RAIN: Refinable Attack Investigation with On-demand Inter-process Information Flow Tracking

Yang Ji, Sangho Lee, Evan Downing, Weiren Wang, Mattia Fazzini, Taesoo Kim, Alessandro Orso, and Wenke Lee

ACM CCS 2017
Oct 31, 2017
More and more data breaches
More and more data breaches

DATA BREACHES
(SOURCE: BREACH LEVEL INDEX BY GEMALTO)

Number of data breaches
Number of breached records (mil)

2013-H1: 658
2013-H2: 513
2014-H1: 1594
2014-H2: 819
2015-H1: 2459
2015-H2: 924
2016-H1: 1029
2016-H2: 853
2017-H1: 1155
2017-H2: 918
2018-H1: 915
2018-H2: 721
2019-H1: 1901
Is attack investigation accurate?
Is attack investigation accurate?
Is attack investigation accurate?
Is attack investigation accurate?
Is attack investigation accurate?
Is attack investigation accurate?

“Hmm, I only want C!”

Compromised FTP addon
Is attack investigation accurate?

“Hmm, I only want C!”

Compromised FTP addon
Is attack investigation accurate?

"Hmm, I only want C!"

Compromised FTP addon

A, B, or C?
Is attack investigation accurate?

Dependency confusion!

“A, B, or C?”

Compromised FTP addon

“Hmm, I only want C!”
“Let me change the offer price.”
“Let me change the offer price.”
"Let me change the offer price."

Compromised FTP addon
"Let me change the offer price."

Compromised FTP addon

Is this file affected?
“Let me change the offer price.”

Is this file affected?

Dependency confusion!

Compromised FTP addon
Related work

- Accuracy
- Runtime Efficiency
- Analysis Efficiency
Related work

• System-call-based
  • DTrace, Protracer, LSM, Hi-Fi
Related work

• System-call-based
  • DTrace, Protracer, LSM, Hi-Fi
Related work

- **System-call-based**
  - DTrace, Protracer, LSM, Hi-Fi

- **Dynamic Information Flow Tracking (DIFT)**
  - Panorama, Dtracker
Related work

- **System-call-based**
  - DTrace, Protracer, LSM, Hi-Fi

- **Dynamic Information Flow Tracking (DIFT)**
  - Panorama, Dtracker
Related work

• System-call-based
  • DTrace, Protracer, LSM, Hi-Fi

• Dynamic Information Flow Tracking (DIFT)
  • Panorama, Dtracker

• DIFT + Record replay
  • Arnold
Related work

• System-call-based
  • DTrace, Protracer, LSM, Hi-Fi

• Dynamic Information Flow Tracking (DIFT)
  • Panorama, Dtracker

• DIFT + Record replay
  • Arnold
RAIN
RAIN

• We use
  • Record replay
  • Graph-based pruning
  • Selective DIFT
RAIN

• We use
  • Record replay
  • Graph-based pruning
  • Selective DIFT

• We achieve
  • High accuracy
  • Runtime efficiency
  • Highly improved analysis efficiency
Threat model

• Trusts the OS
  • RAIN tracks user-level attacks.

• Tracks explicit channels
  • Side or covert channel is out of scope.

• Records all attacks from their inception
  • Hardware trojans or OS backdoor is out of scope.
Architecture

Target host

Analysis host
Architecture

Target host

RAIN
Customized
Kernel

Analysis host
Architecture

Target host
- Customized libc
- RAIN
- Customized Kernel

Analysis host
Architecture

Target host
- Customized libc
- RAIN Customized Kernel

Analysis host

Logs
Architecture

Target host
- Customized libc
- RAIN
- Customized Kernel

Analysis host
- Provenance graph builder

Logs

Coarse-level graph
Architecture

Target host
- Customized libc
- RAIN Customized Kernel

Analysis host
- Provenance graph builder
- Triggering, reachability analysis

Logs

Coarse-level graph

Pruned sub-graph

Prune
Architecture

Target host

Customized libc
RAIN Customized Kernel

Analysis host

Provenance graph builder
Triggering, reachability analysis
Replay and selective DIFT

Logs

Coarse-level graph

Prune

Pruned sub-graph

Refine

Refined sub-graph
OS-level record replay

1. Records **external inputs**
2. Captures the **thread switching** from the pthread interface, not the produced **internal data**
3. Records **system-wide** executions
OS-level record replay

1. Records **external inputs**
2. Captures the **thread switching** from the pthread interface, not the produced **internal data**
3. Records **system-wide** executions
OS-level record replay

1. Records **external inputs**
2. Captures the **thread switching** from the pthread interface, not the produced **internal data**
3. Records **system-wide** executions
OS-level record replay

1. Records **external inputs**
2. Captures the **thread switching** from the pthread interface, not the produced **internal data**
3. Records **system-wide** executions
OS-level record replay

1. Records external inputs
2. Captures the thread switching from the pthread interface, not the produced internal data
3. Records system-wide executions
OS-level record replay

1. Records external inputs
2. Captures the thread switching from the pthread interface, not the produced internal data
3. Records system-wide executions
OS-level record replay

1. Records **external inputs**
2. Captures the **thread switching** from the pthread interface, not the produced **internal data**
3. Records **system-wide** executions
1. Records **external inputs**
2. Captures the thread **switching** from the pthread interface, not the produced **internal data**
3. Records **system-wide** executions
OS-level record replay

1. Records external inputs
2. Captures the thread switching from the pthread interface, not the produced internal data
3. Records system-wide executions
Coarse-level logging and graph building

- Keeps logging system-call events
- Constructs a graph to represent:
  - the processes, files, and sockets as nodes
  - the events as causality edges

A: Attacker site
B: /docs/report.doc
C: /tmp/errors.zip
P1: /usr/bin/firefox
Coarse-level logging and graph building

• Keeps logging system-call events
• Constructs a graph to represent:
  • the processes, files, and sockets as nodes
  • the events as causality edges

A: Attacker site
B: /docs/report.doc
C: /tmp/errors.zip
P1: /usr/bin/firefox
Coarse-level logging and graph building

• Keeps logging system-call events
• Constructs a graph to represent:
  • the processes, files, and sockets as nodes
  • the events as causality edges

A: Attacker site
B: /docs/report.doc
C: /tmp/errors.zip
P1: /usr/bin/firefox
Coarse-level logging and graph building

- Keeps logging system-call events
- Constructs a graph to represent:
  - the processes, files, and sockets as nodes
  - the events as causality edges

A: Attacker site
B: /docs/report.doc
C: /tmp/errors.zip
P1: /usr/bin/firefox
Coarse-level logging and graph building

• Keeps logging system-call events
• Constructs a graph to represent:
  • the processes, files, and sockets as nodes
  • the events as causality edges

A: Attacker site
B: /docs/report.doc
C: /tmp/errors.zip
P1: /usr/bin/firefox
• Does every recorded execution need replay and DIFT?
• Does every recorded execution need replay and DIFT? **No!**
Pruning

• *Does every recorded execution need replay and DIFT? No!*
• Prunes the data in the graph based on trigger analysis results
  • Upstream
  • Downstream
  • Point-to-point
  • Interference
Upstream

A: Attacker site
B: /docs/report.doc
C: /tmp/errors.zip
D: /docs/ctct1.csv
E: /docs/ctct2.pdf
F: /docs/loss.csv

P1: /usr/bin/firefox
P2: /usr/bin/TextEditor
P3: /bin/gzip
A: Attacker site
B: /docs/report.doc
C: /tmp/errors.zip
D: /docs/ctct1.csv
E: /docs/ctct2.pdf
F: /docs/loss.csv

P1: /usr/bin/firefox
P2: /usr/bin/TextEditor
P3: /bin/gzip
A: Tampered file /docs/ctct.csv
B: Seasonal report docs/s1.csv
C: Seasonal report docs/s2.csv
D: Budget report docs/bgt.csv
E: Half-year report docs/h2.pdf

P1: Spreadsheet editor
P2: Auto-budget program
P3: Auto-report program
Downstream

A: Tampered file /docs/ctct.csv
B: Seasonal report docs/s1.csv
C: Seasonal report docs/s2.csv
D: Budget report docs/bgt.csv
E: Half-year report docs/h2.pdf

P1: Spreadsheet editor
P2: Auto-budget program
P3: Auto-report program
A: Tampered file /docs/ctct.csv
B: Seasonal report docs/s1.csv
C: Seasonal report docs/s2.csv
D: Budget report docs/bgt.csv
E: Half-year report docs/h2.pdf

P1: Spreadsheet editor
P2: Auto-budget program
P3: Auto-report program
Point-to-point

A: Tampered file /docs/ctct.csv
B: Seasonal report docs/s1.csv
C: Seasonal report docs/s2.csv
D: Budget report docs/bgt.csv
E: Half-year report docs/h2.pdf
F: Document archive server

P1: Spreadsheet editor
P2: Auto-budget program
P3: Auto-report program
P4: Firefox browser
A: Tampered file /docs/ctct.csv
B: Seasonal report docs/s1.csv
C: Seasonal report docs/s2.csv
D: Budget report docs/bgt.csv
E: Half-year report docs/h2.pdf
F: Document archive server

P1: Spreadsheet editor
P2: Auto-budget program
P3: Auto-report program
P4: Firefox browser
A: Tampered file /docs/ctct.csv
B: Seasonal report docs/s1.csv
C: Seasonal report docs/s2.csv
D: Budget report docs/bgt.csv
E: Half-year report docs/h2.pdf
F: Document archive server

P1: Spreadsheet editor
P2: Auto-budget program
P3: Auto-report program
P4: Firefox browser
A: Tampered file /docs/ctct.csv
B: Seasonal report docs/s1.csv
C: Seasonal report docs/s2.csv
D: Budget report docs/bgt.csv
E: Half-year report docs/h2.pdf
F: Document archive server

P1: Spreadsheet editor
P2: Auto-budget program
P3: Auto-report program
P4: Firefox browser

1. A → P1 → B
2. B → P3 → C
3. C → P4 → D
4. D → P2 → E
5. E → F

Point-to-point
Interference

• Insight: only inbound and outbound files that interfere in a process will possibly produce causality.
  • We determine interference according to the time order of inbound and outbound IO events.
Interference

- Insight: only inbound and outbound files that interfere in a process will possibly produce causality.
  - We determine interference according to the time order of inbound and outbound IO events.
Interference

- Insight: only inbound and outbound files that interfere in a process will possibly produce causality.
  - We determine interference according to the time order of inbound and outbound IO events.
Interference

• Insight: only inbound and outbound files that interfere in a process will possibly produce causality.
  • We determine interference according to the time order of inbound and outbound IO events.
Interference

• Insight: only inbound and outbound files that interfere in a process will possibly produce causality.
  • We determine interference according to the time order of inbound and outbound IO events.
Interference

• Insight: only inbound and outbound files that interfere in a process will possibly produce causality.
  • We determine interference according to the time order of inbound and outbound IO events.
Interference

• Insight: only inbound and outbound files that interfere in a process will possibly produce causality.
  • We determine interference according to the time order of inbound and outbound IO events.
Interference

- Insight: only inbound and outbound files that interfere in a process will possibly produce causality.
  - We determine interference according to the time order of inbound and outbound IO events.
Interference

- Insight: only inbound and outbound files that interfere in a process will possibly produce causality.
  - We determine interference according to the time order of inbound and outbound IO events.
Refinement - selective DIFT

• Replays and conducts DIFT to the necessary part of the execution
  • Aggregation
  • Upstream
  • Downstream
  • Point-to-point
Upstream refinement

- D → Read → P2
- P2 → Write → B
- B → Read → P1
- P1 → Send → A
- P3 → Write → C
- C → Read → B
- B → Read → A
- E → Read → P3
- P3 → Write → F
- F → Mmap → P3
Upstream refinement

D → Read → B → Write → P1 (Send)
E → Read → B
P3 → Write → B
C → Read → P1
F → Mmap → P1
Upstream refinement
Upstream refinement
Implementation summary

• RAIN is built on top of:
  • Arnold, the record replay framework
  • Dtracker (Libdft) and Dytan, the taint engines

<table>
<thead>
<tr>
<th>Host</th>
<th>Module</th>
<th>LoC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target host</td>
<td>Kernel module</td>
<td>2,200 C (Diff)</td>
</tr>
<tr>
<td></td>
<td>Trace logistics</td>
<td>1,100 C</td>
</tr>
<tr>
<td>Analysis host</td>
<td>Provenance graph</td>
<td>6,800 C++</td>
</tr>
<tr>
<td></td>
<td>Trigger/Pruning</td>
<td>1,100 Python</td>
</tr>
<tr>
<td></td>
<td>Selective refinement</td>
<td>900 Python</td>
</tr>
<tr>
<td></td>
<td>DIFT Pin tools</td>
<td>3,500 C/C++ (Diff)</td>
</tr>
</tbody>
</table>
Evaluations

- Runtime performance
- Accuracy
- Analysis cost
- Storage footprint
Runtime overhead: 3.22% SPEC CPU2006
Multi-thread runtime overhead: 5.35% SPLASH-3
IO intensive application: less than 50%
High analysis accuracy

Dependency confusion rate

Scenarios from red team exercise of DARPA Transparent Computing program
Pruning effectiveness: ~94.2% reduction
Storage cost: ~4GB per day (1.5TB per year)
Discussion

• Limitations
  • RAIN trusts the OS that needs kernel integrity protection.
  • Over-tainting issue

• Direction
  • Hypervisor-based RAIN
  • Further reduce storage overhead
Conclusion

• RAIN adopts a multi-level provenance system to facilitate fine-grained analysis that enables accurate attack investigation.

• RAIN has low runtime overhead, as well as significantly improved analysis cost.