Research in Systems Security using Binary Code and Memory Dump Analysis

Mark Vella

CPS 5116
Hardening computer systems to make them resilient to threats.

- Prevention | Detection | Response

Cyber security an overwhelming concern

- Insider security mainly involves abuse of privilege, typically detected through log analysis

Binary code and memory dumps enable threat investigation and form the basis for mitigation.

- Especially when net/host logs are plenty, tampered with or missing. *Or simply, it is all we have got to work with!*
Dynamic Binary Instrumentation

Automating binary code analysis with full access to the program state possibly at an individual instruction granularity

Taint Analysis

Static/dynamic data flow analysis technique focusing just on flows of interest e.g. untrusted sources, sys calls, pointers etc.

<table>
<thead>
<tr>
<th>Line #</th>
<th>Statement</th>
<th>$\Delta$</th>
<th>$\tau_\Delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$x := 2*\text{get_input}(\cdot)$</td>
<td>${}$</td>
<td>${}$</td>
</tr>
<tr>
<td></td>
<td>$x \rightarrow 40$</td>
<td>${x \rightarrow 40}$</td>
<td>${x \rightarrow T}$</td>
</tr>
<tr>
<td>2</td>
<td>$y := 5 + x$</td>
<td>${x \rightarrow 40, y \rightarrow 45}$</td>
<td>${x \rightarrow T, y \rightarrow T}$</td>
</tr>
<tr>
<td>3</td>
<td>$\text{goto } y$</td>
<td>${x \rightarrow 40, y \rightarrow 45}$</td>
<td>${x \rightarrow T, y \rightarrow T}$</td>
</tr>
</tbody>
</table>

Schwartz, Edward J., Thanassis Avgerinos, and David Brumley. "All you ever wanted to know about dynamic taint analysis and forward symbolic execution (but might have been afraid to ask)." In Security and Privacy (SP), 2010 IEEE Symposium on, pp. 317-331. IEEE, 2010.
Symbolic Execution

Single execution with all classes of values that individual data objects may take during multiple concrete executions e.g. solving the possible targets of indirect jumps, test case generation to feed dynamic analysis

<table>
<thead>
<tr>
<th>Statement</th>
<th>Δ</th>
<th>Π</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>{}</td>
<td>true</td>
</tr>
<tr>
<td>$x := 2 \times \text{get_input()}$</td>
<td>${x \rightarrow 2 \times s}$</td>
<td>true</td>
</tr>
<tr>
<td>if $x - 5 == 14$ goto 3 else goto 4</td>
<td>${x \rightarrow 2 \times s}$</td>
<td>$[(2 \times s) - 5 == 14]$</td>
</tr>
<tr>
<td>if $x - 5 == 14$ goto 3 else goto 4</td>
<td>${x \rightarrow 2 \times s}$</td>
<td>$-[(2 \times s) - 5 == 14]$</td>
</tr>
</tbody>
</table>
Threat Analysis

- Threats to security-critical resources in terms of CIA
  - Malware, access/control protocol tamper/bypass, implementation vulnerability exploitation, social engineering

- Binary code analysis to
  - *Detect malware*: Stand-alone/injected binaries, native/scripts
  - *Vulnerability analysis*: is the other side of the same coin

and to support *Memory dump analysis*

- *Binary code analysis link*: Analyze code dumped from raw memory, and Reversing data structures in raw memory

- Automated malware analysis e.g. Cuckoo sandbox

- For incident response – still in its infancy e.g. Volatility, F-Response; and still mainly used with malware hunting in mind
Intrusion detection

- DBI – exposing obfuscated web attack payloads
  - Jennifer Bellizzi and Mark Vella, WeXpose: Towards on-Line Dynamic Analysis of Web Attack Payloads using Just-In-Time Binary Modification. SECRYPT 2015: 5-15

Vulnerability analysis (VA)/automatic exploit generation (AEG)

- DBI/TA – exposing browser UAF bugs

- DBI/TA/SE – extending VA with AEG
Memory artifact identification

- DBI/TA – pointer identification

Memory forensics

- DBI/TA/SE – rootkit infection detection from memory dumps

Pattern matching – scan for artifacts with 'obvious' binary-level patterns in raw memory and report on findings (multiple inaccuracies in paper portray current lack of understanding in academia)

Mobile devices are the new desktop - Just more ubiquitous!

All previous research areas also apply, but with *special* challenges:

- Resource constraints, less control, lesser understood platforms, complex deployment ecosystems

+ Joint research with the dependable computing stream

*Additional* primitive – Runtime Verification

Correctness property checking performed on execution traces

*Sys Security main area – focus on systems-level instrumentation (direct link with the security primitives)*

---

(Appstore) App scanning

- Malware detection - Android permissions misuse (uses model checking but lays the ground for a runtime solution)

Vulnerability analysis by leveraging Intents and DBI

BYOD – Bring Your Own Device

File system virtualization (through OS source modification) to support multiple security profiles (as opposed to the TA approach of the previous version of the same solution)


Yet, in-memory patching offers better portability as in this capability leaks mitigation approach. BUT, requires full understanding of Android internals at the raw memory level


A roadmap to fully exploit DBI is in-place waiting for volunteers!

So...

mark.vella@um.edu.mt