Jackdaw
Automatic, unsupervised, scalable extraction and semantic tagging of (interesting) behaviors

Mario Polino, Andrea Scorti, Federico Maggi, Stefano Zanero

Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria
NECSTLab

HITB Kuala Lumpur, October 15th, 2014
Malware Analysis at Scale
### Need to bridge a gap

#### Static Analysis

**Pros:**
- high code coverage,
- scalability

**Cons:**
- obfuscation,
- labor-intensive,
- skill-intensive

#### Dynamic Analysis

**Pros:**
- no semantic gap to be reversed with human skills

**Cons:**
- low code coverage, can be detected by adversary
Hybrid Analysis

Hybrid Analysis techniques

- Developed by several groups independently
- Try to leverage scalability and completeness of static techniques, combined with semantic insights from dynamic techniques
Example of Hybrid Analysis - Reanimator

Goal: identify “dormant” behavior in a program

**Dynamic analysis:** behavior identification
**Static analysis:** signatures of implementing code on (colored) CFG

Example of Hybrid Analysis - Beagle

Goal: analyzing evolution of malware families

**dynamic analysis:** identification of labeled/unlabeled behaviors as sequences of system calls

**static analysis:** mapping of behaviors on implementing code to check differences by behavior vs. overall

Pivot concept: behavior

behavior ≈ sequence of actions ≈ sequence of API calls (on Win binaries)

download_execute

recv
WriteFile
CreateProcess

Receives data from a network socket
Write a file
Create a new process
Defining Behaviors

Previous work: manual specification of behaviors

- Labor-intensive
- Only a small subset of behaviors can be defined manually
- Biased by previous experience of experts

Objective

Extract (interesting) behavior specifications in an automatic way from a large collection of (untagged) malware

Why?

Support the analyst by providing a list of important behaviors, with a rough explanation, to prioritize the analysis.
Our Approach: Jackdaw
System Architecture

Step 1: Data Collection
Step 2: Clustering
Step 3: Behavior Extraction
Step 4: Semantic Tagging
First step: Data gathering

1. **Dynamic Analysis:** data flow analysis
   - **API functions** name
   - **Parameters** of API functions

2. **Static Analysis:** fingerprint of code associated to data flow
   - sub-graphs of the CFG
   - can be hashed and matched
   - reasonably resilient to polymorphism
Control Flow Graph Fingerprinting

Static Analysis.
Identify **portions of CFG** likely to come from the same source code.

Properties:

- **Unique**
- Robust to *insertion / deletion*
- Robust to *modification*

Both CFGs share a subgraph of given order
First Step: Data Gathering - Data cleaning

- Static data cleaning: remove the fingerprints of benign binaries (e.g., Windows libraries and exes)
- Dynamic data cleaning (Windows API name \textbf{Normalization}):
  
  \begin{itemize}
    \item \textbf{Prefixes}
      
      \begin{itemize}
        \item WSA\texttt{Socket} → \texttt{socket}
      \end{itemize}
    \item \textbf{Suffixes}
      
      \begin{itemize}
        \item CreateEvent\texttt{A}, CreateEvent\texttt{W} → CreateEvent
      \end{itemize}
  \end{itemize}
Second Step: Clustering

Goal: Build clusters of similar data flows

Step 2: Clustering of data-flow information based on features of the CFG
Second step: Clustering

- Clustering of data flow in malware
- Feature: fingerprint
- simple one pass algorithm
- Threshold
- Similarity metrics

\[
J(A, B) = \frac{|A \cap B|}{|A \cup B|}
\]
Third Step: Behavior Extraction

Goal: find API functions that represent each cluster (behavior model).
Third step: Behavior Extraction - MFR Heuristic

MFR Heuristic (Most Frequent Rule)
Model = API functions that appear often. How often? We set a threshold.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>API</th>
<th>NtClearEvent</th>
<th>CreateEvent</th>
<th>NtSetEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Flow 1</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>Data Flow 2</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>Data Flow 3</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Data Flow 13</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>Data Flow 14</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>

Behavior Specification: $NtClearEvent \land CreateEvent$
Fourth Step: Semantic Tagger

Use **Crawler** to get knowledge and build significant **tag** for **behaviors**
For Each behavior:

Based on semantic, for example we are interested in posts in which there are tags like "windows", "winapi"
We are not interested in some tags like "python", "php" etc.
We weight the importance of posts according to those wl/bl tags
Fourth step: Semantic Tagger

- We look for **tags** searching **API function name**, each* element of powerset of **API function** in a model.
- Compute a **score** for each tag (based on **post relevance** and **frequency** of tags in post related to the search).
- Build a **ranking** of tags.
System Evaluation
### Dataset

The dataset:

- 1,272 samples from 17 malware families
Evaluation of behavior extraction: approach

- **Unsupervised learning** (no ground truth)
- We built a pseudo **ground truth**, asking experts to **manually** describe a model.
- We compare these **manually defined behavior models** with **behavior models automatically identified** by Jackdaw.
Evaluation of behavior extraction: results

Correctness:

Ground-truth Behavior: 
firewall_settings

ShellExecute (advfirewall firewall add rule name: 1)

Automatic Behavior

ShellExecute (advfirewall firewall add rule name: 1)

RegOpenKey (hku\s-1-5-21-842925246-1425521274-308236825-500\software\microsoft\internet explorer\main)

GetProcAddress

Completeness:

34 over 45 behavior models manually created by experts have been identified also by Jackdaw.
Empirical evaluation.
Example of behavior HTTP connection:

```
Malware Analysis at Scale
Jackdaw
Evaluation
Conclusions

InternetOpen (szAgent: atlsys13.exe: 1)
InternetOpenUrl,MapMemRegion,connect,recv,send
(szUrl: http://robertokunihira.sites.uol.com.br/nordeste.jpg, ForeignPort: ['80']: 1, LocalAddress: (['tcp'], public, ['1029']): 1, ForeignIP: public: 1)
'Banload_09af6de40ab414f41ba48b447345e75d'
```

<table>
<thead>
<tr>
<th>Position</th>
<th>Tag (hint)</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>http</td>
<td>18</td>
</tr>
<tr>
<td>2</td>
<td>proxy</td>
<td>13.5</td>
</tr>
<tr>
<td>3</td>
<td>ftp</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>file</td>
<td>6.8</td>
</tr>
<tr>
<td>5</td>
<td>mfc</td>
<td>6.8</td>
</tr>
<tr>
<td>6</td>
<td>post</td>
<td>6.2</td>
</tr>
<tr>
<td>7</td>
<td>internet</td>
<td>5.6</td>
</tr>
<tr>
<td>8</td>
<td>upload</td>
<td>5.6</td>
</tr>
<tr>
<td>9</td>
<td>file-download</td>
<td>4.5</td>
</tr>
<tr>
<td>10</td>
<td>arrays</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>download</td>
<td>3.9</td>
</tr>
<tr>
<td>12</td>
<td>rich-internet-application</td>
<td>3.3</td>
</tr>
<tr>
<td>13</td>
<td>networking</td>
<td>3.3</td>
</tr>
<tr>
<td>14</td>
<td>httprequest</td>
<td>2.8</td>
</tr>
<tr>
<td>15</td>
<td>httpwebrequest</td>
<td>2.2</td>
</tr>
<tr>
<td>16</td>
<td>internet-explorer</td>
<td>1.7</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
</tbody>
</table>
Recognizing behaviors in unknown malware
Limitations and Future works

Limitations:
- needs buckets of variants of each malware family
- analyzed malware needs to be unpacked

Future Works:
- Introduce sequence/time concept in behavior models
- NLP to improve semantic tagging
Third step: Behavior Extraction - PLR Heuristic

PLR Heuristic (Propositional Logic Rule)

Let $T$ be a set of elements; given a set of elements $L \subseteq \mathcal{P}(T)$, the solution is all sets $Q \subseteq \mathcal{P}(T)$ such that:

- $\forall l \in L, \forall q_i, q_j \in Q$ with $q_i \neq q_j$, if $q_i \subset l$ then $q_j \cap l = \emptyset$
- $\forall l \in L, \exists! q \in Q$, $q \subset l$

<table>
<thead>
<tr>
<th>Cluster</th>
<th>API</th>
<th>...Atom API\textsubscript{1} ...</th>
<th>...Atom API\textsubscript{2}...</th>
<th>RegCloseKey</th>
<th>...NtKey API\textsubscript{1}...</th>
<th>...NtKey API\textsubscript{2}...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taint 1</td>
<td></td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Taint 5</td>
<td></td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>Taint 6</td>
<td></td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Taint 8</td>
<td></td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Behavior Specification:

$$(\text{AtomAPI}_1 \land \text{AtomAPI}_2) \oplus (\text{RegCloseKey} \land \text{NtKeyAPI}_1 \land \text{NtKeyAPI}_2)$$
Conclusions

Jackdaw:

- **Automatically** extracts **behavior models** of widespread behaviors, exploiting both dynamic and static analysis.
- Assigns a set of **semantic tags** to each model to help analyst
- **Maps** behavior model on **binary code**, building a catalog of implementations of same behavior which can be used to attribute to family/group
Thanks!

- Thanks to you for your attention, please get in touch: stefano.zanero@polimi.it or @raistolo
- Thanks to my student Mario who is the main author and coder of JackDaw, and to my colleague Federico for 10 years of work together.
- Thanks to my brother Dhillon and the awesome HITB crew for one of the most enjoyable events in the scene worldwide. I’ve spoken in Malaysia in 2006, then I was back in 2012 for the ten years celebration, in 2013 and this year, and I look forward to whatever you guys do next. Kudos!