2015

Improving the detection and validation of inland revenue numbers

Henry Gee
University of Otago

Thomas Laurenson

Hank Wolfe
University of Otago

DOI: 10.4225/75/57b3f6edfb88a
13th Australian Digital Forensics Conference, held from the 30 November – 2 December, 2015 (pp. 62-69), Edith Cowan University Joondalup Campus, Perth, Western Australia.

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. Otago, New Zealand
geehenry123@gmail.com, thomas@thomaslaurenson.com, hank.wolfe@otago.ac.nz

Abstract

Forensic analysis commonly involves searching an investigation target for personal identifiable information. An Inland Revenue Department (IRD) number is used for taxation purposes in New Zealand and can provide evidence of perpetrator identity, transaction information or electronic fraud. This research has designed and implemented a bulk_extractor feature scanner to detect and validate IRD numbers (features). The IRD scanner has been tested on a known data set to ensure tool functionality. A large real world data set was then used to determine scanner effectiveness in a realistic investigation scenario. Real world data set testing highlighted a high number of unrelated features detected by the scanner. To combat this, a novel post-processing technique was implemented to identify forensically interesting IRD numbers by performing feature context searching. The post-processing findings proved that feature context searching is an effective data reduction technique that identified a low number of directly relevant IRD numbers.

Keywords
Digital forensic analysis, Personally Identifiable Information, Inland Revenue number, IRD, bulk_extractor

INTRODUCTION

The size of digital investigation targets continues to grow at an exponential rate due to increased digital storage capacity and the number of devices seized per case (Quick & Choo, 2014). Furthermore, the complexity of digital investigations is also escalating caused by the proliferation of operating systems and file formats (Garfinkel, 2010). As a consequence, advanced forensic analysis techniques are required to combat these digital investigation challenges.

A common forensic analysis task is to search digital media for Personally Identifiable Information (PII). The United States (U.S.) Government Accountability Office (GAO) defines PII as any personal information that can be used to locate, link or identify an individual (GAO, 2008). Examples include names, aliases, Social Security Number, passport number, driver's license number, taxpayer identification number, credit card number and email addresses (McCallister, Grance and Scarfone, 2010).

A person's taxpayer identification number is an important personal identifier which is used for general tax purposes. The identification of taxpayer information is useful in digital investigations to determine perpetrator identification, attribution of financial information or potential electronic fraud. In New Zealand, taxpayer identification is controlled by the Inland Revenue Department (IRD) in the form of an IRD number: a unique 8 or 9 digit number. An IRD number is also used for Goods and Services Tax (GST), a tax implemented on most products sold and/or services rendered in New Zealand. Australia has a similar tax number system where each person has a Tax File Number (TFN) issued by the Australian Taxation Office (ATO), while the U.S. equivalent is the Social Security Number (SSN).

Previous Research

Forensic analysis commonly involves performing text-based string searching to identify PII. Digital forensic string searching analyses each byte of digital evidence, at the physical level, to locate specific text strings of interest to an investigation (Beebe & Clark, 2007). Take the case of an electronic fraud investigation; potentially interesting searches may include credit card numbers, bank account numbers and tax identification numbers. Research projects in the past have investigated, designed and evaluated a variety of forensic analysis techniques and tools to retrieve such information.

"An examiner can use the UNIX utilities `strings` and `grep` to perform keyword searches … `strings` will print the human-readable ASCII text it finds in a given input file, and `grep` will search through an input file or stream for a string matching a user-supplied regular expression or string" (Altheide, 2004). The Sleuth Kit (TSK) includes the Windows compatible `srch_strings` program, a modified version of the `strings` command

found in the GNU `binutils` package (Altheide & Carvey, 2011). The `Identity Finder` tool from Cornell University has the ability to scan hard drives, web sites and collections of files to identify social security, credit card and bank account numbers (Cornell University, 2014). Popular forensic tool kits have built-in functionality to identify common PII artifacts. EnCase Forensic has built-in functionality to search for credit card numbers, phone numbers, email addresses and social security numbers (Guidance Software, 2011).

Garfinkel (2006) authored a program that scans media for credit card numbers (with different delimiters) and validates the output with the Luhn algorithm and a variety of other validation checks; for example, no major credit card number begins with the number eight. This was achieved by developing a variety of *feature extractors* to extract potentially interesting information including SSNs, email information and credit card numbers. Garfinkel (2013) continued this development resulting in the `bulk_extractor` tool; a stream forensic tool that reads the entire target media from start to end without performing disk seeking. The `bulk_extractor` tool contains a wide variety of feature extractors, or scanners, which search and extract a variety of PII information such as credit card numbers, telephone numbers and email addresses.

The dilemma is that the discussed techniques and tools have limited usefulness as the resultant search hits are only identified based on the structure of the string being searched. A credit card number, for example, is usually stored in the following structure; NNNN-NNNN-NNNN-NNNN, four octets of four numbers. However, not all number combinations in this structure may be a valid credit card number. Thus, PII numbers usually have validation algorithms that can verify a value, such as the Luhn algorithm (Garfinkel, 2006).

Research Problem

Previous research surrounding forensic analysis of PII has primarily been designed for U.S. based information. There exists a need for verified techniques and tools to detect and validate PII from other countries. In New Zealand, IRD numbers can aid digital investigations involving financial information and attribution. Currently, there are no viable solutions available to detect and validate IRD numbers. This leads to practitioners having to develop their own tools that lack testing and verification. Implementation of techniques and tools must demonstrate design and evaluation to ensure that the resultant system produces reliable and credible results to aid forensic analysis.

Paper Structure

The aim of this paper is to present the design and implementation of a `bulk_extractor` scanner plug-in to detect and validate IRD numbers and a stoplist to filter irrelevant results. Experimental testing is conducted on a known data set to determine the functionality and capability of the implemented design. A selection of real world data sets are included in experimental testing to determine the effectiveness and efficiency of the system design using realistic investigation examples. Finally, a conclusion is presented and future research areas are suggested.

IRD SCANNER DESIGN AND IMPLEMENTATION

A new `bulk_extractor` scanner plug-in was developed to perform automated detection and validation of IRD numbers. This section outlines the development of an IRD number scanner, the output format of the IRD scanner and the creation of a stoplist to remove false positive results present in default operating system installations.

IRD Scanner Plug-in Development

The `bulk_extractor` tool was selected as the platform to aid tool development for the following reasons:

- Designed specifically for digital forensic analysis
- Supports a plug-in architecture to provide ease of software development
- Provides optimistic decompression of compressed data
- Open source software provides the ability to freely use, modify and redistribute

An IRD feature scanner plug-in (`scan_ird.flex`) was authored for `bulk_extractor` (version 1.4.0 revision 10884) using GNU Fast LEXical analyser (FLEX) programming language; a tool to generate C++ programs to perform complex textual search patterns. The IRD scanner has two main functions: *1)* Extract characters from the `bulk_extractor` stream buffer; and *2)* Validate the extracted number. Similar to U.S.

SSNs, IRD numbers do not have a standardised structure and may be encountered in a variety of patterns. Table 1 displays an overview of potential IRD number structures.

IRD Number Description	IRD Number Structure	Example
8 digits	NNNNNNNN	12345678
9 digits	NNNNNNNNN	123456789
8 digits with space delimiter	NN NNN NNN	12 345 678
8 digits with dash delimiter	NN-NNN-NNN	12-345-678
9 digits with space delimiter	NNN NNN NNN	123 465 789
9 digits with dash delimiter	NNN-NNN-NNN	123-456-789

Table 1: Overview of possible IRD number structures with examples (Inland Revenue, 2015).

The IRD scanner was designed and implemented to include the six potential IRD number structures displayed in Table 1. To perform automated detection of potentially valid IRD numbers, six FLEX search patterns were authored. Figure 1 displays an example of one IRD number scanner rule to detect an 8 digit number separated with a dash delimiter.

```
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 */
}
```

Figure 1: Example of a FLEX scanner rule to detect a potential IRD number.

As stated earlier, credit card numbers can be validated using the Luhn algorithm, also known as the modulus 10 algorithm. IRD numbers have a very similar validation model which uses the modulus 11 algorithm (Inland Revenue, 2015). Furthermore, an IRD number must be within the following range: 10,000,000 and 150,000,000. This results in 140 million possible numbers of which approximately only 13.8 million are valid. Using the validation algorithm, an 8 or 9 digit number can be processed, validated and included in scanner results. Any 8 or 9 digit number that does not validate is discarded. Figure 2 displays an overview of the IRD number validation process implemented in the scanner plug-in.

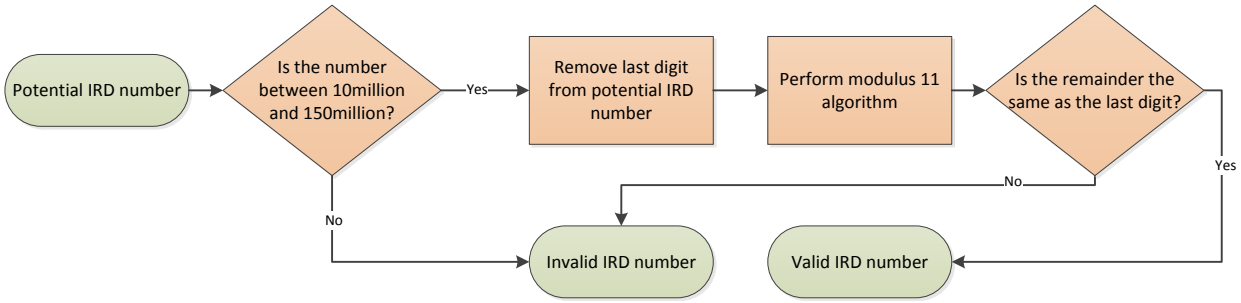

Figure 2: Overview of IRD number validation technique (Inland Revenue, 2015).

IRD Scanner Plug-in Output

Once the target data set has been processed the `bulk_extractor` tool produces three different output files for the IRD scanner: *1)* A feature file; *2)* A histogram file; and *3)* A stoplist feature file. According to Bradley & Garfinkel (2015) the feature file is a tab-delimited text file that contains the offset where the feature was found, the feature itself (in this case the IRD number) and a configurable number of bytes that precede and follow the feature (referred to as the feature context). Figure 3 displays a snippet of a populated IRD feature 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: {1:16 b:a04b76264279d00118000000cc02
107618283    22-220-616   N<BR>IRD no.<BR>22-220-616 <BR>Tax Code \x0D\x0A
```

Figure 3: Example of a feature file output from the IRD scanner.

A stoplist feature file is presented in the same format and is populated with IRD numbers that have been deemed irrelevant as they appear in the specified stoplist. Documentation of excluded features is an important function to have later access to if required.

Stoplist Implementation

Baier & Breitinger (2011) state that blacklists are a document or database that contain known content which are used to filter irrelevant results. For example, a blacklist may contain a collection of known illicit material that is represented by file names and cryptographic hash values. A `bulk_extractor` stoplist operates on a very similar principle. The `bulk_extractor` tool implements a novel blacklisting technique called context-sensitive stoplists which only filter a feature if it appears in exactly the same context of a known operating system file (Garfinkel, 2013). Stoplists are important as they perform data reduction, therefore, decreasing the amount of information that requires further analysis.

A stoplist was generated by performing a fresh default installation of various Microsoft Windows operating systems in a Virtual Machine (VM) testing environment. Multiple VMs were created using a variety of operating systems installed with default options. The VM disk was forensically imaged using `FTK Imager` and then processed using the IRD scanner. This resulted in a feature file of known IRD numbers for each default operating system install. All feature files were processed and the feature (IRD number) and feature context extracted and then populated in a master stoplist (`ird_stoplist.txt`). The stoplist can be included when running `bulk_extractor` by specifying the -w argument and the stoplist file name. The following Windows operating systems were selected for stoplist generation (all systems were 32-bit unless stated):

- MSDOS622
- Windows 3.1
- Windows 95
- Windows 98
- Windows ME
- Windows 2000
- Windows NT 4.0
- Windows XP (32 bit)
- Windows XP (64 bit)
- Windows Server 2003
- Windows Vista
- Windows 7 (32 bit)
- Windows 7 (64 bit)
- Windows 8
- Windows 8.1
- Windows 10

Post-processing IRD Scanner Output

Post-processing involves performing additional forensic analysis of the IRD scanner output. The `bulk_extractor` tool implements stream-based forensic analysis and therefore, provides no information regarding file system content. A framework was developed to perform a variety of post-processing tasks to help identify data files that contain validated and forensically interesting IRD numbers. The implemented framework is comprised of the following phases:

- Determine the data file associated with each detected and validated feature
- For each feature, search the feature context using keywords
- Extract files that contain search hits

Determining the data files associated with the identified IRD numbers was achieved by processing the IRD feature file using the `identify_filenames.py` script. This process produces an *annotated* feature file which appends the logical file system location for each feature. A post-processing script (`ird_search_context.py`) was authored to parse and search the annotated feature file. Each feature context was extracted and a keyword search performed to identify potentially interesting IRD numbers. The following keywords were used: *1)* IRD; and *2)* GST. The 'GST' keyword was included because a company's IRD

number is the same as that used for GST purposes. For each feature with a matching keyword, the feature was extracted for an investigator to perform manual analysis.

KNOWN DATA SET TESTING

The first data set used for experimental testing was an authored realistic computer system with known content. The data set was populated with a variety of common file types populated with known valid IRD numbers. This section discusses the data set generation method and results obtained.

Data Set Generation

The creation of a known data set was accomplished by implementing a Virtual Machine (VM) testing environment using the `VirtualBox` software. The VM was installed with Microsoft Windows 7 32-bit using default installation options. `VirtualBox` Guest Additions were installed and a shared folder was created to copy data to the guest system. A forensic image of the VM was collected using `FTK Imager` and then used to create an IRD stoplist using the IRD scanner and implementing the resultant feature file as a stoplist.

A collection of common document formats (DOC, DOCX, XLS, XLSX, RTF and PDF) were authored using a variety of Microsoft Office versions on Windows and Apple OS X. A selection of text encodings and PDF printing methods were implemented to ensure scanner effectiveness. A collection of folders and archive files were also created. This resulted in a total of 24 files. Each file created had a valid IRD number inserted to provide ground truth data. The known data was copied to the VM by using the shared folder functionality. To ensure the data files were copied correctly a Message Digest version 5 (MD5) hash value was calculated before and after transfer. The final stage in data set generation was to create a forensic image of the VM using `FTK Imager`.

Known Data Set Results

The known data set was processed using the IRD scanner, firstly without the stoplist and then together with the stoplist. The known data set was also processed using the `strings` and `grep` utilities, at first without any validation and then with IRD number validation. Table 2 displays an overview of results from known data set testing for the four different forensic analysis approaches.

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	0	0%
Strings and grep with validation	10,904	N/A	N/A
IRD Scanner with blacklist	62	21,554	99.71%

Table 2: IRD Scanner VS Strings and Grep: Overview of IRD number structures found in the known data set.

A very high number of potential IRD numbers were found by `strings` and `grep` (approximately 300,000) as this method did not perform IRD number validation. In comparison, the IRD scanner without a stoplist discovered approximately 21,000 validated IRD numbers. This result demonstrates the effectiveness of IRD number validation. Testing with `strings` and `grep` including IRD number validation detected approximately 11,000 validated IRD numbers. This was lower than the IRD scanner (with validation) due to the capability of the `bulk_extractor` tool to decompress data during processing, thus, resulting in more search hits. The `strings` and `grep` tools cannot perform any decompression. Finally, the IRD scanner with a stoplist proved the most effective technique which discovered a total of 62 validated IRD numbers. The stoplist proved very effective by removing 99.71% of potential IRD numbers. These IRD numbers were filtered as they are known to reside in operating system files from a default installation. Of the 62 identified numbers, 24 were true positives from the known data set. The remainder (38 IRD numbers) were manually classified as false positives. In summary, the results from the IRD scanner with an authored stoplist proved effective at identifying the correct IRD numbers from the known data set while dramatically reducing false positive matches.

REAL WORLD DATA SET TESTING

Real world data testing involved evaluating the IRD scanner on a variety of hard drives purchased on the second-hand market. This section presents an overview of the real world data set, experimental testing results using the authored IRD scanner and findings achieved by performing post-processing of scanner output.

Data Set Overview

Real world data set testing for computer forensic tool development and testing is advantageous due to data diversity and unpredictability, thereby, providing more robust research findings (Garfinkel et al., 2009). The real world data set used for this testing was from second-hand hard drives primarily sourced from the New Zealand TradeMe auction website. This provided a useful testing data set as the hard drives would most likely have NZ IRD numbers. A total of 77 hard drives were sourced from Roberts (2013) and an additional 75 were purchased for this research. Attempts were made to forensically image all of the 152 hard drives. However, only 122 were able to be read, resulting in a total of 122 forensic images applicable for real world data set testing.

Real World Data Set Results

The total of 122 forensic images were processed using the `bulk_extractor` tool and specifying the IRD scanner plug-in. Processing revealed that only 92 of the forensic images had data, the remainder (30 forensic images) had previously been sanitised (overwritten with zeroes). The 92 forensic images equated approximately 3.29 Terabytes of raw data. Table 3 displays a total count of the detected features (IRD numbers), a total count of features filtering using the stoplist and the percentage of stopped features.

IRD Number Description	Total Features	Relevant Features	Stopped Features	Percentage Stopped
8 digits	8,625,477	8,315,957	309,520	3.59%
9 digits	7,590,536	6,663,180	927,356	12.22%
8 digits with dash delimiter	148,032	148,012	20	0.01%
8 digits with space delimiter	242,043	238,882	3,161	1.31%
9 digits with dash delimiter	4,687	2,359	2,328	49.67%
9 digits with space delimiter	83,090	82,651	439	0.53%
Total	16,693,865	15,451,041	1,242,824	7.44%

Table 3: Overview of IRD number structures found in the real world data set.

The results obtained from real world testing illustrate the variability of tool performance on real data. Even with a master stoplist comprised of 16 default operating system installations there was an overwhelming number of detected and validated IRD numbers; a total of approximately 17 million. This is a formidable number of search hits for a practitioner to manually analyse. Approximately 1 million IRD numbers were filtered using the stoplist, this being approximately 7% of all detected features. However, the stoplist did prove more effective on a small number of hard drives, filtering over 60% of all detected IRD numbers on six different hard drives. Further processing is essential to identify IRD numbers of interest and to reduce manual analysis.

Post-processing was thus performed on the results from the IRD scanner in an attempt to identify true positive IRD numbers of forensic interest. The post-processing framework was executed on each forensic image, specifically the feature file output (see Figure 3 for an example). The feature context, the data immediately surrounding the detected IRD number (default of 16 bytes either side), was searched using 'IRD' and 'GST' as keywords. The technique processes previously detected and validated IRD numbers and attempts to find IRD numbers with a recognisable prefix (e.g., IRD Number: NNN-NNN-NNN). Post-processing resulted in a dramatic reduction in IRD numbers deemed to be relevant. Table 4 displays a summary of search hits found. It includes the total number of keyword hits and whether the IRD number was found in an allocated file or unallocated space.

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

Table 4: Overview of feature context search results from the real world data set.

The post-processing keyword search proved effective in identifying IRD numbers of potential interest. A total of 5,700 IRD numbers were identified. Of these, a total of 792 allocated files were identified and extracted including: Microsoft Word and Excel documents, PDF files, Lotus Notes database files, CSV files, Outlook Express files and Exchange Server EDB files. The post-processing searching technique found IRD numbers

stored in all six specified number structures (see Table 1). An important finding was that a wide variety of IRD number prefixes were discovered, as illustrated in the following examples identified from testing output:

- IRD
- IRD No
- IRD No is
- GST
- GST Number
- GST No
- GST#
- GST REG No

The prefix examples above were also found in a variety of conventions including: all upper-case, all lower-case, sentence case, with or without colons and a variety of spacing between the keyword and actual IRD number. The variety of prefix conventions make incorporating potential IRD number prefixes into the scanner exceptionally difficult. However, this highlights the effectiveness and flexibility of performing keyword searching on previously validated IRD numbers using feature context searching.

CONCLUSION

This research has contributed to the forensic analysis techniques used to detect and validate IRD numbers. A system was designed and implemented by authoring a scanner plug-in for the `bulk_extractor` tool. The IRD scanner was evaluated against known content and real world data sets to determine effectiveness. A novel post-processing search technique was implemented by searching the IRD number context (data before and after the detected and validated IRD number) which proved effective at performing data reduction of irrelevant results, reporting only forensically interesting results to the investigator.

Forensic analysis and detection of PII remains an active research area. Additional research and testing to detect and validate other important personal information such as bank account numbers, driver's license numbers, passport numbers and telephone numbers would prove useful to digital forensic practitioners in countries where techniques and tools are limited and not verified.

Resource Availability

To aid future research and development the IRD scanner plug-in, the master stoplist and post-processing framework have been made available and hosted on the authors GitHub repository: https://github.com/geehe732

REFERENCES

Altheide, C. (2004). Forensic analysis of windows hosts using UNIX-based tools. Digital Investigation, 1(3), 197-212.

Altheide, C., & Carvey, H. (2011). Digital forensics with open source tools: Using open source platform tools for performing computer forensics on target systems. Elsevier.

Baier, H., & Breitinger, F. (2011). Security aspects of piecewise hashing in computer forensics. In Sixth International Conference on IT Security, Incident Management and IT Forensics (p. 21-36).

Beebe, N., & Clark, J. (2007). Digital forensic text string searching: Improving information retrieval effectiveness by thematically clustering search results. Digital Investigation, 4(Supplement), 49-54.

Bradley, J., & Garfinkel, S. (2015). Bulk extractor 1.4 user's manual. Retrieved 9th August, 2015, from http://digitalcorpora.org/downloads/bulk_extractor/BEUsersManual.pdf

Cornell University. (2014). IT: About identity finder. Retrieved 9th August, 2015, from http://www.it.cornell.edu/services/idfinder/

GAO. (2008). GAO Report 08-536 - Information Security: Protecting personally identifiable information (Tech. Rep.).

Garfinkel, S. (2006). Forensic feature extraction and cross-drive analysis. Digital Investigation, 3(Supplement), 71-81.

Garfinkel, S., Farrel, P., Roussev, V., Dinolt, G. (2009). Bringing science to digital forensics with standardized forensic corpora. Digital Investigation, 6(Supplement), S2-S11.

Garfinkel, S. (2010). Digital forensics research: The next 10 years. Digital Investigation, 7 (Supplement), S64-S73.

Garfinkel, S. (2013). Digital media triage with bulk data analysis and bulk extractor. Computers & Security, 32, 56-72.

Guidance Software. (2011). EnCase Forensic Version 7.02: User's Guide. Guidance Software.

Inland Revenue. (2015). Non-Resident Withholding Tax and Resident Withholding Tax Specification Document. Retrieved 28th September, 2015 from http://www.ird.govt.nz/resources/2/a/2ab4a5b0-43d5-4ebd-b0d1-9b5958737b25/rwt-nrwt-spec-2015.pdf

McCallister, E., Grance, T., & Scarfone, K. (2010). SP 800-122. Guide to protecting the confidentiality of Personally Identifiable Information (PII) (Tech. Rep.). Gaithersburg, MD, United States.

Roberts, D. (2013). Data Remanence in New Zealand. (Thesis, Doctor of Philosophy). University of Otago. Retrieved 9th August from http://hdl.handle.net/10523/3768

Quick, D., & Choo, K. (2014). Impacts of increasing volume of digital forensic data: A survey and future research challenges. Digital Investigation, 11 (4), 273-294.