AmCache-EvilHunter: Automating Evidence of Execution Extraction from the Amcache.hve Artifact

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

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

{cristian.souza, eduardo.ovalle}@kaspersky.com, batista@ime.usp.br

Abstract. *Digital Forensics and Incident Response (DFIR) investigations frequently rely on native Windows artifacts to identify evidence of execution, reconstruct attacker activity, and generate indicators of compromise. Among these artifacts,* `Amcache.hve` *is especially valuable because it stores metadata about executables, installed applications, drivers, and shortcuts (such as file paths, timestamps, publisher information, and SHA-1 hashes). However, manually inspecting Amcache data can be time-consuming during incident response, especially when analysts need to filter large numbers of records and quickly verify suspicious hashes against threat intelligence sources. This paper presents* `AmCache-EvilHunter`*, an open-source command-line tool that parses Windows* `Amcache.hve` *registry hives, extracts relevant execution-related metadata, identifies suspicious executable names, filters records, and integrates Kaspersky OpenTIP and VirusTotal lookups. The tool was designed to reduce repetitive analysis tasks and support faster triage during real-world investigations.*

1. Introduction

Digital Forensics and Incident Response (DFIR) investigations depend on the ability to quickly identify what was executed, installed, copied, or present on a compromised system [Gnanasekaran et al. 2025, Jagathbala et al. 2025]. This information is important because attackers frequently remove their tools after execution, rename malicious files using legitimate Windows process names, or use remote access and administrative tools to maintain access to an environment (e.g., `PsExec`). In these scenarios, forensic artifacts can preserve traces that are no longer available on disk. The Windows operating system contains several artifacts that can help analysts identify evidence of execution, such as Prefetch, ShimCache, UserAssist, event logs, and the Amcache [Easttom et al. 2024, Neyaz and Shashidhar 2022].

The `Amcache.hve` artifact is especially useful because it stores rich metadata about executables and drivers, including file paths, compilation timestamps, publisher information, file sizes, and SHA-1 hashes. These hashes are valuable during investigations because they allow analysts to search public or private threat intelligence sources and generate indicators of compromise to hunt the same files (e.g., malware) across the infrastructure [Souza et al. 2025]. Amcache records files discovered by the Microsoft Compatibility Appraiser, executables and drivers copied during program execution, and applications that required compatibility shimming. It may preserve information about files that were deleted, renamed, or otherwise lost during an attack [Tokarev and Tokareva 2023].

This work introduces `AmCache-EvilHunter`, an open-source command-line tool to parse and analyze Windows `Amcache.hve` registry hives. The tool extracts relevant records, normalizes important fields, filters entries according to the analyst's needs, identifies suspicious executables, and enriches the output using threat intelligence information obtained from Kaspersky OpenTIP[1] and VirusTotal[2]. The main goal is to reduce manual effort during triage and help analysts quickly identify records that deserve deeper investigation.

The remainder of this paper is structured as follows: Section 2 presents background on the Amcache forensic artifact, as well as its limitations and related tools. Section 3 describes the architecture of the proposed solution in detail. Section 4 provides guidance on installation and usage, detailing the various available options. Section 5 highlights the planned demonstration during the conference. Finally, Section 6 concludes the paper with a summary of the tool's capabilities and its potential impact on the digital forensics community.

2. Background and Related Tools

Application Activity Cache, commonly known as Amcache, was introduced in Windows 7 and became more relevant in Windows 8 and later versions [Joo et al. 2023]. It is stored as a Windows Registry hive named `Amcache.hve`, usually located at:

```
C:\Windows\AppCompat\Programs\Amcache.hve
```

This artifact is managed by Windows compatibility components and contains metadata about applications, executable files, loaded drivers, shortcuts, and installed software. The Amcache format can vary depending on the libraries responsible for filling the cache, rather than only the operating system version. Known libraries involved in this process include `aecache.dll`, `aeevts.dll`, `aeinv.dll`, `aelupsvc.dll`, `aepdu.dll`, and `aepic.dll`. From a DFIR perspective, the most relevant keys in Amcache include `InventoryApplicationFile`, `InventoryApplication`, `InventoryDriverBinary`, and `InventoryApplicationShortcut`. These keys store different classes of information and should be interpreted according to their forensic meaning [Lagny 2019, Souza 2025]:

- The `InventoryApplicationFile` key is one of the most important keys for forensic triage. It stores metadata about executables discovered on the system, including the file name, full path, publisher, version, binary type, product name, link date, file size, SHA-1 hash, and whether the file is considered an operating system component.
- The `InventoryApplication` key records information about applications that were formally installed on the system. When a `ProgramId` appears both in `InventoryApplicationFile` and `InventoryApplication`, it can help analysts distinguish between a file that was only present on disk and a program that was installed.

[1]https://opentip.kaspersky.com
[2]https://www.virustotal.com

- The `InventoryDriverBinary` key stores information about kernel-mode drivers loaded by the system. This is especially relevant because malicious or vulnerable drivers are often abused by attackers to disable security products, hide activity, or operate with kernel privileges[3].
- The `InventoryApplicationShortcut` key stores information about shortcut files. It can help identify applications that were present in user-facing locations such as the Desktop or Start Menu. Although this key does not prove execution by itself, it can provide useful supporting evidence when correlated with other artifacts.

2.1. Amcache limitations

It is worth noting that the Amcache mechanism computes the SHA-1 hash over only the first 31,457,280 bytes, approximately 31 MB, of each executable. Therefore, for files larger than this size, the hash stored in Amcache may not correspond to the SHA-1 hash of the complete file. This is relevant during threat intelligence lookups because the stored hash may fail to match the hash of the full binary in online databases. Attackers could also abuse this characteristic by creating large malicious binaries, making hash-based identification harder when only Amcache data is available.

2.2. Related tools

Several tools can be used to collect or parse Windows forensic artifacts, as summarized in Table 1. `KAPE`[4] and `Velociraptor`[5] can collect `Amcache.hve` from endpoints during triage. `AmcacheParser`[6] is widely used to parse Amcache data and export structured output. `Registry Explorer`[7] can be used to manually inspect registry hives through a graphical interface.

However, incident response procedures often require quick conclusions, which requires tools to perform more than parsing. Modern DFIR tools should also implement fast filtering, suspicious pattern identification, removal of known-good operating system components, and threat intelligence enrichment. `AmCache-EvilHunter` was developed to complement existing approaches by focusing on quick suspicious-record identification and integrated OpenTIP and VirusTotal lookups.

3. Architecture and Implementation

`AmCache-EvilHunter` is implemented in Python and was designed as a command-line tool for forensic triage. It receives an `Amcache.hve` registry hive as input, parses its records, normalizes relevant fields, applies user-selected filters, enriches hashes using threat intelligence services, and displays or exports the results. Figure 1 presents the general architecture of the tool.

The parser opens the registry hive and iterates through the relevant subkeys. The tool keeps a set of core fields that are useful for DFIR analysis, including:

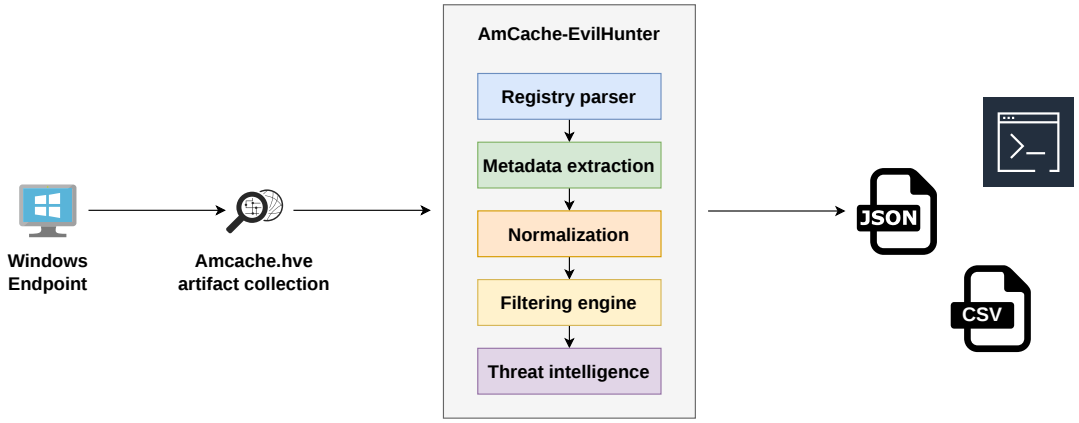
[3]https://securelist.com/av-killer-exploiting-throttlestop-sys/117026/
[4]https://www.sans.org/tools/kape
[5]https://github.com/Velocidex/velociraptor
[6]https://github.com/EricZimmerman/AmcacheParser
[7]https://www.sans.org/tools/registry-explorer

Table 1. Comparison between AmCache-EvilHunter and related tools.

Feature	This work	AmcacheParser	KAPE	Velociraptor
Parse `Amcache.hve`	✓	✓	✓*	✓*
CSV export	✓	✓	✓	✓
JSON export	✓	✓	✓	✓
Date filtering	✓	–	–	Limited
Keyword search	✓	–	–	Limited
Suspicious executable detection	✓	–	–	–
Threat intelligence enrichment	✓	–	–	Limited

*KAPE and Velociraptor can collect and/or process `Amcache.hve` through modules or artifacts, but they are not dedicated Amcache analysis tools.

Figure 1. General architecture of AmCache-EvilHunter.



- `Name` and `OriginalFileName`;
- `LowerCaseLongPath` and computed `FilePath`;
- `Publisher`, `Version`, and `ProductName`;
- `BinaryType`, `LinkDate`, and `Size`;
- `IsOsComponent`;
- `SHA-1`;
- `RecordDate`.

The normalization step trims string values and removes the four leading zeroes commonly stored before Amcache SHA-1 hashes. This makes the values easier to compare with public sources such as VirusTotal and Kaspersky OpenTIP.

The filtering module supports different investigation workflows. Date filters allow analysts to focus on a specific incident window. Keyword search allows fast hunting for known filenames, paths, or attacker naming conventions. Suspicious-name detection identifies records whose file names match patterns frequently observed during incidents, such as typos of legitimate Windows processes, one-character executable names, numeric executable names, common malware names, and random hexadecimal strings. The tool

also supports filters for missing publisher information and non-operating-system components. These filters are useful because malicious binaries often lack publisher metadata and are not marked as operating system components. While these conditions do not prove maliciousness, they help reduce noise and prioritize files for collection and analysis.

The user can enable OpenTIP lookups with `--opentip` and VirusTotal lookups with `-v` or `--vt`. API keys are read from environment variables, avoiding hardcoded credentials in command lines or scripts. Lookup results are cached during execution to avoid repeated requests for the same hash. Finally, records with detections are highlighted and the user can choose to export results to CSV or JSON Lines for reporting, timeline analysis, or further processing.

4. Installation and Usage

`AmCache-EvilHunter` is available as an open-source project on `https://github.com/cristianzsh/amcache-evilhunter`. The tool requires Python 3.7 or higher and the installation can be performed with the following commands:

```
git clone https://github.com/cristianzsh/amcache-evilhunter.git
cd amcache-evilhunter
pip3 install -r requirements.txt
```

The main options of the proposed solution are summarized in Table 2. Basic `Amcache.hve` processing can be performed using:

```
python3 amcache_evilhunter.py -i path/to/Amcache.hve [OPTIONS]
```

Table 2. Main AmCache-EvilHunter options.

Option	Description
`-i, --input PATH`	Path to the `Amcache.hve` registry hive.
`--start YYYY-MM-DD`	Include only records on or after this date.
`--end YYYY-MM-DD`	Include only records on or before this date.
`--search TERMS`	Search for comma-separated, case-insensitive terms.
`--find-suspicious`	Filter records matching suspicious executable name patterns.
`--missing-publisher`	Filter records with missing publisher information.
`--exclude-os`	Include only files that are not marked as operating system components.
`--opentip`	Enable Kaspersky OpenTIP hash lookups. Requires `OPENTIP_API_KEY`.
`-v, --vt`	Enable VirusTotal hash lookups. Requires `VT_API_KEY`.
`--only-detections`	Show or save only files with detections.
`--json PATH`	Export results to JSON Lines.
`--csv PATH`	Export results to CSV.

To filter records by date range and search for a specific term:

```
python3 amcache_evilhunter.py -i Amcache.hve \
--start 2025-06-19 --end 2025-06-19 \
--search notepad
```

To identify suspicious executable names and export the results to CSV:

```
python3 amcache_evilhunter.py -i Amcache.hve \
--find-suspicious --csv suspicious.csv
```

To search suspicious records and query VirusTotal:

```
export VT_API_KEY=YOUR_API_KEY

python3 amcache_evilhunter.py -i Amcache.hve \
--find-suspicious -v
```

To query Kaspersky OpenTIP and show only malicious detections:

```
export OPENTIP_API_KEY=YOUR_API_KEY

python3 amcache_evilhunter.py -i Amcache.hve \
--opentip --only-detections
```

The project also includes a build script to generate standalone binaries for Linux and Windows systems:

```
chmod +x build.sh; ./build.sh
```

5. Demonstration

During the conference, `AmCache-EvilHunter` will be demonstrated using real-world `Amcache.hve` files collected from Windows systems. The demonstration will focus on practical DFIR workflows, showing how an analyst can move from raw Amcache data to suspicious records and threat intelligence results. A complete video demonstration about the tool's usage is available at `https://www.youtube.com/watch?v=BgxEUM797tA`.

The first part of the demonstration will show how to parse a complete `Amcache.hve` file and export its contents to CSV. This is useful when the analyst wants to preserve the parsed data, inspect it manually, or import it into other tools. Figure 2 shows the expected basic execution workflow.

Figure 2. Basic usage of AmCache-EvilHunter to parse an `Amcache.hve` file.

```
$ ./AmCache-EvilHunter.py -i Amcache1.hve --csv output.csv
```

SHA-1	Name	RecordDate	OS?
11eba7b1e26cc7d492a2c161ac48370811d0b01e	csrss.exe	2024-09-04T19:25:16.972623	Yes
3d26b0dc519e04999118f4a02ea8acd5f1db8feb	MicrosoftEdgeUpdate.exe	2024-09-04T19:25:16.988337	No
010db07461e45b41c886192df6fd425ba8d42d82	svchost.exe	2024-09-04T19:25:16.988337	Yes
e3e0f810ef3e0a36c13c97eee8246784fb7cbfca	TiWorker.exe	2024-09-04T19:25:17.004080	No
77f2e744c92417653b5abd6ccb3b5e521111979a	CompatTelRunner.exe	2024-09-04T19:25:17.092058	Yes
0646f86530602a6d5980ad9d460e45f77ef46399	DeviceCensus.exe	2024-09-04T19:25:17.092058	Yes
7b2e3990de386df08a049ecb611bc9ab17c1497c	winlogon.exe	2024-09-04T19:26:42.749811	Yes
647ff5972f7b41ceec7572ef051fd8dde39d9993	msedge.exe	2024-09-04T19:26:45.559471	No
ce8669d8826c8795115d58c62e726ae53943dce9	OneDriveSetup.exe	2024-09-04T19:27:08.364932	Yes
5d6102f5a170e982c7735bfc2b9c1a0a0d435fd1	msiexec.exe	2024-09-04T19:28:07.277472	Yes
2de3263c0aac35367b7db9de2b2ca05f20068a52	drvinst.exe	2024-09-04T19:28:07.321125	Yes
2ff161a1185b5716ade6b895127d561299e7cafe	OneDriveSetup.exe	2024-09-04T19:30:08.018908	No
cab6666a502e4c945f95f78d77e60e025f5c592f	ChromeSetup.exe	2024-09-04T19:33:08.934714	No

The second part will show how date filtering can help analysts focus on a specific incident window. For example, if suspicious activity happened on June 19, 2025, the analyst can restrict the output to records associated with that date:

```
python3 amcache_evilhunter.py -i Amcache.hve \
--start 2025-06-19 --end 2025-06-19 \
--csv output.csv
```

The third part of the demonstration will focus on suspicious executable identification. Analysts are often faced with a large number of executables and need to reduce noise quickly. The `--find-suspicious` option filters records based on known suspicious patterns, including common malware names, typos of legitimate Windows processes, one-letter or one-digit executable names, and random hexadecimal names. Figure 3 shows the expected execution workflow.

Figure 3. Suspicious artifacts identification.

```
$ ./AmCache-EvilHunter.py -i Amcache2.hve --find-suspicious
```

SHA-1	Name	RecordDate	OS?
11eba7b1e26cc7d492a2c161ac48370811d0b01e	csrss.exe	2024-09-04T20:04:09.746658	Yes
010db07461e45b41c886192df6fd425ba8d42d82	svchost.exe	2024-09-04T20:04:09.746658	Yes
4db5a8e237937b6d7b435a8506b8584121a7e9e3	svchost.exe	2024-09-04T20:12:49.216136	No
bd59d7c734ca2f9cbaf7f12bc851f7dce94955d4	123.exe	2024-09-04T20:19:22.164874	No

This feature is useful, for example, when two files named `svchost.exe` appear in the results: one marked as an operating system component and another located outside the expected system path. The second record becomes a good candidate for collection and deeper analysis, especially if it has no publisher information.

The fourth part will combine suspicious-record filtering with missing publisher and operating system component exclusion (Figure 4). This allows the analyst to focus on files that are more likely to be suspicious.

Figure 4. Suspicious artifacts identification.

```
$ ./AmCache-EvilHunter.py -i Amcache2.hve --find-suspicious --missing-publisher --exclude-os
```

SHA-1	Name	RecordDate	OS?
4db5a8e237937b6d7b435a8506b8584121a7e9e3	svchost.exe	2024-09-04T20:12:49.216136	No
bd59d7c734ca2f9cbaf7f12bc851f7dce94955d4	123.exe	2024-09-04T20:19:22.164874	No

Finally, the demonstration will show how `AmCache-EvilHunter` can enrich the output with threat intelligence information by using Kaspersky OpenTIP or VirusTotal, as shown in Figure 5. This is especially useful when the binary is no longer available on disk but the Amcache record still contains a SHA-1 hash.

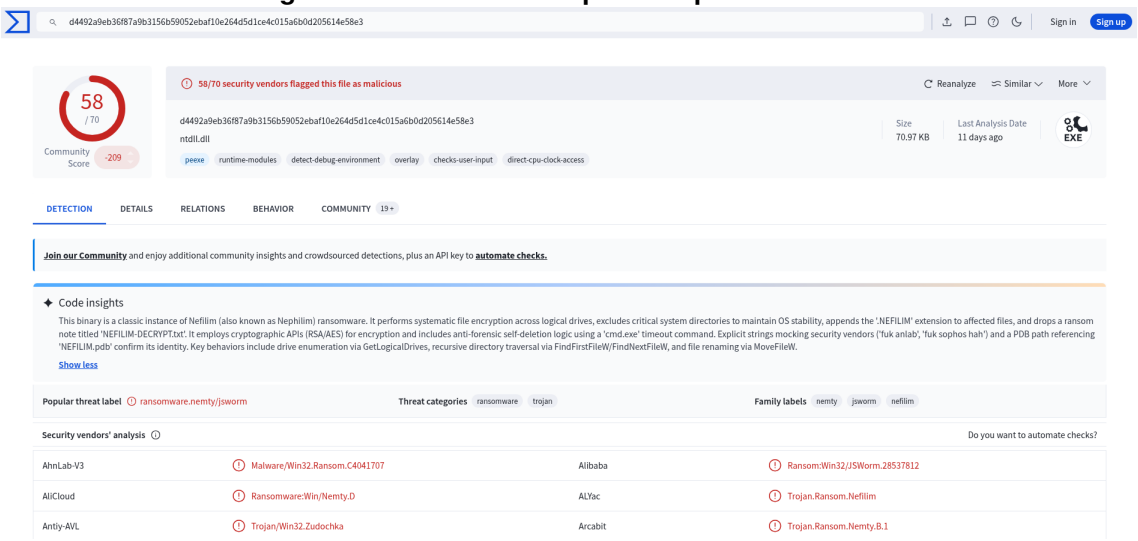
Figure 5. Threat intelligence lookup using Kaspersky OpenTIP.

```
$ ./AmCache-EvilHunter.py -i Amcache2.hve --find-suspicious --opentip --only-detections
```

SHA-1	Name	RecordDate	OS?	OT
4db5a8e237937b6d7b435a8506b8584121a7e9e3	svchost.exe	2024-09-04T20:12:49.216136	No	Malware
bd59d7c734ca2f9cbaf7f12bc851f7dce94955d4	123.exe	2024-09-04T20:19:22.164874	No	Malware

Figure 6 shows a VirusTotal query for a suspicious SHA-1 hash. Using this information, an analyst can retrieve the corresponding artifact from threat intelligence feeds for further static and dynamic analysis.

Figure 6. VirusTotal lookup for suspicious SHA-1.



6. Conclusion

`Amcache.hve` is a cornerstone artifact for Windows forensics. It captures rich metadata about executables, drivers, installed applications, and shortcuts, including full paths, SHA-1 hashes, compilation timestamps, publisher information, version details, and operating system component flags. Its value lies in preserving evidence of file presence and execution-related metadata that may remain available even after malicious tools are deleted from disk.

This paper presented `AmCache-EvilHunter`, an open-source tool designed to support Amcache-based triage during DFIR investigations. The tool parses `Amcache.hve` registry hives, filters records by date and keywords, identifies suspicious executable names, highlights files without publisher information, excludes operating system components, exports results to CSV and JSON, and integrates Kaspersky OpenTIP and VirusTotal lookups.

In practical incident response, these features help analysts reduce noise, prioritize suspicious binaries, generate indicators of compromise, and quickly decide which artifacts require deeper analysis. The tool is especially useful in cases involving deleted malware, renamed binaries, remote access tools, and suspicious drivers or executables that are no longer available on disk. As future work, we intend to expand support for additional Amcache keys, improve suspicious-pattern detection, add more threat intelligence providers, and support correlation with other Windows artifacts (e.g., Prefetch, ShimCache, UserAssist, and event logs). We also intend to improve the tool based on feedback from DFIR practitioners and real-world incident response scenarios.

References

Easttom, C., Butler, W., Phelan, J., Bhagavatula, R. S., Steuber, S., Rodriguez, K., Balkissoon, V. I., and Naseer, Z. (2024). *Windows Forensics*. Springer.

Gnanasekaran, V., Neudert, R., Heegaard, P. E., and Pernul, G. (2025). A role taxonomy in security-safety incident response. In *International Conference on Availability, Reliability and Security*, pages 286–304. Springer.

Jagathbala, R., Kumar, G. B., Geethanjali, D., and Nanthini, N. (2025). Forenxplorer: A smart digital forensic triage tool for fast and automated incident response. In *2025 10th International Conference on Smart Structures and Systems (ICSSS)*, pages 1–7. IEEE.

Joo, D., Lee, J., and Jeong, D. (2023). A reference database of windows artifacts for file-wiping tool execution analysis. *Journal of forensic sciences*, 68(3):856–870.

Lagny, B. (2019). Analysis of the amcache. *ANSSI-DFIRSummit*.

Neyaz, A. and Shashidhar, N. (2022). Windows prefetch forensics. In *Breakthroughs in Digital Biometrics and Forensics*, pages 191–210. Springer.

Souza, C. (2025). Forensic journey: hunting evil within amcache. Technical report, Kaspersky Securelist.

Souza, C. H., Pascoal, T., Neto, E. P., Sousa, G. B., SL Filho, F., Batista, D. M., and Silva, F. S. D. (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.

Tokarev, A. and Tokareva, V. (2023). Comparative analysis of amcache trace formation mechanisms in windows 10 and windows 11. In *2023 IEEE Ural-Siberian Conference on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, pages 289–292. IEEE.