Evaluating Atomicity, and Integrity of Correct Memory Acquisition Methods

Michael Gruhn, Felix Freiling

2016-30-03

Department Computer Science
IT Security Infrastructures
Friedrich-Alexander-University Erlangen-Nürnberg
Erlangen, Germany

DFRWS
DIGITAL FORENSIC RESEARCH WORKSHOP EU

FAU
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Outline

Introduction

Motivation

Atomicity, Integrity and Correctness per [Vömel and Freiling 2012]

Atomicity Violation

Integrity Violation

Estimating Atomicity and Integrity

Payload Application

Atomicity and Integrity Deltas

Results

Take-Home and Future Research

Motivation

- Memory Analysis becomes more and more important:
  - Memory resident malware
  - Disk-less clients
  - Persistent Disk Encryption
- To do proper analysis memory must be acquired forensically sound
  - Correctness
    - captured value at address X must represent the value in memory at address X
  - Atomicity
  - Integrity

Atomicity Violation per [Vömel and Freiling 2012]

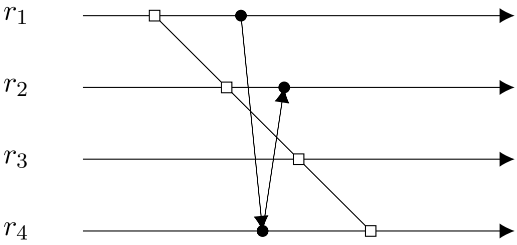

Figure: Space-time diagram of imaging procedure creating non-atomic snapshot.

Integrity Violation per [Vömel and Freiling 2012]

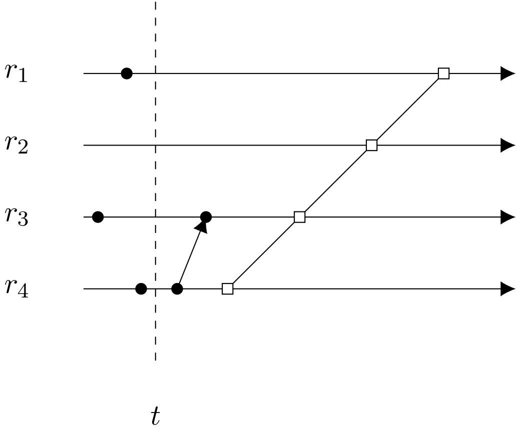

Figure: Integrity of a snapshot with respect to a specific point in time t.

Outline

Introduction
- Motivation

Atomicity, Integrity and Correctness per [Vömel and Freiling 2012]
- Atomicity Violation
- Integrity Violation

Estimating Atomicity and Integrity
- Payload Application
- Atomicity and Integrity Deltas

Results

Take-Home and Future Research

Estimating Atomicity and Integrity via Payload Application

- Application constantly increments counters placed in memory regions
- Start:

Memory Region	Counter
1	0
2	0
3	0
4	0

Estimating Atomicity and Integrity via Payload Application

- Application constantly increments counters placed in memory regions
- Running:

Memory Region	Counter
1	**1**
2	0
3	0
4	0

Estimating Atomicity and Integrity via Payload Application

- Application constantly increments counters placed in memory regions
- Running:

Memory Region	Counter
1	1
2	**1**
3	0
4	0

Estimating Atomicity and Integrity via Payload Application

- Application constantly increments counters placed in memory regions
- Running:

Memory Region	Counter
1	1
2	1
3	**1**
4	0

Estimating Atomicity and Integrity via Payload Application

- Application constantly increments counters placed in memory regions
- Running:

Memory Region	Counter
1	1
2	1
3	1
4	**1**

Estimating Atomicity and Integrity via Payload Application

- Application constantly increments counters placed in memory regions
- Running:

Memory Region	Counter
1	**2**
2	1
3	1
4	1

Estimating Atomicity and Integrity via Payload Application

- Application constantly increments counters placed in memory regions
- Running:

Memory Region	Counter
1	2
2	**2**
3	1
4	1

Estimating Atomicity and Integrity via Payload Application

- Application constantly increments counters placed in memory regions
- Running:

Memory Region	Counter
1	2
2	2
3	**2**
4	1

Estimating Atomicity and Integrity via Payload Application

- Application constantly increments counters placed in memory regions
- Running:

Memory Region	Counter
1	2
2	2
3	2
4	2

Estimating Atomicity and Integrity via Payload Application

- Application constantly increments counters placed in memory regions
- Running:

Memory Region	Counter
1	3
2	2
3	2
4	2

Estimating Atomicity and Integrity via Payload Application

- Application constantly increments counters placed in memory regions
- Running:

Memory Region	Counter
1	3
2	3
3	2
4	2

Estimating Atomicity and Integrity via Payload Application

- Application constantly increments counters placed in memory regions
- Running:

Memory Region	Counter
1	3
2	3
3	3
4	2

Estimating Atomicity and Integrity via Payload Application

- Application constantly increments counters placed in memory regions
- Running:

Memory Region	Counter
1	3
2	3
3	3
4	3

Estimating Atomicity and Integrity via Payload Application

- Application constantly increments counters placed in memory regions
- Running:

Memory Region	Counter
1	3
2	3
3	3
4	3

- Perfect atomic capture has only **two consecutive** counter values
- Perfect integer when counter values from when capture was started
- Details in the paper

Estimating Atomicity and Integrity via Deltas

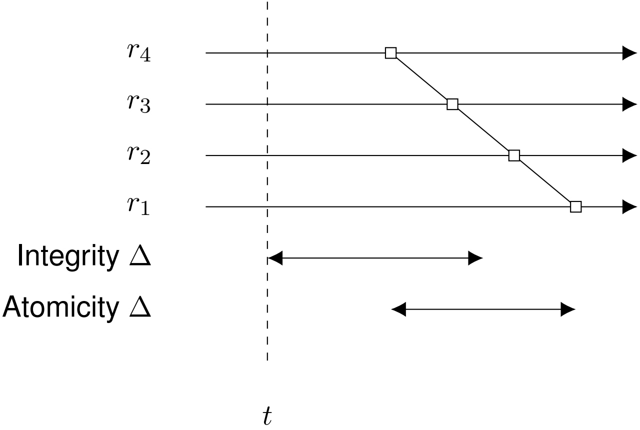

Figure: Atomicity and integrity in a maximum load scenario.

Atomicity and Integrity Upper Bounds

	(Worst Case) Atomicity Delta	(Worst Case) Integrity Delta
msramdump	1	43.84
memimager	1	63.28
VirtualBox	1	26.64
QEMU	1	35.24
ProcDump (-r)	0	39.75
ProcDump	1	36.50
Windows Task Manager	1	728.54
pmdump	37	136.62
WinPMEM	13230	5682.24
FTK Imager	13151	5917.24
win64dd	15039	8077.54
win64dd (/m 1)	15039	8172.28
DumpIt	15711	8500.09
inception	43898	22056.77

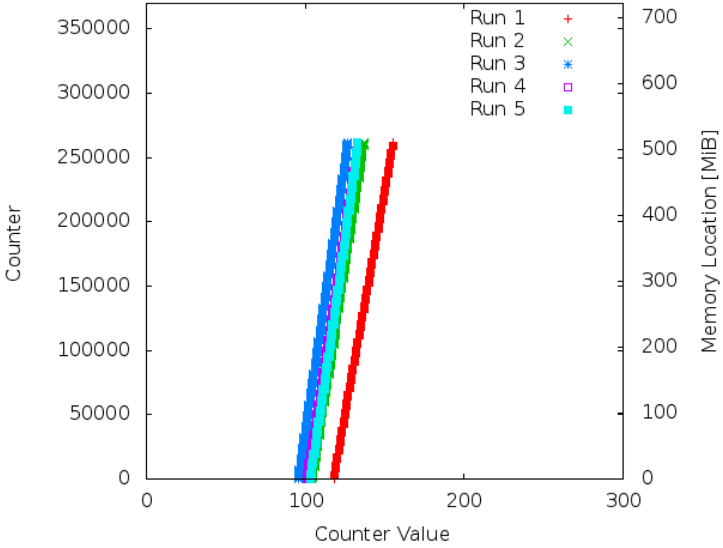

Figure: Acquisition plot of pmdump

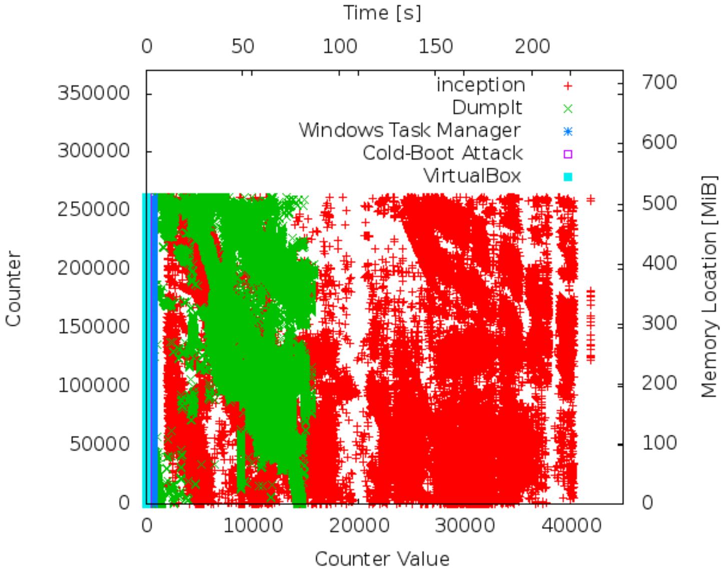

Figure: Memory acquisition technique comparison (acquisition plot)

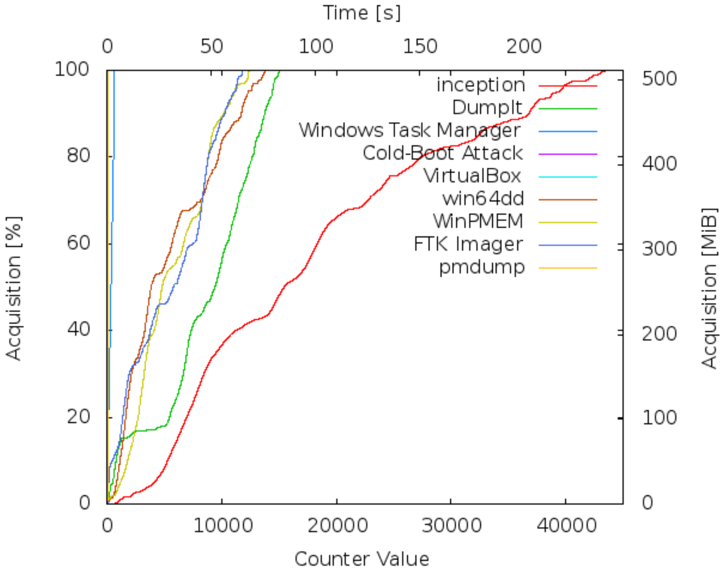

Figure: Memory acquisition technique comparison (acquisition density plot)

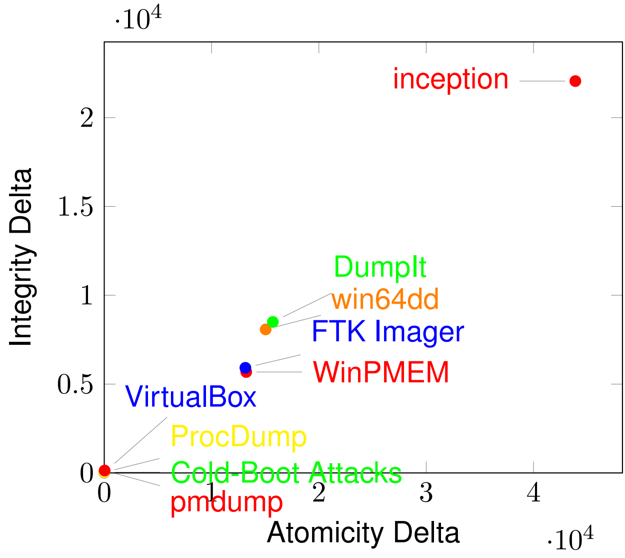

Figure: Each acquisition position inside an atomicity/integrity-Matrix

Take-Home and Future Research

- DMA exhibited the greatest memory smear
  - Is inception/Python the issue?
  - Will PCI DMA perform better?
  - Does DMA increase concurrency?
- How do state-of-the-art research methods (Body-Snatcher) perform?

Take-Home and Future Research

- DMA exhibited the greatest memory smear
    - Is inception/Python the issue?
    - Will PCI DMA perform better?
    - Does DMA increase concurrency?
- How do state-of-the-art research methods (Body-Snatcher) perform?
- **What is the impact of non-atomic memory captures on analysis?**
    - 2-Take Approach solution?

Take-Home and Future Research

- DMA exhibited the greatest memory smear
    - Is inception/Python the issue?
    - Will PCI DMA perform better?
    - Does DMA increase concurrency?
- How do state-of-the-art research methods (Body-Snatcher) perform?
- **What is the impact of non-atomic memory captures on analysis?**
    - 2-Take Approach solution?

Source Code available at
`https://www1.cs.fau.de/projects/rammangler`
Slides and Paper available at
`https://http://www.dfrws.org/2016eu/program.shtml`

Warning about "Source Code": It's what they call "research" code:

```
for(i=0; /*FIXME ... we assume success */; i++)
```

Questions?

42.