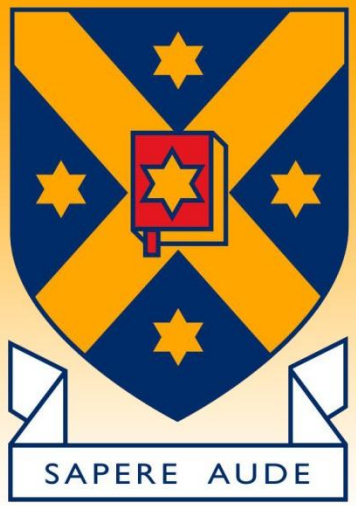


Improving the Detection and Validation of Inland Revenue Numbers

Henry Gee
Thomas Laurenson
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

Research Problem

- **Inland Revenue (IRD) numbers**
  - Used for taxation purposes in New Zealand
  - Similar to Social Security Number (SSN) and Tax File Number (TFN)
  - 8 or 9 digit number
- **Personally Identifiable Information (PII)**
  - Used to identify, link or locate a person
  - Names, alias, email address, passport number
- String searching
- No solutions for non-US personal information
- No verified techniques or tools for IRD numbers

Research Objective

To enable **effective** and **efficient automated detection** and **validation** of Inland Revenue numbers

- Design Science Research Methodology
- System Design and Implementation
- System Evaluation
  - Known data set testing
  - Real world data set testing

System Requirements

- **Detect IRD Numbers**
- **Validate potential IRD numbers**
- Adhere to digital forensic requirements
- Build on similar solutions
- Search common forensic data abstractions
  - Disk images (RAW, E01), RAM dumps, network traffic captures, files
- Search unallocated (slack) space
- Search compressed data
- Open source

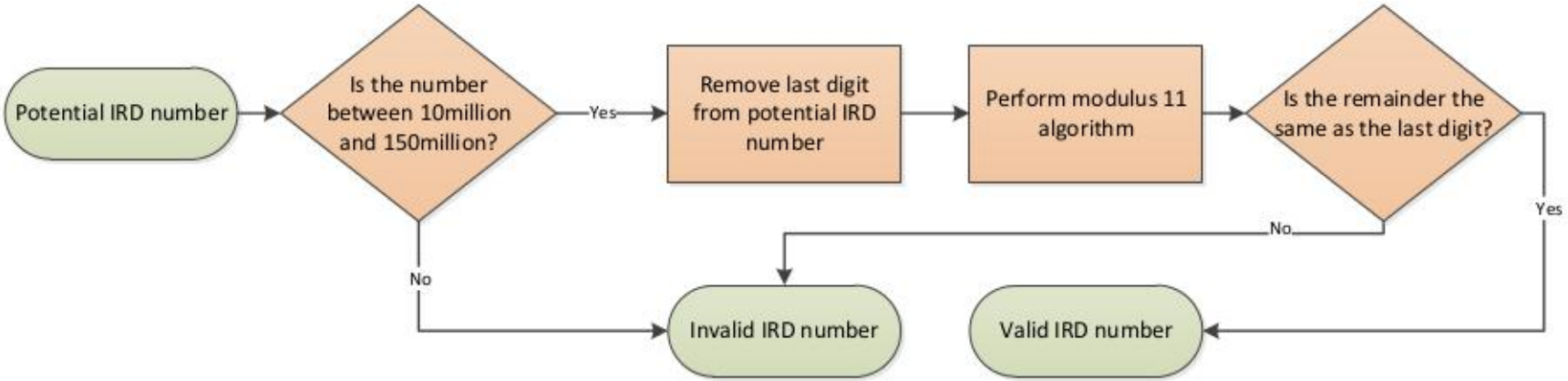
IRD Numbers

- No standardised structure
- 8 or 9 digit number

IRD Number Description	IRD Number Structure	Example
8 digits	NNNNNNNN	12345678
8 digits with space delimiter	NN NNN NNN	12 345 678
8 digits with dash delimiter	NN-NNN-NNN	12-345-678
9 digits	NNNNNNNNN	123456789
9 digits with space delimiter	NNN NNN NNN	123 456 789
9 digits with dash delimiter	NNN-NNN-NNN	123-456-789

IRD Number Validation

- Validate using modulus 11 algorithm
- Total possible numbers: $10^8 + 10^9$ (> 1 billion)
- Within Range: 140 million possible numbers
- With validation: 13.8 million possible numbers



System Implementation Platform

- Selected the `bulk_extractor` tool
  - Stream forensic tool
  - Authored by Simson Garfinkel
  - https://github.com/simsong/bulk_extractor
- Specifically designed to extract ***features***
  - Email addresses, credit card numbers, telephone numbers
- Processes ***bulk data***
  - Disk images, RAM dumps, network traffic captures, files
- Multithreaded
- Optimistic data decompression
- Plug-in architecture

Scanner Plug-in Development

- Authored a new scanner plug-in
- FLEX search patterns to detect the **six defined structures**
- IRD number validation

```
DELIM           ([-])
START8          [0-9]{2}
BLOCK           [0-9]{3}

[^0-9]{START8}{DELIM}{BLOCK}{DELIM}{BLOCK}  {
        /* Number structure: NN-NNN-NNN */
        /* IRD scanner processing code goes here */
}
```

Scanner Plug-in Output

1. Feature file
2. Histogram file

```
# BULK_EXTRACTOR-Version: 1.4.0 ($Rev: 10844 $)
# Feature-Recorder: ird
# Filename: /media/forensic/HDD/WindowsTestHDD.001
# Feature-File-Version: 1.1
107594013       112233445       plugininstaller_1122334455667788_6.1.7600
107598558       80-137-249      llageGST Reg No:80-137-2494B Titoki Place
107598608       76264279        id: {l:16 b:a04b76264279d00118000000cc02
107618283       22-220-616      N<BR>IRD no.<BR>22-220-616 <BR>Tax Code
```

LOCATION (OFFSET)

FEATURE (IRD NUMBER)

FEATURE CONTEXT

Stoplist Implementation (1)

- False positives results are unavoidable!
  - Especially true when searching for an 8 or 9 digit number
  - True for most string searching
- Leveraged `bulk_extractor` stoplist technique
  - Context sensitive stoplist

```
# BULK_EXTRACTOR-Version: 1.4.0 ($Rev: 10844 $)
# Feature-Recorder: ird
# Filename: /media/forensic/HDD/WindowsTestHDD.001
# Feature-File-Version: 1.1
107594013       112233445       plugininstaller_1122334455667788_6.1.7600
107598558       80-137-249      llageGST Reg No:80-137-2494B Titoki Place
107598608       76264279        id: {l:16 b:a04b76264279d00118000000cc02
107618283       22-220-616      N<BR>IRD no.<BR>22-220-616 <BR>Tax Code
```

FEATURE CONTEXT

Stoplist Implementation (2)

- Stoplist generation method:
  1. Install a default operating system, acquire forensic image
  2. Process using IRD scanner
  3. All discovered ***features*** are deemed irrelevant

- Selected 16 operating systems to create a master stoplist
  - Total number of features: 572,057
  - Number of **unique** features : 116,120

1) MSDOS622
2) Windows 3.1
3) Windows 95
4) Windows 98
5) Windows ME
6) Windows 2000
7) Windows NT 4.0
8) Windows XP (32 bit)
9) Windows XP (64 bit)
10) Windows Server 2003
11) Windows Vista
12) Windows 7 (32 bit)
13) Windows 7 (64 bit)
14) Windows 8
15) Windows 8.1
16) Windows 10

Known Data Set Testing: Method (1)

- Use a Virtual Machine (VM) testing environment
- Installed with Microsoft Windows 7 (default options)
- Created 24 files with known valid IRD numbers
  - doc, docx, xls, xlsx, rtf, pdf, folders, archives
  - Variety of encodings, different office versions/platforms
- Copied documents to VM using shared folder
  - Performed MD5 before and after to ensure data integrity
- Acquired a forensic image of VM
- Ready for testing…

Known Data Set Testing: Method (2)

- We need a experimental testing baseline
- `strings` and `grep`
  - Commonly used to perform text-based string searches
  - `strings`: extract text from raw data
  - `grep`: search input (text) using regular expressions

- Processed known data set using four (4) techniques:
  1. `strings` and `grep` with no validation
  2. IRD Scanner with validation
  3. `strings` and `grep` with validation
  4. IRD Scanner with validation and stoplist

Known Data Set Testing: Results

IRD Number Extraction Method	Total Found	Total Stopped	Percentage Stopped
Strings and grep with no validation	290,772	N/A	N/A
IRD Scanner	21,616	N/A	N/A
Strings and grep with validation	10,904	N/A	N/A
IRD Scanner with stoplist	62	21,554	99.71%

Real Data Set Testing: Method

- Second-hand HDDs sourced from online auction (TradeMe)
  - Total of 152
  - 122 were readable
  - 30 were wiped (zeroed)
  - This left 92 used for testing
  - Approximately 3TB raw data

- Processed using **IRD scanner** plug-in and **master stoplist**

Real Data: Results Overview

- Huge number of false positives
- Detected 15 million IRD numbers
- Stopped 1 million IRD numbers (7.44%)
- Why did the stoplist fail us?
  - Some drives it worked (6 HDDs the stopped features were > 60%)
  - Many features found in ‘Program Files’
  - Many features found in ‘user’ data (user folders and slack space)

- More data reduction is essential
- Performed **post-processing** techniques

Post-processing: Framework

- Social Security Number (SSN) detection techniques

  SSN: NNN-NNN-NNN

  SSN NNN-NNN-NNN

- Our approach is different:
  - Find and validate number (IRD Scanner does this)
  - Search context (data surrounding *feature*)

- Prefixes:
  - IRD
  - IRD No
  - IRD No is
  - IRD #
  - IRD:
  - IRD No:
  - IRD no
  - Ird Number
  - IRD num

Post-processing: Method

- Parse the feature file
- Search the feature context for keywords:
  - ‘IRD’
  - ‘GST’

```
# BULK_EXTRACTOR-Version: 1.4.0 ($Rev: 10844 $)
# Feature-Recorder: ird
# Filename: /media/forensic/HDD/WindowsTestHDD.001
# Feature-File-Version: 1.1
107594013       112233445       plugininstaller_1122334455667788_6.1.7600
107598558       80-137-249      llageGST Reg No:80-137-2494B Titoki Place
107598608       76264279        id: {l:16 b:a04b76264279d00118000000cc02
107618283       22-220-616      N<BR>IRD no.<BR>22-220-616 <BR>Tax Code
```

KEYWORD HIT IN FEATURE CONTEXT

Post-processing Results

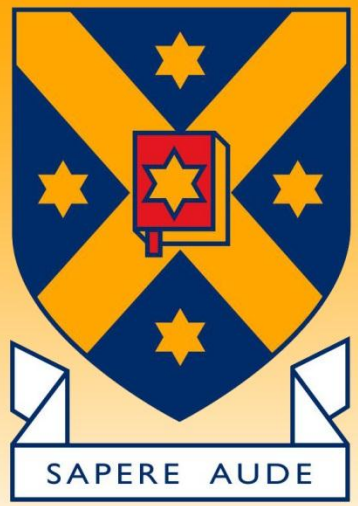
- Post-processing removed 99.96% IRD numbers
- Resulted in 5,700 ***interesting*** IRD numbers

Search Term	Keyword Hit	Allocated File	Unallocated File
IRD	172	37	135
GST	5,528	755	4,773
Total	5,700	792	4,908

- Found IRD numbers in many file formats of forensic interest:
  - Microsoft Word documents, Excel spreadsheets, PDF files, CSV files,
  - Lotus Notes database files, Outlook Express files, Exchange Server EDB files

Conclusions

- There are no robust tools for non-US personal information
- Effective and efficient tool for IRD number detection
- IRD numbers are hard to detect
  - No standardised number structure
  - Found with many different prefixes
- Novel post-processing technique
- Research project is openly available:
  - IRD Scanner plugin (scan_ird.flex)
  - IRD stoplist (ird_stoplist.txt)
  - Post-processing scripts
  - https://github.com/thomaslaurenson/IRDNumberScanner



Thank you for your attention
Any Questions?