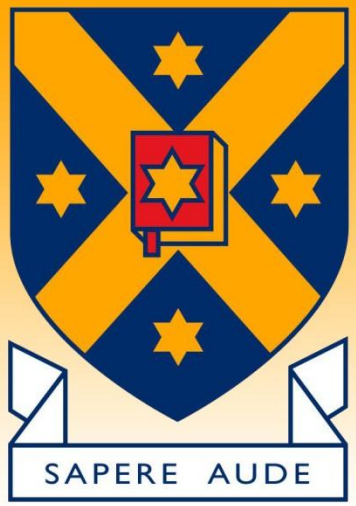


Towards A Standardised Strategy to Collect and Distribute Application Software Artifacts

Thomas Laurenson
Supervisors: Stephen MacDonnel, Hank Wolfe
Department of Information Science
School of Business
University of Otago
Dunedin, New Zealand

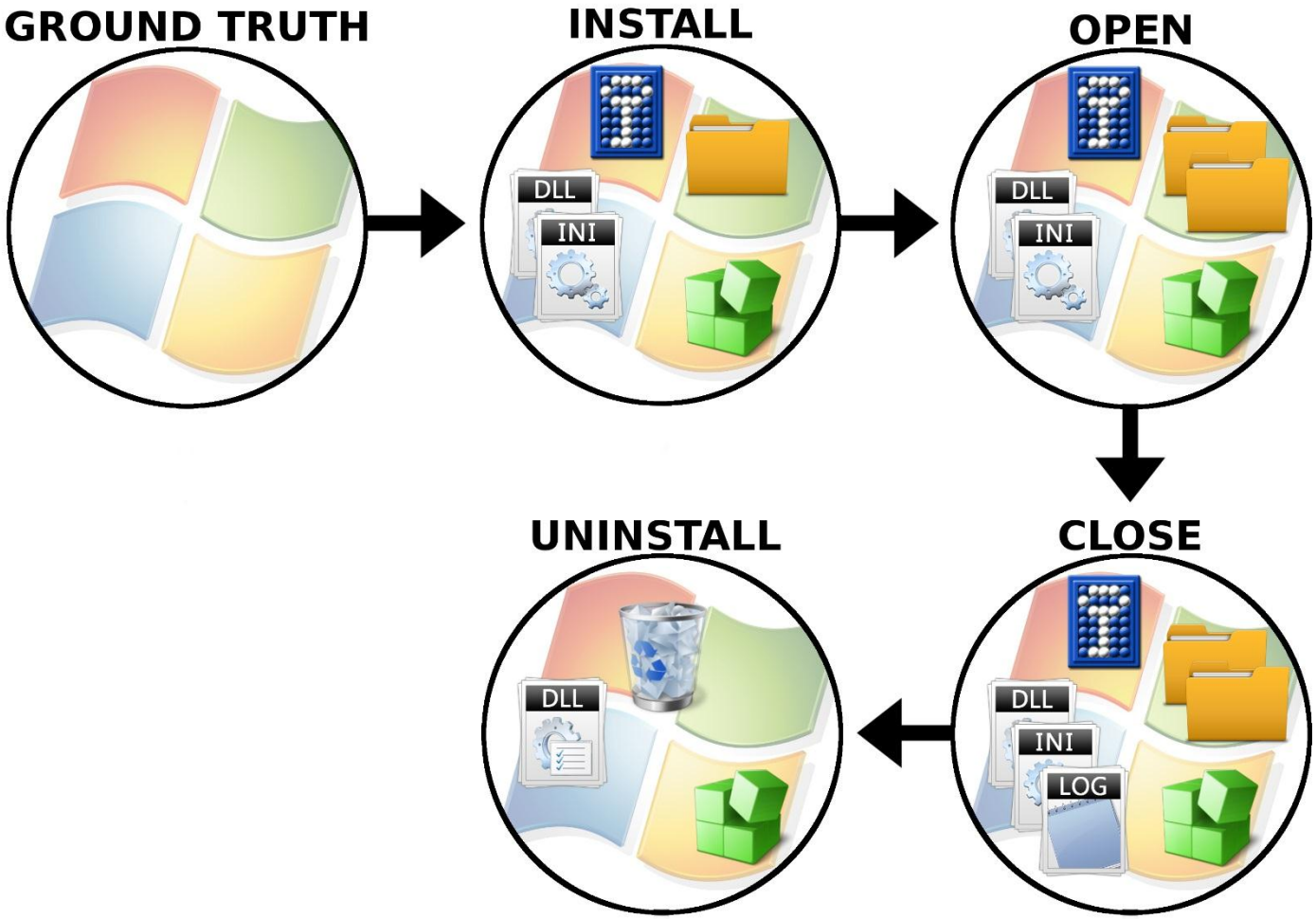
2015 SRI Security Congress
13th Australian Digital Forensics Conference
Perth, Western Australia

Application Software Artifacts

- Different types of digital artifacts

- Application software causes system-level changes
- Applications have various life cycle phases

Application Life Cycle



Reference Sets

- Contain known content represented by metadata
- Usually data files and crypto hash values (MD5 and SHA-1)
- Can identify **relevant** content
- Can filter **irrelevant** content
- Reference sets for application software have a variety of names
  - ***Application profile***
  - Application fingerprint
  - Application footprint

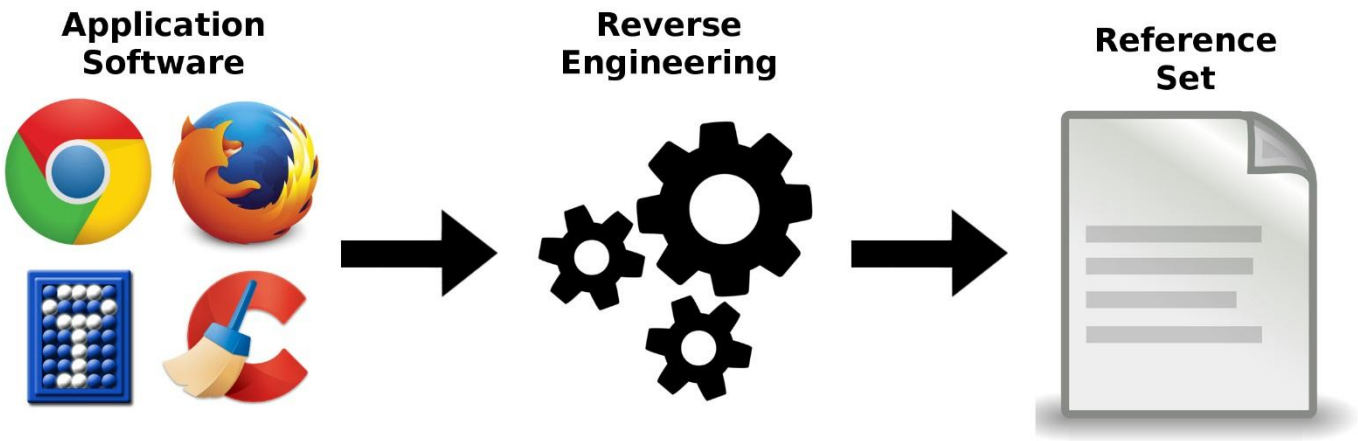
Research Problem

- Application software needs reverse engineering
- Reverse engineering lacks standardisation
- **Data collection** problems:
  - No systematic approach
  - No standard set of tools
  - No tool automation
- **Data distribution** problems:
  - No standardised data abstraction to store or distribute results
  - Challenges incorporating multiple evidence sources
- Present research is a standalone endeavour

Research Objective

To facilitate a **standardised strategy** to **collect** and **distribute** **application software artifacts**

System Design



Design Science Research Methodology

System Design:

- **Data collection** (reverse engineering)
- **Data distribution** (reference set)

System Implementation:

- Software Development

System Requirements

- Data collection:
  - Portable Microsoft Windows tool
  - Perform **data collection** on a live system
  - Use an **automated procedure** for data collection
  - Remove irrelevant digital artifacts
  - Inclusion of data file hashing
- Data distribution:
  - Standardised data abstraction
  - Store, distribute and automate processing
- Build on similar solutions
- Open Source

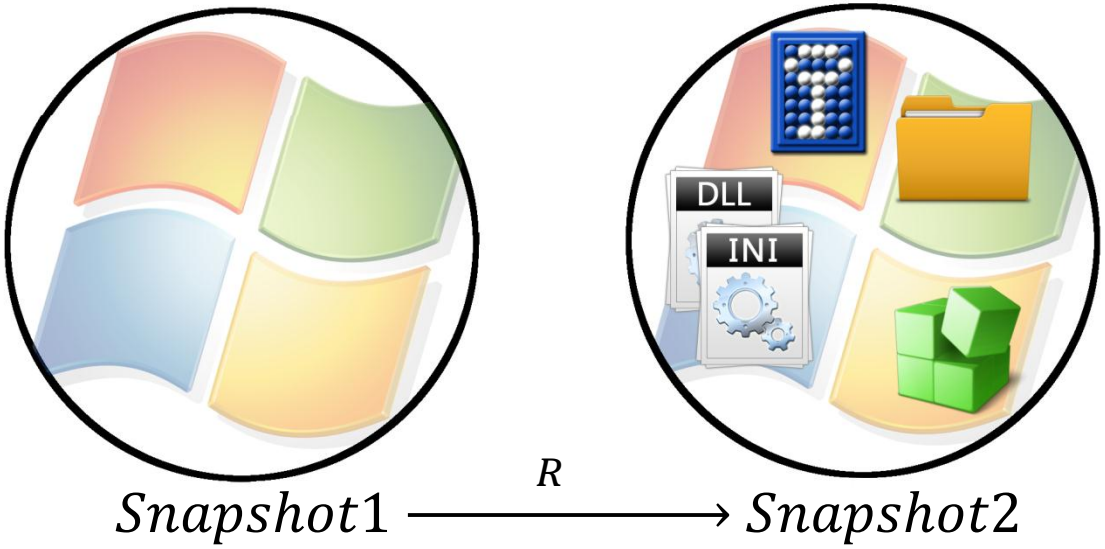
Data Collection: Method

- Authored a new system-level differencing tool
  - Named `LiveDiff`
  - Portable application
  - Run on Microsoft Windows
  - Based on `Regshot` project

- Data collection is performed on a live running system
- Data collection is performed by capturing a system ***snapshot***:
  - Scan every file system entry
  - Scan every Registry entry

Differential Analysis

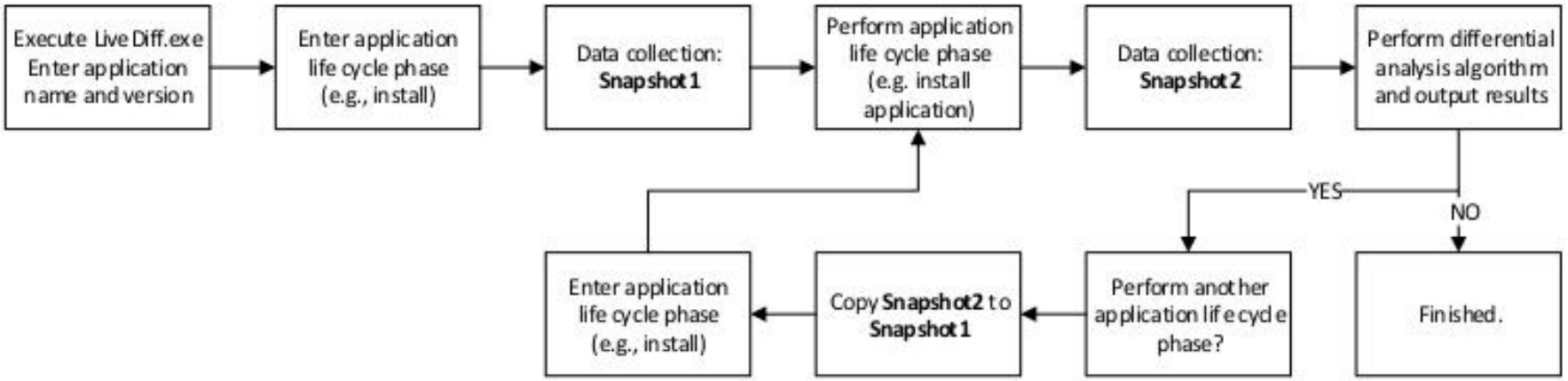
- ***Snapshot*** before and after an action (e.g., install)
- Compare snapshots
  - Formalised **differential forensic analysis** strategy



- R = new, changed (properties), modified (content), deleted

Data Collection: Procedure

- **Automated procedure** for data collection
- Command line interface
- Minimal input required by user



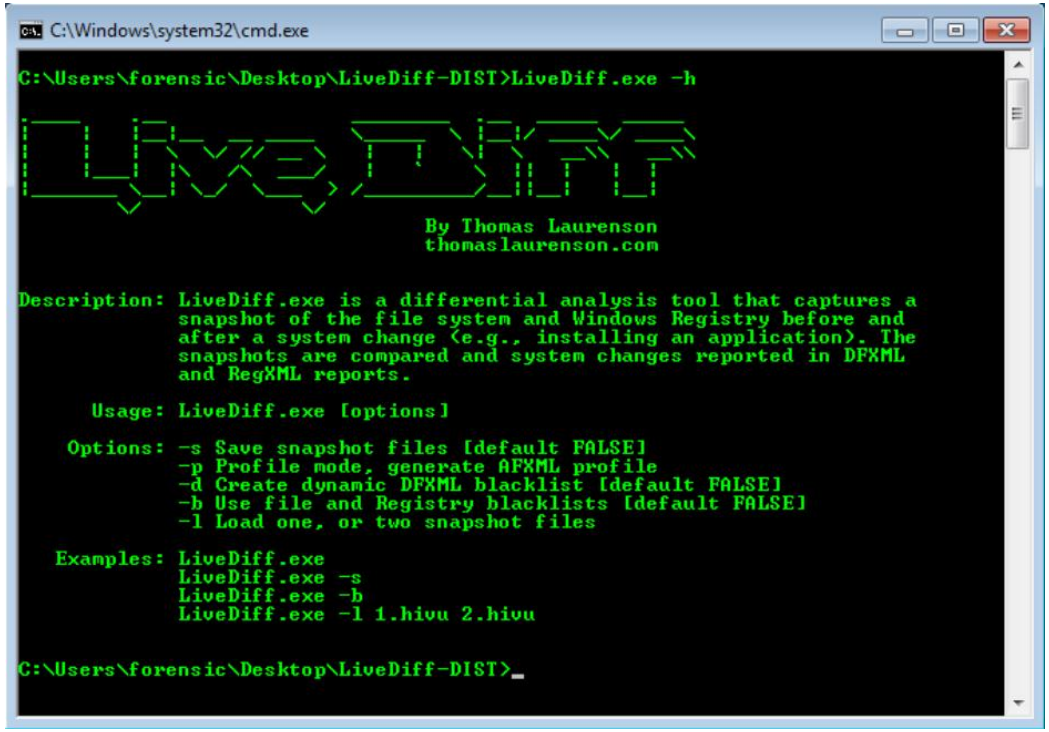
Dynamic Blacklisting

- Collect ***snapshot*** before data collection
- Populate blacklist using snapshot
  - Using logical path
  - All entries are known
  - Blacklist stored in memory using a ***prefix tree***

1. Blacklist queried when performing ***snapshots***:
   - If artifact path found, skip the entry as it is known

2. Blacklist can help data ***file hashing***:
   - Cannot hash every data file (> 10 minutes)
   - Only hash data files not in blacklist

Tool Demonstration



Data Abstraction

- A formalised data abstraction structure should support:
  - Storage
  - Distribution
  - Automated processing

- **Application Profile XML (APXML)**
  - Leverages Digital Forensic XML (DFXML)
  - File system entries (`FileObject`)
  - Registry entries (`CellObject`)

- Standardised using XML schema: `apxml.xsd`

Metadata Properties

- Selected only unique object properties

File System (`FileObjects`)		Windows Registry (`CellObjects`)	
Directory	**File**	**Key**	**Value**
Full path	Full path	Full path	Full Path
Type (2)	Type (1)	Type (k)	Type (v)
Allocation	SHA1 Hash	Allocation	Value name
	Allocation		Value data type
			Data (contents)
			Raw data (contents)
			Allocation

APXML Structure

```
<apxml version='1.0.0' encoding='UTF-16' ?>
  <metadata/>
  <creator/>
  <install>
       <!-- FileObjects -->
       <!-- CellObjects -->
  </install>
  <execute>
       <!-- FileObjects -->
       <!-- CellObjects -->
  </execute>
  <uninstall>
       <!-- FileObjects -->
       <!-- CellObjects -->
  </uninstall>
</apxml>
```

Automated Processing

- Python API: `apxml.py`
  - Read, Write APXML documents
  - Automated processing using simple scripts

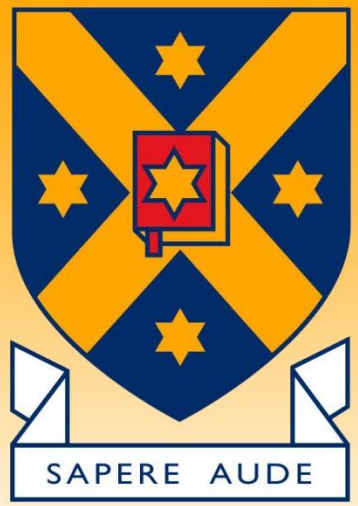
```
import sys, apxml, Objects

# Read Application Profile XML document
apxml_object = apxml.iterparse(sys.argv[1])

# Create a list of known hashes (SHA-1) from profile
known_sha1s = {f.sha1: f for f in apxml_object
               if isinstance(f, Objects.FileObject) and f.meta_type == 1}

# Find known files in target disk image
for (event, obj) in Objects.iterparse(sys.argv[2]):
    if isinstance(obj, Objects.FileObject):
        if (obj.filename in known_sha1s):
            print(obj.filename, obj.sha1)
```

Conclusion

- Data collection > Differential Analysis > `LiveDiff`
- Data distribution > Reference Set > `APXML`
- Implemented using standardised and accepted DF techniques

- https://github.com/thomaslaurenson/livediff/
- https://github.com/thomaslaurenson/apxml_schema/

- My future research
  - Removing irrelevant digital artifacts
  - Matching digital artifacts
  - Advanced hashing algorithms (block hashing, similarity digests)
- Future Research
  - Additional evidence sources (volatile memory, network traffic)



Thank you for your attention
Any Questions?