Science, Sharing, and Repeatability in Memory Forensics

Brendan Dolan-Gavitt
Columbia University
whoami

- @moyix
- Did some early work in memory forensics:
  - VAD, registry in memory, GDI, VMI
- Other software: pdbparse, PANDA
- Currently postdoc researcher at Columbia
Memory Forensics as a Scientific Field

- Still very young
- First DFRWS memory forensics challenge less than a decade ago!
- The gap between research and practice is not very large
Scientific Method

• In developing memory forensic tools:
  • Form some hypothesis about software artifacts
  • Investigate target – run experiments, disassemble, etc. to confirm/disprove
  • New understanding becomes crystallized into tools, plugins, etc. that practitioners use
Reproducibility

“The first principle is that you must not fool yourself—and you are the easiest person to fool.”
Reproducibility

• Correctness is critical

• Forensics moves fast, however
  • ~1.5 years between OS versions
  • Just describing results (without code) is very slow

• Validation needs to move quickly too
Sharing

- Note that this is the *Open* Memory Forensics Workshop
- Making code available is critical!
  - Redeveloping from scratch takes too long
  - Direct examination of code is better
- Sharing *data* is also necessary (i.e., memory images for testing)
Reproducibility in Memory Forensics

- Standard reference images
- NIST CFReDS, DFRWS challenge images
- Tool testing (NIST)
- Work on validating acquisition (Vömel & Stütten, 2013)
A Missing Piece

• Many investigations involve *dynamic* analyses of executing programs

• Particularly malware

• What does it mean to reproduce a malware analysis?
Challenges

• Dynamic analyses depend on a runtime environment
  • Network servers may go down
  • Behavior dependent on software & library versions
  • May trigger on certain dates/times
Record / Replay

• We want to instead share a specific execution of a program

• Observation: if we record all the nondeterministic inputs to the system, we can then replay the exact execution later

• Technique has been around ~20 years, used mainly for debugging (i.e. reverse execution)
Record / Replay

CPU

== Friday?

== 0x45?

>= 0x80?

Outside World
Record / Replay

CPU

==
Friday?

==
0x45?

>=
0x80?

Get Current Date

Outside World

Fri May 23 11:33:27
Record / Replay

CPU

==
== Friday?
>=
0x45?

Outside World

Fri May 23 11:33:27
Record / Replay

CPU

Get Current Date

Outside World

Fri May 23 11:33:27

Recv Packet

== Friday?

0x0000:  4500 002c 0000 4000
0x0008:  4006 6b48 127e 0021
0x0010:  5dae 5f37 01bb bed4
0x0018:  fccd 820f d690 0847
0x0020:  6012 3908 cfa2 0000
0x0028:  0204 05b4

== 0x45?

>= 0x80?
Get Current Date
Fri May 23 11:33:27

Recv Packet
0x0000: 4500 002c 0000 4000
0x0008: 4006 6b48 127e 0021
0x0010: 5dae 5f37 01bb bed4
0x0018: fccd 820f d690 0847
0x0020: 6012 3908 cfa2 0000
0x0028: 0204 05b4

Record / Replay

CPU

= Friday?

= 0x45?

>= 0x80?
Record / Replay

CPU

Get Current Date

Outside World

Fri May 23 11:33:27

Recv Packet

0x0000: 4500 002c 0000 4000
0x0008: 4006 6b48 127e 0021
0x0010: 5dae 5f37 01bb bed4
0x0018: fccd 820f d690 0847
0x0020: 6012 3908 cfa2 0000
0x0028: 0204 05b4

== Friday?

== 0x45?

>= 0x80?
Record / Replay

Time

- rdtsc
- Interrupt
- DMA
Reproducible Dynamic Analysis with PANDA

- PANDA - Platform for Architecture Neutral Dynamic Analysis
- Supports *shareable* recordings of whole-system execution
- Write *plugins* to analyze replays as they execute
• Record / replay critical:
  • Heavy analyses don’t disrupt execution
  • Analyses don’t have to worry about memory layout changing between runs
This site stores recordings made with the PANDA dynamic analysis platform. To find out more about PANDA's record/replay features, you can peruse the documentation. After downloading, the .rr files can be extracted using scripts/rrunpack.py in the PANDA distribution.

<table>
<thead>
<tr>
<th>Name</th>
<th>Summary</th>
<th>Download</th>
<th>Size</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>cve-2012-4792-exploit</td>
<td>Exploitation of cve-2012-4792</td>
<td>rrlogs/cve-2012-4792-exploit.rr</td>
<td>130.1 MB</td>
<td>968.8 million</td>
</tr>
<tr>
<td>cve-2012-4792-crash</td>
<td>Crashing instance of cve-2012-4792</td>
<td>rrlogs/cve-2012-4792-crash.rr</td>
<td>129.9 MB</td>
<td>608.8 million</td>
</tr>
<tr>
<td>cve-2011-1255-exploit</td>
<td>Exploitation of cve-2011-1255</td>
<td>rrlogs/cve-2011-1255-exploit.rr</td>
<td>126.6 MB</td>
<td>2.1 billion</td>
</tr>
<tr>
<td>cve-2014-1776-crash</td>
<td>Crashing instance of cve-2014-1776</td>
<td>rrlogs/cve-2014-1776-crash.rr</td>
<td>155.9 MB</td>
<td>1.2 billion</td>
</tr>
<tr>
<td>dia2dump</td>
<td>Parsing a PDB with dia2dump</td>
<td>rrlogs/dia2dump.rr</td>
<td>190.8 MB</td>
<td>5.4 billion</td>
</tr>
<tr>
<td>line2</td>
<td>Sending an IM using LINE for Android</td>
<td>rrlogs/line2.rr</td>
<td>64.6 MB</td>
<td>10.4 billion</td>
</tr>
<tr>
<td>win7_64bit_install_STOP_D1</td>
<td>Failure during boot to install CD of Win7 64bit. DRIVER_IRQL_NOT_LESS_OR_EQUAL</td>
<td>rrlogs/win7_64_install_fail.rr</td>
<td>203.3 MB</td>
<td>5.3 billion</td>
</tr>
<tr>
<td>carberp2</td>
<td>Running custom RU_Az build of the Carberp malware</td>
<td>rrlogs/carberp2.rr</td>
<td>91.9 MB</td>
<td>2.9 billion</td>
</tr>
</tbody>
</table>
### Log Size

<table>
<thead>
<tr>
<th>Replay</th>
<th>Instructions</th>
<th>Log Size</th>
<th>Instr/Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>freebsdboot</td>
<td>9.3 billion</td>
<td>533 MB</td>
<td>17</td>
</tr>
<tr>
<td>spotify</td>
<td>12 billion</td>
<td>229 MB</td>
<td>52</td>
</tr>
<tr>
<td>haikuurl</td>
<td>8.6 billion</td>
<td>119 MB</td>
<td>72</td>
</tr>
<tr>
<td>carberp1</td>
<td>9.1 billion</td>
<td>43 MB</td>
<td>212</td>
</tr>
<tr>
<td>win7iessl</td>
<td>8.6 billion</td>
<td>9.4 MB</td>
<td>915</td>
</tr>
<tr>
<td>Starcraft</td>
<td>60 million</td>
<td>1.8 MB</td>
<td>33</td>
</tr>
</tbody>
</table>
Other PANDA Features

- Android emulation
- Lifting binary code to LLVM
- Taint analysis
- System call tracing
Plugin Architecture

- Extend PANDA by writing plugins (C/C++)
- Implement functions that take action at various instrumentation points
- Can also instrument generated code in LLVM mode
- Plugin-plugin interaction: compose simple tools for complex functionality
Android Emulation

- Supports Android 2.x – 4.2
- Can make phone calls, send SMS, run native apps
- Record/replay
- Introspection into Android apps (Dalvik-level) for Android 2.3 (from DroidScope)
- System-level introspection supported on all Android versions
Memory Forensics on Replays

- In some ways, best of both worlds between debugging and memory image analysis
- All memory accessible throughout entire lifetime of
- Can pause, dump memory, run Volatility, etc.
- But can still be triggered by things happening in execution
Conclusions

• Reproducibility is critical to achieving valid forensic results

• For some areas we have decent solutions – code sharing, testing, standard images

• For ephemera such as software execution, we propose record and replay, and a system, PANDA
Credits

• PANDA devs
  • Tim Leek (MIT Lincoln Lab)
  • Patrick Hulin (MIT Lincoln Lab)
  • Josh Hodosh (MIT Lincoln Lab)
  • Ryan Whelan (MIT Lincoln Lab)
  • Sam Coe (Northeastern University)
  • Andy Davis (MIT Lincoln Lab)
Contact

• Get in touch! @moyix on Twitter
  brendan@cs.columbia.edu

• Join the mailing list: panda-users@mit.edu

• IRC Channel: #panda-re on Freenode

• Contribute code:
  https://github.com/moyix/panda