The state of bulk_extractor

Bit Curator Forum

Wednesday, June 17

11:45 a.m. - 1:45 p.m. ET

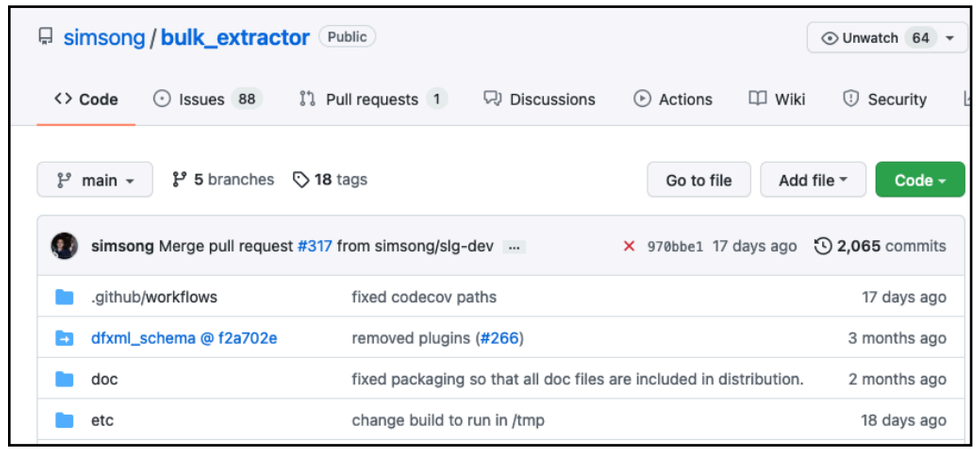
Simson Garfinkel, PhD*

These slides can be downloaded from https://simson.net/ref/2026/2026-06-15_bulk_extractor.pdf

bulk_extractor - A open source digital forensics tool

Version 1 Developed between 2003 and 2014

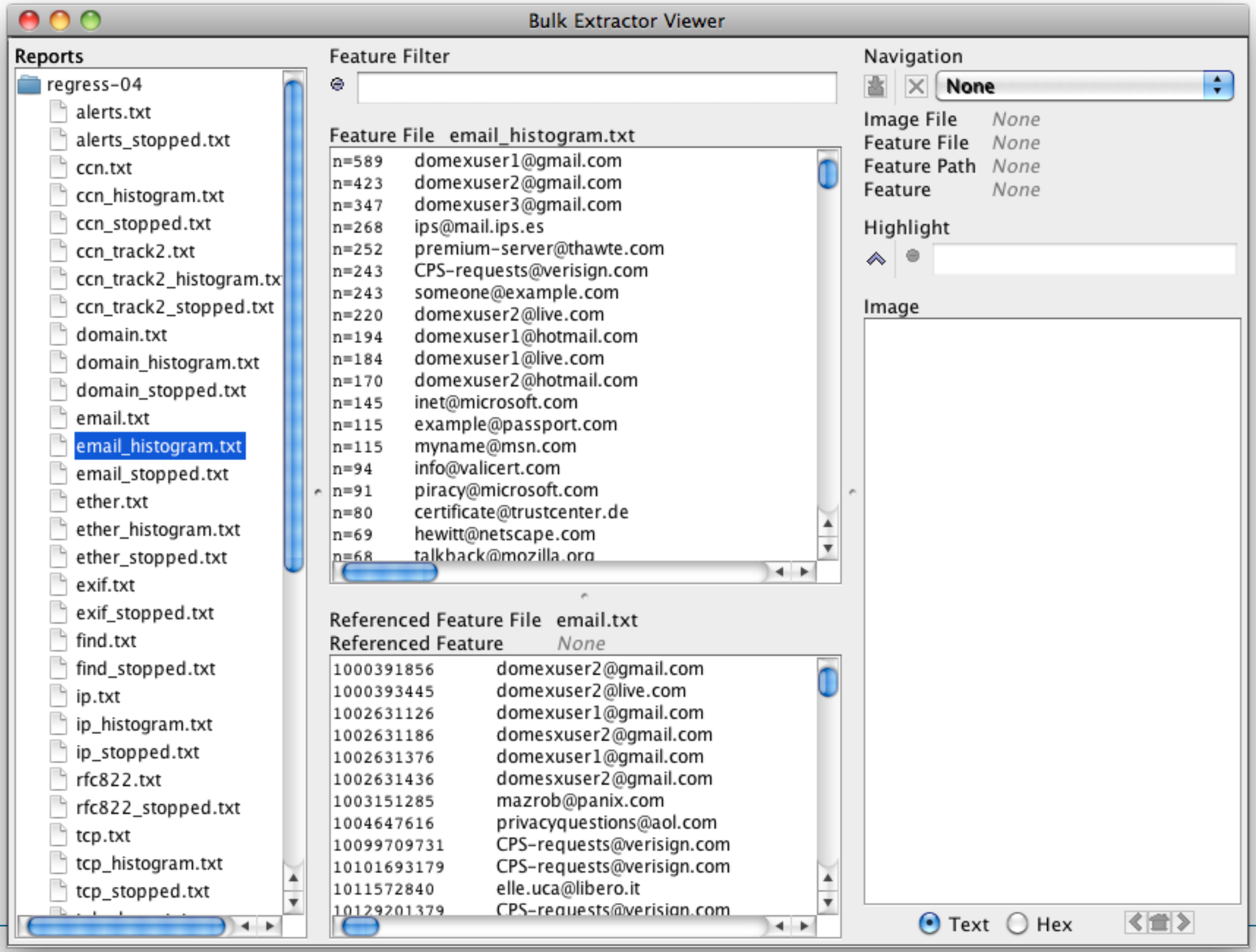
- Funded by US DOD & FBI
- command-line tool: ~ 59K lines of C++98
  - —*GUI: ~ 18K lines java*
  - —*Compiled with Autoconf toolchain*
- Runs on macOS, Linux and Windows
- Multi-threaded carving and identity “extraction” tool



Key features:

- “Scanners” look for information of interest in typical investigations.
- Recursively re-analyzes compressed data.
- Results stored in “feature files”
- Embedded in at least one commercial product
- User base: research, education, LE, defense, digital humanities

bulk_extractor legacy Java GUI



bulk_extractor — Key features for digital humanities

Strengths:

- Rapidly analyzes disk images or files in any format.
- Identifies email addresses, phone numbers, SSNs, credit card numbers, etc.
- Recursively analyzes compressed data
- Plug-in architecture

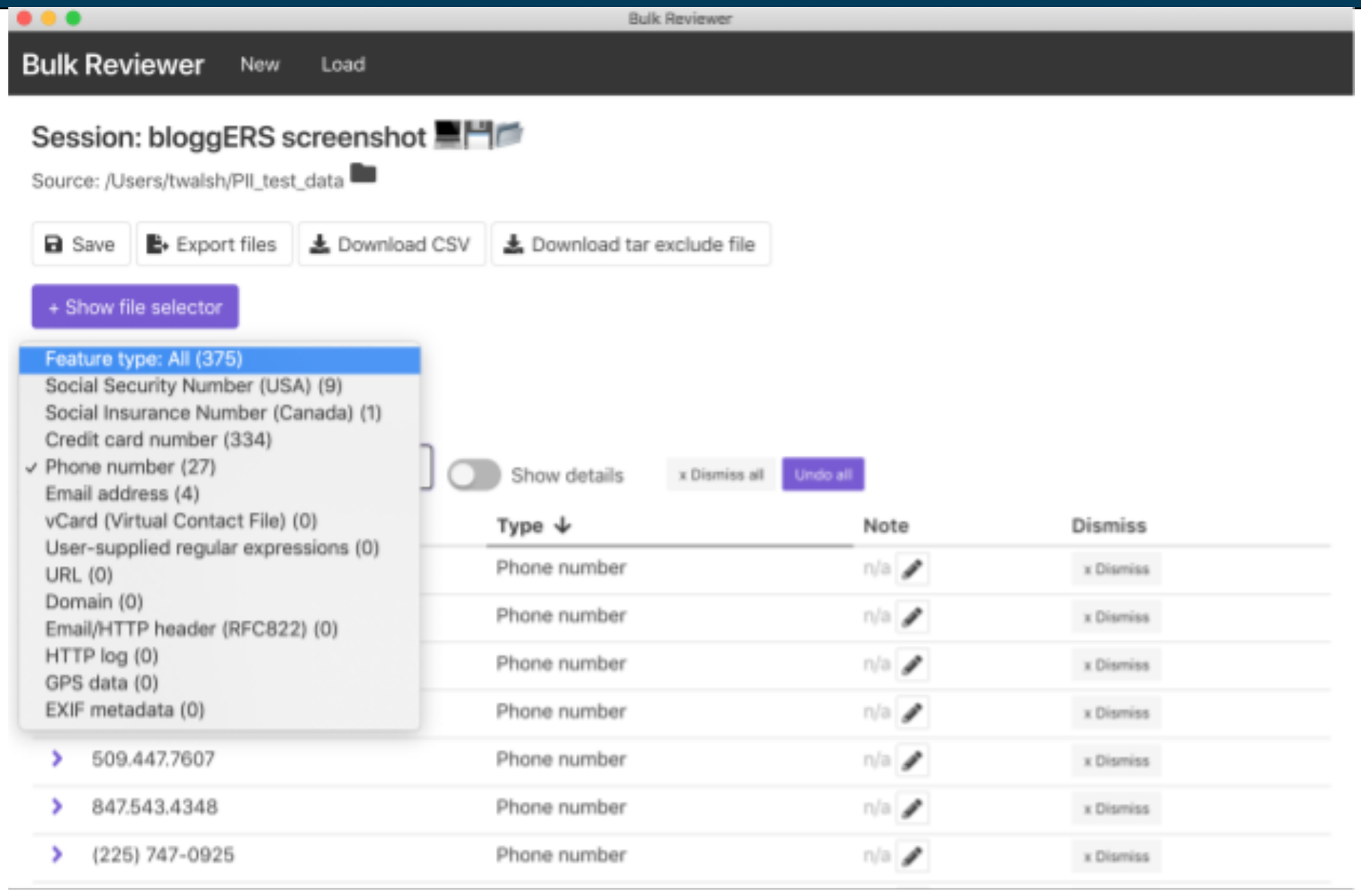
Weaknesses:

- Does not organize output
- Does not process file-by-file when processing a disk image

Integrations:

- brunnhilde — Can run bulk_extractor
- Bulk Reviewer — Tessa Walsh's user interface — electron — needs to be updated

Tessa Walsh's Bulk Reviewer…



https://saaers.wordpress.com/2020/05/26/spotlight-interview-with-tessa-walsh-artefactual/
https://github.com/bulk-reviewer/bulk-reviewer

Sharpening your tools: Updating bulk_extractor for the 2010s

Software quality improvements:

- C99 → C++20
- Added CI/CD.
- Increased coverage tests from 0% → 49%

Performance improvements:

- Increased parallelism
- Decreased memory requirements

practice

DOI:10.1145/3600098

Article development led by acmqueue
queue.acm.org

Updating `bulk_extractor` for the 2020s.

BY SIMSON GARFINKEL AND JON STEWART

Sharpening Your Tools

DIGITAL FORENSICS (DF) is a fast-moving field with a huge subject area. A digital investigator must be able to analyze "any data that might be found on any device anywhere on the planet."[12] As such, developers must continually update DF tools to address new file formats, new encoding schemes, and new ways that the subjects of investigations use their computers. At the same time, tools must retain the ability to analyze legacy data formats—all of them, in fact.

Most DF tools run on consumer desktop operating systems, adding another layer of complexity: These operating systems are also continually evolving. Analysts must update and upgrade their systems, lest they risk compromise by malware, which decreases productivity and can discredit an analysis in court. This is true even for workstations that are "air gapped" (not connected to the Internet), since malware in evidence can exploit bugs in forensic software.[19]

Surprisingly, open source forensic tools distributed as source code face a greater challenge when the underlying operating system is upgraded: Software compatibility layers typically emphasize compatibility for the application binary interface (ABI), not source code. Software compiled from source must cope with upgraded compilers, libraries, and new file locations. As a result, older open source software frequently does not run on modern systems without updating. One way around this problem is to run the old software inside a virtual machine—but older virtual machines won't be protected against modern malware threats.

One advantage of open source software is the end user has the source code and is therefore able to update the application (or pay for a programmer to update the application). In practice, many users of DF tools lack the expertise, financial resources, and time to update the collection of open source tools they rely upon to do their jobs. Instead, that task falls upon tool developers, who must simultaneously cope with essential changes in DF best practices as well as in operating systems, compilers, and libraries, while avoiding inadvertent changes to important functionality. Developers must also resist the urge for aggressive rewrites that add new expansive functionality, lest they succumb to the "second-system effect."[5]

This article presents our experience updating the high-performance DF tool BE (`bulk_extractor`)[16] a decade after its initial release. Between 2018 and 2022, we updated the program from C++98 to C++17. We also performed a complete code refactoring and adopted a unit test framework.

The new version typically runs with 75% more throughput than the previous version, attributable to improved multithreading. This article provides lessons and recommendations for other DF tool maintainers. All developers can benefit from the detailed discussion of how embracing features in the C++17 standard and modern software engineering practices can improve the correctness, reliability, and throughput of forensic software. Businesses and funding agencies can use this experience to help justify the substantial cost

IMAGE BY ZINE TRON

Clock time comparison of running BE1.6 and BE2 on a variety of hardware and software configurations.

All executables are compiled -O3. Times for the nps-2009-ubnist1 and nps-2009-domexusers are average of three runs. The nps-2009-ubnist1 and nps-2009-domexusers read and write to the system SSD, while the nps-2013-2tb reads from the system SSD and writes to an external USB3 HD due to storage considerations. BE1.6 speeds reported for runs with the standard 30 default scanners enabled: accts, aes, base64, elf, email, evtx, exif, find, gps, gzip, hiberfile, httplogs, json, kml, msxml, net, ntfsindx, ntfslogfile, ntfsmft, ntfsusn, pdf, rar, sqlite, utmp, vcard, windirs, winlnk, winpe, winprefetch and zip. BE2 for runs with the standard 29 scanners enabled (hiberfile is disabled) and with AES192 key searching disabled, and with the BE1.6 configuration that adds hiberfile and AES192 key searching. The Apple M1 Pro 10 core processor has 8 “performance” cores and 2 “efficiency” cores. “Throughput” is normalized to the speed of BE1.6 on the same hardware with the same disk image; a throughput of 200% means the disk image will be analyzed in half the time.

		Scanners				
		29			30 + AES192	
Computer	**Disk Image (+ config)**	**BE1.6**	**BE2**	**SPEEDUP**	**BE2**	**SPEEDUP**
MacBook Pro (Retina, 13 inch, late 2013)*	nps-2009-ubnist1	140s	109s	128%	120s	117%
	nps-2009-domexusers	1420s	837s	170%	1208s	118%
Mac mini (2018)**	nps-2009-ubnist1	43s	35s	123%	33s	130%
	nps-2009-domexusers	428s	319s	134%	428s	100%
MacBook Pro (16-inch, 2021)†	nps-2009-ubnist1	20s	16s	125%	17s	118%
	nps-2009-domexusers	221s	126s	175%	172s	128%
	nps-2013-2tb	20142s	10944s	184%	11184s	180%

* 2.8GHz Dual-core Intel Core i7; 16GB, 1600MHz DDR3; two physical cores (four with hyperthreading); macOS 11.6.3

** 3GHz 6-core i5; 2667MHz DDR4; macOS 12.1

† Apple M1 Pro 10 core; 32GB RAM; macOS 12.1

State of bulk_extractor 2026

Current “release” is 2.1.1

- Installed from source
- good support for JPEGs, document files, email messages, web browser caches, PDFs
- Pretty reliable; occasionally people discover crashes
- Little validation for exhaustiveness of collection.

Easy improvements:

- Make easier to install

Harder improvements:

- Natively handle new document types
- Better support for running on Windows

Critical issue:

- Project has no funding

Announcing the bulk_extractor icon contest!

We need an icon for bulk_extractor.

Please submit your icons as a “PR” on the bulk_extractor GitHub!

- close this issue: https://github.com/simsong/bulk_extractor/issues/6

https://github.com/simsong/bulk_extractor

First prize — a free copy of bulk_extractor!

Second prize — two free copies of bulk_extractor!

What else is needed from the user community…

Are there specific features that would be useful?

- Write your own modules in python?
- New compression types? (RAR? xz? 7zip?)
- SQLite3 output?
- Structured output?

Does we need better support for installing bulk_extractor?

- ‘apt install’
- ‘snap install’
- Microsoft Windows installer?

Are there people interested in supporting and maintaining bulk_extractor?