

# **The Role of Operating System in Computer Forensics**

**Ewa Huebner**  
**University of Western Sydney**  
**Australia**

# My background

## → 1972

→ graduated with Masters of Electronic Engineering PW

## → 1972 – 1976

→ worked for Instytut Maszyn Matematycznych

## → 1977 – 1996

→ systems programmer for various institutions in Australia

## → 1996 – 1999

→ PhD candidate at the University of Sydney, Australia

→ graduated in July 1999

## → 1999 till now

→ academic at University of Western Sydney, Australia

# Computer Forensics at UWS

## → Started in 2004

### → Teaching

- new subject Computer Forensics Workshop
- new specialisation in Computer Forensics for the bachelor of Computer Science degree
- project work in Computer Forensics at various levels

### → Research

- started exploring the field to identify areas suitable for further research
- published a number of papers in forensics journals and conferences

# Historical note

## → Conventional computer forensics

- focuses on investigation of file systems
- it is possible to acquire the content in a forensically sound way
- i.e. without modifying the data under investigation

## → Next generation

- attempt to capture volatile data
- memory forensics
- live system investigation



# OS in Computer Forensic Investigations

## → Natural phenomenon?

- OS is an artifact, but
- it is a complex system with non-deterministic behaviour
- very difficult (if not impossible) to examine analytically

## → Black box

- have to accept the OS as is
- even if it is a hindrance to investigation
- many 3<sup>rd</sup> party tools which provide additional functionality

## OS continued

### → **Similar to early efforts in increasing security**

→ but now OS's built with security features in mind

- not a later addition to existing system

→ for example in Windows

- free pages in memory are cleared by a background zero thread
- Encrypting File System is easily available to unsophisticated users
- CTRL-ALT-DEL combination of keys to avoid login spoofing

# OS in conventional computer forensics

## → File system

- structure defined independently of OS
- but how this structure is used is OS dependent
- in particular so called non-essential data
- For example
  - maintenance of the time stamps in files
  - stale information in deallocated blocks

## → System logs

- can be deactivated
- have a fixed length (will be truncated by the system)
- can be manipulated to hide some events
  - somewhat harder on Windows with binary format of event log file

# OS in live system investigations

## → Availability of native utilities and API crucial to successful investigation

- this is easier in Unix-derivative systems where OS code is in public domain
- for Windows there are many toolkits built on top of the system
  - the level of confidence cannot be absolute

## → The real problem – memory forensics

# OS in Memory Forensics

- **To investigate memory forensically we need to**
  - acquire a complete image of system physical memory
  - interpret the content of this image

**All this without changing the system  
under investigation in any way!!! (\*)**

(\*) Locard principle

# Memory Image Acquisition

**→ Two fundamental problems:**

- 1. How to do it without changing the very memory we wish to acquire?**
- 2. How to do it to ensure that memory image is complete and self-consistent?**

# 1<sup>st</sup> problem

## → All system activities require memory

- also taking an image of memory content
- the image changes as a consequence of acquisition

## → Hardware supported acquisition

- PCI Tribble card
- Fire Wire protocol
- both based on DMA (Direct Memory Access)

## → Will it work?

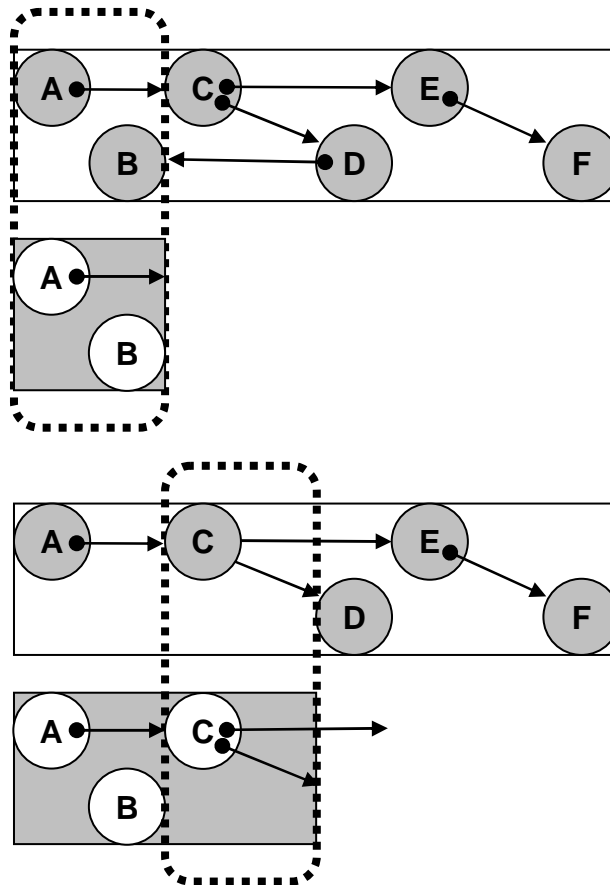
- Tribble needs to be installed before the incident
- the processor continues to run while DMA transfer is in progress, changing the contents of memory

## 2<sup>nd</sup> problem

### → **memory image acquired while the system is active**

- the process requires significant time to complete
  - counted in seconds
- contains objects from different moments in time
- the image may not be self-consistent

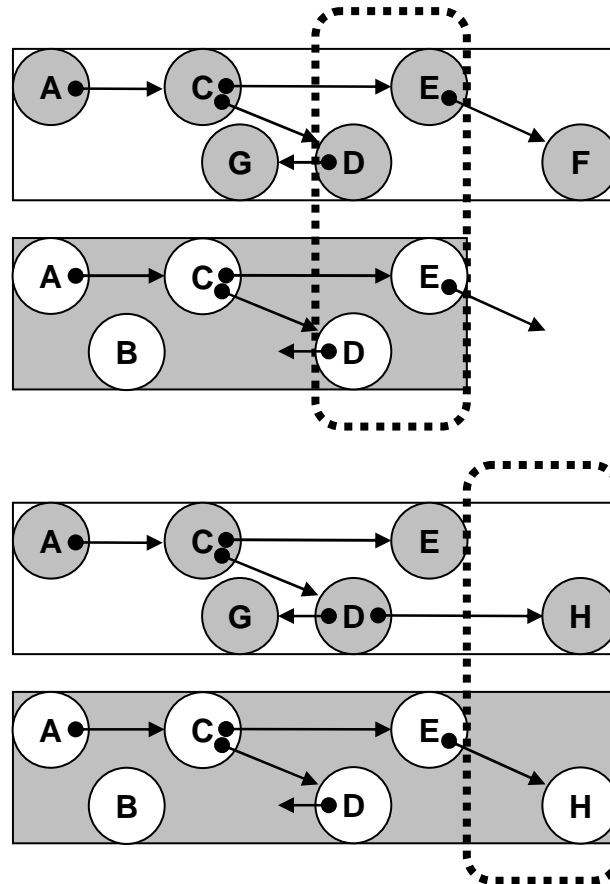
# Not self-consistent memory image



a) Objects A and B are copied.

b) Object B is deleted, object C is copied.

## Not self-consistent memory image (2)



c) Object G is created, objects D (with a pointer to object G) and object E (with a pointer to object F) are copied.

d) Object F is deleted. Object H (pointed to by object D) is created.

# Consequences

- **The image acquired does not represent contents of memory in any moment of time**
- **Information is missing, for example**
  - connection between object D and object G
  - connection between object E and object F
- **Information may be wrong, for example**
  - object E appears to be connected with object H, while in reality it was object D which had connection to object H
- **This may lead to incomplete and even wrong analysis of data**

## Possible solution

→ **Forensically sound acquisition is not possible without direct support from the kernel of the OS**

→ such support is not present in the main stream operating systems today

→ **The same problems were solved**

→ for so called persistent systems

→ We can borrow the way of reasoning leading to a solution of our problem

# Persistent Systems

## → Persistent systems features

- The system must support creation and maintenance of persistent objects as a basic abstraction.
- The state of processes must be contained within the persistent objects.
- A protection mechanism for persistent objects must be provided.
- Persistent objects must be stable and resilient, i.e. they must survive system malfunction

# New functionality in OS

## → Segregate a small area in physical memory

- fixed range of addresses
- not used in normal operation of the system

## → This area contains

- optimised code capable of copying the remaining content of physical memory to external media (for example a USB drive)
- all necessary buffers and device drivers are also part of this code
- the code runs with interrupts disabled
- the code is invoked by entering an unusual combination of keys captured by the keyboard driver
- it may also copy the swap areas (can be acquired conventionally)

## Is this all?

### → This approach guarantees that

- memory image is self-consistent
- it should be possible to interpret the part of memory allocated to OS and user processes

### → What else is in physical memory?

- free pages (allocated to past or current processes)
- file block cache pages used to improve performance of the file system

# Interpretation of non-allocated memory

## → OS offers no assistance here

- these pages are not logically a part of the address spaces maintained by OS
- they are remnants of some past activities
- irrelevant from the OS point of view
- very important from forensic point of view

## → Some things OS could do

- maintain a log of memory allocation with some history on how the page frame was used
- keep track how long each page resides in memory (can also be useful for the purpose of page eviction strategy)

# File Cache Blocks

## → Can be identified

- if PCB (Process Control Block) information in kernel is available
- we can determine which file these blocks belong to

## → Other gains

- some blocks may be available for files recently deleted
- fragments of encrypted files may appear in clear text

# Interpretation of memory image

## → Old fashioned OSs

→ used to have memory dump analysis tools, mostly used to analyse the system after crash

## → UNIX derivatives

→ have some remnants of this functionality, geared to crash analysis

## → WINDOWS

→ such utilities may exist, but are not in public domain

## → 3<sup>rd</sup> party tools

→ difficult to create and maintain

→ structure of memory may differ between various versions of the same system

# Ultimate interpreter

## → The OS itself is the ultimate interpreter of memory contents

→ it should be possible to construct tools which use parts of the OS to interpret the contents of memory image statically without actually running the system

## → Another way – virtual systems

→ memory image is loaded in a virtual environment

→ the host system can control the behaviour of the virtual guest system to interpret the contents of the memory image

## Is there more?

### → Operating systems should be designed and implemented with awareness of requirements of forensic investigations

- much more research is needed to explore all problems
- it is no longer satisfactory to accept that other aspects of design will determine all design decisions
- it may be on collision course with security requirements

### → The starting point

- April 2008 issue of ACM Operating Systems review dedicated to these very issues <http://www.sigops.org/osr.html>

**Thank you for your patience.**

**Any questions?**