Classification (and Detection) of Metamorphic Malware Using Value Set Analysis

Felix Leder – leder@cs.uni-bonn.de
Bastian Steinbock – steinboc@cs.uni-bonn.de
Peter Martini – martini@cs.uni-bonn.de
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encrypted malware:</td>
<td>Encrypted virus body is decrypted at run-time</td>
</tr>
<tr>
<td></td>
<td>(example: UPX)</td>
</tr>
<tr>
<td></td>
<td>Decryptor</td>
</tr>
<tr>
<td></td>
<td>Virus body</td>
</tr>
<tr>
<td>Polymorphic malware:</td>
<td>Morphing/varying decryptor stub</td>
</tr>
<tr>
<td></td>
<td>Decryptor</td>
</tr>
<tr>
<td></td>
<td>Virus body</td>
</tr>
<tr>
<td></td>
<td>Decryptor</td>
</tr>
<tr>
<td></td>
<td>Decryptor</td>
</tr>
<tr>
<td>Metamorphic malware:</td>
<td>Morphes the whole virus body.</td>
</tr>
<tr>
<td></td>
<td>V iru s b o dy</td>
</tr>
<tr>
<td></td>
<td>V</td>
</tr>
<tr>
<td></td>
<td>iru o dy s b V</td>
</tr>
</tbody>
</table>
Metamorphic Malware: Is there really a threat?

Metamorphic Malware is **hardly detectable with regular string signatures**.

Virus scanners use customized detection engines for each family.

**Problem:**

In 2009 Symantec detected more than 2.8 million new malware specimen. (170% growth)

**Impossible to analyze every sample or by hand. Pre-classification needed.**

**Bad detection example – Lexotan32:**

- File infecting virus from 2002
- Virus total detection rate in 2009: 12.9%
- None of 40 scanners detected all of the samples
• Code changes completely
• Common subsequences have sizes of max. 5 bytes
• Every infection looks completely different

```plaintext
1: jmp 4
2: reg_2 = reg_1+2
3: jmp ...
4: reg_1 = 5
5: jmp 2

reg_1 = 5
reg_2 = reg_1+2

Result always 7
```
• While structure changes, behavior has to stay the same
• Existing approaches:
  – Code normalization: Standard representations
    • metamorphism can be very complex
    • only shown for W32/Evol, self-made examples
  – Execution traces / blackboxing: Possible but easy to defeat
    (Waiting, changing execution order, environment detection)
Static Behavioral Detection

- **Static analysis** investigates whole sample without execution
- Behavior is reflected by values / memory contents
- Each program contains characteristic values it cannot change:

```plaintext
For I = 0 to 1000 do:
...
socket(AF_INET = 2, SOCK_STREAM = 6, PF_INET = 2)
sockaddr_in.sin_port = 80
connect(...)

If var_1 > 9:
  var_1 = 10
```
Methodology

VALUE SET ANALYSIS
Value Set Analysis - VSA:

“What values are possible for a specific variable/memory-location at a specific location inside the program?”

- **Static data flow tracking** and approximate memory contents
- Scalability: Over-approximate when too complex

Start (0x100):

```assembly
mov eax, 1
mov ebx, start
add eax, ebx
```

Value Sets:

- eax = {1}; ebx = {}
- eax = {1}; ebx = {0x100}
- eax = {0x101}; ebx = {0x100}
Var_1 := 1
Var_1 = {1}

Var_2 := Var_1
Var_2 = {1}

Var_1 := 2
Var_1 = {2}

Var_2 := Var_1
Var_2 = {1, 2}

Var_1 := 3
Var_1 = {3}

Var_2 := Var_1
Var_2 = {1, 2, 3}
VSA - Examples – Arithmetic Operations

\[
\begin{align*}
\text{Var}_1 & := 1 \\
\text{Var}_2 & := \text{Var}_1 + 1 \\
\text{Var}_3 & := \text{Var}_1 \times 2
\end{align*}
\]

\[
\begin{align*}
\text{Var}_1 & := 2 \\
\text{Var}_2 & = \{2,3,4\}; \text{Var}_1 = \{1,2,3\} \\
\text{Var}_3 & = \{2,4,6\}; \text{Var}_2 = \{2,3,4\}; \text{Var}_1 = \{1,2,3\}
\end{align*}
\]

\[
\begin{align*}
\text{Var}_1 & := 3 \\
\text{Var}_2 & = \{2,3,4\}; \text{Var}_1 = \{1,2,3\} \\
\text{Var}_3 & = \{2,4,6\}; \text{Var}_2 = \{2,3,4\}; \text{Var}_1 = \{1,2,3\}
\end{align*}
\]
\[
\text{Var}_1 := 1 \\
\text{Var}_1 = \{1\}
\]

\[
\text{Var}_2 := \text{Var}_1 + 1 \\
\text{If Var}_2 > 2 \\
\text{Var}_2 = \{2\} \\
\text{Var}_3 := 56 \\
\text{Var}_3 = \{56\} \\
\text{Var}_2 = \{2, 3\} \\
\text{Var}_1 = \{1, 2\}
\]

\[
\text{Var}_3 := 34 \\
\text{Var}_3 = \{34\} \\
\text{Var}_2 = \{3\} \\
\text{Var}_2 = \{2, 3\} \\
\text{Var}_1 = \{1, 2\}
\]

\[
\text{Var}_2 = \{2, 3\}; \text{Var}_1 = \{1, 2\}
\]

\[
\text{Var}_1 := 2 \\
\text{Var}_1 = \{2\}
\]

\[
\text{Var}_2 = \{2, 3\}; \text{Var}_1 = \{1, 2\}
\]

\[
\text{Var}_2 = \{3\} \\
\text{Var}_3 := 34 \\
\text{Var}_3 = \{34\} \\
\text{Var}_2 = \{4\} \\
\text{Var}_1 = \{1, 2\}
\]

\[
\text{Var}_3 = \{34, 56\} \\
\text{Var}_2 = \{2, 3\} \\
\text{Var}_1 = \{1, 2\}
\]
Determine Characteristic Value Sets

REFINEMENT
• Metamorphic malware is often file infecting
• Challenge: Distinguish host/malware Value Sets
METHODOLOGY
• **Matching**: How to quantify the similarity of two ...

**Data Objects**

\[ \text{eax} = \{1, 2, 3\} \times \text{eax} = \{1, 2, 99\} = ?? \%

**Value Sets**

\[ \text{eax} = \{\ldots\} \text{ebx} = \{\ldots\} \times [\text{sp-4}] = {} \text{esi} = \{\ldots\} = ?? \%

**Files**

\[ \text{ } \times \text{ } = ?? \% \]
Different matching strategies possible:

- **Average matching**: Score = % of equal elements

- **Threshold matching**: Score = \[
\begin{cases} 
1 & \text{if } \% \text{ of equal elements} > \Delta \\
0 & \text{otherwise}
\end{cases}
\]

\[
\text{eax} = \{1, 2, 3\} \times \text{eax} = \{1, 2, 99\} = 66\% 
\]

Average score : 66%
Threshold score with (\(\Delta = 0.6\)) : 100%
Threshold score with (\(\Delta = 0.7\)) : 0 %

Have to be used for all layers: Data Obj., Value Sets, Files
Penalties for (unsimilar) Data Objects - Data Object Adjustments

- \(|\text{Refined set}\, | < \, |\text{Infected set}\, |
  \)
  \[
  \text{eax} = \{1\} \quad \times \quad \text{eax} = \{1,3,5,10,99,1010,445,110,22,1337\} \quad \Rightarrow \text{score} - x% \]

- Location of the value (stack, heap, global memory)
  \[
  \text{[global\_12345678]} = \{13\} \quad \times \quad \text{[esp-4]} = \{13\} \quad \Rightarrow \text{score} - y% \]

- Similar stack offsets indicate more similarity
  \[
  \text{[ebp - 4]} = \{13\} \quad \times \quad \text{[ebp - 128]} = \{13\} \quad \Rightarrow \text{score} - z% \]
PARAMETER DERIVATION
• Lexotan32 (sophisticated metamorphic engine)
• 25 variants, 25 benign programs
• Sensitivity Analysis → best combination
24 setups with perfect separation (out of 192)

Question: Luck, small test set, or parameter influence?
Which parameters have strongest impact?  
(or need to be set specifically)

Correlation - Parameters to Quality

- False Neg.
- False Pos.
Results – Best Parameters

- DO Adjustments (Penalty) of 30%
- Threshold matching
- High threshold for Files and Data Objects
- Threshold for Value Sets almost irrelevant

Are those parameters family dependent?
Evaluation for Parameter Generality

MALWARE DETECTION USING VSA
• 7 different metamorphic malware: W32...
  – Lexotan32
  – Evol
  – AOC
  – Blackbat
  – Bolzano
  – Hatred
  – Hezhi

• Each test set: 55 files:
  – 5 infected for refinement
  – 25 infected
  – 25 clean
100 % Separation = 0 FALSE NEGATIVES, 0 FALSE POSITIVES

All instruction detection results

[Bar chart showing detection results for different instructions with 100% separation and no false negatives or positives]
• False Positive distance
• How characteristic are the Value Sets?
• False positive estimation for real-world

**Parameter set is suitable for other families, too!**
Decreasing complexity → Only specific Points Of Interest in prog.

<table>
<thead>
<tr>
<th>Malware</th>
<th>All instruction POIs</th>
<th>Jump POIs</th>
<th>Call POIs</th>
<th>Function POIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>W32/AOC</td>
<td>✓</td>
<td>1 f.p. / 0 f.n.</td>
<td>no value sets</td>
<td>no value sets</td>
</tr>
<tr>
<td>W32/BlackBat</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>W32/Bolzano</td>
<td>✓</td>
<td>✓</td>
<td>no value sets</td>
<td>no value sets</td>
</tr>
<tr>
<td>W32/Evol</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1 f.p. / 0 fn</td>
</tr>
<tr>
<td>W32/Hatred</td>
<td>✓</td>
<td>✓</td>
<td>no value sets</td>
<td>no value sets</td>
</tr>
<tr>
<td>W32/Hezhi</td>
<td>✓</td>
<td>no value sets</td>
<td>no value sets</td>
<td>no value sets</td>
</tr>
</tbody>
</table>

✓- 100% detection, 0 false positives

Summary:
- 2 False positives in 120 samples
- No false negatives
- Refinement to strict for special POI types
Larger sample sets

CLASSIFICATION OF METAMORPHIC FAMILIES
**Setup**

- 4197 samples from MWCollect database
- 7 metamorphic families
- Same parameter set as before
- (All Instructions)

**Classification goal – perfect separation**

- Variant classification as family members
- All other samples as non-members
Classification - Results

Similarity Scores per Family

Similarity Scores > 0 per Family

- Evol
- AOC
- Bolza

- Evol
- Bolzano
- Hatred
- Lexotan32
- Blackbat
- Hezhi

0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Competition

RUNTIME PERFORMANCE
Best classification is unusable if too slow for use case

**Existing approaches and use-cases**

- **Classification (mostly blackbox):**
  - Run time: 2 to 10 minutes + classification time

- **Virus-Detection (on-demand)**
  - Application slow-down: 100% - 200% overhead for most AV
  - Data throughput: 3.6 – 18 MB/s

- **Mail gateways**
  - Greylisting introduces delays of 5 - 15 minutes
Run-Time measurements give upper bound...

- IDA Pro – unnecessary analysis steps
- Value Set Analysis – Python (IDAPython Integration)
- Matching – Python
- C up to 280 times faster than Python [Armin Rigo - Psycho]
Run-Time - Results Overview

- Average total analysis time / sample: 7.9 s
- Average match time +0.28 s

- Data throughput: 20 KB/s

Current implementation...
+ Faster than Blackbox
+ Ok for mail gateways
- Too slow for on-demand scanning
  (may be an additional means to AV)
Average match time per family

- AOC
- Beesee
- Devil
- Helmed
- Hertzli
- Lexicoder2

Time [s]

0.0  0.1  0.2  0.3  0.4  0.5
Summary

• Structure of metamorphic malware changes but (general) behavior stays similar

• Static Analysis can estimate behavior based on data flow relations and values → Value Set Analysis

• Strict parameters allow for good differentiation
• Detection possible
• Classification/differentiation from other families perfect

• Run-time ok for classification, mail gateways, ...
• Too slow for on-demand scanning
Malware Boot Camp
• Summer- and winter-school
• February and September
• 5 weeks hardcore fun
Number of values in real malware

W32/Lexotan32

- Original (primary) file: 188 VS - 933 Data Objects
- 1st refinement: 108 VS - 238 Data Objects
- 2nd refinement: 108 VS - 225 Data Objects
- 3rd – 9th refinement: no changes
Points of Interest (POI)

- Points in Executable that are likely to contain characteristic Value Sets
  - **All instruction POIs**: Every instruction may be interesting
  - **Jump POIs**: Decision dependent values
  - **Call POIs**: Function parameters and caller state
  - **Function POIs**: State at the beginning of an (internal) function
Aiming for **PERFECTION**...

(100% detection, 0 false positives)
• Outliers (up to 2000 s)
• Real-world: Impl. time limit