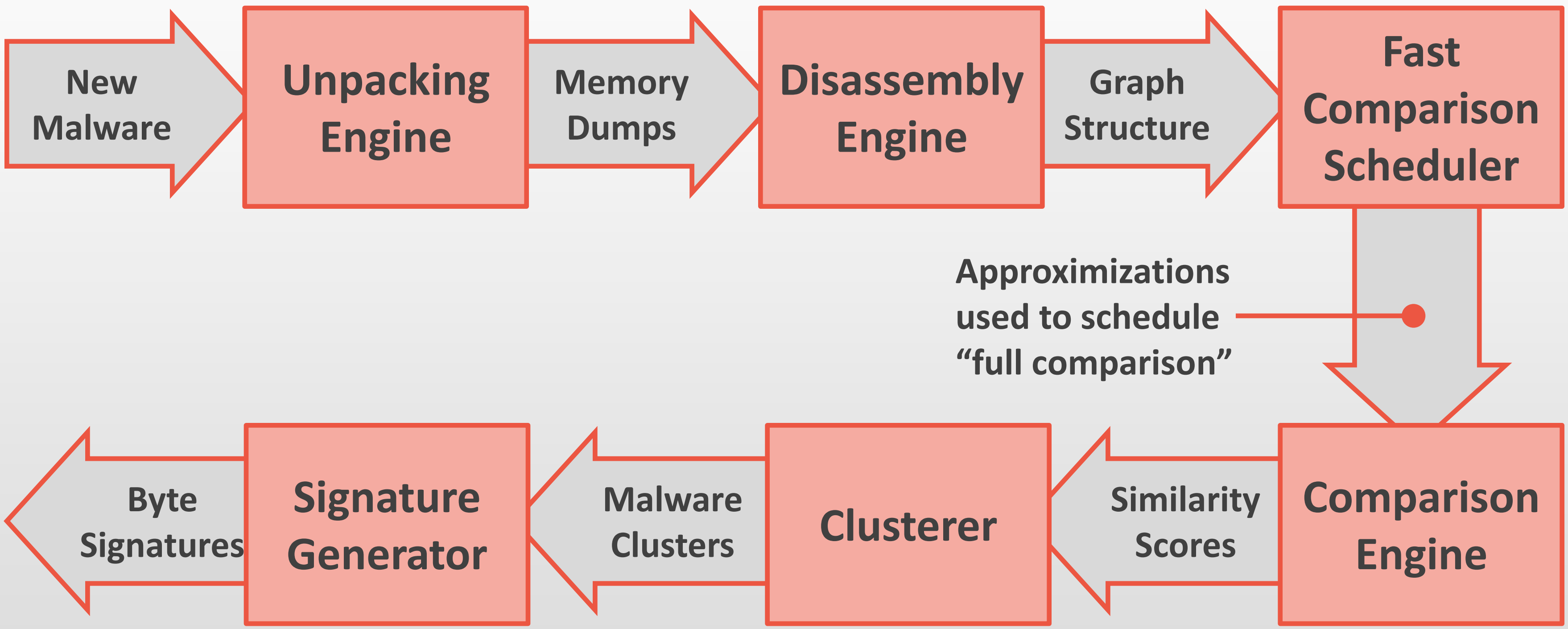
VxClass

zynamics VxClass

- Full infrastructure for processing new malware
  - Automated generic unpacking to remove crypters
    based on full-system emulation
  - Comparison engine to detect similarities between executables (based on zynamics BinDiff)
    often better than a human analyst
  - Clustering algorithms for grouping into "families"
    allows to visualize the malware phylogeny
  - Generation of byte-based AV-signatures for each cluster

Malware Pipeline

Malware is processed in the following stages



Unpacking Engine

zynamics VxClass

- Full-system Emulation
  - Uses snapshot of fully booted Win XP SP3 (32-bit)
  - Malware is injected into the system
  - Different execution modes
    - Execute a pre-defined number of instructions
    - Examine repeated snapshots until “good enough”
  - Acquires new processes and new kernel memory

    injected memory into other processes is not yet acquired

Full-system emulation solves many problems

legacy APIs, API detections, …

Disassembly Engine

- Processes full executables or memory dumps
- Scan for and recognize typical function prologues
- Generates
  - Flowgraphs
  - Executable Callgraphs
- Compiler and library filtering (FLIRT)

Comparison Engine

zynamics VxClass

The "lifeblood" of VxClass

- Based on algorithms developed for the industry-standard zynamics BinDiff
- Disregard byte sequences, focus on structural comparison
- Operates on function flowgraphs and the callgraph structure

  performs comparison based on these structures instead of concrete byte/instruction sequences

→Highly resilient against compiler/platform changes

  different compilers and settings, even different CPUs (!)

Comparison Engine

zynamics VxClass

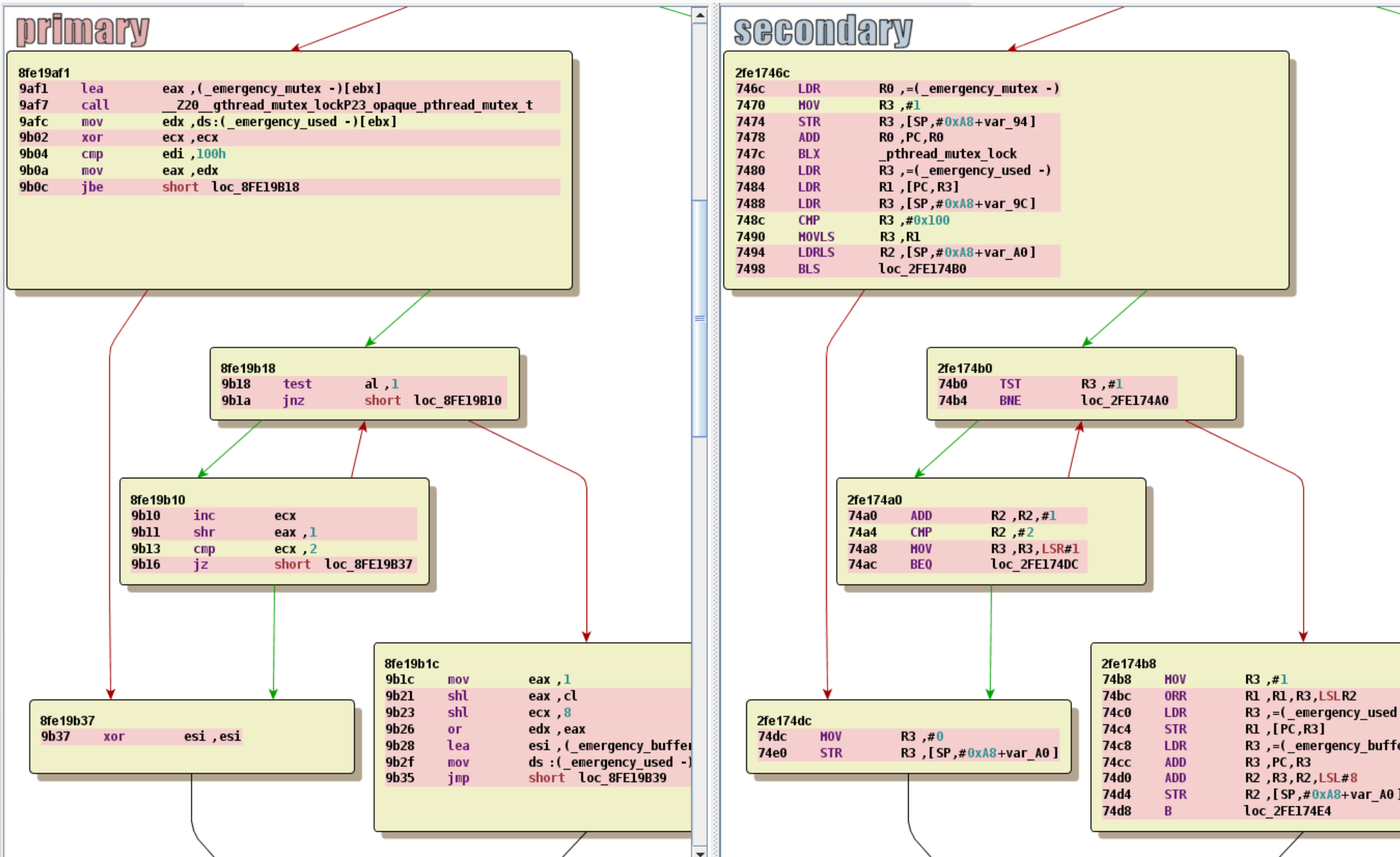
Example: GCC vs. Visual C++

below is SpiderMonkey versus escript.dll (Adobe Reader)

Comparison Engine

Example: Mac OS X vs. iPhone

allows cross-CPU comparison



Scheduling Engine

- Calculating a full similarity matrix is $O(n^2)$

    prohibitively expensive for large sample sets

- Need to reduce complexity

    →Filter samples that are not similar at all

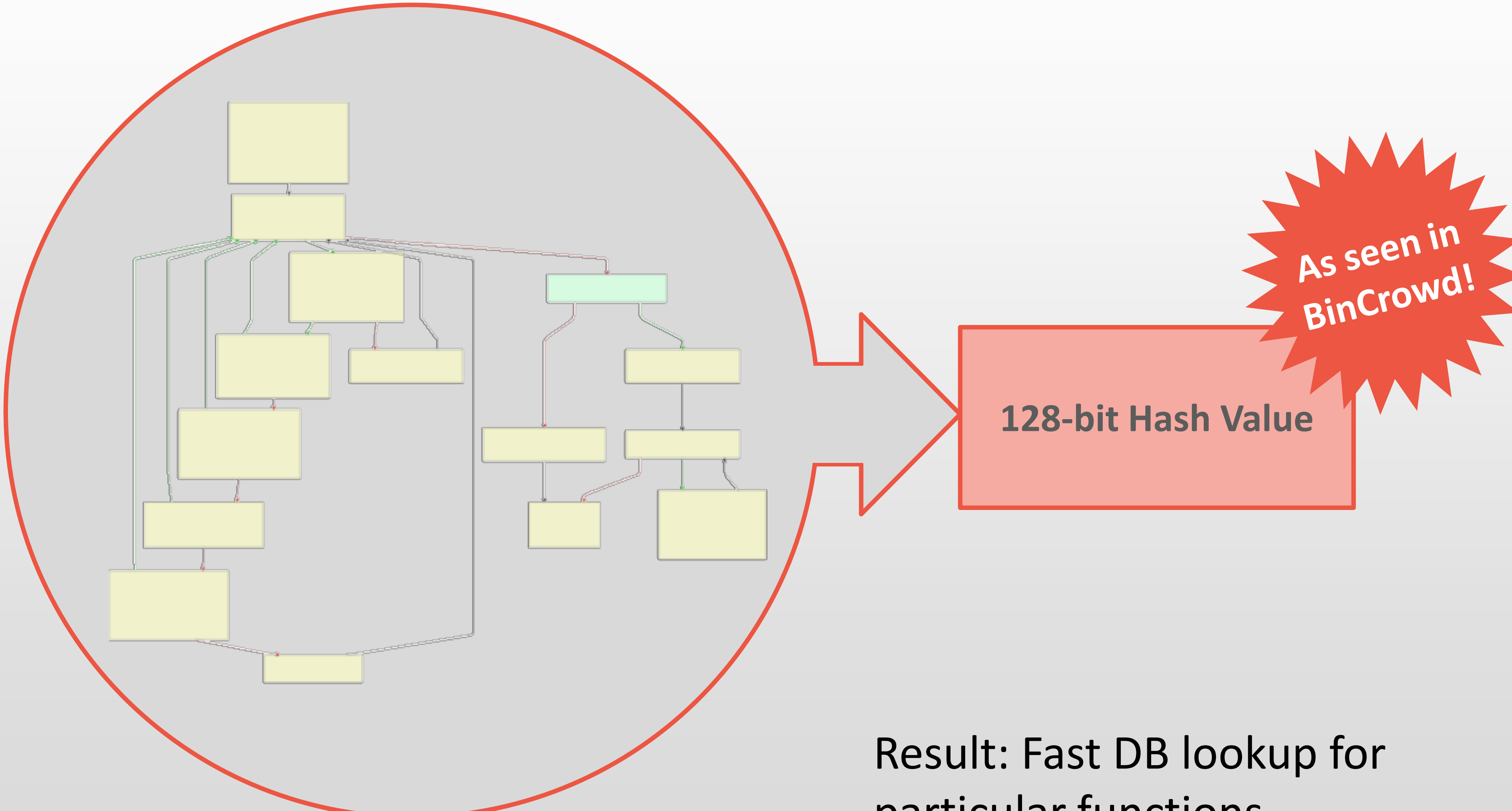
- Fast comparison engine using 128-bit flowgraph "hashes" (MD-Index)
    - Calculating MD-Indices is fast
    - Fast comparison via search for common hashes
- Result of fast comparison prioritizes samples

    yields the complete matrix eventually

MD-Index

zynamics VxClass

Hashing flowgraphs for fast database lookup



Result: Fast DB lookup for particular functions

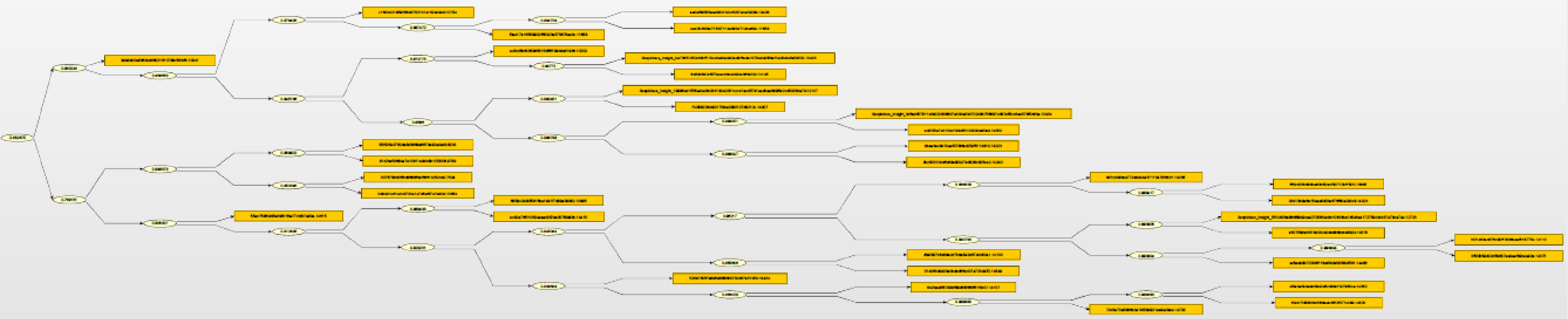
Clustering

zynamics VxClass

- Based on the similarity matrix generate clusters in different ways
  - Compute connected components
    use a similarity threshold for graph edges
  - Apply phylogeny algorithms (bioinformatics)
    yields a family tree
  - Use any other clustering algorithm

Clustering

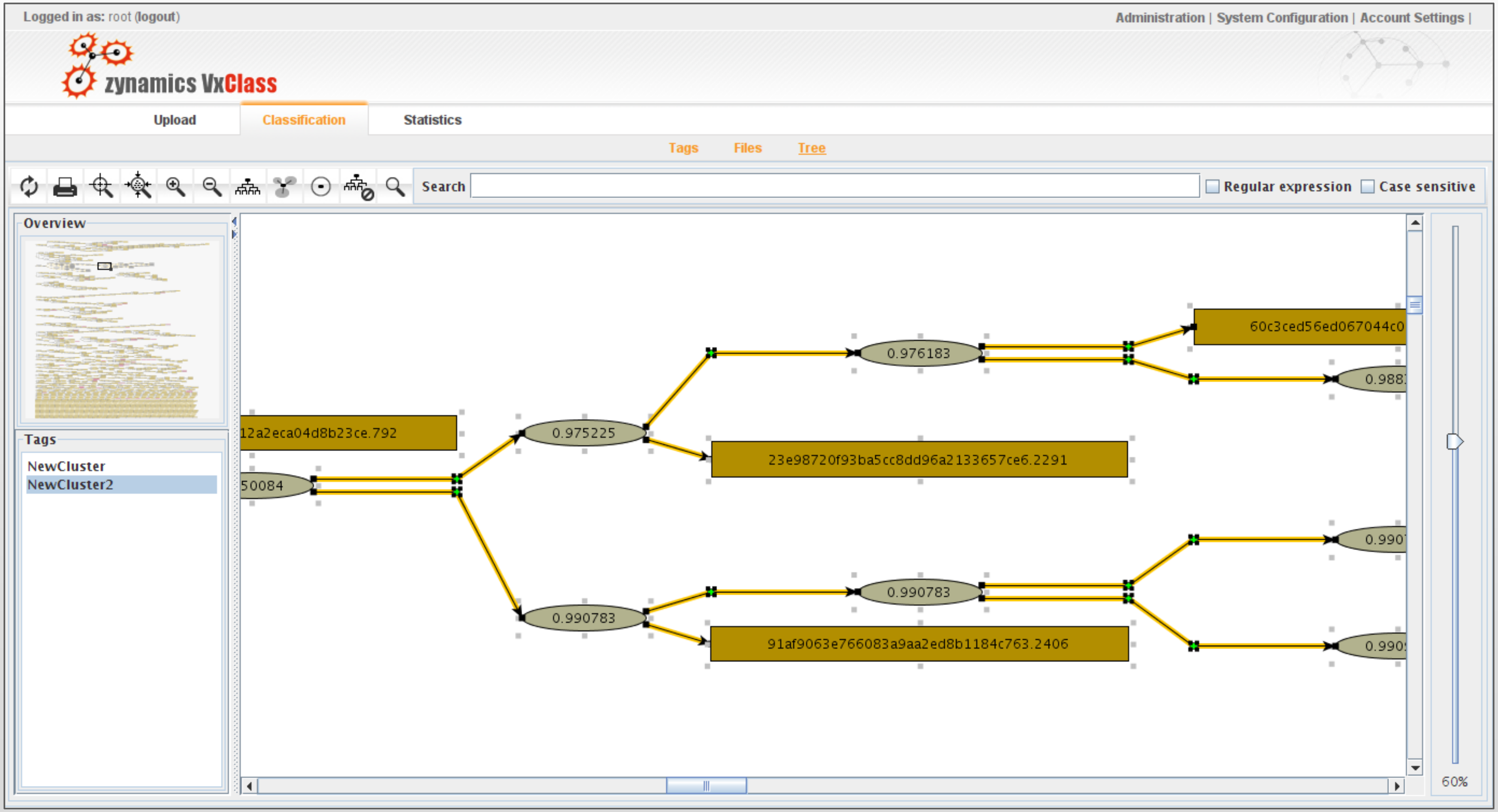
Example tree using phylogeny algorithms



Clustering

What to do with those clusters?

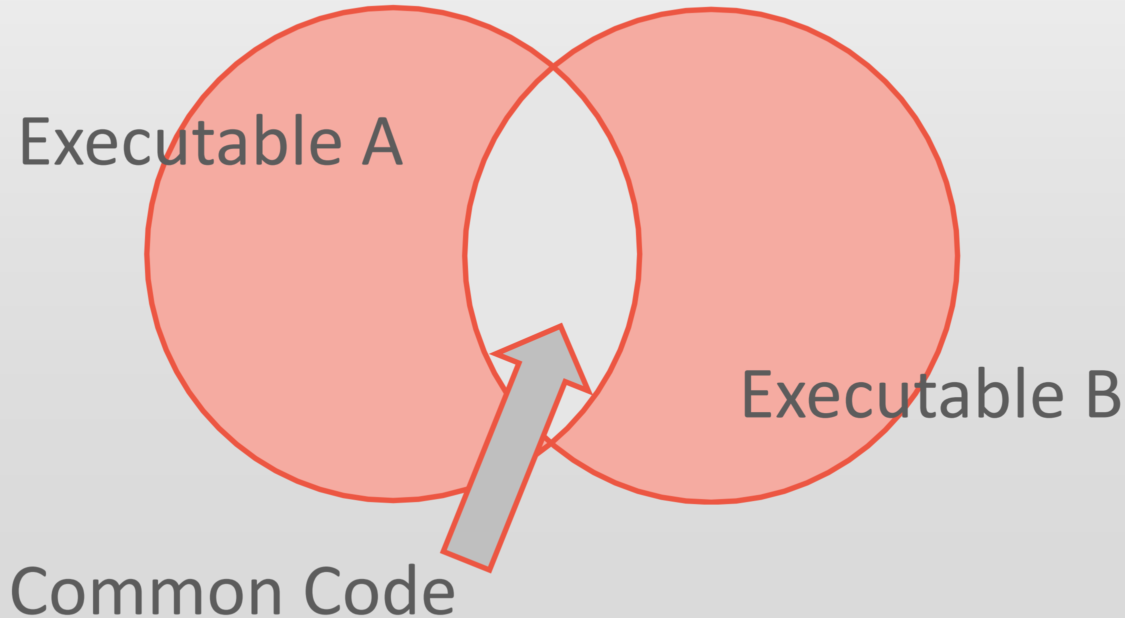
tag, name and otherwise analyze them using the web interface



Signature Generator

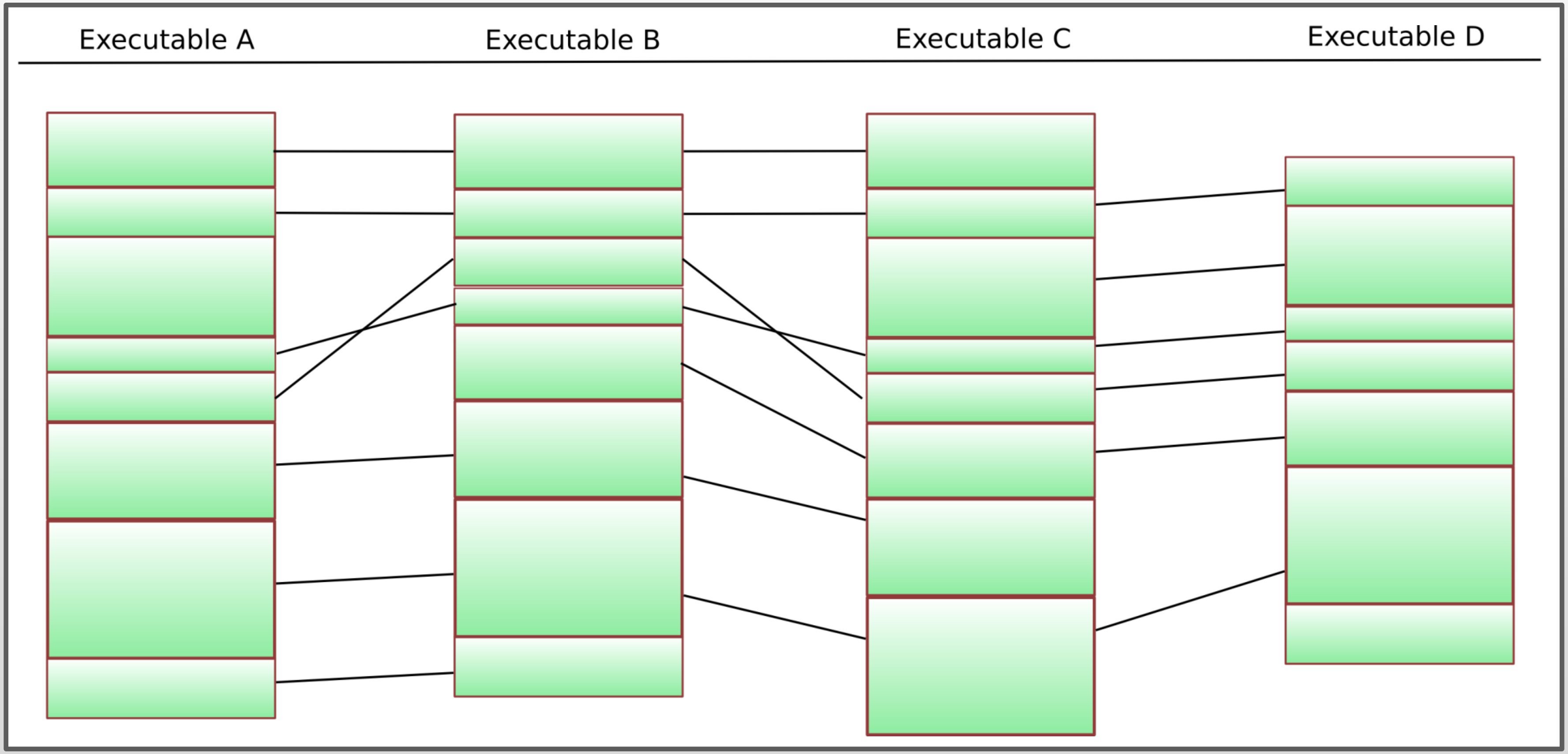
zynamics VxClass

- Fact: Items in the same cluster/family usually share a lot of code
  - Comparison algorithms work like an "intersection operator":



Signature Generator

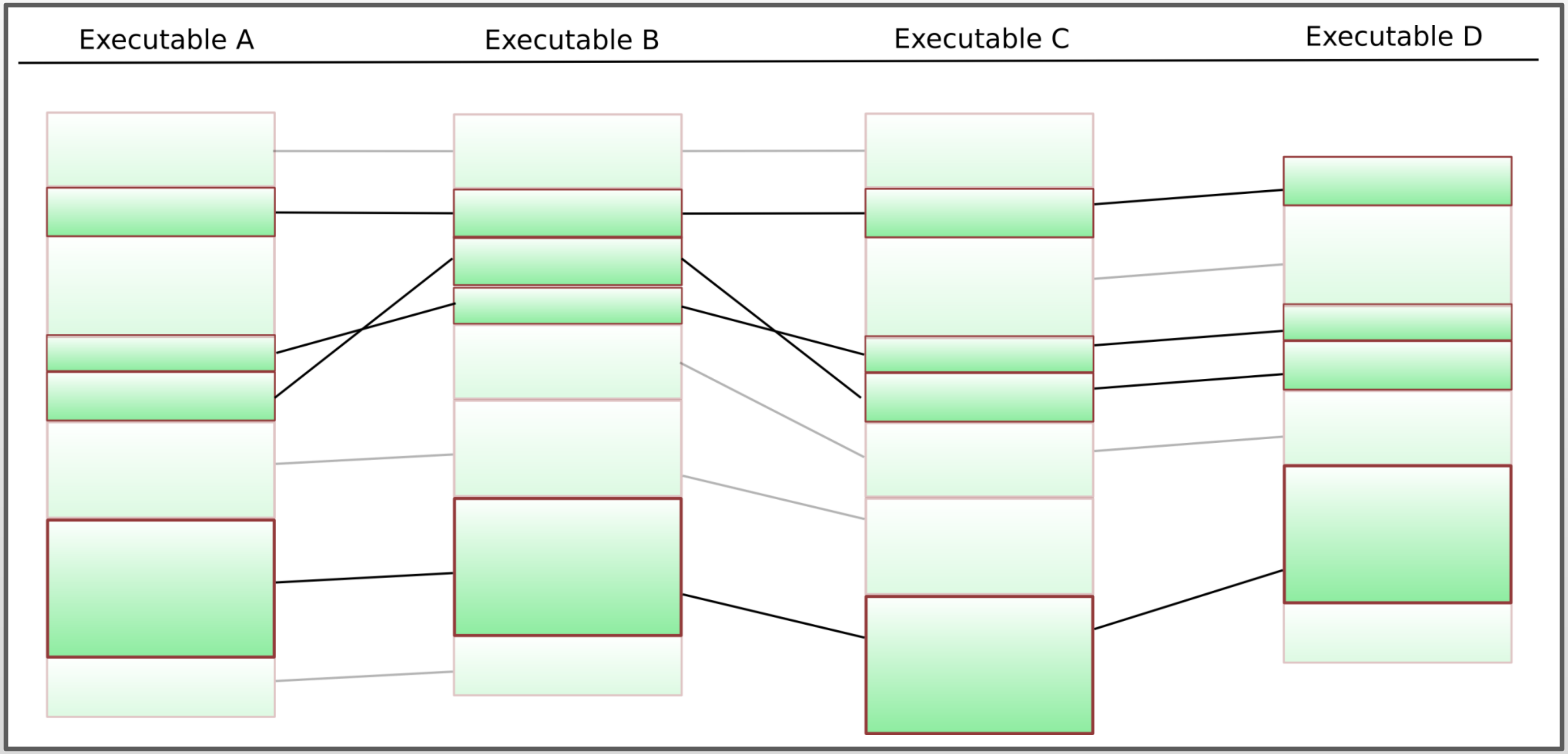
- To build a signature, find functions "common to all elements of a cluster"
- Map matched similar functions in each executable:



Signature Generator

zynamics VxClass

- Eliminate functions not present everywhere

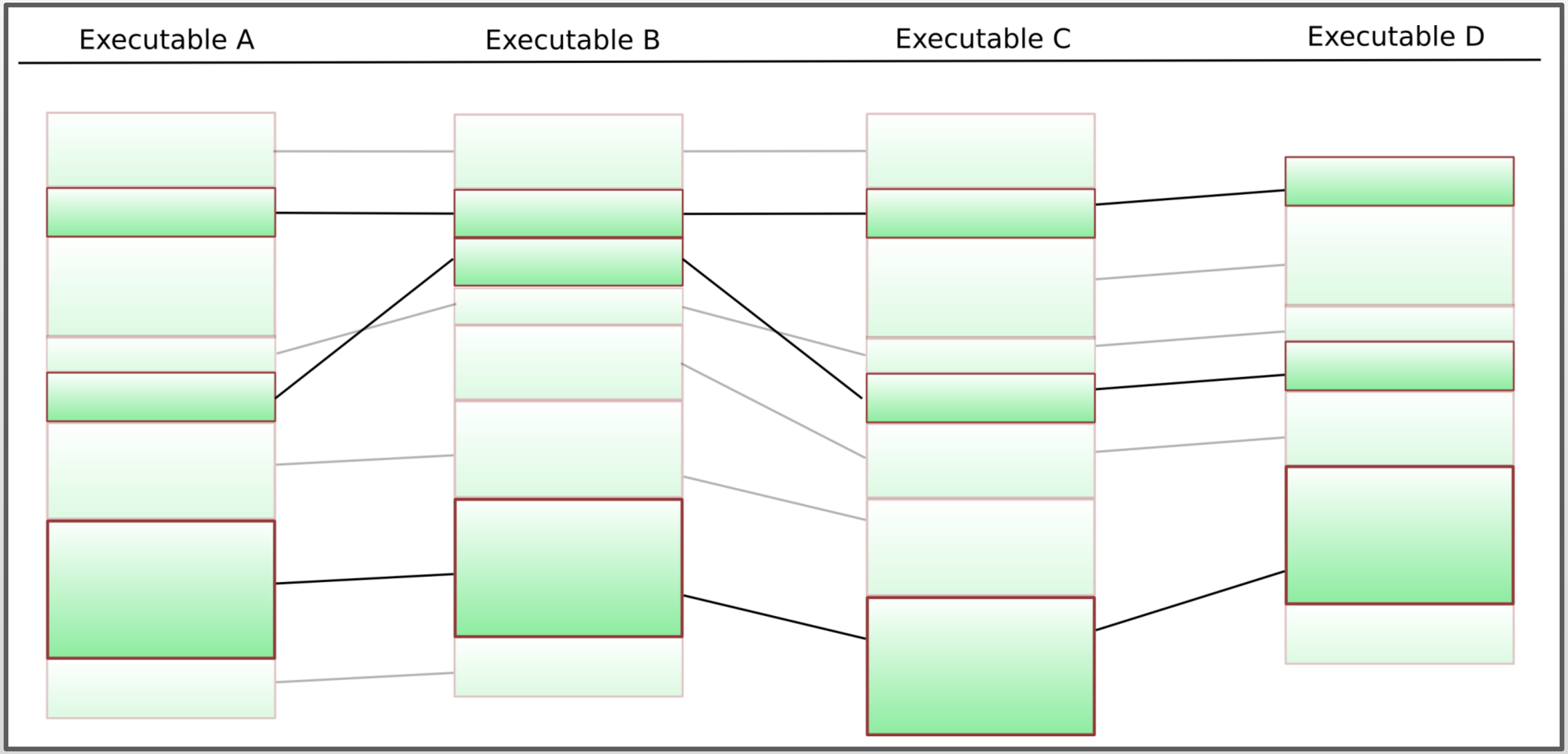


Signature Generator

zynamics VxClass

- A k-LCS algorithm makes sure only functions that are in the same order retained

  this works because functions are identified by their address



Signature Generator

zynamics VxClass

- Repeat for the basic blocks in each executable

→ Results in a "common core" of basic blocks

    these occur in all items of the cluster and in the same order

- Compute regular k-LCS to determine common byte sequences

    approximate (exact k-LCS has exponential complexity)

- Fill the "gaps" with wildcards:

```
Worm.SigGen-20080603193253-1876:0:*:03ff*0100*4424*8d54
24*8d4fff*525051*0100*83c418*85c07c*8b4424*8d4c24*4100*
51c64424*02e8*5424*83c40c8d4424*52685c*4100*8b4424*8b46
048b4e08*d1e085c9894604*4100*535657a8018bf175*84c074*8a
d0b9*4100...
```

Converted to ClamAV format

Signature Generator

zynamics VxClass

Stats: Classified 5000 random executables from VirusTotal and named resulting clusters.
Signatures were generated and applied to 15000 new executables.

Cluster Name	# executables	sig. size in bytes	new detected variants
Win32.KillAV.Variants	183	1785	111
Win32.Bacuy.Variants	599	27942	863
Win32.SkinTrim	173	290318	356
Win32.SwizzorA	15	69286	929
Win32.WinTrim	114	3925	126
FakeAlert	54	460	0
Win32.Chifrax	12	40098	26

Signature Generator

zynamics VxClass

- What about false positives?
  - Scanning 22239 known-good executables (ClamAV)
  - Seemingly false positives were always due to bugs
    bugs in FLIRT, bugs in our library filtering
  - False positive rates empirically around 0.005%

- What about false negatives, then?
  - By construction always either valid signature or none

Signature Generator

zynamics VxClass

- Signatures consist of malware bytes minus the "variable" bits
  - → generated signatures carry some "predictive power"
  - → All except one of the generated signatures caught some "new" variant of the same malware

Performance

- One (beefy) machine processes ≈1400 samples/day
  - Includes unpacking
  - Higher performance if specific unpacking happens first
- Scalable: VxClass routinely runs on a 4-machine cluster
  - Scaling to 20-25 machines should be possible

Limitations

zynamics VxClass

- Heavy obfuscation of control flow
  breaks classification and leads to empty signatures
- Virtualizing packers
- Unpacking only works on 32-bit Windows
  - No Linux/Mac OS X/Mobile unpacking
  - 64-bit support is in the works
- Using the generated signatures in a AV product requires good unpacking capability
  signatures are generated post-unpacking

But: Manual intervention possible (upload IDBs)

Questions?

http://zynamics.com/

http://blog.zynamics.com/

christian.blichmann@zynamics.com