App-Agnostic Post-Execution Semantic Analysis of Android In-Memory Forensics Artifacts

Aisha Ali-Gombe Ph.D.*, Alexandra Tambaoan*, Angela Gurfolino*
Golden G. Richard III Ph.D.†
Mobile Forensics – The need
Memory Forensics

Kernel-land Memory Forensics

- Clipboard data
- Volatile registry branches
- Network connections
- Running processes
- Open files
- Encryption keys
- Private browsing data
- Kernel structures
- Application structures

Userland Memory Forensics (UMF)

- Apps
- Android Framework
- Native Libraries
- Android Runtime
- HAL/HIDL
- Linux Kernel

I stole the squirrels 😳
Prior Work on UMF

App-Specific Techniques
(Anglano et. al, 2017)...

Domain-Specific Techniques
(Saltaformaggio et. al, 2015)...

Limited to only specialized scenario
Prior Work on UMF (2)

Generalized Techniques

No context and no relationship between recovered data
Research Objective

• Generalized and app-agnostic UMF technique called OAGen
  • Automated object relationship (OAG) generation
  • Robust semantic analysis modules (Paths, Context and Scope)

GOAL:
Our technique can be applied to any kind of forensics investigation and can be use for Android program analysis
OAGen – Design

\[ G = (N, E) \]
where \( n \in N \) and \( n_1 \rightarrow n_2 \in E \)

```java
MessageData msgData = new MessageData()
msgData.setURI (new URI())
```
OAG Generation

Algorithm 1: Illustrating the Initial Phase of Object Allocation Graph (OAG) Generation

1. parameters: String path
2. nodeList = HeadDump(path)
3. G = Graph(strict=False, directed=True)
4. for node in nodeList do
5.     G.addNode(node.id, node.label, node.data)
6. end
7. for node in G.itemnodes() do
8.     recToObject(G, node)
9. end

Algorithm 2: Recursively Searching and Adding Nodes and Edges to the Graph

1. parameters: Node node, Address heapBeg
2. decoded = decodeObject(node, heapBeg)
3. if decoded is String then
4.     node.label = "String Class"
5.     node.data = decoded
6. else if decoded is Array then
7.     if decoded is Primitive then
8.         node.label = "Primitive Array Class"
9.         node.data = decoded
10. else if decoded is Object then
11.     node.label = "Object Array Class"
12.     arr = getAllRef(decoded)
13.     for x in arr do
14.         G.addEdge(node, G.addNode(x, node.id=x))
15.     end
16.     node.data = decoded
17. else if decoded is Object then
18.     node.label = "Object Class"
19.     ObjectFields = getAllRef(decoded)
20.     for x in ObjectFields do
21.         if x is ref then
22.             G.addEdge(node, G.addNode(x, node.id=x))
23.         end
24.     node.data = decoded
Semantic Analysis

Path Exploration

*Given directed graph* $G = (N, E)$, and two nodes $s$ and $t$, *k-Edge-disjoint Paths* are paths with no common edge.

---

**Context Determination**

**Algorithm 3:** Algorithm for Context Determination

1. **parameters:** Node:target, Graph:G
2. nodePred=[target]
3. for $n$ in nodePred do
   4.   for $i$ in G.iterpred($n$) do
   5.     nodePred.append($n$)
   6.   end
7. end
8. edgeList=G.edges(nodePred)
9. H = G.subgraph(edgeList)
10. return H

---

**Object Scope**

**Algorithm 4:** Algorithm for Object Scope Search

1. **parameters:** Node:target, Graph:G
2. nodeList = G.getJThreads() || G.getGCRoot() || G.getComponents()
3. for node in nodeList do
   4.   nodeSuccessors=[node]
7.   for $n$ in nodeSuccessors do
   8.     for $i$ in G.itersucc($n$) do
   9.       nodeSuccessors.append($n$)
10.   end
11. end
12. H = G.subgraph(nodeSuccessors)
13. if H.has_node(target) then
    14.   Flow(target, node, H)
15. end

Search Utility – Strings and Objects
OAGen – Implementation & Evaluation

• Python - ~ 3000 loc
• Five modules (OAG Gen, Search utility, path exploration, context and scope determination)
• Prototype - git@github.com:apphackuno/OAGen.git
• Object recovery is based on the default Region-Space memory allocation algorithm
  • Utilizes the base Heapdump of DroidScraper (Ali-Gombe et al. 2019)
• Evaluation - MacBook Pro with a 2.6GHz Intel processor and 16GB RAM
  • Performance of the graph generation algorithm
  • Robustness of the recursive search
  • Application of the semantic analysis algorithms
# Evaluation - Processing Time

<table>
<thead>
<tr>
<th>Applications</th>
<th>Image Size (in Mb)</th>
<th>Number of Nodes</th>
<th>Number of Edges</th>
<th>Processing Time (in Minutes)</th>
<th>Throughput (in Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.whatsapp</td>
<td>1338</td>
<td>318632</td>
<td>476586</td>
<td>1025.5</td>
<td>0.03</td>
</tr>
<tr>
<td>com.facebook.katana</td>
<td>1733</td>
<td>282120</td>
<td>411522</td>
<td>847</td>
<td>0.04</td>
</tr>
<tr>
<td>com.callpod.android_apps.keeper</td>
<td>1237</td>
<td>227776</td>
<td>330240</td>
<td>899.5</td>
<td>0.03</td>
</tr>
<tr>
<td>org.thoughtcrime.securesms</td>
<td>1249</td>
<td>150824</td>
<td>229308</td>
<td>381.5</td>
<td>0.06</td>
</tr>
<tr>
<td>com.peakel.gothicwallpapers</td>
<td>1272</td>
<td>128256</td>
<td>183276</td>
<td>206.5</td>
<td>0.11</td>
</tr>
<tr>
<td>com.klinker.android.evolve_sms</td>
<td>1208</td>
<td>108248</td>
<td>154014</td>
<td>280</td>
<td>0.08</td>
</tr>
<tr>
<td>com.android.messaging</td>
<td>1257</td>
<td>106304</td>
<td>156318</td>
<td>311.5</td>
<td>0.07</td>
</tr>
<tr>
<td>com.dd.monkey</td>
<td>1169</td>
<td>80512</td>
<td>116472</td>
<td>133</td>
<td>0.15</td>
</tr>
<tr>
<td>com.android.tencent.zdevs.bah</td>
<td>1192</td>
<td>59232</td>
<td>81462</td>
<td>157.5</td>
<td>0.13</td>
</tr>
<tr>
<td>com.easyhin.usereasyhin</td>
<td>1192</td>
<td>57504</td>
<td>85212</td>
<td>63</td>
<td>0.32</td>
</tr>
<tr>
<td>com.huosDebei</td>
<td>1211</td>
<td>39320</td>
<td>50964</td>
<td>63</td>
<td>0.33</td>
</tr>
<tr>
<td>org.chromium.webview_shell</td>
<td>1289</td>
<td>38560</td>
<td>50070</td>
<td>45.5</td>
<td>0.48</td>
</tr>
<tr>
<td>com.yandex226.yandex</td>
<td>1201</td>
<td>38528</td>
<td>49590</td>
<td>56</td>
<td>0.36</td>
</tr>
<tr>
<td>com.yuxinglin.xxd</td>
<td>1195</td>
<td>35832</td>
<td>45294</td>
<td>49</td>
<td>0.41</td>
</tr>
</tbody>
</table>

**Average throughput = 0.19Mbps**
Evaluation – Macro Benchmark

Objective: Robustness of OAGen in searching and finding object references

<table>
<thead>
<tr>
<th></th>
<th>com.whatsapp</th>
<th>com.android.messaging</th>
<th>com.yandex226.yandex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heap’s objects_allocated_field</td>
<td>301,742</td>
<td>92,046</td>
<td>11,172</td>
</tr>
<tr>
<td>Droidscraper</td>
<td>301,467</td>
<td>90,053</td>
<td>10,328</td>
</tr>
<tr>
<td>OAGen</td>
<td>318,632</td>
<td>106,304</td>
<td>35,528</td>
</tr>
</tbody>
</table>

*OAGen recovers objects in heap, stack, initialized data and code segments*
Evaluation – Semantic Analysis 1: Cybercrime Investigation

• End-to-end encryption
• Complete destruction of evidence
Evaluation – Semantic Analysis 2: Data Exfiltration

Object Search Utility

Object Scope within RefQueueWorker thread

Memfetch

OAG Generation

>128K nodes

Context Determination

Android Object Allocation Graph
org.apache.http.impl.client.DefaultHttpClient 0x12d57830
org.apache.http.impl.client.DefaultHttpClient 0x12d5f490
org.apache.http.impl.client.DefaultHttpClient 0x12d42ea8
org.apache.http.impl.client.DefaultHttpClient 0x132c0ba0

java.lang.ref.FinalizerReference 0x12d40118
android.os.BinderProxy 0x12d40170
java.lang.ref.WeakReference 0x12d40140
java.lang.ref.FinalizerReference 0x12d40770
com.android.internal.telephony.ITelephony$Stub$Proxy 0x12d40158
String - 0x18010155 0x12d40168
java.lang.StringBuilder 0x12d40188

Context Determination
Challenges and Future Work

• Require considerable amount of time to weed out noise

• Tentative execution path reconstruction
  • mapping OAG to in-memory code

• Need more testing to explore the effects:
  • garbage collection/process state on recovered data
  • program obfuscation
Key Takeaway

• OAGen - a tool for post-execution semantic analysis of Android applications
  • Object Allocation Graph Generation
  • Algorithms
    • Path exploration
    • Context
    • Object Scope Determination
Acknowledgement

This work is supported by the U.S. National Science Foundation under the awards -

NSF-CRII-SaTC-1850054
THANK YOU! QUESTIONS!

git@github.com:apphackuno/OAGen.git
aaligombe@towson.edu