Name – Soham Kadu

Intern ID – 235

🛠 Tool Name:

GoStringUngarbler + HashMyFiles

📜 **History:**

- **GoStringUngarbler** is a community-developed Go-based tool used in malware analysis to decode garbled or obfuscated strings found in binary files.
- **HashMyFiles** is a trusted Windows utility from NirSoft, active for over a decade, used to generate cryptographic hashes of files.

📄 **Description:**

A powerful combination of two lightweight tools. GoStringUngarbler is used to extract and decode suspicious strings (such as URLs, file paths, C2 domains) from malware or binaries. HashMyFiles is used to compute and verify file hashes (MD5, SHA1, SHA256) for integrity verification or forensic tracking.

📌 **What Is This Tool About?**

These tools are commonly used in malware analysis, reverse engineering, and digital forensics.

- GoStringUngarbler helps investigators decode hidden or obfuscated strings in malicious executables.
- HashMyFiles is used to verify the integrity of these suspicious files or create hash reports for forensic evidence.

Key Characteristics / Features:

GoStringUngarbler

1. Written in Go, open source
2. Supports common obfuscation decoding
3. Works on binary dumps, strings files
4. Fast, CLI-based
5. Supports decoding with custom logic
6. Minimal dependencies

HashMyFiles

1. Portable – no install needed
2. Supports MD5, SHA1, SHA256, SHA384
3. Export results in HTML, TXT, CSV
4. Drag & drop interface
5. Supports folder & batch hashing
6. Easy integration into forensic workflows

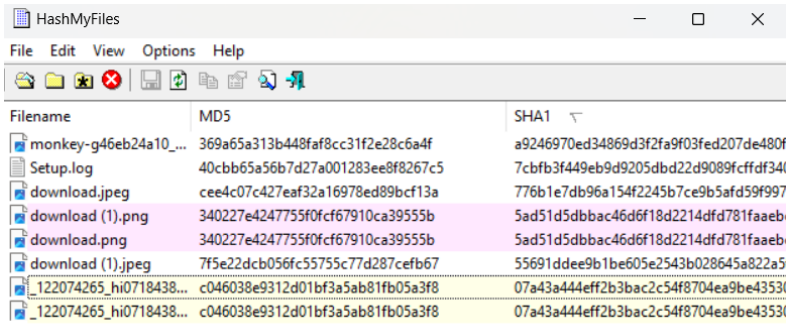
Types / Modules Available:

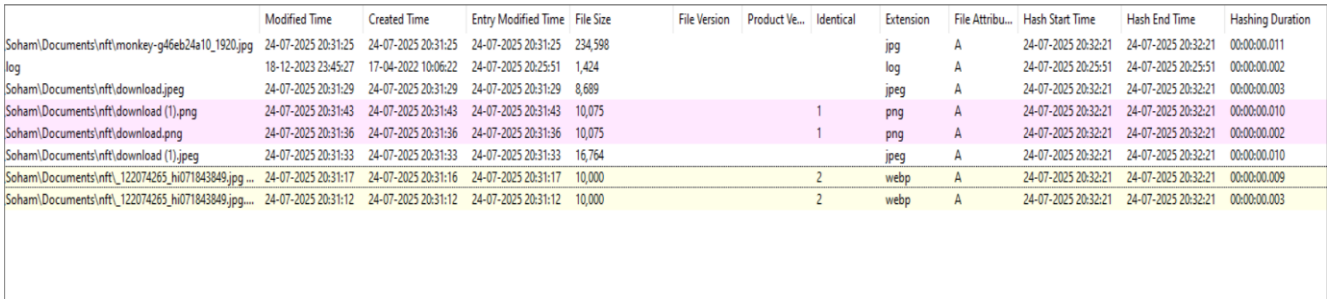
- GoStringUngarbler CLI tool
- Obfuscation decoding modules (e.g., Base64, XOR)
- HashMyFiles GUI & Command-line versions
- Context menu integration in Windows
- Timestamp & hash export modules

🎯 **How Will This Tool Help?**

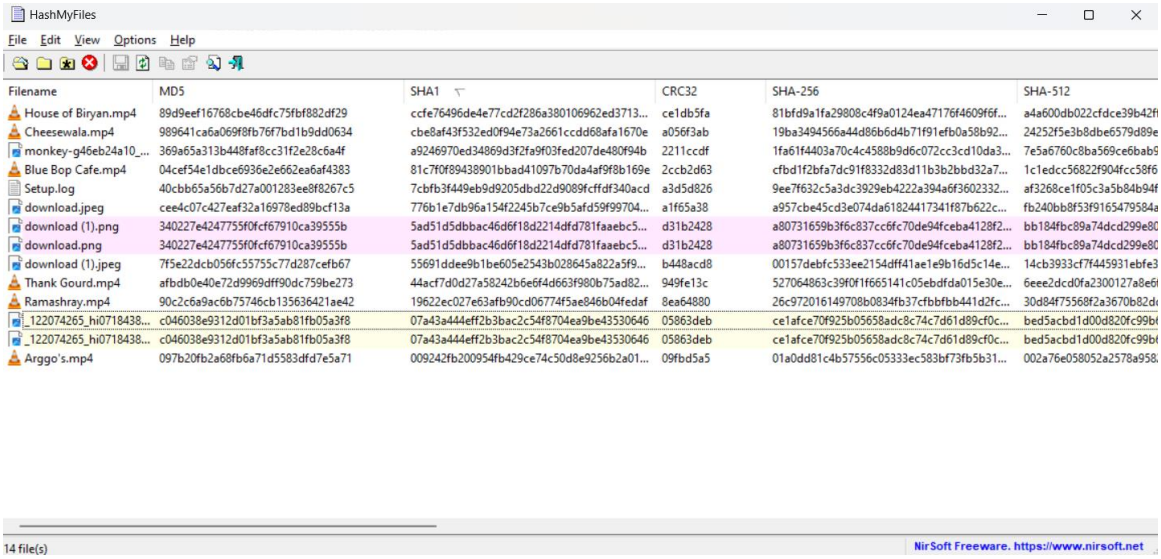
- Extracts hidden strings from malware
- Decodes hardcoded indicators (C2s, payloads)
- Verifies file integrity for chain of custody
- Helps in malware variant comparison
- Generates quick hash reports for evidence
- Easy to automate in investigation scripts

Proof of Concept (PoC) Images:





	Modified Time	Created Time	Entry Modified Time	File Size	File Version	Product Ve...	Identical	Extension	File Attribu...	Hash Start Time	Hash End Time	Hashing Duration
Soham\Documents\nft\monkey-g46eb24a10_1920.jpg	24-07-2025 20:31:25	24-07-2025 20:31:25	24-07-2025 20:31:25	234,598				jpg	A	24-07-2025 20:32:21	24-07-2025 20:32:21	00:00:00.011
log	18-12-2023 23:45:27	17-04-2022 10:06:22	24-07-2025 20:25:51	1,424				log	A	24-07-2025 20:25:51	24-07-2025 20:25:51	00:00:00.002
Soham\Documents\nft\download.jpeg	24-07-2025 20:31:29	24-07-2025 20:31:29	24-07-2025 20:31:29	8,689				jpeg	A	24-07-2025 20:32:21	24-07-2025 20:32:21	00:00:00.003
Soham\Documents\nft\download (1).png	24-07-2025 20:31:43	24-07-2025 20:31:43	24-07-2025 20:31:43	10,075			1	png	A	24-07-2025 20:32:21	24-07-2025 20:32:21	00:00:00.010
Soham\Documents\nft\download.png	24-07-2025 20:31:36	24-07-2025 20:31:36	24-07-2025 20:31:36	10,075			1	png	A	24-07-2025 20:32:21	24-07-2025 20:32:21	00:00:00.002
Soham\Documents\nft\download (1).jpeg	24-07-2025 20:31:33	24-07-2025 20:31:33	24-07-2025 20:31:33	16,764				jpeg	A	24-07-2025 20:32:21	24-07-2025 20:32:21	00:00:00.010
Soham\Documents\nft_122074265_hi071843849.jpg ...	24-07-2025 20:31:17	24-07-2025 20:31:16	24-07-2025 20:31:17	10,000			2	webp	A	24-07-2025 20:32:21	24-07-2025 20:32:21	00:00:00.009
Soham\Documents\nft_122074265_hi071843849.jpg....	24-07-2025 20:31:12	24-07-2025 20:31:12	24-07-2025 20:31:12	10,000			2	webp	A	24-07-2025 20:32:21	24-07-2025 20:32:21	00:00:00.003




```
INFO:root:678 in 693 | result at 0x53c0e0: 'UnlockFileEx'
INFO:root:679 in 693 | result at 0x53cc80: 'NetUserGetLocalGroups'
INFO:root:680 in 693 | result at 0x53cfe0: 'GetProcessMemoryInfo'
INFO:root:681 in 693 | result at 0x53d2c0: 'CreateEnvironmentBlock'
INFO:root:682 in 693 | result at 0x53d5a0: 'DestroyEnvironmentBlock'
INFO:root:683 in 693 | result at 0x53f080: 'file type does not support deadline'
INFO:root:684 in 693 | result at 0x53f420: 'not pollable'
INFO:root:685 in 693 | result at 0x542840: 'internal error: unknown network type '
INFO:root:686 in 693 | result at 0x542be0: 'wsaioctl'
INFO:root:687 in 693 | result at 0x548020: 'os: process already finished'
INFO:root:688 in 693 | result at 0x548940: 'winreadlinkvolume'
INFO:root:689 in 693 | result at 0x548be0: 'pattern contains path separator'
INFO:root:690 in 693 | result at 0x54a1c0: 'syntax error scanning complex number'
INFO:root:691 in 693 | result at 0x552f60: '0123456789ABCDEFX'
INFO:root:692 in 693 | result at 0x554140: '0123456789abcdefx'
INFO:root:693 in 693 | result at 0x5557a0: 'Hello, world.'
INFO:root:Error occured: 3
INFO:root:Stack obfuscated string count: 360 strings.
INFO:root:Split obfuscated string count: 152 strings.
INFO:root:Seed obfuscated string count: 181 strings.
INFO:root:Processing took 5817 ms!
```

```go
// Generate a random key with the same length as the input string
key := make([]byte, len(data))

// Fill the key with random bytes
obfRand.Read(key)

// Select a random operator (XOR, ADD, SUB) to be used for encryption
op := randOperator(obfRand)

// Encrypt each byte of the data with the key using the random operator
for i, b := range key {
    data[i] = evalOperator(op, data[i], b)
}
```

```
__int64 __fastcall sub_48D780()
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS NUMPAD "+" TO EXPAND]

  if ( &retaddr <= *(v0 + 16) )
    sub_46A400();
  *data = 0xAF7E8C66BECA0046LL;
  for ( i = 0LL; i < 8; ++i )
    data[i] -= KEY[i];
  return runtime_slicebytetostring(data, 8);
}
```

📜 **15-Liner Summary:**

1. CLI-based decoding with Go
2. GUI-based file hashing
3. Supports malware string extraction
4. Works on Windows/Linux
5. Lightweight and fast
6. Good for YARA rule crafting
7. Easy file hash reporting
8. Export results to TXT, HTML, CSV
9. Useful in static malware analysis
10. Ideal for integrity validation
11. Custom decoding logic support
12. Drag & drop functionality
13. No installation required
14. Low resource usage
15. Open source / freeware

Time to Use / Best Case Scenarios:

- During malware reverse engineering
- Before dynamic analysis or sandboxing
- For file comparison after patching
- In evidence packaging for courts
- Before uploading files to threat intel platforms

When to Use During Investigation:

- Post-infection artifact analysis
- Unknown binary behavior tracking
- Creating IoCs (Indicators of Compromise)
- Timeline correlation with file hashes
- Investigating persistence mechanisms

Best Person to Use This Tool & Required Skills:

Best Users:

- Malware Analyst
- Digital Forensics Investigator
- Threat Intel Researcher

Required Skills:

- Basic command-line usage
- Familiarity with obfuscation techniques
- Understanding of file hashing concepts
- Comfort with static malware analysis

Flaws / Suggestions to Improve:

- GoStringUngarbler lacks GUI for beginners
- Needs more decoding logic (ROT13, AES)
- HashMyFiles doesn’t support recursive sub-folder hashing by default
- No direct integration with threat intel platforms
- Needs centralized log generation

Good About the Tools:

- Extremely lightweight and portable
- Effective in early-stage malware triage
- No installation needed – plug and play
- High speed for bulk processing

- Reliable outputs for forensic reporting