Foremost-NG: An Open-Source Toolkit for Advanced File Carving and Analysis

Cristian H. M. Souza[1,2] **, Daniel M. Batista**[1]

[1]Department of Computer Science, University of São Paulo
[2]Global Emergency Response Team (GERT), Kaspersky

cristianmsbr@gmail.com, batista@ime.usp.br

Abstract. *File carving and data recovery are critical tasks in digital forensics, incident response, and malware analysis. The original* `Foremost` *tool, while powerful, has long suffered the lack of updates and limited file formats. This paper introduces* `Foremost-NG`*, a community-driven fork that significantly refactors the core structure, adds new file-format parsers (including EVTX, script files, Mach-O, and ELF executables), and integrates with VirusTotal for on-the-fly threat intelligence. By modernizing the carving engine, decoupling format-specific handlers, and embedding an HTTP-based lookup for hash-based analysis,* `Foremost-NG` *delivers a robust, extensible platform for modern forensic workflows, as well as facilitates the inclusion of new file-formats.*

1. Introduction

In the rapidly evolving landscape of cybersecurity and digital investigations, the ability to recover and analyze deleted or hidden artifacts can make the difference between a successful incident response and an unresolvable case [Casino et al. 2022]. Recovered artifacts can also be used for tasks such as malware analysis, with the aim of improving detection mechanisms by understanding the malware's behavior [Souza et al. 2025].

Traditional file-carving tools, while indispensable, often struggle with newer file formats, complex binary structures, and the diversity of data types encountered on modern endpoints [Ramli et al. 2021]. The original `Foremost` project laid important groundwork, pioneering a signature-based carving engine that could recognize common file headers and reconstruct deleted files from raw disk images[1]. However, as new operating systems, executable formats, and log containers emerged, extending `Foremost` proved increasingly needed. At the same time, its codebase was proved to be tightly coupled, making it difficult for third-party developers to add support for formats such as Windows EVTX logs, Mach-O binaries, and ELF executables without risking regressions or lengthy integration cycles.

To the best of our knowledge, the last official release of `Foremost` (version 1.5.7) dates back to 2009 [2]. Since then, changes in the GNU C Library (`glibc`) have introduced compatibility issues that prevent the original source code from compiling on modern operating systems[3]. `Foremost-NG` was conceived to address these limitations by

[1]`https://foremost.sourceforge.net`
[2]`http://foremost.sourceforge.net/pkg/foremost-1.5.7.tar.gz`
[3]`https://github.com/korczis/foremost/issues/8`

refactoring and modernizing the tool's architecture from the ground up. `Foremost-NG` organizes the carving engine into a clear separation of concerns: a central dispatcher handles raw data ingestion and chunk management, while individual parser modules register themselves via a uniform interface. This decoupled design enables developers to add or remove format support simply by dropping a new parser into the source and updating a registration list. The result is a robust, maintainable codebase where growth is encouraged and regressions are minimized.

Beyond refactoring, `Foremost-NG` introduces first-class support for file types that until now were difficult to carve. Windows EVTX event logs (rich sources of forensic evidence) are now parsed natively, allowing investigators to extract and interpret event records directly from a raw partition or memory dump. Similarly, script files (shell scripts, Python, PowerShell) are identified by their characteristic shebangs and Unicode signatures. On macOS and Linux systems, `Foremost-NG` detects Mach-O and ELF headers, respectively, reconstructing binary executables even when fragmented across disk sectors. Each of these new modules not only recovers complete files but also verifies structural integrity, minimizing false-positive fragments.

Arguably the most notable new feature is the integration with VirusTotal's public API[4], which permits `Foremost-NG` to perform an on-demand lookup based on the cryptographic hashes (SHA-256, MD5) it computes for each recovered artifact. Within a few seconds of carving a file, the user gains insight into whether that artifact is known to threat actors, what tags or AV detections exist, and whether it merits deeper reverse engineering. This real-time enrichment transforms `Foremost-NG` from a simple file recovery utility into an intelligence-driven forensics assistant, guiding analysts toward high-priority items and reducing time wasted on benign fragments.

The design philosophy behind `Foremost-NG` is twofold: first, make it trivially easy for practitioners to extend support to future or proprietary file formats; second, tightly integrate external intelligence sources so that each carved object is automatically contextualized. In the sections that follow, we detail the revised architecture, enumerate the supported formats and their parsing strategies, and explain how the VirusTotal integration operates under the hood . Our goal is to show that `Foremost-NG` not only inherits the best aspects of the original carving engine but also decisively moves the needle toward an extensible, intelligence-ready forensics toolkit.

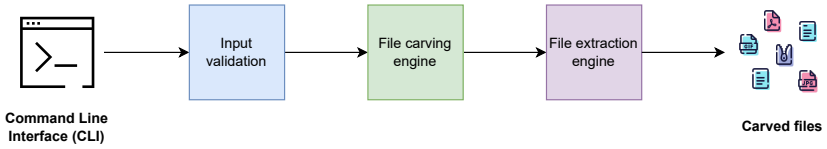
The remainder of this paper is organized as follows. Section 2 presents the architecture and implementation of `Foremost-NG`. Section 3 describes each supported format, including EVTX, scripting files, Mach-O, ELF, and legacy image/document types, and explains how parsers are structured. Section 4 covers installation procedures, configuration of the VirusTotal lookup, and common command-line workflows. Finally, Section 5 summarizes the contributions of `Foremost-NG` and outlines directions for future enhancements.

2. Architecture and Implementation

Figure 1 shows an overview of the `Foremost-NG` architecture, composed by an input validation module, the carving engine itself, and the extraction engine.

[4]`https://docs.virustotal.com/reference/overview`

Figure 1. Foremost-NG architecture



`Foremost-NG` retains the lightweight C implementation of the original tool but rearranges core components and provides integration with VirusTotal for threat intelligence lookups. At launch, a central dispatcher initializes a registry of available parsers embedded in the source code. Each parser contains a set of characteristics for identifying valid files:

- A unique format identifier (e.g., `"EVTX"`, `"MACHO"`, `"ELF"`).
- Byte signatures or heuristic checks for header detection.
- A callback to validate extracted headers/trailers.
- A recovery function that writes out the reconstructed file and its metadata.

Upon reading raw input (disk image, memory dump, or file container), the dispatcher continuously feeds byte-ranges to each parser's signature check routine. When a match is detected, the corresponding parser's recovery function is invoked to carve out full length and perform sanity checks. This separation ensures that adding, removing, or updating any parser does not require modifying the dispatcher logic itself, dramatically reducing merge conflicts and easing code reviews.

The VirusTotal lookup in `Foremost-NG` is implemented as a lightweight, self-contained module that operates as follows. First, when a file has been successfully carved, `Foremost-NG` computes its cryptographic checksum (either SHA-1 or SHA-256) using OpenSSL's EVP interface [Sivanantham et al. 2024]. Next, it retrieves the user's API key from the `VT_API_KEY` environment variable (returning an error if this variable is unset or empty). With both checksum and key in hand, the tool formats an HTTP GET request to `https://www.virustotal.com/api/v3/files/hash`, setting the `x-apikey` header and requesting JSON in return.

Using `libcurl` [Kalmukov 2024], the module performs the request with SSL verification disabled (to avoid unnecessary blocking), streaming the response into a dynamically allocated buffer. Once the HTTP transaction completes, the code searches for the `last_analysis_stats` object in the returned JSON. If found, it uses `sscanf` to extract the number of malicious and undetected engine results. These two counts populate a `VTResult` struct, and a simple flag is set if malicious is greater than zero. Finally, any allocated memory is freed, `libcurl` is cleaned up, and `Foremost-NG` writes a brief verdict (malicious count, undetected count, and a clean/malicious flag) into the output logs. This workflow ensures that every carved artifact can be automatically enriched with real-time threat intelligence, allowing investigators to immediately prioritize suspicious files without leaving the carving phase.

3. Format Support and Carving Modules

`Foremost-NG`'s format support section outlines the diverse range of file types that can be accurately recovered from raw data sources. Leveraging a modular extraction frame-

work, each format handler encapsulates the logic needed to recognize headers, footers, and size information, ensuring that both legacy formats and modern or platform-specific formats are carved reliably. By separating format-specific parsing into individual routines, `Foremost-NG` allows investigators to recover a wide array of artifacts. The supported files are highlighted below.

- **JPEG (JFIF/EXIF)**: When carving JPEG images, the tool identifies standard JFIF and EXIF headers and ensures the presence of both quantization and Huffman tables before extracting. It then locates the end-of-image marker and reconstructs the entire JPEG file, appending any relevant metadata (such as JFIF or EXIF) to the output filename, so that analysts receive a complete, valid image even if fragments were scattered on disk.
- **PNG (IHDR/IEND)**: PNG extraction begins by verifying the 8-byte PNG signature and then reading the IHDR chunk to capture image dimensions. `Foremost-NG` iterates through PNG chunks until encountering the IEND marker, at which point it outputs the full PNG file, including width and height metadata appended as a comment, ensuring that even partially overwritten files are reconstructed whenever possible.
- **BMP (Bitmap)**: For BMP images, the tool reads the file size from the header and validates basic dimensions before carving. Once confirmed, it extracts the exact number of bytes specified in the header, producing a usable BMP file. This allows recovery of raw bitmap data, even in cases where the filesystem has marked portions of the image as deleted.
- **GIF (GIF87a/GIF89a)**: The solution detects GIF images by matching the GIF87a or GIF89a signature, reads the width and height for context, and searches for the trailer byte to mark the end of the file. By reconstructing from header through trailer, it ensures a complete GIF, including any embedded frames, is recovered, making it possible to view animated or static GIFs even if they were fragmented.
- **RIFF Containers (AVI, WAVE)**: In RIFF-based formats, such as AVI and WAV, `Foremost-NG` reads the container identifier (AVI or WAVE) and the size field to determine how many bytes to carve. It then extracts the entire container, preserving audio or video streams intact. This approach reliably recovers multimedia files for playback or analysis, even when file table entries have been removed.
- **WMV (Windows Media Video)**: WMV carving relies on recognizing Windows Media header objects and reading the embedded size field. The tool locates the file's header sequence, validates required fields, and extracts the exact amount of data corresponding to the video stream. This enables investigators to recover WMV files for playback and further forensic inspection, even if the file was partially overwritten.
- **OLE Containers (DOC, XLS, PPT)**: For legacy Office documents stored in OLE format, `Foremost-NG` parses the OLE header, enumerates directory entries, and follows the FAT chain to determine the total size of all streams. It then reconstructs the entire container as a single file, automatically inferring whether it is a Word document, Excel spreadsheet, or PowerPoint presentation. This makes it possible to recover Office files that would otherwise be inaccessible due to missing

filesystem metadata.

- **ZIP (OOXML/OpenOffice)**: ZIP and modern Office archives (OOXML or OpenOffice) are handled by locating local file headers and then scanning through each entry until the central directory footer is reached. `Foremost-NG` recognizes OOXML-specific paths (e.g., `word/document.xml`) or OpenOffice mimetypes to assign appropriate extensions. By extracting the full ZIP container, it enables forensic analysts to recover archives and Office files even when they are fragmented or partially corrupted.
- **RAR Archives**: RAR support involves parsing RAR block headers, handling both standard and encrypted or multi-volume archives. The solution follows each file header block until the end-of-archive marker, appending comments when multi-volume or encrypted headers are detected. This ensures that complete RAR archives can be recovered for later decompression or analysis, even if some header information is missing.
- **PDF (Linearized or Non-Linear)**: PDF extraction begins by checking for linearization markers; if found, `Foremost-NG` reads the declared file size and extracts the entire linearized document. If linearization is absent or fails, it searches for the "%%EOF" footer within a bounded range. By supporting both linearized and standard PDFs, it reliably recovers PDF files, preserving objective-oriented structure and embedded content despite fragmentation.
- **C/C++ Source Files (CPP)**: Text-based source code is identified by scanning for "`#include`" directives. Once detected, the tool captures contiguous printable ASCII characters backward to include file prologues and forward until non-printable characters are encountered. It then verifies the presence of C/C++ markers (such as closing braces), ensuring that only genuine source files (and not random ASCII fragments) are carved. This approach recovers human-readable code for analysis of embedded scripts or programs.
- **HTML (HTM)**: `Foremost-NG` locates HTML pages by recognizing the `<HTML>` tag and verifying that subsequent bytes are printable. It then extracts bytes through the matching HTML tag, reconstructing the complete document. This allows recovery of web pages that were browsed or downloaded but later deleted, preserving forensic evidence of online activity.
- **MPEG-1 Video (MPG)**: The tool identifies MPEG-1 video by scanning for packet start codes (0xBB) and traversing PES packet size fields until it reaches a sequence end marker. By accumulating these packets, `Foremost-NG` reconstructs the entire MPG file. Even if video segments are fragmented, this method yields a playable MPEG-1 file for forensic review.
- **MP4 Video**: MP4 extraction reads each header to determine the total media length. `Foremost-NG` iterates through box sizes until a terminating box is found or size becomes zero, then outputs the entire MP4 fragment. Although still under development, this approach aims to recover modern MP4 videos by preserving container structure and associated metadata.
- **EXE/DLL (PE Executables)**: `Foremost-NG` locates Windows PE executables by finding the "MZ" and "PE" signatures. It reads the section table to calculate

the maximum extent of all loadable sections, appends compile timestamps as comments, and then extracts the full file. This ensures that complete EXE or DLL files (including any embedded resources) are recovered for malware analysis or reverse engineering.

- **Windows Registry Hives (REGF)**: To carve Windows registry hives, the tool checks for the REGF signature and reads the hive size field. It then extracts exactly the specified number of bytes, ensuring that the entire registry file (including key and value tables) is reconstructed. This makes it possible to recover registry hives for detailed forensic parsing of system and user activity.
- **Windows Event Logs (EVTX)**: `Foremost-NG` identifies EVTX files by matching the "ElfFile" magic in the event log header. It reads the number of 64 KiB chunks in the file and either locates the next EVTX header (to carve only until that point) or extracts all chunks indicated. The resulting EVTX file can then be parsed with standard event log tools, allowing investigators to recover crucial audit data from unallocated space.
- **ELF (Linux/Unix Binaries)**: For ELF binaries, the tool checks the ELF magic and identifies whether the file is 32-bit or 64-bit. It reads the program header table to determine all loadable segments' extents and then extracts the union of those segments. The carved ELF file retains the correct executable layout and symbol information, enabling subsequent analysis on Unix-style systems.
- **Mach-O (macOS/iOS Binaries)**: Mach-O extraction first detects the FAT (universal) magic to handle multi-architecture binaries; it then reads each architecture's offset and size to compute the maximal range. If the FAT header is absent, it checks for thin Mach-O magic and reads load command entries to locate segment extents. The complete Mach-O binary, whether fat or thin, is written out, preserving both 32-bit and 64-bit slices for macOS analysis.
- **Script Files (Bash, Python, PowerShell)**: Script carving begins by locating a shebang (`#!`) at the file's start. `Foremost-NG` then scans forward to include all printable characters up to a non-printable sentinel, ensuring the file is at least 32 bytes long and contains more than comments. It extracts the entire script (complete with interpreter directives) so that analysts can examine shell scripts, Python code, or PowerShell scripts that may have been deleted.

Each module logs detailed metadata into an `audit.txt` file in the output directory. Parsers that support internal integrity checks flag corrupt or truncated extractions, allowing analysts to discard incomplete results at a glance.

4. Installation and Usage

To install `Foremost-NG`, clone the GitHub repository and compile with a standard GCC toolchain. Example on a Unix-like system:

```
git clone https://github.com/cristianzsh/foremost-ng.git
cd foremost-ng
make
sudo make install
```

After installation, the `foremost-ng` binary is available system-wide. Users must export an environment variable called `VT_API_KEY` and include a valid VirusTotal API key if they wish to enable the VT lookup:

```
export VT_API_KEY=<YOUR_VT_API_KEY>
```

The tool's documentation can be read in the `README.md` file, or using `man foremost-ng`. Carving test samples can be downloaded from `https://dftt.sourceforge.net`. A video demonstrating how to install, run, and interpret the tool's output is available at `https://www.youtube.com/watch?v=RXGGKduSzJ0`.

4.1. Basic Usage

To carve a disk image using `Foremost-NG`, invoke the tool with the `-i` option to specify the input device or image and the `-o` option to designate an output directory. For example:

```
foremost-ng -i disk.img -o /temp/output/carved
```

In this default mode, every registered parser (such as EVTX, script files, Mach-O, and ELF) will inspect the raw data sequentially. Carved artifacts are written into subdirectories under specified output directory, each named after the corresponding file format.

When it is necessary to restrict carving to a subset of formats (for example, focusing on executable or log files), the `-t` flag accepts a comma-separated list of up to all registered format identifiers. For instance:

```
foremost-ng -t evtx,elf,exe -i disk.img
-o /temp/output/carved
```

In this invocation, only the EVTX, ELF, and EXE modules will run, reducing both runtime and extraneous output. Each carved file is still placed in a format-specific subdirectory within the specified output path. To obtain real-time threat intelligence on recovered files, one can include the `-x` switch, provided a valid VirusTotal API key. For example:

```
foremost-ng -x -t elf -i memory.dump -o ./carved-files
```

Here, `Foremost-NG` will compute SHA-256 (or SHA-1) hashes for each extracted ELF binary, query VirusTotal's API, and annotate the `audit.txt` with malicious and undetected counts. If the API key is missing or invalid, `Foremost-NG` will still perform carving but will emit a warning and skip all VT lookups.

4.2. Output Structure

Each run generates:

- A top-level output directory (e.g., `carved/`).
- Subdirectories for each format (e.g., `carved/evtx`, `carved/elf`, `carved/exe`).
- Individual recovered files named `00000001.evtx`, `00000002.sh`, `00000003.elf`, and so on.
- A `recovery.log` summarizing each extraction with byte offsets, hash values, and VT information (if enabled).

Figure 2 shows an execution example and the expected output.

Figure 2. `Foremost-NG` execution.

```
$ ./foremost-ng -v -x evidence2.dd
foremost-ng version 1.5.7
Audit File

foremost-ng started at Mon Jun  9 23:33:41 2025
Invocation: ./foremost-ng -v -x evidence2.dd
Output directory: /tmp/foremost-ng/src/output
Configuration file: /tmp/foremost-ng/src/foremost.conf
[INFO] Processing: evidence2.dd
------------------------------------------------------------------
File: evidence2.dd
Start: Mon Jun  9 23:33:41 2025
Length: 2 GB (2845109643 bytes)

  ID  Name (bs=512)              Size       Offset  Comment                  VT
   0: 00004352.elf              57 KB      2228224                           Clean (0)
   1: 00004820.elf              57 KB      2467840                           Clean (0)
   2: 00005108.elf               1 KB      2615296                           Clean (0)
   3: 00005492.script            3 KB      2811904                           Clean (0)
   4: 00399884.exe             600 KB    204740608  01/01/1970 00:00:00    Malicious (2)
   5: 00405060.exe              52 KB    207390720  01/01/1970 00:00:00    Clean (0)
   6: 00414932.exe               2 MB    212445184  01/01/2015 00:00:00    Clean (0)
   7: 00428728.exe               2 MB    219508736  01/01/2015 00:00:00    Clean (0)
   8: 00442520.exe             536 KB    226570240  01/01/1970 00:00:00    Clean (0)
   9: 00447356.exe             600 KB    229046272  01/01/1970 00:00:00    Malicious (2)
  10: 01786101.zip             470 KB    914484119                         Clean (0)
  11: 01787042.zip             350 KB    914965615                         Clean (0)
```

5. Conclusion

`Foremost-NG` modernizes the original carving engine by refactoring its core and making it easy to add or update format handlers without affecting other components. It extends support beyond legacy formats to include Windows EVTX logs, script files, Mach-O binaries, and ELF executables. Built-in VirusTotal integration enriches every carved file with real-time threat intelligence, allowing analysts to focus on suspicious artifacts immediately. By combining modular extensibility with automated intelligence lookups, `Foremost-NG` offers a straightforward yet powerful solution for today's digital forensics and incident-response needs. As future work, we plan to extend the supported file types and improve the overall tool's performance.

References

Casino, F., Dasaklis, T. K., Spathoulas, G. P., Anagnostopoulos, M., Ghosal, A., Borocz, I., Solanas, A., Conti, M., and Patsakis, C. (2022). Research trends, challenges, and emerging topics in digital forensics: A review of reviews. *Ieee Access*, 10:25464–25493.

Kalmukov, Y. (2024). Experimental evaluation of the php's curl library performance. *arXiv preprint arXiv:2405.00001*.

Ramli, N. I. S., Hisham, S. I., and Badshah, G. (2021). Analysis of file carving approaches: A literature review. In *Advances in Cyber Security: Third International Conference, ACeS 2021, Penang, Malaysia, August 24–25, 2021, Revised Selected Papers 3*, pages 277–287. Springer.

Sivanantham, G., Krishnan, A., et al. (2024). Experimenting integration of custom ecdsa algorithm in openssl. In *2024 IEEE International Conference on Public Key Infrastructure and its Applications (PKIA)*, pages 1–6. IEEE.

Souza, C. H., Pascoal, T., Neto, E. P., Sousa, G. B., Filho, F. S., Batista, D. M., and Dantas Silva, F. S. (2025). Sdn-based solutions for malware analysis and detection: State-of-the-art, open issues and research challenges. *Journal of Information Security and Applications*, 93:104145.