Tracking Memory Writes for Malware Classification and Code Reuse Identification

André Ricardo Abed Grégio[1,2], Paulo Lício de Geus[2], Christopher Kruegel[3], Giovanni Vigna[3]

[1]CTI Renato Archer, [2]University of Campinas, [3]UC Santa Barbara

DIMVA 2012

July 27, 2012

Agenda

Motivation

Current Hard Problems in INFOSEC Research. DHS Report, Nov/2009, Issue 7 of 11: Combatting Malware and Botnets, p.43.

"A/V and IDS/IPS approaches are becoming less effective because malware is becoming increasingly sophisticated (...) "

Proposal

- An approach to capture and model malware behavior by tracking instructions writing into registers/memory.
- A two-step procedure to cluster malware (based on their traces similarity) and to identify code reuse.

Agenda

Behavioral Extraction

PoC Tracer: PyDBG

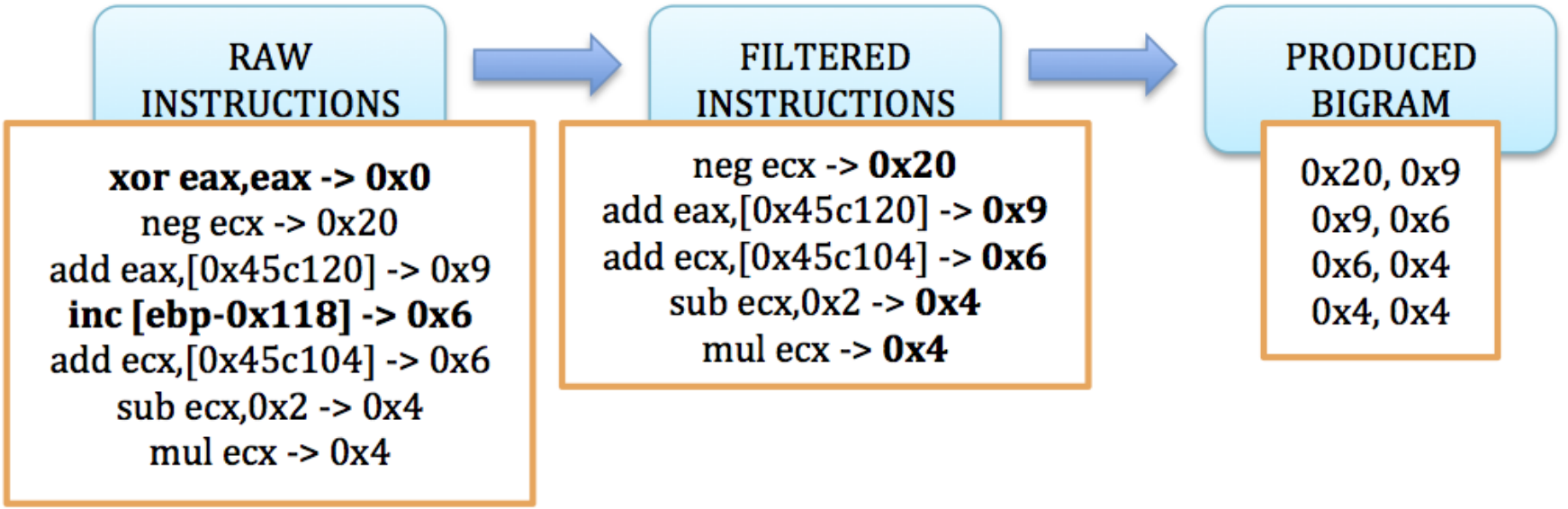
If `dbg.context.Eip > 0x70000000 and ret_addr < 0x70000000`
then `dbg.bp_set(ret_addr)` *[Win XP DLL range]*

- Single-stepping, software breakpoints and basic "hiding".

Subset of Instructions

- **Logged:** `add`, `adc`, `sub`, `sbb`, `mul`, `imul`, `div`, `idiv`, `neg`, `xadd`, `aaa`, `cmpxchg`, `aad`, `aam`, `aas`, `daa`, `das`, `not`, `xor`, `and`, `or`. (logic and arithmetic operations)
- **Removed:** `inc`, `dec`; write value == "0". (simple counter, little aggregated information)

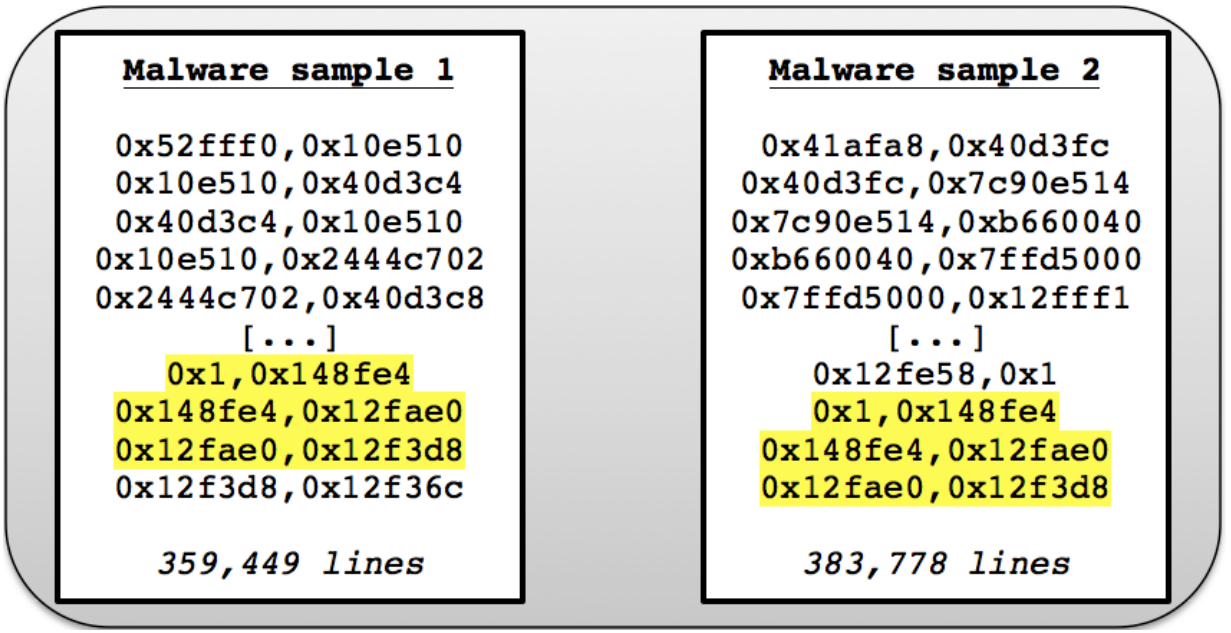
Data Processing



Agenda

Two-step Algorithm

- **INPUT:** Two data value traces (malware samples).
  1. Quick Comparison (faster, decisory)
  2. Full Similarity (computes the overlap)
- **OUTPUT:** Similarity measure ranging from 0 (completely different) to 1 (identical)



Step 1: Quick Comparison

- Each trace receives the 100 least-frequent bigrams as an identifier (experimentally determined).
- It discards the most common bigrams and focus on the specifics of a certain family fingerprint.

Moving forward...

ID_{M_1}, ID_{M_2}: M_1, M_2 (traces) 100-least frequent bigrams.

$J(ID_{M_1}, ID_{M_2}) = ID_{M_1} \cap ID_{M_2} / ID_{M_1} \cup ID_{M_2}$ (Jaccard).

IF $J(ID_{M_1}, ID_{M_2}) > 0.3$ **THEN**
    Next step!
ELSE
    Similarity == $J(ID_{M_1}, ID_{M_2})$

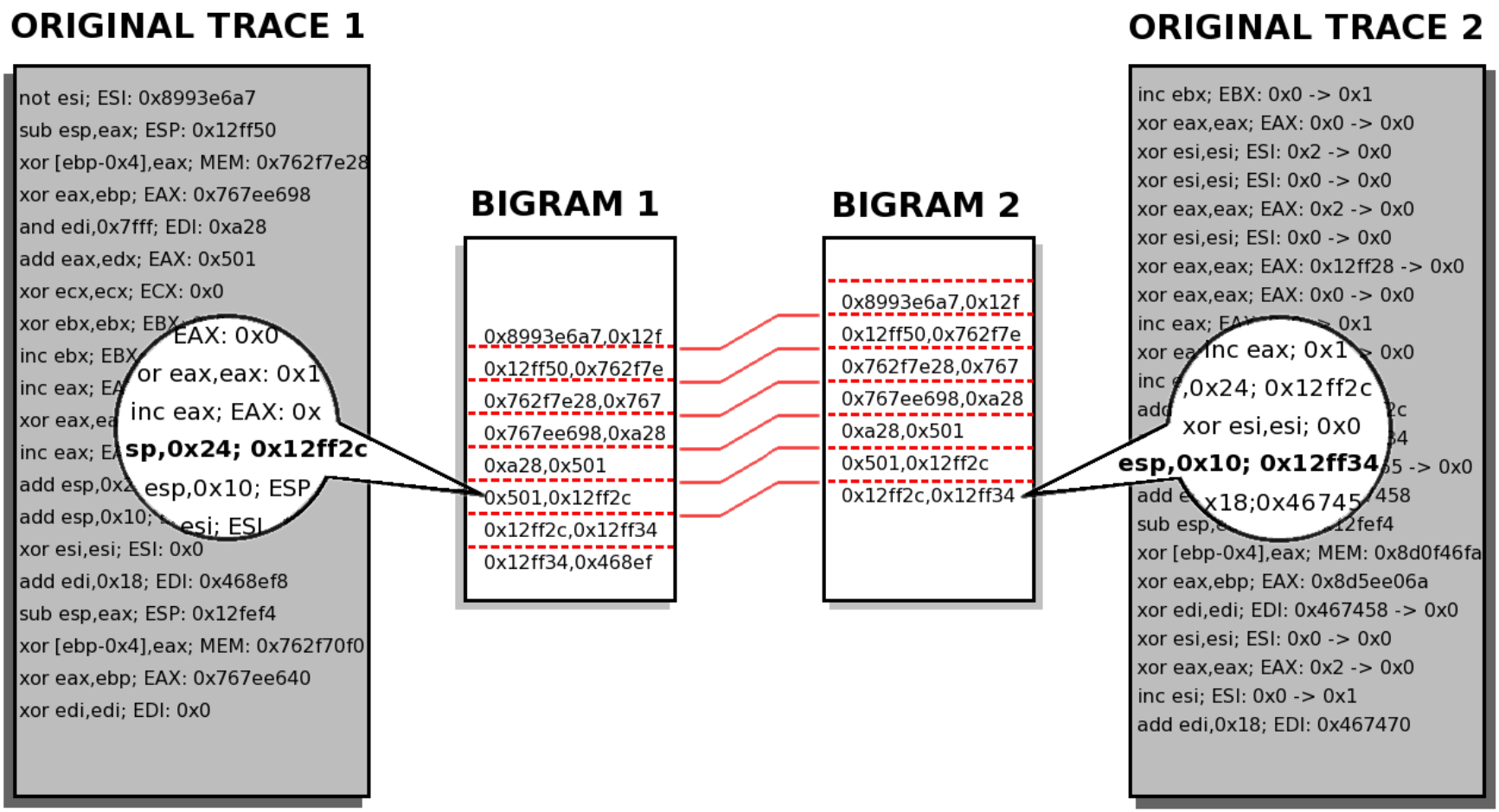
Step 2: Full Similarity Computation

- Longest Common Subsequence (LCS) of malware traces T_1, T_2, whose lengths are L_1, L_2.
- Similarity of M_1, M_2 is then the "containment rate" $C(M_1, M_2) = LCS(T_1, T_2)/min(L_1, L_2)$.

eDiff

Approximates LCS computation. Marks differing regions between traces. Maps the shared subsequences to the original instructions.

eDiff


ORIGINAL TRACE 1
not esi; ESI: 0x8993e6a7
sub esp,eax; ESP: 0x12ff50
xor [ebp-0x4],eax; MEM: 0x762f7e28
xor eax,ebp; EAX: 0x767ee698
and edi,0x7fff; EDI: 0xa28
add eax,edx; EAX: 0x501
xor ecx,ecx; ECX: 0x0
xor ebx,ebx; EBX:
inc ebx; EBX:
inc eax; EA
xor eax,ea
inc eax; EA
add esp,0x2
add esp,0x10;
xor esi,esi; ESI: 0x0
add edi,0x18; EDI: 0x468ef8
sub esp,eax; ESP: 0x12fef4
xor [ebp-0x4],eax; MEM: 0x762f70f0
xor eax,ebp; EAX: 0x767ee640
xor edi,edi; EDI: 0x0
EAX: 0x0
or eax,eax: 0x1
inc eax; EAX: 0x
sp,0x24; 0x12ff2c
esp,0x10; ESP
esi; ESI
BIGRAM 1
0x8993e6a7,0x12f
0x12ff50,0x762f7e
0x762f7e28,0x767
0x767ee698,0xa28
0xa28,0x501
0x501,0x12ff2c
0x12ff2c,0x12ff34
0x12ff34,0x468ef
BIGRAM 2
0x8993e6a7,0x12f
0x12ff50,0x762f7e
0x762f7e28,0x767
0x767ee698,0xa28
0xa28,0x501
0x501,0x12ff2c
0x12ff2c,0x12ff34
ORIGINAL TRACE 2
inc ebx; EBX: 0x0 -> 0x1
xor eax,eax; EAX: 0x0 -> 0x0
xor esi,esi; ESI: 0x2 -> 0x0
xor esi,esi; ESI: 0x0 -> 0x0
xor eax,eax; EAX: 0x2 -> 0x0
xor esi,esi; ESI: 0x0 -> 0x0
xor eax,eax; EAX: 0x12ff28 -> 0x0
xor eax,eax; EAX: 0x0 -> 0x0
inc eax; EA 0x1
xor ea inc eax; 0x1 0x0
inc ,0x24; 0x12ff2c
xor esi,esi; 0x0
esp,0x10; 0x12ff34
x18;0x46745
sub esp,
xor [ebp-0x4],eax; MEM: 0x8d0f46fa
xor eax,ebp; EAX: 0x8d5ee06a
xor edi,edi; EDI: 0x467458 -> 0x0
xor esi,esi; ESI: 0x0 -> 0x0
xor eax,eax; EAX: 0x2 -> 0x0
inc esi; ESI: 0x0 -> 0x1
add edi,0x18; EDI: 0x467470

Agenda

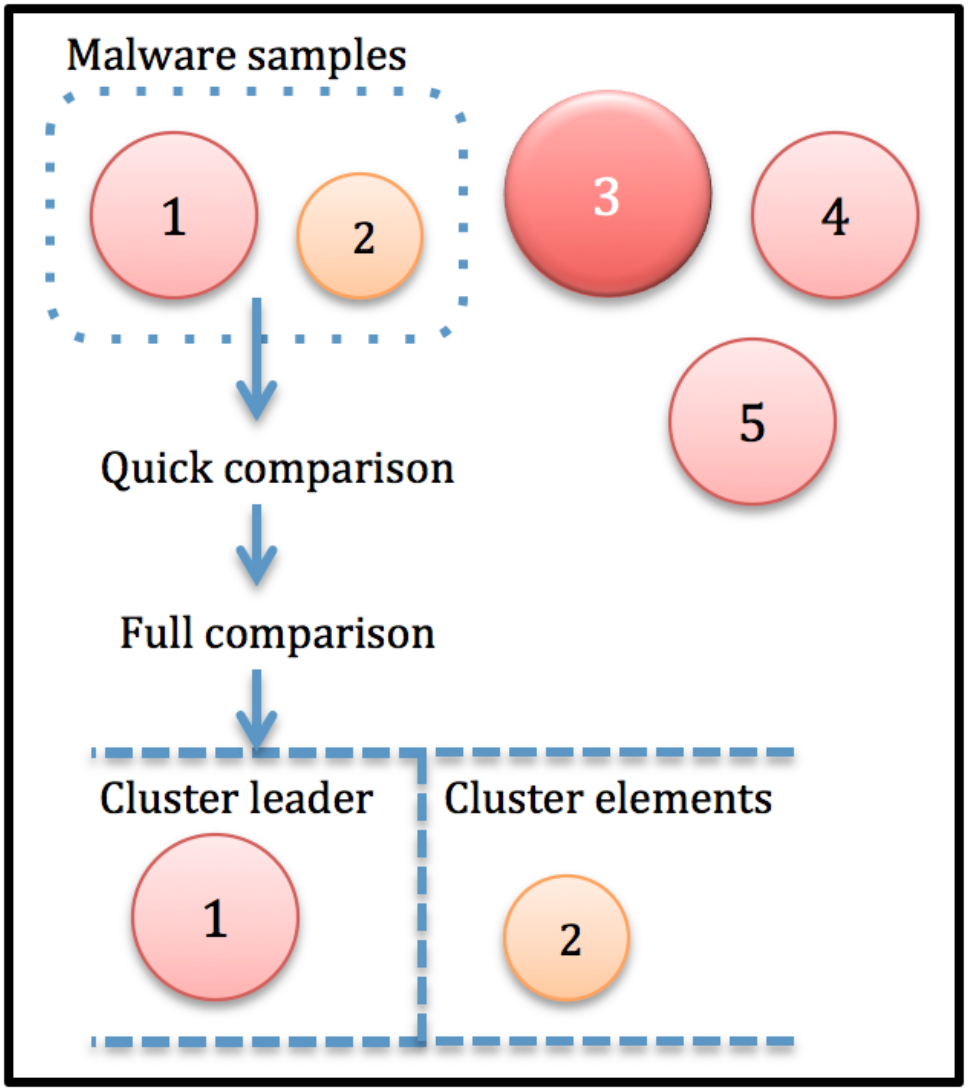
Clustering

- **INPUT:** a set of N malware traces to be clusteres.
- **OUTPUT:** groups of malware samples that are similar.

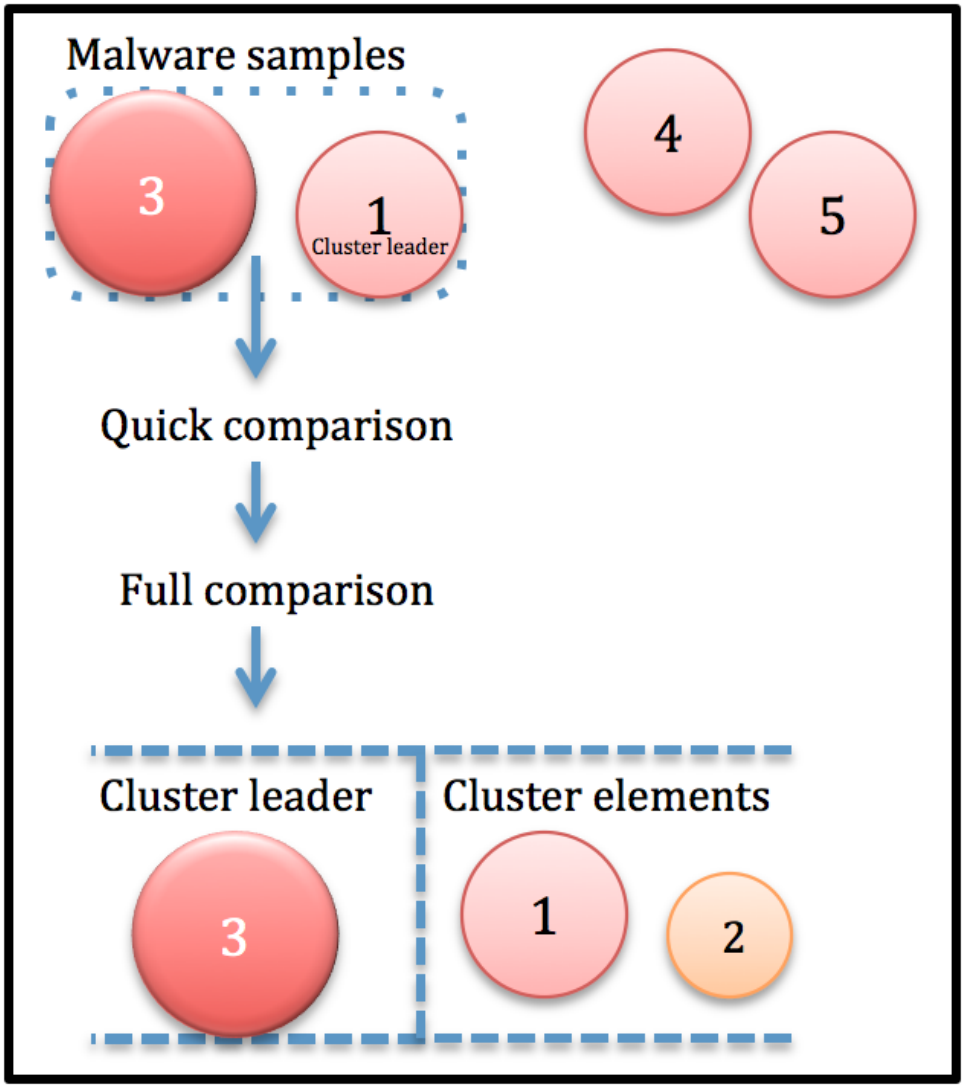
The clustering process is implemented in two steps:

1. **Pre-clustering:** quickly generates an initial clustering whereas avoiding $\frac{N^2}{2}$ comparisons. It defines cluster leaders (longest traces) and groups samples that exhibit over 70% of similarity.
2. **Inter-cluster merging:** merges clusters whose traces are quite similar but whose least-frequent bigrams are too different to pass the 70% threshold.

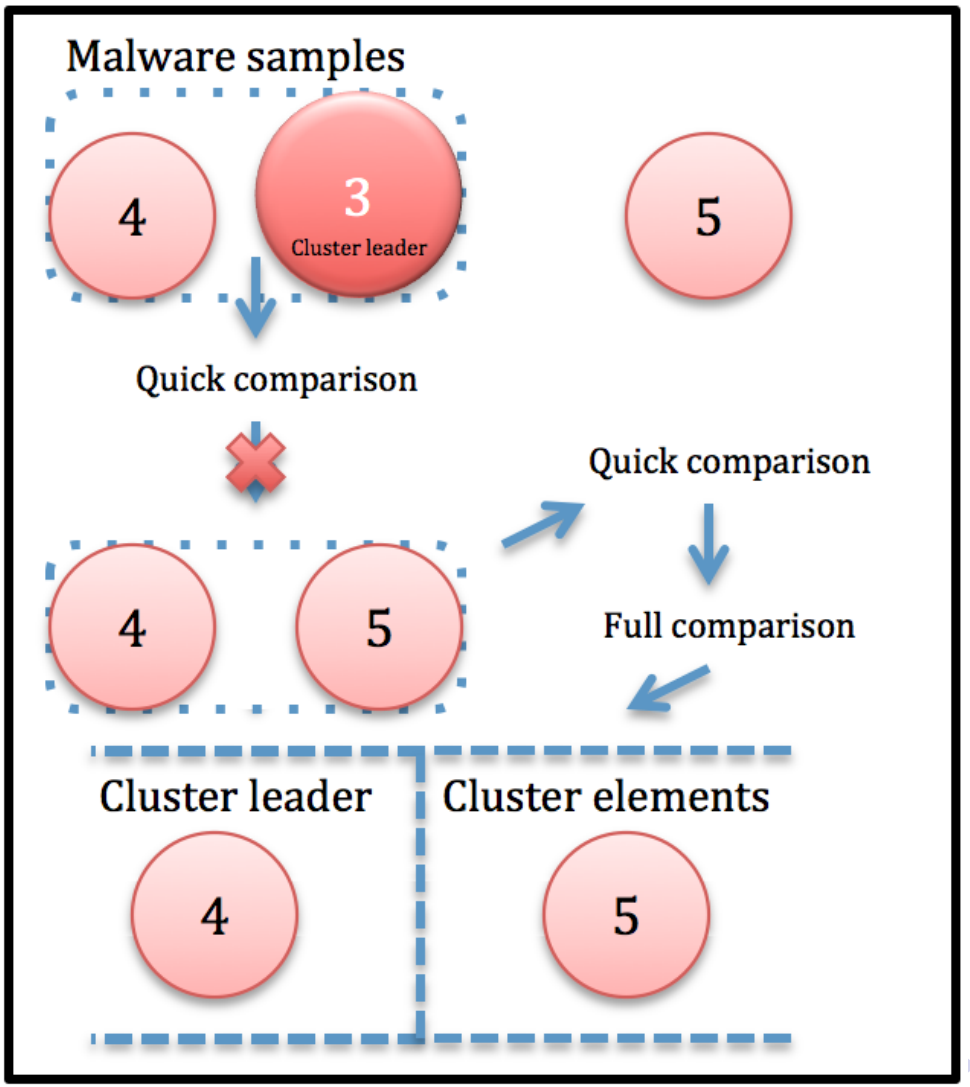
Pre-clustering [I]



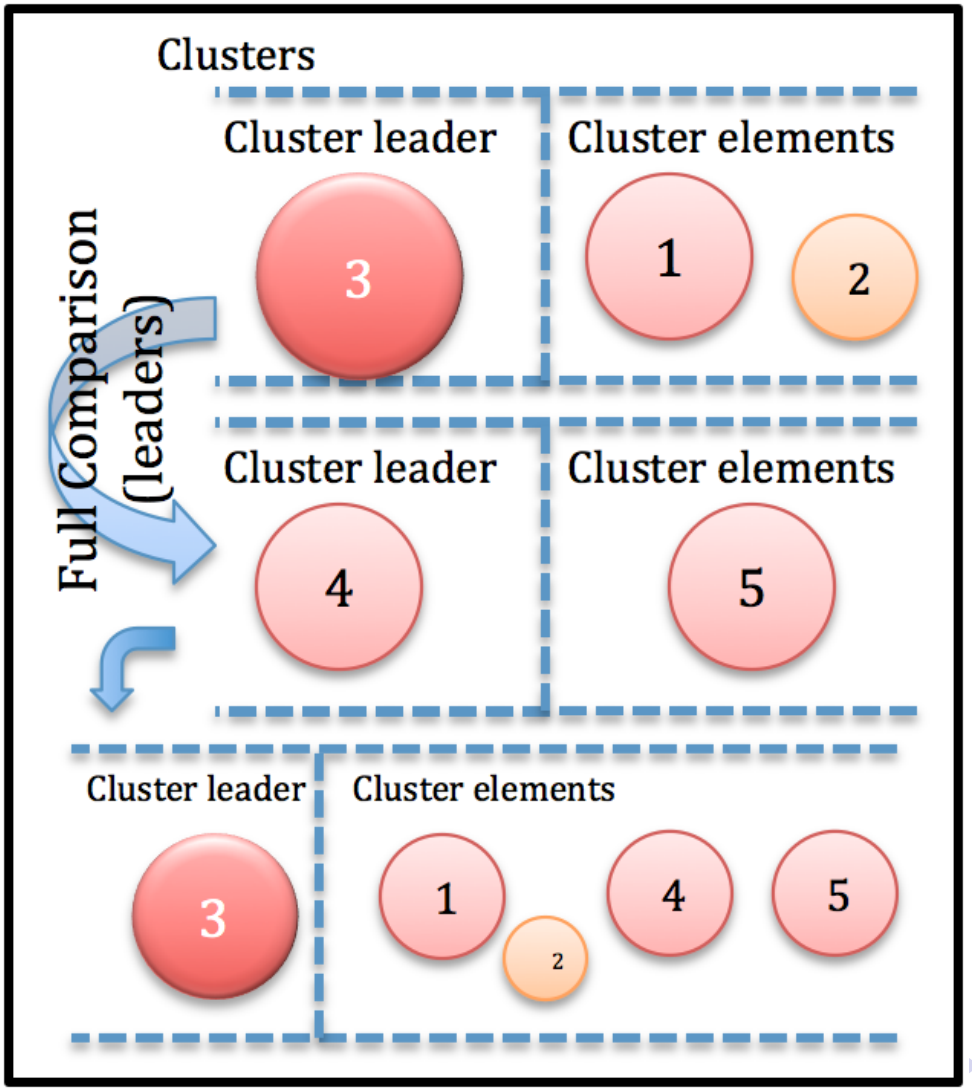
Pre-clustering [II]



Pre-clustering [III]



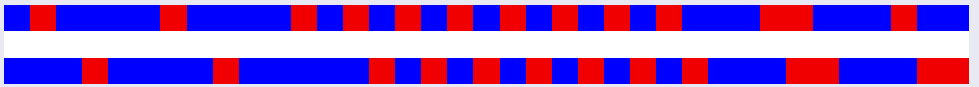
Inter-cluster merging



Code Reuse Identification

- Identical values in two traces $\Longrightarrow$ similar code?
- Bigrams: *eDiff*; Instructions: RegExp!

Blocks of shared code



Agenda

Environment/Data:

- Windows XP SP3 emulated over QEMU-KVM.
- 16,248 malware samples (execution traces).

Clustering evaluation

Complicated task (lack of official ground truth); metrics required!

- Precision: Clusters contain only samples from the same family?
- Recall: Samples from the same family are clustered together?
- Quality: $Q = P \times R$

Ground Truth

Static Filter

Clusters from information obtained by malware static analysis.

G. Jacob et al. A Static, Packer-Agnostic Filter to Detect Similar Malware Samples. 1 hour ago...

Behavioral Clustering

Clusters based on behavior extracted from dynamic execution.

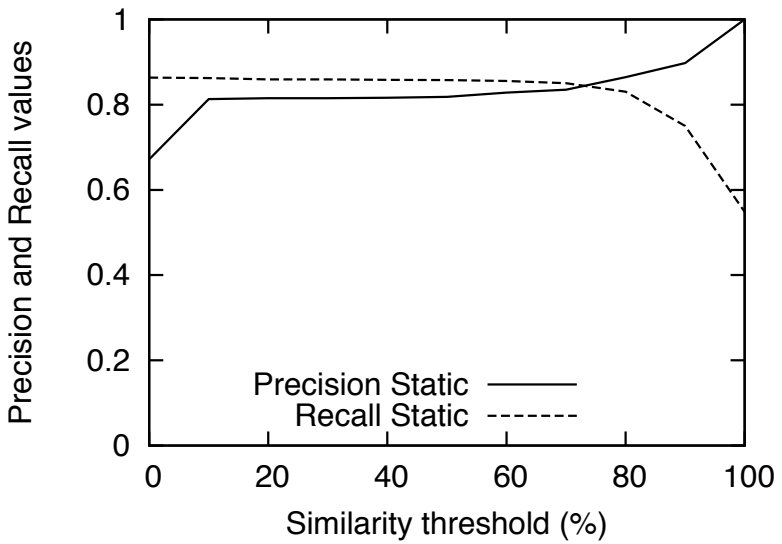
U. Bayer et al. Scalable, Behavior-Based Malware Clustering. NDSS 2009.

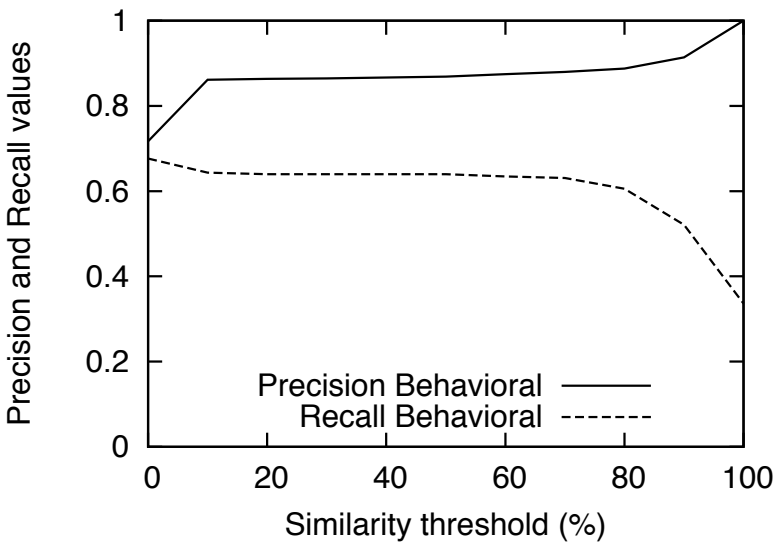
AV Labels

Level of Agreement: AV vendors "normalized" assigned labels.

MALWARE MD5	AVG,AVIRA,F-PROT
072cb45db4b7e34142183bc70bf8b489	agent_r,agent,busky
050a0b8b78cad111352b372417e467fe	agent_r,agent,busky
063d85386df0edb28b3f0182b83a4fe3	agent_r,**runner**,busky

Results - Reduced dataset (1000 samples)





Quality, amount of clusters (#), similarity **T**hresholds.

T	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
#	551	583	586	586	588	591	602	612	639	734	1000
Q_S	.579	.701	.700	.700	.700	.701	.708	.710	.717	.673	.548
Q_B	.485	.554	.552	.552	.554	.555	.554	.554	.537	.476	.336

AV label's Level of Agreement ($T = 70\%$): .894

Results - Full dataset (16,248 samples)

- Static Filters: 7,900 clusters
- Behavioral Clustering: 3,410 clusters
- Our approach 7,793 clusters

Comparison - Reference clustering sets

Ref. Clustering	Precision	Recall	Quality
Static	0.758	0.810	0.614
Behavioral	0.846	0.572	0.485
AV labeling	-	-	0.871

- More specialized $\Longrightarrow$ better recall (split samples according to the structural information, packer etc.)
- More generalized $\Longrightarrow$ better precision (general behavior of a family after unpacking tends to be similar)
- AV total agreement: 8%; AV "clean": 13.93%.

Agenda

Limitations and Future Work

- Classification algorithms can be subverted once you know the *rules*, leading to bad clustering.
- Debuggers are usually detectable (test with other approaches).
- Internal components (dynamic analyzer) can be subverted.
- This is PoC.

BUT...

- Malware analysis are subject to split personalities, bogus instructions, detection and evasion, stalling, crashing due to interaction with some component etc.

SO...

- New ideas and techniques are useful to defense purposes.

Thanks

UCSB SecLab, Unicamp Institute of Computing, CTI Information Systems Security Division. Contact: argregio@cti.gov.br.