INTERACTIVE GRAPHICAL EXPLORATION OF MALWARE BEHAVIOR
Agenda

- Motivation
- Technical Background
- Implementation
- Examples
Motivation

- AV-TEST received over 140 million new malware samples in 2015
- AV-TEST processed hundreds of thousands malware samples with its dynamic analysis in the last few months
- Are we still identifying the new, interesting and important malware?
Motivation

AV-TEST dynamic analysis tool
## Motivation

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Module Name</th>
<th>Function Name</th>
<th>Flags</th>
<th>Base Address</th>
<th>FileName</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016-05-02 20:38:53,193</td>
<td>VirtualProtectEx</td>
<td>Protection</td>
<td>0x00000040</td>
<td>0x3000000001</td>
<td>0x000000004</td>
<td>success</td>
</tr>
<tr>
<td>2016-05-02 20:38:53,193</td>
<td>VirtualProtectEx</td>
<td>ProcessHandle</td>
<td>0xffffffff</td>
<td></td>
<td></td>
<td>success</td>
</tr>
<tr>
<td>2016-05-02 20:38:53,193</td>
<td>VirtualProtectEx</td>
<td>Address</td>
<td>0x30001634</td>
<td></td>
<td></td>
<td>success</td>
</tr>
<tr>
<td>2016-05-02 20:38:53,193</td>
<td>VirtualProtectEx</td>
<td>Size</td>
<td>0x00000004</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2016-05-02 20:38:53,203</td>
<td>LdrLoadDll</td>
<td>Flags</td>
<td>0x244912</td>
<td>0x3124000000</td>
<td>wlib.dll</td>
<td>success</td>
</tr>
<tr>
<td>2016-05-02 20:38:53,203</td>
<td>LdrGetProcedureAddress</td>
<td>Ordinal</td>
<td>0</td>
<td></td>
<td></td>
<td>success</td>
</tr>
<tr>
<td>2016-05-02 20:38:53,203</td>
<td>LdrGetProcedureAddress</td>
<td>FunctionName</td>
<td>Main</td>
<td>0x31244562</td>
<td></td>
<td>success</td>
</tr>
<tr>
<td>2016-05-02 20:38:53,203</td>
<td>LdrGetProcedureAddress</td>
<td>ModuleHandle</td>
<td>0x31240000</td>
<td></td>
<td></td>
<td>success</td>
</tr>
<tr>
<td>2016-05-02 20:38:53,203</td>
<td>LdrGetProcedureAddress</td>
<td>FunctionAddress</td>
<td>0x31621275</td>
<td></td>
<td></td>
<td>success</td>
</tr>
<tr>
<td>2016-05-02 20:38:53,203</td>
<td>LdrGetProcedureAddress</td>
<td>ModuleHandle</td>
<td>0x31240000</td>
<td></td>
<td></td>
<td>success</td>
</tr>
<tr>
<td>2016-05-02 20:38:53,203</td>
<td>LdrGetProcedureAddress</td>
<td>FunctionName</td>
<td>wdCommandDispatch</td>
<td>0x315c1c80</td>
<td></td>
<td>success</td>
</tr>
<tr>
<td>2016-05-02 20:38:53,203</td>
<td>LdrGetProcedureAddress</td>
<td>ModuleHandle</td>
<td>0x31240000</td>
<td></td>
<td></td>
<td>success</td>
</tr>
<tr>
<td>2016-05-02 20:38:53,203</td>
<td>LdrGetProcedureAddress</td>
<td>FunctionName</td>
<td>wdGetApplicationObject</td>
<td>0x315c1c80</td>
<td></td>
<td>success</td>
</tr>
<tr>
<td>2016-05-02 20:38:53,203</td>
<td>LdrGetProcedureAddress</td>
<td>ModuleHandle</td>
<td>0x31240000</td>
<td></td>
<td></td>
<td>success</td>
</tr>
<tr>
<td>2016-05-02 20:38:53,203</td>
<td>RegOpenKeyExW</td>
<td>Handle</td>
<td>0x0000000000</td>
<td></td>
<td></td>
<td>success</td>
</tr>
<tr>
<td>2016-05-02 20:38:53,203</td>
<td>RegOpenKeyExW</td>
<td>Registry</td>
<td>0x0000000002</td>
<td></td>
<td></td>
<td>success</td>
</tr>
<tr>
<td>2016-05-02 20:38:53,203</td>
<td>RegOpenKeyExW</td>
<td>SubKey</td>
<td>Software\Microsoft\Windows\Current</td>
<td></td>
<td></td>
<td>success</td>
</tr>
</tbody>
</table>

Cuckoo Sandbox on malwr.com
Raw Output of dynamic malware analysis can be
- Huge chunks of text
- Not intuitive
- Not nice to view
  → Not nice to work with as a human

Processing the data is only possible with automation, manual analysis is difficult and time consuming

Automated processing performs a defined task
- Does this very well and fast, e.g. classifying behavior as malicious or benign
  → Often doesn’t find (interesting) anomalies
Motivation

- Even abstracted data is difficult to process as human
Motivation

- Even abstracted data is difficult to process as human.

```
Features | Ergebnis
---------|----------
createsExecutablesInNonstandardDirectories | true
createsFilesInNonstandardDirectories | true
createsTemporaryFiles | true
createsTemporaryFiles.1 | true
createsTemporaryFiles.2 | true
numberOfCreatedFiles | 26
numberOfCreatedValues | 1
numberOfDeletedProcesses | 15
numberOfExecutableFiles | 18
numberOfModifiedValues | 10
numberOfUsedTopLevelDirectories | 1
ratioOfExecutables | RATIO 0.3
ratioOfNonstandardDirectories | RATIO 1
ratioOfSystemValues | RATIO 0.18191819
ratioOfTemporaryFiles | RATIO 1
ratioOfValues2Files | RATIO 0.3548387
```

Visualize it!
Technical Background - Sunshine

- ~ 100 Machines
- **Dynamic Data Analysis**: controlled execution of malware samples on non-infected systems
- Can run on both virtual and physical hardware
- Monitors
  - Modifications of the *filesystem*
  - Modifications of the *registry*
  - *Processes* and their modules
  - System areas in the *memory*
  - Incoming and outgoing *network traffic*
- Assigns a **flag for every action**: CREATED, MODIFIED, DELETED, NOFLAG
Data Analysis

- Behavior report is usually huge
- Features are extracted as a layer of abstraction and to reduce the data size
- We use three different types of features
  - Boolean features
  - Aggregation features
  - Ratio features
- Features can have sub-features
- These Features are combined into a feature vector describing the behavior of a sample
- Further usage of the feature vector: classification, …
Visualization

Source: Shaid, Maarof 2014

Source: Gove et al. 2014

Source: Zhuo, Nadji 2012
Visualization

- Features can have sub-features → **hierarchical visualization**
- Selection of hierarchical visualization techniques is limited
- Options: **node-link** and **space-filling** Diagram (e.g. a Treemap)
Visualization

- **Treemaps** generally unfit since comparison between them is hard
  - Aggregation features usually dominate the Treemap, while Boolean features disappear

- Common **top-down tree visualizations** are also unsuitable because of the high number of nodes

- → **radial trees**
Differentiating between nodes might still be difficult.

To make it easier, triggered nodes (value > 0) are highlighted and colored.

The used colors depend on the distribution of the associated flags.

The filling level depends on the maximum value inside the data set.
Visualization

- Edges are colored according to the **most frequently used flag** of the node the edge is pointing at.
- Node size depends on its **level**.
- Nodes further away from the root node are drawn **tinier**.
Implementation

- Frontend: Javascript (p5JS)
- Backend: Java (Spring Framework)
- 4 months of development
- ~8000 Lines of Code
- 40 files
#### Examples

- Several **mail attachments** from April 27th, same kind of mail
- Varying static detection, varying dynamic classification

→ Which file can be used for testing?

<table>
<thead>
<tr>
<th>Incident Name</th>
<th>Sample Date</th>
<th>Sha256</th>
<th>Size</th>
<th>Sample note</th>
<th>PEDumpHash</th>
<th>VT Gl</th>
<th>Mars</th>
<th>Class.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMG4251067555-JPG.scr</td>
<td>16-04-27 10:03</td>
<td>e525683a09a61</td>
<td>33219</td>
<td>view *</td>
<td>43b09d20d7</td>
<td>22 %</td>
<td>MALW</td>
<td>GOODWARE</td>
</tr>
<tr>
<td>IMG7431582762-JPG.scr</td>
<td>16-04-27 09:29</td>
<td>d730910c0cf028</td>
<td>41472</td>
<td>view *</td>
<td>6850a60f01</td>
<td>22 %</td>
<td>MALW</td>
<td>MALWARE</td>
</tr>
<tr>
<td>IMG5360612155-JPG.scr</td>
<td>16-04-27 08:25</td>
<td>250fb8dc150e68</td>
<td>41472</td>
<td>view *</td>
<td>406ea02061</td>
<td>26 %</td>
<td>MALW</td>
<td>MALWARE</td>
</tr>
<tr>
<td>IMG1208335660-JPG.scr</td>
<td>16-04-27 07:01</td>
<td>21953719df1f8a</td>
<td>41472</td>
<td>view *</td>
<td>6ae6d46f70</td>
<td>22 %</td>
<td>ANY</td>
<td>MALWARE</td>
</tr>
<tr>
<td>IMG9773080219-JPG.scr</td>
<td>16-04-27 06:48</td>
<td>52576443cb2c11t</td>
<td>41472</td>
<td>view *</td>
<td>6a314ca24d</td>
<td>17 %</td>
<td>ANY</td>
<td>MALWARE</td>
</tr>
<tr>
<td>IMG4593570354-JPG.scr</td>
<td>16-04-27 05:24</td>
<td>0ed2d08e694c9b6</td>
<td>41472</td>
<td>view *</td>
<td>75e0299ac</td>
<td>17 %</td>
<td>ANY</td>
<td>MALWARE</td>
</tr>
<tr>
<td>IMG0130661721-JPG.scr</td>
<td>16-04-27 04:37</td>
<td>90dd943b322a</td>
<td>41472</td>
<td>view *</td>
<td>9e2a1d65</td>
<td>17 %</td>
<td>ANY</td>
<td>MALWARE</td>
</tr>
<tr>
<td>IMG0522787902-JPG.scr</td>
<td>16-04-27 04:14</td>
<td>0086a4a4c7732</td>
<td>41472</td>
<td>view *</td>
<td>e7627d264</td>
<td>22 %</td>
<td>ANY</td>
<td>MALWARE</td>
</tr>
<tr>
<td>IMG9760768309-JPG.scr</td>
<td>16-04-27 03:56</td>
<td>fd5a604a4c4a0a6</td>
<td>9879</td>
<td>view *</td>
<td>0ed029a35</td>
<td>9 %</td>
<td>ANY</td>
<td>MALWARE</td>
</tr>
<tr>
<td>IMG0949848712-JPG.scr</td>
<td>16-04-27 03:56</td>
<td>c0e2e4f1c9a0290</td>
<td>3111</td>
<td>view *</td>
<td>da3c973428</td>
<td>9 %</td>
<td>ANY</td>
<td>MALWARE</td>
</tr>
<tr>
<td>IMG2051530909-JPG.scr</td>
<td>16-04-27 02:58</td>
<td>c5baddeb04f1a091</td>
<td>9803</td>
<td>view *</td>
<td>0ed029b35</td>
<td>9 %</td>
<td>ANY</td>
<td>GOODWARE</td>
</tr>
<tr>
<td>IMG8513652687-JPG.scr</td>
<td>16-04-27 02:53</td>
<td>397c0953e03b31</td>
<td>43008</td>
<td>view *</td>
<td>15e91f9d91</td>
<td>48 %</td>
<td>MALW</td>
<td>MALWARE</td>
</tr>
</tbody>
</table>
Examples

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
Examples

- Selecting a random set of 53 different dynamic analyses
- Anything interesting in there? Any clusters? Which behaviors are currently often used?
Examples

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image 1" /></td>
<td><img src="image2.png" alt="Image 2" /></td>
<td><img src="image3.png" alt="Image 3" /></td>
<td><img src="image4.png" alt="Image 4" /></td>
</tr>
<tr>
<td><img src="image5.png" alt="Image 5" /></td>
<td><img src="image6.png" alt="Image 6" /></td>
<td><img src="image7.png" alt="Image 7" /></td>
<td><img src="image8.png" alt="Image 8" /></td>
</tr>
<tr>
<td><img src="image9.png" alt="Image 9" /></td>
<td><img src="image10.png" alt="Image 10" /></td>
<td><img src="image11.png" alt="Image 11" /></td>
<td><img src="image12.png" alt="Image 12" /></td>
</tr>
<tr>
<td><img src="image13.png" alt="Image 13" /></td>
<td><img src="image14.png" alt="Image 14" /></td>
<td><img src="image15.png" alt="Image 15" /></td>
<td><img src="image16.png" alt="Image 16" /></td>
</tr>
</tbody>
</table>

---

INTERACTIVE GRAPHICAL EXPLORATION OF MALWARE BEHAVIOR

2016-05-20
Examples

- Sort by activity from little to much
Examples

- Spotting interesting samples

- Lots of modifications in filesystem and registry

- Very active samples
Examples

- Closer look at one interesting sample
Examples

- Looking for certain behaviors
Examples

- Malware vs. PUA vs. Goodware

<table>
<thead>
<tr>
<th>Malware (Bjlog)</th>
<th>PUA (Bundler)</th>
<th>PUA (Bundler)</th>
<th>Malware (Locky)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Graph" /></td>
<td><img src="image2.png" alt="Graph" /></td>
<td><img src="image3.png" alt="Graph" /></td>
<td><img src="image4.png" alt="Graph" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUA (Installcore)</th>
<th>Clean App</th>
<th>Clean App</th>
<th>PUA (?)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image5.png" alt="Graph" /></td>
<td><img src="image6.png" alt="Graph" /></td>
<td><img src="image7.png" alt="Graph" /></td>
<td><img src="image8.png" alt="Graph" /></td>
</tr>
</tbody>
</table>
SUMMARY

- Fun to work with, but real use cases still rare
  - Sample verification works well
  - Illustrating PR material
  - Doing first research/investigation steps on interesting samples

- More extensions planned/possible
  - Include network traffic
  - Include static attributes
  - Certain use cases, e.g. displaying only malicious features to find Goodware or PUA that behaves suspicious

- Visualizing output of other dynamic analysis systems possible
  - Raw output → Feature Vector → Visualization
Thank you for your attention!

@avtestorg (English) & @avtestde (German)

Follow us on facebook.com/avtestorg

Current test results at https://www.av-test.org