

Date: 25th July 2025

Author: Ruby Girjashankar kannaujiya Intern ID:-309

Purpose: Assignment as an intern at Digisuraksha Parhari Foundation.

---

# Tools POC Report – Binwalk & Foremost

---

## ❖ Tool Name: Binwalk

---

### History:

Binwalk was developed by the ReFirm Labs team to aid in reverse-engineering firmware images. It's widely used by security researchers, especially in embedded systems.

---

### Description:

Binwalk is an open-source tool used to analyze, extract, and reverse-engineer firmware images. It scans binary files for embedded file systems and other data using signature analysis.

---

### What Is This Tool About?

Binwalk extracts files or data embedded within firmware images. It is essential in digital forensics, malware analysis, and embedded security research.

---

### Key Characteristics / Features:

1. Scans binary images for embedded files
2. Supports entropy analysis
3. Extracts file systems (e.g., SquashFS, JFFS2)
4. Custom signature support
5. Fast and accurate matching

6. Integrates with gzip, tar, lzma, etc.
  7. Useful for reverse-engineering
  8. Command-line interface
  9. Plugin extensibility
  10. Works on Linux/macOS
- 

### **Types / Modules Available:**

- Signature Analysis Module
  - Extraction Module
  - Entropy Analysis
  - Plugin Support
  - Recursive Extraction
- 

### **How Will This Tool Help?**

- Extracts hidden firmware content
  - Speeds up forensic analysis of IoT devices
  - Helps discover malware in firmware
  - Supports embedded systems reverse engineering
- 

### **15-Liner Summary:**

1. Scans binary files
2. Firmware reverse-engineering
3. CLI based, fast
4. Plugin support
5. Linux/macOS compatible
6. File carving
7. Detects compression/encryption
8. Recursive extraction
9. Open-source
10. Forensics-friendly
11. Embedded file system detection
12. Bulk extraction
13. Malware investigation aid
14. Saves time in analysis
15. Easy automation

---

## Time to Use / Best Case Scenarios:

- During firmware analysis
  - After embedded system acquisition
  - While auditing IoT devices
  - During malware investigation in devices
- 

## When to Use During Investigation:

- Post device seizure
  - While profiling IoT threats
  - When investigating smart devices
  - After firmware dump extraction
- 

## Best Person to Use & Required Skills:

**User:** Embedded Security Researcher / Forensic Analyst

**Skills:**

- Linux command-line
  - Firmware structure knowledge
  - Basic file systems
  - Forensics awareness
- 

## Flaws / Suggestions to Improve:

- No GUI interface
  - May not support proprietary formats
  - Needs better visualization
  - Doesn't detect encrypted partitions
  - Output could be more readable
- 

## Good About the Tool:

- Lightweight and fast

- Very accurate
  - Highly automatable
  - Great for reverse engineering
  - Open-source with community support
- 

## ❖ Tool Name: Foremost

---

### History:

Foremost was developed by the U.S. Air Force Office of Special Investigations and the Naval Postgraduate School for file recovery in digital forensics.

---

### Description:

Foremost is a command-line program used to recover deleted files from raw disk images using file headers and footers. It is powerful in forensic recovery.

---

### What Is This Tool About?

Foremost scans disks for known byte patterns of file headers/footers and reconstructs deleted or lost files — even from damaged drives.

---

### Key Characteristics / Features:

1. File carving based on headers/footers
2. Recovers deleted data
3. Works without needing file system structure
4. CLI interface
5. Lightweight and fast
6. Supports JPG, PNG, PDF, DOC, ZIP, etc.
7. Generates recovery logs
8. Very accurate pattern matching
9. Configurable via `foremost.conf`
10. Works on Linux

---

## **Types / Modules Available:**

- File Carving Module
  - Configuration File Module
  - Logging & Output Module
- 

## **How Will This Tool Help?**

- Recovers lost or deleted files
  - Works on disk images, memory cards, USBs
  - Useful in law enforcement and cybercrime
  - Doesn't rely on file system integrity
- 

## **15-Liner Summary:**

1. CLI-based file recovery
  2. Carves files from raw images
  3. Lightweight tool
  4. Works on Linux
  5. Easy config file
  6. Fast scan & extract
  7. No need for file system
  8. Recovers photos/docs/zips
  9. Used in forensics
  10. Output with logs
  11. No installation needed
  12. Effective on corrupted data
  13. Open-source
  14. Trusted by experts
  15. Great for deep data recovery
- 

## **Time to Use / Best Case Scenarios:**

- After accidental deletion
- Post-drive format
- Memory card recovery
- USB and disk corruption

---

## **When to Use During Investigation:**

- File recovery from seized devices
  - Inspecting formatted drives
  - Child protection cases
  - Cybercrime involving hidden data
- 

## **Best Person to Use & Required Skills:**

**User:** Forensics Investigator / Data Recovery Specialist

**Skills:**

- Linux terminal usage
  - File header structure understanding
  - Disk image analysis
- 

## **Flaws / Suggestions to Improve:**

- No GUI
  - Doesn't support encrypted recovery
  - Might miss fragmented files
  - Basic configuration
  - Limited to known file types
- 

## **✓ Good About the Tool:**

- Open-source and free
  - Reliable in recovery
  - Lightweight
  - Fast scanning
  - Simple usage
- 

## **Binwalk in Action**

Figure 1: Firmware Extraction Using Binwalk

```
student@kali:~/firmware$ ./firmware.bin.extracted/squashfs-root$ cat /etc/init.d/firewalld
#!/bin/sh

NUM="64"
START=50
STOP=50
PROCESS="base"
CMD="bin/gzcat|bc2n|sh"
PROG=/usr/bin/firewalld
NAME=firewall
PIDCOUNT=8

prepare_cmd () {
    echo "$1" | $PROCESS$NUM -d
}

load_interfaces()
{
    config_get interface "$1" Interface
    interfaces=" ${interface} ${interfaces}"
}

start_service()
{
    [ -s /etc/dropbear/dropbear_rsa_host_key ] || keygen
    . /lib/functions.sh
```

This screenshot shows the **Binwalk tool scanning a firmware binary image**. The terminal output reveals recognized embedded file systems and data types like gzip, squashfs, or ELF binaries. It also auto-extracts the files into separate folders for detailed investigation, making it easier for analysts to reverse-engineer embedded software or find suspicious components.

---

## Entropy Analysis with Binwalk

Figure 2: Entropy Graph of Firmware Regions

```
student@kali:~/firmware$ strings FW_WRT1900ACSV2_2.0.3.201002_prod.ing
Linksys WRT1900ACS Router
B R
!IC
-- System halted
Attempting division by 0!
Uncompressing Linux...
decompressor returned an error
done, booting the kernel.
invalid distance too far back
invalid distance code
invalid literal/length code
incorrect header check
unknown decompression method
invalid window size
invalid block type
invalid stored block lengths
too many length or distance symbols
invalid code lengths set
invalid bit length repeat
invalid literal/lengths set
invalid distances set
incorrect data check
Out of memory while allocating output buffer
Out of memory while allocating input buffer
Out of memory while allocating z_stream
Out of memory while allocating workspace
Not a gzip file
header error
read error
uncompression error
phandle
linux.phandle
```

The entropy graph generated by Binwalk visually represents regions of **high entropy**, often indicating compressed or encrypted data. This helps investigators quickly focus on areas that may contain obfuscated or sensitive files such as malware payloads or encrypted configurations.

---

## Foremost CLI Usage

Figure 3: Foremost Command-Line Help Output

```
root@Kali:~# binwalk -h
Binwalk v2.1.2
Craig Heffner, ReFirmLabs
https://github.com/ReFirmLabs/binwalk

Usage: binwalk [OPTIONS] [FILE1] [FILE2] [FILE3] ...

Disassembly Scan Options:
  -Y, --disasm           Identify the CPU architecture of a file using the capstone disassembler
  -T, --minsn=<int>      Minimum number of consecutive instructions to be considered valid (default: 500)
  -k, --continue         Don't stop at the first match

Signature Scan Options:
  -B, --signature        Scan target file(s) for common file signatures
  -R, --raw=<str>         Scan target file(s) for the specified sequence of bytes
  -A, --opcodes          Scan target file(s) for common executable opcode signatures
  -M, --magic=<file>     Specify a custom magic file to use
  -b, --dumb              Disable smart signature keywords
  -I, --invalid           Show results marked as invalid
  -X, --exclude=<str>     Exclude results that match <str>
  -y, --include=<str>     Only show results that match <str>

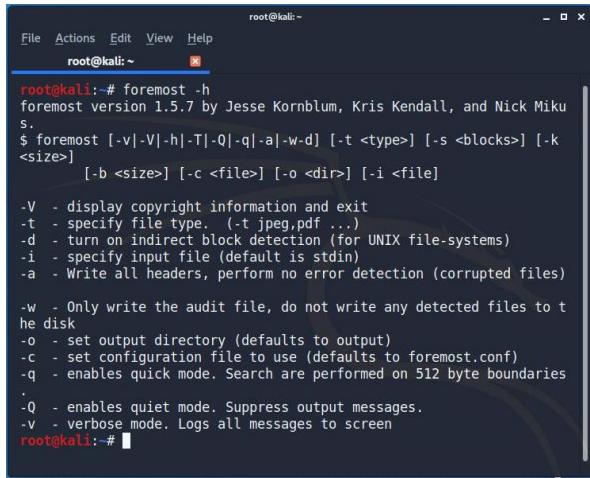
Extraction Options:
  -e, --extract            Automatically extract known file types
  -D, --dd=<type>:<ext>:<cmd> Extract <type> signatures, give the files an extension of <ext>, and execute <cmd>
  -M, --matryoshka          Recursively scan extracted files
  -d, --depth=<int>         Limit matryoshka recursion depth (default: 8 levels deep)
  -C, --directory=<str>     Extract files/folders to a custom directory (default: current working directory)
  -j, --size=<int>          Limit the size of each extracted file
  -n, --count=<int>         Limit the number of extracted files
  -r, --rm                  Delete carved files after extraction
  -z, --carve               Carve data from files, but don't execute extraction utilities
```

This image displays the `foremost -h` command output, listing all available options. It highlights supported file types, configuration flags, and output settings. This quick reference is essential for forensic investigators who rely on **Foremost's accuracy in recovering deleted files** from raw disk images.

---

## Recovered Files Using Foremost

Figure 4: Directory View of Carved Files



The screenshot shows a terminal window titled "root@kali:~". The user has run the command "foremost -h" which displays the help documentation for the Foremost tool. The help text includes the version information ("foremost version 1.5.7 by Jesse Kornblum, Kris Kendall, and Nick Mikus."), usage examples, and a detailed list of command-line options and their descriptions.

```
root@kali:~# foremost -h
foremost version 1.5.7 by Jesse Kornblum, Kris Kendall, and Nick Mikus.
$ foremost [-v|-V|-h|-T|-Q|-q|-a|-w-d] [-t <type>] [-s <blocks>] [-k <size>]
           [-b <size>] [-c <file>] [-o <dir>] [-i <file>]

-v  - display copyright information and exit
-t  - specify file type. (.t jpeg,pdf ...)
-d  - turn on indirect block detection (for UNIX file-systems)
-i  - specify input file (default is stdin)
-a  - Write all headers, perform no error detection (corrupted files)

-w  - Only write the audit file, do not write any detected files to the disk
-o  - set output directory (defaults to output)
-c  - set configuration file to use (defaults to foremost.conf)
-q  - enables quick mode. Search are performed on 512 byte boundaries
-Q  - enables quiet mode. Suppress output messages.
-v  - verbose mode. Logs all messages to screen
root@kali:~#
```

After executing Foremost, recovered files are automatically saved in categorized folders (e.g., jpg, pdf). This screenshot shows a typical output directory after carving files from a USB or disk image. It demonstrates **Foremost's usefulness in recovering evidence from deleted or corrupted data sources**.

---

## Sample Use Case:

### Scenario:

A cybersecurity team is handed a suspicious firmware dump from a smart doorbell suspected to be leaking user data.

- They first use **Binwalk** to extract the embedded file system and identify suspicious scripts.
- Then, they use **Foremost** to **carve deleted image and document files** from the raw memory of the firmware to check for evidence of stored logs or user images.

Both tools help reconstruct the attacker's trail and present valid forensic evidence.

---

## Comparison – Binwalk vs. Foremost:

Feature	Binwalk	Foremost
Tool Type	Firmware analysis & extraction	File recovery from raw images
Usage Level	Intermediate to Advanced	Beginner to Intermediate
Interface	CLI	CLI
Best For	Embedded devices, firmware	Disk/memory file carving

Feature	Binwalk	Foremost
Output Format	Extracted folders, logs	Recovered files + logs
Speed	Fast	Fast
Extensibility	Plugin support	Config file based
GUI Support	✗	✗
Forensics Role	Reverse engineering	Data recovery

---

## Real-World Applications:

### ◆ Binwalk:

- Analyzing firmware in **IoT devices** for hidden malware
- Auditing embedded systems in **cybersecurity firms**
- Used in **vulnerability research labs**

### ◆ Foremost:

- Recovering deleted files in **child exploitation cases**
- Used by **law enforcement** to inspect suspect USB drives
- Helps in **accidental data recovery** and **evidence collection**

---

## Why Use Both Together?

Using Binwalk and Foremost in **combination** gives forensic analysts a powerful edge:

- Binwalk uncovers structured embedded data (e.g., scripts, configs)
- Foremost recovers **hidden/deleted** files that Binwalk might miss

Together, they provide both surface and deep-dive insights into suspicious digital evidence.

---

### ✓ Conclusion:

Binwalk and Foremost are essential tools in the **digital forensics toolkit**. While Binwalk is ideal for analyzing firmware and embedded content, Foremost excels in recovering deleted files from raw storage. Their combined usage can significantly **enhance the accuracy and depth of any investigation**.

---