

bulk_extractor: A Stream-Based Forensics Tool

Simson L. Garfinkel Associate Professor, Naval Postgraduate School June 14, 2011 http://afflib.org/

NPS is the Navy's Research University.

Location: Monterey, CA

Students: 1500

- US Military (All 5 services)
- US Civilian (Scholarship for Service & SMART)
- Foreign Military (30 countries)
- All students are fully funded

Schools:

- Business & Public Policy
- Engineering & Applied Sciences
- Operational & Information Sciences
- International Graduate Studies

NCR Initiative:

- 8 offices on 5th floor, 900N Glebe Road, Arlington
- FY12 plans: 4 professors, 2 postdocs
- IMMEDIATE OPENINGS FOR RESEARCHERS
- IMMEDIATE SLOTS FOR .GOV PHDs!





Current NPS research thrusts in computer forensics

Area #1: End-to-end automation of forensic processing

- Digital Forensics XML Toolkit
- Disk Image -> Power Point

Area #2: Bulk Data Analysis

- Statistical techniques (sub-linear algorithms)
- Similarity Metrics

Area #3: Data mining for digital forensics

Automated social network analysis (cross-drive analysis)



Area #4: Creating Standardized Forensic Corpora

Freely redistributable disk and memory images, packet dumps, file collections.







Stream-based forensics with bulk_extractor

Stream-Based Disk Forensics: Scan the disk from beginning to end; do your best.



- 1. Read all of the blocks in order.
- 2. Look for information that might be useful.
- 3. Identify & extract what's possible in a single pass.

No disk seeking.

Potential to read and process at disk's maximum transfer rate.

Potential for intermediate answers.

Reads all the data — allocated files, deleted files, file fragments.

Separate metadata extraction required to get the file names.





Primary Disadvantage: Completeness



Fragmented files won't be recovered:

- Compressed files with part2-part1 ordering (possibly .docx)
- Files with internal fragmentation (.doc but not .docx)

Fortunately, most files are not fragmented.

Individual components of a ZIP file can be fragmented.

Most files that *are* fragmented have carvable internal structure:

Log files, Outlook PST files, etc.



This talk describes bulk_extractor, a tool for performing stream-based forensics.

Why you should care: a bulk_extractor success story



History of bulk_extractor

Internal design

Suppressing false positives with context sensitive stop lists.

Extending bulk_extractor with plug-ins

Future Plans







San Luis Obispo is "the happiest place in America"

Watch the video to find out why National Geographic named San Luis Obispo the top spot.

http://www.sanluisobispovacations.com/



A bulk_extractor Success Story

City of San Luis Obispo Police Department, Spring 2010

District Attorney filed charges against two individuals:

- Credit Card Fraud
- Possession of materials to commit credit card fraud.



- Arrested with a computer.
- Expected to argue that defends were unsophisticated and lacked knowledge.

Examiner given 250GiB drive the day before preliminary hearing.

Typically, it would take several days to conduct a proper forensic investigation.





bulk_extractor found actionable evidence in 2.5 hours!

Examiner given 250GiB drive the day before preliminary hearing.



Bulk_extractor found:

- Over 10,000 credit card numbers on the HD (1000 unique)
- Most common email address belonged to the primary defendant (possession)
- The most commonly occurring Internet search engine queries concerned credit card fraud and bank identification numbers (intent)
- Most commonly visited websites were in a foreign country whose primary language is spoken fluently by the primary defendant.

Armed with this data, the DA was able to have the defendants held.



Faster than conventional tools. Finds data that other tools miss.

Runs 2-10 times faster than EnCase or FTK on the same hardware.

• bulk_extractor is multi-threaded; EnCase 6.x and FTK 3.x have little threading.

Finds stuff others miss.

- "Optimistically" decompresses and re-analyzes all data.
- Finds data in browser caches (downloaded with zip/gzip), and in many file formats.

Presents the data in an easy-to-understand report.

- Produces "histogram" of email addresses, credit card numbers, etc.
- Distinguishes primary user from incidental users.







History of bulk_extractor

bulk_extractor: 20 years in the making!

In 1991 I developed SBook, a free-format address book.



SBook used "Named Entity Recognition" to find addresses, phone numbers, email addresses while you typed.



Today we call this technology Named Entity Recognition

SBook's technology was based on:

- Regular expressions executed in parallel
 - -US, European, & Asian Phone Numbers
 - Email Addresses
 - -URLs
- A gazette with more than 10,000 names:
 - -Common "Company" names
 - -Common "Person" names
 - -Every country, state, and major US city
- Hand-tuned weights and additional rules.

Implementation:

- 2500 lines of GNU flex, C++
- 50 msec to evaluate 20 lines of ASCII text.
 - -Running on a 25Mhz 68030 with 32MB of RAM!

😝 😁 💮 🔛 demo
Q < > + Auto 🛟
Saved
Simson L. Garfinkel
MIT Computer Science and Artificial Intelligence La
Simson's mac.com account
Simson L. Garfinkel
 MIT Computer Science and Artificial Intelligence Laboratory 77 Massachusetts Ave Cambridge, MA 02139 617-253-1000 http://www.sbook5.com/ Creator of SBook® Al-based Address Book Software
Created Nov 24 22:20 2003 by simsong



In 2003, I bought 200 used hard drives

The goal was to find drives that had not been properly sanitized.

First strategy:

- DD all of the disks to image files
- run strings to extract printable strings.
- grep to scan for email, CCN, etc.
 - VERY SLOW!!!!
 - -HARD TO MODIFY!

Second strategy:

- Use SBook technology!
- Read disk 1MB at a time
- Pass the raw disk sectors to flex-based scanner.
- Big surprise: scanner didn't crash!





Simple flex-based scanners required substantial post-processing to be useful

Techniques include:

- Additional validation beyond regular expressions (CCN Luhn algorithm, etc).
- Examination of feature "neighborhood" to eliminate common false positives.



The technique worked well to find drives with sensitive information.



Between 2005 and 2008, we interviewed law enforcement regarding their use of forensic tools.

Law enforcement officers wanted a *highly automated* tool for finding:

- Email addresses
- Credit card numbers (including track 2 information)
- Search terms (extracted from URLs)
- Phone numbers
- GPS coordinates
- EXIF information from JPEGs
- All words that were present on the disk (for password cracking)

The tool had to:

- Run on Windows, Linux, and Mac-based systems
- Run with no user interaction
- Operate on raw disk images, split-raw volumes, E01 files, and AFF files
- Allow user to provide additional regular expressions for searches
- Automatically extract features from compressed data such as gzip-compressed HTTP
- Run at maximum I/O speed of physical drive
- Never crash



Starting in 2008, we made a series of limited releases. Today we are releasing bulk_extractor 1.0.0

- January 2008 Created Subversion Repository
- April 2010 Initial public release 0.1.0
- May 2010 Initial multi-threading release 0.3.0

-Each thread runs in its own process

- Sept. 2010 Stop lists 0.4.0
- Oct. 2010 Context-based stop-lists 0.5.0
- Dec. 2010 Switch to POSIX-based threads 0.6.0
- Dec. 2010 Support for WIndows HIBERFIL.SYS decompression 0.7.0
- Jun. 2010 First 1.0.0 Release (TODAY)

Tool capabilities result from substantial testing and user feedback. Moving technology from the lab to the field has been challenging:

- Must work with evidence files of any size and on limited hardware.
- Users can't provide their data when the program crashes.
- Users are *analysts* and *examiners*, not engineers.







Inside bulk_extractor

bulk_extractor: architectural overview

Written in C, C++ and GNU flex

- Command-line tool.
- Linux, MacOS, Windows (compiled with mingw)

Key Features:

- "Scanners" look for information of interest in typical investigations.
- Recursively re-analyzes compressed data.
- Results stored in "feature files"
- Multi-threaded

Java GUI

Runs command-line tool and views results



bulk_extractor extracts "features" from disk images.





bulk_extractor: system diagram





image processing C++ iterator handles disks, images and files

Works with multiple disk formats.

- E01
- AFF
- raw
- split raw
- individual disk files

```
Produces sbuf_t object:
```

```
class buf_t {
    ...
public:;
    uint8_t *buf; /* data! */
    pos0_t pos0; /* forensic path */
    size_t bufsize;
    size_t pagesize;
    ...
};
```



Evidence

We chop the 1TB disk into 65,536 x 16MiB "pages" for processing.

The "pages" overlap to avoid dropping features that cross buffer boundaries.

The overlap area is called the margin.

- Each sbuf can be processed in parallel they don't depend on each other.
- Features start in the page but end in the margin are *reported*.
- Features that start in the margin are *ignored* (we get them later)
 - -Assumes that the feature size is smaller than the margin size.



Entire system is automatic:

- Image_process iterator makes sbuf_t buffers.
- Each buffer is processed by every scanner
- Features are automatically combined.



Scanners process an sbuf and extract features

scan_email is the email scanner.

inputs: sbuf objects





The *feature recording system* saves features to disk.

Feature Recorder objects store the features.

- Scanners are given a (feature_recorder *) pointer
- Feature recorders are *thread safe*.



Features are stored in a feature file:

48198832	<u>domexuser2@gmail.com</u>	<pre>tocol><name>domexuser2@gmail.com/Home</name></pre>
48200361	<u>domexuser2@live.com</u>	<pre>tocol><name>domexuser2@live.com</name><pass< pre=""></pass<></pre>
48413829	<pre>siege@preoccupied.net</pre>	siege) O'Brien < <u>siege@preoccupied.net</u> >_ <u>hp://meanwhi</u>
48481542	<u>danilo@gnome.org</u>	Daniloegan < <u>danilo@gnome.org</u> >_Language-Team:
48481589	gnom@prevod.org	: Serbian (sr) < <u>gnom@prevod.org</u> >_MIME-Version:
49421069	<u>domexuser1@gmail.com</u>	<pre>server2.name", "domexuser1@gmail.com");user_pref("</pre>
49421279	<u>domexuser1@gmail.com</u>	er2.userName", " <u>domexuser1@gmail.com</u> ");user_pref("
49421608	<u>domexuser1@gmail.com</u>	tp1.username", " <u>domexuser1@gmail.com</u> ");user_pref("



offset

feature in evidence context

Histograms are a powerful tool for understanding evidence.

Email histogram allows us to rapidly determine:

- Drive's primary user
- User's organization
- Primary correspondents
- Other email addresses



Drive #51 (Anonymized)

8133
3504
2956
2108
1579
1206
1118
1030
989
960
947
845
802
790
763



The feature recording system *automatically* makes historgrams.

Simple histogram based on feature:

- n=579 <u>domexuser1@gmail.com</u>
- n=432 <u>domexuser2@gmail.com</u>
- n=340 <u>domexuser3@gmail.com</u>
- n=268 <u>ips@mail.ips.es</u>
- n=252 premium-server@thawte.com
- n=244 <u>CPS-requests@verisign.com</u>
- n=242 <u>someone@example.com</u>

Based on regular expression extraction:

- For example, extract search terms with .*search.*q=(.*)
 - n=18 pidgin hotmail+thunderbird n=10 n=3 Grey+Gardens+cousins dvd n=3 n=2 %TERMS% n=2 cache: n=2 p pi n=2 pid n=2
 - n=1 Abolish+income+tax
 - n=1 Brad+and+Angelina+nanny+help
 - n=1 Build+Windmill
 - n=1 Carol+Alt





bulk_extractor has *multiple* feature extractors. Each scanner runs in order. (Order doesn't matter.)

Scanners can be turned on or off

- Useful for debugging.
- AES key scanner is very slow (off by default)

Some scanners are recursive.

- e.g. scan_zip will find zlib-compressed regions
- An sbuf is made for the decompressed data
- The data is re-analyzed by the other scanners
 - This finds email addresses in compressed data!

Recursion used for:

- Decompressing ZLIB, Windows HIBERFILE,
- Extracting text from PDFs
- Handling compressed browser cache data



SBUFs



Recursion requires a *new way* to describe offsets. bulk_extractor introduces the "forensic path."

Consider an HTTP stream that contains a GZIP-compressed email:



We can represent this as:

11052168704-GZIP-3437	live.com
11052168704-GZIP-3475	live.com
11052168704-GZIP-3512	live.com

eMn='domexuser1@live.com';var srf_sDispM
pMn='domexuser1@live.com';var srf_sPreCk
eCk='domexuser1@live.com';var srf_sFT='<</pre>



GUI: 100% Java Launches bulk_extractor; views results

Uses bulk_extractor to decode forensic path





Crash Protection

Every forensic tool crashes.

- Tools routinely used with data fragments, non-standard codings, etc.
- Evidence that makes the tool crash typically cannot be shared with the developer.

Crash Protection: checkpointing!

- Bulk_extractor checkpoints current page in the file config.cfg
- After a crash, just hit up-arrow and return; bulk_extractor restarts at next page.



Integrated design, but compact. 2726 lines of code; 33 seconds to compile on an i5











Suppressing False Positives

Modern operating systems are *filled* with email addresses.

Sources:

- Windows binaries
- SSL certificates
- Sample documents

- n=579 <u>domexuser1@gmail.com</u>
- n=432 <u>domexuser2@gmail.com</u>
- n=340 <u>domexuser3@gmail.com</u>
- n=268 <u>ips@mail.ips.es</u>
- n=252 premium-server@thawte.com
- n=244 <u>CPS-requests@verisign.com</u>
- n=242 <u>someone@example.com</u>

It's important to suppress email addresses not relevant to the case.

Approach #1 — Suppress emails seen on many other drives. Approach #2 — Stop list from bulk_extractor run on clean installs.

Both of these methods stop list commonly seen emails.

- Operating Systems have a LOT of emails. (FC12 has 20,584!)
- Problem: this approach gives Linux developers a free pass!



Approach #3: Context-sensitive stop list.

Instead of a stop list of features, use features+context:

- Offset: **351373329**
- Email: <u>zeeshan.ali@nokia.com</u>
- Context: ut_Zeeshan Ali <<u>zeeshan.ali@nokia.com</u>>, Stefan Kost <</p>
- Offset: **351373366**
- Email: <u>stefan.kost@nokia.com</u>
- Context: >, Stefan Kost <<u>stefan.kost@nokia.com</u>>_____sin

-Here "context" is 8 characters on either side of feature.

-We put the feature+context in the stop list.

The "Stop List" entry is the feature+context.

- This ignores Linux developer email address in Linux binaries.
- The email address is reported if it appears in a different context.



We created a context-sensitive stop list for Microsoft Windows XP, 2000, 2003, Vista, and several Linux.

Total stop list: 70MB (628,792 features; 9MB ZIP file)

Sample from the stop list:

tzigkeit <gord@gnu.ai.mit.edu> * tests/demo tzigkeit <gord@gnu.ai.mit.edu> Reported by u-emacs-request@prep.ai.mit.edu (or the corresp u:/pub/rtfm/" "/ftp@rtfm.mit.edu:/pub/usenet/" " ub/rtfm/" "/ftp@rtfm.mit.edu:/pub/usenet/" " udson <<u>ghudson@mit.edu</u>>', "leftv" <u>uq-fortran-mode@erl.mit.edu</u> This list coll uke Mewburn <<u>lm@rmit.edu.au</u>>, 931222 AC ARG kit@expo.lcs.mit.edu */ #ifndef As um * kit@expo.lcs.mit.edu */ #ifndef A um kit@expo.lcs.mit.edu */ #ifndef S um *

sl3/fedora12-64/domain.txt sl3/fedora12-64/domain.txt sl3/redhat54-ent-64/domain.txt sl3/redhat54-ent-64/email.txt sl3/redhat54-ent-64/domain.txt sl3/redhat54-ent-64/domain.txt sl3/redhat54-ent-64/domain.txt sl3/fedora12-64/domain.txt sl3/redhat54-ent-64/email.txt sl3/redhat54-ent-64/email.txt



The context-sensitive stop list prunes the OS-supplied features.

Applying it to domexusers HD image:

• # of emails found: $9143 \rightarrow 4459$

without stop list

- n=579 domexuser1@gmail.com n=432 domexuser2@gmail.com
- n=340 domexuser3@gmail.com
- n=268 ips@mail.ips.es
- n=252 premium-server@thawte.com
- n=244 CPS-requests@verisign.com
- n=242 someone@example.com
- n=237 inet@microsoft.com
- n=192 domexuser2@live.com
- n=153 domexuser2@hotmail.com
- n=146 domexuser1@hotmail.com
- n=134 domexuser1@live.com
- n=115 example@passport.com
- n=115 myname@msn.com
- ca@digsigtrust.com n=110

You can download the list today:

http://afflib.org/downloads/feature context.1.0.zip

with stop list

- domexuser1@qmail.com **n=579**
 - domexuser2@gmail.com n=432
 - n=340 domexuser3@gmail.com
 - n=192 domexuser2@live.com
 - n=153 domexuser2@hotmail.com
 - n=146 domexuser1@hotmail.com
 - n=134 domexuser1@live.com
 - n=91 premium-server@thawte.com
 - n=70 talkback@mozilla.org
 - n=69 hewitt@netscape.com
 - DOMEXUSER2@GMAIL.COM n=54
 - domexuser1%40gmail.com@imap.gmail.com n=48
 - domex2@rad.li n=42
 - n=39 lord@netscape.com
 - talkback@mozilla.org and other email n=37 49091023.6070302@gmail.com HellKUdlikker nrocont on the head of installed

auuresses were not eiiminated Decause they were present on the base OS installs.





Extending bulk_extractor with Plug-ins

Filenames can be added through post-processing.

bulk_extractor reports the *disk blocks* for each feature.



To get the file names, you need to map the disk block to a file.

- Make a map of the blocks in DFXML with fiwalk (<u>http://afflib.org/fiwalk</u>)
- Then use python/identify_filenames.py to create an annotated feature file.



bulk_diff.py: compare two different bulk_extractor reports

The "report" directory contains:

- DFXML file of bulk_extractor run information
- Multiple feature files.

bulk_diff.py: create a "difference report" of two bulk_extractor runs.

- Designed for timeline analysis.
- Developed with analysts.
- Reports "what's changed."
 - -Reporting "what's new" turned out to be more useful.
 - "what's missing" includes data inadvertently overwritten.





IP Carving and Network Reassembly plug-in

bulk_extractor extended to recognize and validate network data.

Automated extraction of Ethernet MAC addresses from IP packets in hibernation files.

We then re-create the physical networks the computers were on:





C++ programmers can write C++ plugins

Plugins are distributed as shared libraries.

- Windows: scan_bulk.DLL
- Mac & Linux: scan_bulk.so

Plugins must support a single function call:

- scanner_params Describes what the scanner should do.
 - *—sp.sbuf SBUF to scan*
 - *—sp.fs Feature recording set to use*
 - -sp.phase==0 initialize
 - -sp.phase==1 scan the SBUF in sp.sbuf
 - -sp.phase==2 shut down
- recursion_control_block Provides information for recursive calls.

The same plug in system will be used by a future version of fiwalk.

The same plug-in will be usable with multiple forensic tools.





bulk_extractor future

NPS

bulk_extractor is an open source program! You can help make it better.

Better handling of text:

- MIME decoding (e.g. user=40localhost should be user@localhost)
- Improved handling of Unicode.

More scanners

- RAR & RAR2
- LZMA
- BZIP2
- MSI & CAB
- NTFS
- VCARD

Reliability and conformance testing.

GET PAID TO WORK ON BULK_EXTRACTOR: ASK ME HOW!



In conclusion, bulk_extractor is a powerful stream-based forensic tool.

Bulk_extractor demonstrates the power of:

- Bulk data processing.
- Carving EVERYTHING
- Multi-threading (we can process data with 100% CPU utilization)

Bulk_extractor is 100% free software

- Public Domain (work of US Government)
- Please use the ideas in other programs!
 - -DFXML
 - -Job Distribution
 - -Forensic Path
 - -SBUF
- Let's keep the plug-in system consistent.
- Download from http://afflib.org/

Questions?

			1				
lications Processes	Services	Performance	Networking	Users			
PU Usage	CPU Usage	History					
100 %							
	ph						
Memory	Physical Me	emory Usage Hi	story				_
					و و و و و د		
2.04 GB							
Physical Memory (MB)		System					
Total	8103	Handles		19603			
Cached	1503	Threads		1017			
Available	6010	Processes	1	71			
ree	4543	Up Time	0:0	0:06:24			
		Commit (G	iB)	2/15			
(eroel Memory (MR)							
Kernel Memory (MB)	235		Resource Monitor				
Kernel Memory (MB) Paged	235		and the state of the second se				

