8-17-2021

Forensic Artifact Finder (ForensicAF): An Approach & Tool for Leveraging Crowd-Sourced Curated Forensic Artifacts

Tyler Balon
University of New Haven

Krikor Herlopian
University of New Haven

Ibrahim Baggili
University of New Haven, ibaggili@newhaven.edu

Cinthya Grajeda-Mendez
University of New Haven

Publisher Citation

Tyler Balon, Krikor Herlopian, Ibrahim Baggili, and Cinthya Grajeda-Mendez. 2021. Forensic Artifact Finder (ForensicAF): An Approach & Tool for Leveraging Crowd-Sourced Curated Forensic Artifacts. In The 16th International Conference on Availability, Reliability and Security (ARES 2021). Association for Computing Machinery, New York, NY, USA, Article 43, 1–10. DOI:https://doi.org/10.1145/3465481.3470051

Comments

This is the Author's Accepted Manuscript.
Article part of the International Conference Proceeding Series (ICPS), *ARES 2021: The 16th International Conference on Availability, Reliability and Security*, published by ACM.

Forensic Artifact Finder (ForensicAF): An Approach & Tool for Leveraging Crowd-Sourced Curated Forensic Artifacts

Tyler Balon
University of New Haven
Connecticut Institute of Technology
United States

Krikor Herlopian
University of New Haven
Connecticut Institute of Technology
United States

Ibrahim Baggili
University of New Haven
Connecticut Institute of Technology
United States

Cinthya Grajeda-Mendez
University of New Haven
Connecticut Institute of Technology
United States

ABSTRACT

Current methods for artifact analysis and understanding depend on investigator expertise. Experienced and technically savvy examiners spend a lot of time reverse engineering applications while attempting to find crumbs they leave behind on systems. This takes away valuable time from the investigative process, and slows down forensic examination. Furthermore, when specific artifact knowledge is gained, it stays within the respective forensic units. To combat these challenges, we present ForensicAF, an approach for leveraging curated, crowd-sourced artifacts from the Artifact Genome Project (AGP). The approach has the overarching goal of uncovering forensically relevant artifacts from storage media. We explain our approach and construct it as an Autopsy Ingest Module. Our implementation focused on both File and Registry artifacts. We evaluated ForensicAF using systematic and random sampling experiments. While ForensicAF showed consistent results with registry artifacts across all experiments, it also revealed that deeper folder traversal yields more File Artifacts during data source ingestion. When experiments were conducted on case scenario disk images without apriori knowledge, ForensicAF uncovered artifacts of forensic relevance that help in solving those scenarios. We contend that ForensicAF is a promising approach for artifact extraction from storage media, and its utility will advance as more artifacts are crowd-sourced by AGP.

KEYWORDS

Cyber Forensics, Digital Forensics, Artifacts, Triage, Autopsy, AGP, CuFA, Artifact Genome Project

ACM Reference Format:
Tyler Balon, Krikor Herlopian, Ibrahim Baggili, and Cinthya Grajeda-Mendez. 2021. Forensic Artifact Finder (ForensicAF): An Approach & Tool for Leveraging Crowd-Sourced Curated Forensic Artifacts. In *The 16th International Conference on Availability, Reliability and Security (ARES 2021), August 17–20, 2021, Vienna, Austria.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3465481.3470051

1 INTRODUCTION

Digital Forensics (DF) emerged over the last twenty years as an independent branch of forensic science through the empowerment of academia and industry. Practitioners rely on advancements in the domain to aid in deciphering, segmenting, and analyzing the cascade of events in forensic artifacts. The forensic Acquisition, Authentication, & Analysis (AAA) of digital evidence is imperative for its admissibility into the court of law. Technology's widespread use has led to challenges and opportunities in performing forensic analysis due to the volume, velocity and variety of data [6, 39].

DF is backed by a growing market that is estimated to grow from USD $1.72 billion in 2018 to $4.24 billion by 2025 [27]. This growth has led to a variety of DF tools both commercial and open source that practitioners can leverage to provide "layers of abstraction" to aid in upholding integrity when acquiring, authenticating, and analyzing data [37].

However, as technology becomes more complex and storage mediums expand to larger sizes, tools have not kept up with the need for faster analysis. Disk sizes continue to grow, while emerging systems such as: the Internet of Things (IoT), Personal Electrical Vehicles (PEV), and embedded devices, complicate the market. Examiners are no longer able to keep up due to limited time and resources with current investigative techniques that largely rely on manual processing [31]. This manual processing has encouraged isolation between forensic examiners, which has in turn housed their knowledge in their respective forensic units.

To share knowledge, early efforts for crowd sourcing forensic data have been explored. This started by collecting publicly available information on widely popular websites such as 4Chan, Reddit, & YouTube [17]. However, in 2014, the AGP was founded as a shared repository of Curated Forensic Artifacts (CuFA) (vetted forensic artifacts). A CuFA is an artifact of forensic relevance such as a file that stores the chat history of a communication application like Skype. AGP aids practitioners in locating potential evidence that may have been uncovered through past research by a community of academics, industry partners, and practitioners [26].

Our work outlines the development of a promising approach, implemented through a plugin for Autopsy (a popular open source

ARES 2021, August 17–20, 2021, Vienna, Austria

ACM ISBN 978-1-4503-9051-4/21/08…$15.00
https://doi.org/10.1145/3465481.3470051

DF toolkit) to aid in automating artifact triage of digital devices. We make the following contributions:

- We conceptualize and implement Forensic Artifact Finder (ForensicAF) as an open source Autopsy plugin that aids in the extraction of forensically relevant artifacts from a storage medium and presenting them in a Hypertext Markup Language (HTML) document
- We leverage a vetted, crowd-sourced repository of CuFAs found in AGP to help practitioners locate artifacts that have been deemed as noteworthy by past researchers
- We provide primary results that evaluate ForensicAF's ability to locate artifacts on digital media, and compare analysis times through multiple trials to gauge the speed and usefulness of our approach
- We explore the usefulness of artifacts uncovered using ForensicAF that aid in solving publicly available DF scenarios
- We share our implemented approach publicly via GitHub

The rest of this paper is organized as follows. In Section 2 we share our ForensicAF approach and implementation. Section 3 presents our evaluation, followed by the results in Section 4. We discuss our findings in Section 5 and follow them with related work in Section 6. We conclude in Section 7 and build a path for future work in Section 8.

2 FORENSICAF

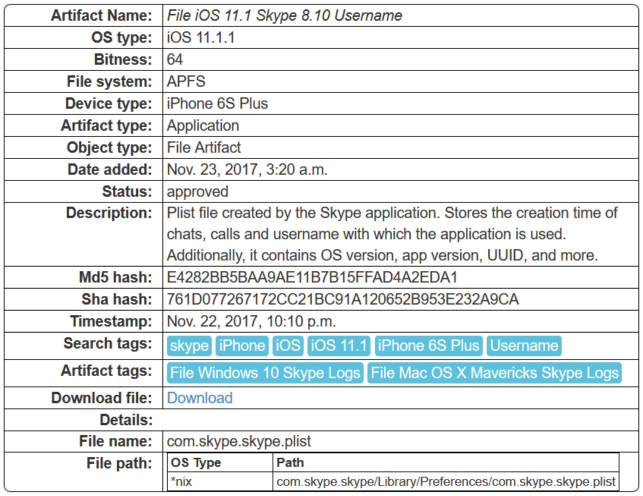
ForensicAF is our proposed approach for finding forensic artifacts on a system under investigation. Figure 2 illustrates the difference between ForensicAF and traditional manual artifact extraction methods. Traditional manual methods can equate to hours of research via search engines, looking up archived information in databases, as well as pulling out old files and reference materials to locate past research. Forensic examiners do not have the luxury of time in terms of downloading applications, reverse engineering them, and examining the unique and encoded artifacts that they produce. Lastly, the traditional method relies on individuals in forensic units that have gained experience in finding artifacts during investigations, where knowledge is not shared and crowd-sourced.

ForensicAF on the other hand employs AGP to aid in automating the process of finding artifacts during forensic examination, while potentially cutting down hours of research – all thanks to the availability of vetted, crowd-sourced artifacts. As shown in Figure 2, contributors from academia, law enforcement, and industry all share an array of specific artifact findings with their respective forensic metadata.

ForensicAF was implemented as an Ingest Module based on the Autopsy Module Development and Application Programming Interfaces (API) reference documentation. While the plugin is not an end-all solution to DF analysis, it serves as an automation tool which employs crowd-sourced CuFAs [30] to report interesting finds.

2.1 Artifact Analysis

AGP [26] allows users to contribute artifacts to their database. While artifacts submitted can be a variety of types, our work employs two common types: (1) File Artifacts and (2) Windows Registry Artifacts.



<table>
<tr><td><b>Artifact Name:</b></td><td><i>File iOS 11.1 Skype 8.10 Username</i></td></tr>
<tr><td><b>OS type:</b></td><td>iOS 11.1.1</td></tr>
<tr><td><b>Bitness:</b></td><td>64</td></tr>
<tr><td><b>File system:</b></td><td>APFS</td></tr>
<tr><td><b>Device type:</b></td><td>iPhone 6S Plus</td></tr>
<tr><td><b>Artifact type:</b></td><td>Application</td></tr>
<tr><td><b>Object type:</b></td><td>File Artifact</td></tr>
<tr><td><b>Date added:</b></td><td>Nov. 23, 2017, 3:20 a.m.</td></tr>
<tr><td><b>Status:</b></td><td>approved</td></tr>
<tr><td><b>Description:</b></td><td>Plist file created by the Skype application. Stores the creation time of chats, calls and username with which the application is used. Additionally, it contains OS version, app version, UUID, and more.</td></tr>
<tr><td><b>Md5 hash:</b></td><td>E4282BB5BAA9AE11B7B15FFAD4A2EDA1</td></tr>
<tr><td><b>Sha hash:</b></td><td>761D077267172CC21BC91A120652B953E232A9CA</td></tr>
<tr><td><b>Timestamp:</b></td><td>Nov. 22, 2017, 10:10 p.m.</td></tr>
<tr><td><b>Search tags:</b></td><td>skype
iPhone
iOS
iOS 11.1
iPhone 6S Plus
Username</td></tr>
<tr><td><b>Artifact tags:</b></td><td>File Windows 10 Skype Logs
File Mac OS X Mavericks Skype Logs</td></tr>
<tr><td><b>Download file:</b></td><td>Download</td></tr>
<tr><td><b>Details:</b></td><td></td></tr>
<tr><td><b>File name:</b></td><td>com.skype.skype.plist</td></tr>
<tr><td><b>File path:</b></td><td><table><tr><th>OS Type</th><th>Path</th></tr><tr><td>*nix</td><td>com.skype.skype/Library/Preferences/com.skype.skype.plist</td></tr></table></td></tr>
</table>

Figure 1: Example iOS Skype File Artifact in AGP. ***Note: Omits artifact metadata and showcases artifacts representation in AGP.***

AGP is built with CyBOX [9] as an underlying data model. CyBOX is "a standardized language for encoding and communicating high-fidelity information about cyber observables". CyBOX extends the possibilities of attribute types to store as metadata for each artifact. Per AGP, File Artifacts are considered "Artifacts that take the form of a file with an extension" and Windows Registry Artifacts are considered, "Artifacts within the Windows Registry". Once an artifact is approved by AGP administrators, the artifact can be considered validated and becomes a CuFA.

File Artifacts in AGP contain a multitude of fields, including, but not limited to: artifact name, artifact type, type of device the artifact was found on, file hashes of the artifact, discovery information, file name, file extension, file format, file path, creation and modification dates, etc. An example iOS Skype Username File Artifact from AGP is shown in Figure 1. ForensicAF uses the file name, file extension, and file path to locate interesting finds on the disk to report. If a CuFA does not have a file name, the entire path is considered an artifact.

Windows Registry Artifacts in AGP contain significantly less fields for metadata compared to File Artifacts. Users that create a Windows Registry Artifact can submit up to, but not limited to: artifact name, artifact type, type of device the artifact was found on, hashes of the artifact, discovery information, a Windows Registry Key, registry values, modification times, sub-keys, etc. ForensicAF uses the Windows Registry Key information to locate known Windows Registry Artifacts on a disk.

Despite AGP storing metadata for each artifact, anomalous artifact metadata may still occur. While AGP has significant validation efforts in place to ensure each artifact is curated, artifact metadata may be vague, erroneous, or incomplete. Instances as such may produce a large number of false positives during analysis. To improve accuracy of artifact finding in ForensicAF, several considerations were adhered to:

- Artifacts may not have file names

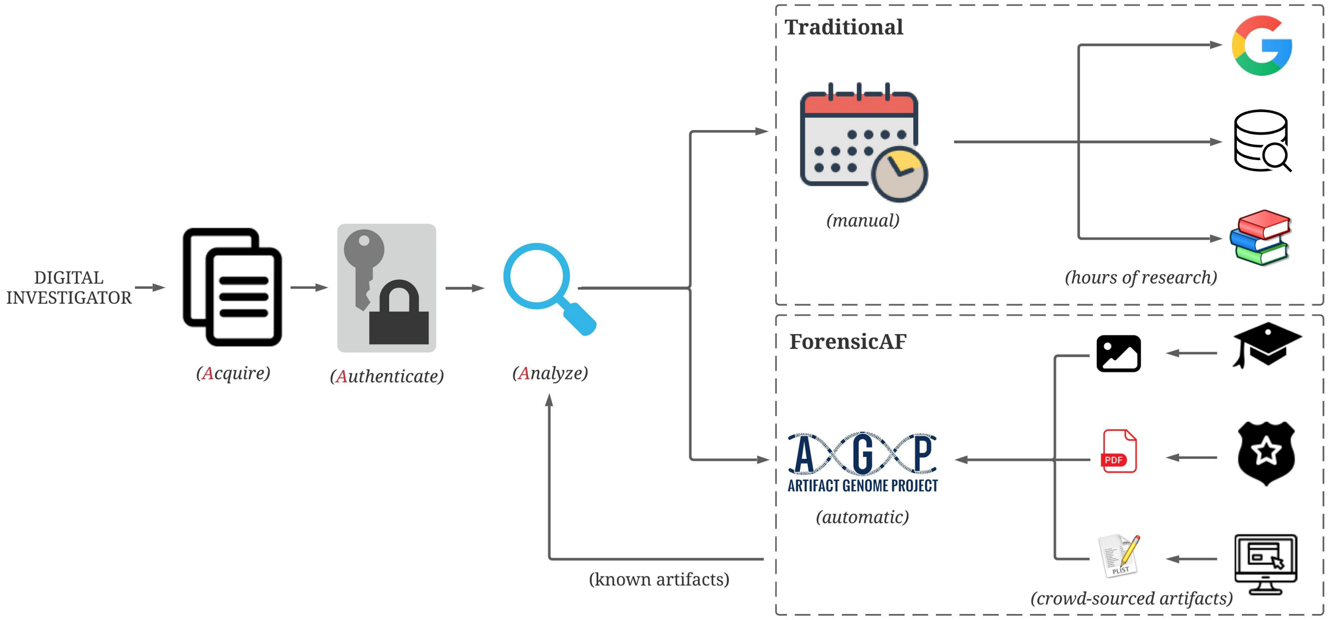


Figure 2: Traditional Manual Artifact Analysis v.s. ForensicAF

  - Entries may be a file path to a known directory. For example, the image path to the photos on an iOS backup, that would contain random file names
- Artifact details may be too vague
  - An artifact cataloged to be in a common directory of a device may return a false positive
  - File names may be common in multiple directories
- User submission errors may still happen
  - Users may mis-type information when submitting an artifact
  - Users may supply misinformation while submitting an artifact

Since ForensicAF is dependent on the amount of cataloged CuFAs when ingesting data sources, we explored user data generated on AGP. At the time of writing, there were 484 users registered, belonging to 256 different organizations. Of those organizations, they spanned geographically over 52 different countries and other territories. The majority of organizations are registered from The United States. At the time of writing, AGP hosted (n=1130) CuFAs, with the largest contributing sector being academia. AGP is actively maintained with new features being added to the project and new CuFAs being created.

2.2 Plugin Construction

ForensicAF was implemented for Autopsy versions 4.16 and greater. We selected Autopsy for its open source nature and wide adoption. Per the Module Development Overview and API reference, ForensicAF is written as an Ingest Module, which can be run in Autopsy when a new data source is added to a case. Ingest Modules will process every file that is in a data source, allowing ForensicAF to search for CuFAs provided by AGP.

The plugin is written in Python 2.7. Autopsy utilizes Jython, a Java implementation of Python that will convert the Python plugin into Java byte code, allowing it to run on the Java Virtual Machine (JVM) [10]. ForensicAF is limited to Python 2.7, and cannot include libraries with native functions. We use several external libraries, built as native Java, or Jython code as shown in Table 1.

Table 1: Libraries used to create ForensicAF

Libraries imported
import org.apache.poi.xssf.usermodel
from com.williballenthin.rejistry import RegistryKey
from com.williballenthin.rejistry import RegistryValue
from com.williballenthin.rejistry import RegistryHiveFile
from com.williballenthin.rejistry import RegistryParseException

ForensicAF runs on Windows, Linux, macOS, and any system capable of running Autopsy & the JVM. Our tool is capable of searching for files and registry keys on a system. We discuss plugin usage in Section 2.4. Systems without a Windows Registry will have that component of the plugin ignored. XSSFWorkbook [5] is used to parse the AGP artifact data. File analysis on the device is performed using all native Python functions (traversal of directories on system, copying files, etc). Registry analysis uses Rejistry [8], to access the hive file and search for CuFAs.

Due to conflicting functions between versions of Autopsy, we settled on designing ForensicAF to support version 4.16 or greater, as recommended by developers we were in contact with through The Sleuth Kit Forum[1].

[1] Autopsy Forum (url: https://sleuthkit.discourse.group)

2.3 Algorithm Design

This subsection breaks down the method used to search for Windows Registry CuFAs and the algorithm that was implemented to search for files during digital forensic analysis. On non-Windows systems, the portion of searching for Windows Registry CuFAs is ignored, and only the File Analysis occurs. AGP is capable of storing artifacts for a variety of Operating System (OS)s, including, but not limited to: Windows, macOS, iOS, Linux and Android.

To locate Windows Registry CuFAs on a data source, we search for the registry hive containing the registry key in the artifacts downloaded from AGP. ForensicAF makes temporary copies of the registry hive to a temp directory before searching for the key. This ensures that data is not deleted nor altered during this process for forensic integrity. We search the hive for the key using Rejistry [8] and output the results to the practitioner.

As explained, Ingest Modules in Autopsy will analyze each file in a data source as Autopsy indexes the data source and displays results to the practitioner. We loop through each artifact in our AGP CuFAs list and search for them using the function `findFiles()`, an Autopsy provided method. If there is nothing found at a specific path, we traverse up one directory and search again. The user can define how many directories to traverse backwards from the location stored in AGP, this ensures maximum discovery of important finds on a device. We implement Algorithm 1, into ForensicAF.

Algorithm 1 File Artifacts

```
 1: procedure FINDCUFAs(t, cufaPaths, cufaNames)
 2:     $i \leftarrow 0$                                   ▷ iterator of traversed
 3:     $traverse \leftarrow t$                            ▷ num dirs to traverse
 4:     $paths \leftarrow cufaPaths$                       ▷ list of CuFA file paths
 5:     $names \leftarrow cufaNames$                       ▷ list of CuFA file names
 6:     $results \leftarrow []$                            ▷ blank list for results
 7:     for $paths, names = 1, 2, \ldots, N$ do
 8:         $path \leftarrow paths$                        ▷ current file path
 9:         $nm \leftarrow names$                          ▷ current file name
10:         for $i$ in range($traverse$) do
11:             $\%foundFiles\%$ = findFiles(path, nm)
12:             if count($foundFiles$) != 0 then
13:                 $results \leftarrow \%foundFiles\%$
14:                 break
15:             else
16:                 path = path[path[1:].find('/')+1:]
17:             end if
18:         end for
19:     end for
20: end procedure
```

Algorithm 1 explains our Find Artifacts procedure we use to locate CuFAs on a data source. Lines 2 - 6 initialize the iterator, how many directories to traverse, the paths/names we will search for, and a place to store results. Line 7 iterates each file path and name in the list provided by AGP. We then begin to traverse upwards **N directories** from the intended location, according to AGP, on line 10. When we call `findFiles`, we use '%' to search for leading or trailing characters added to the file name listed in AGP. If a file is found, we add it to the results and continue searching the remainder of our files, otherwise, we traverse up a directory and search again, continuing until we have traversed as many times as user defined. Depending on the traversal level, we will search for the file traversed up that number of directories, and to all subfolders in that directory. CuFA paths and names are searched for separately, to see if a known file is in a different path, or a known path contains different files. Some CuFA entries are simply known paths, not specific files.

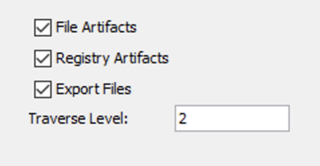
2.4 Usage

In this section, we provide basic usage information for ForensicAF. The plugin is available on GitHub[2] as an open source tool that can be modified to meet user needs.

How To: Install ForensicAF. After Autopsy is installed on the system, navigate to '`Tools > Python Plugins`' to install ForensicAF. The folder, '`%appdata%\Roaming\autopsy\python_modules`' contains python plugins for Autopsy. Create a new folder named '`ForensicAF`' and move '`ForensicAF.py`' to the new folder. This allows Autopsy to use the Ingest Module when re-launched and a data source is added.

How To: Download AGP CuFAs. Users should register online at '`agp.newhaven.edu`'. AGP administrators will manually approve new accounts [26]. Specific artifacts can be searched via AGP using '`Artifact > Search`' or all known artifacts in AGP can be retrieved using the wildcard '`*`'. Select the '`Export Results`' button to download the file '`SearchResults.xls`'. Move the file '`SearchResults.xls`' to the '`ForensicAF`' folder, found at the path: '`%appdata%\Roaming\autopsy\python_modules`', do not change the file name.

How To: Use ForensicAF. Once Autopsy is restarted after ForensicAF is installed, it can be run as an Ingest Module to analyze new data sources. Navigate to '`Tools > Run Ingest Modules`' to select '`ForensicAF`'. The available settings can be seen in Figure 3. '`File Artifacts`' will enable searching the data source for file artifacts from the AGP exported search query. '`Registry Artifacts`' will enable searching the data source for registry items, and should be disabled if the image is from a non-Windows machine. '`Export Files`' will enable exporting the files when the report is generated. '`Traverse Level`' will allow the user to select how many directories up the plugin should traverse to search for files, relative to the path in AGP. As this value is increased the time to run the plugin will increase.



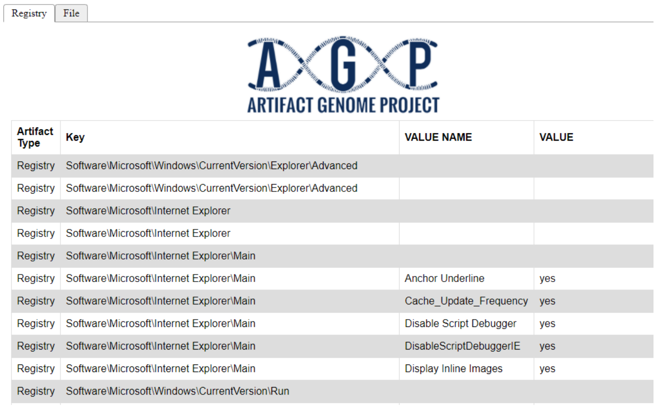
☑ File Artifacts
☑ Registry Artifacts
☑ Export Files
Traverse Level: 2

Figure 3: ForensicAF Ingest Module settings

Usage: What to expect. Depending on '`Traverse Level`' and the size of the data source, the plugin can take several minutes to hours. Once ForensicAF finishes running on the data source, a folder

[2] ForensicAF GitHub (url: https://github.com/unhcfreg/AGP-Autopsy-Plugin)

named ‘`files extracted`’ and a report document in HTML named ‘`report<UNIX_TIMESTAMP>`’ will be saved to the plugin folder in python_modules. Users can retrieve these files by navigating to the directory ‘`%appdata%\Roaming\autopsy\python_modules`’. An example report generation can be seen in Figure 4.



Artifact Type	Key	VALUE NAME	VALUE
Registry	Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced		
Registry	Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced		
Registry	Software\Microsoft\Internet Explorer		
Registry	Software\Microsoft\Internet Explorer		
Registry	Software\Microsoft\Internet Explorer\Main		
Registry	Software\Microsoft\Internet Explorer\Main	Anchor Underline	yes
Registry	Software\Microsoft\Internet Explorer\Main	Cache_Update_Frequency	yes
Registry	Software\Microsoft\Internet Explorer\Main	Disable Script Debugger	yes
Registry	Software\Microsoft\Internet Explorer\Main	DisableScriptDebuggerIE	yes
Registry	Software\Microsoft\Internet Explorer\Main	Display Inline Images	yes
Registry	Software\Microsoft\Windows\CurrentVersion\Run		

Figure 4: Example ForensicAF Report

3 EVALUATION

We evaluated our approach using Random and Systematic sampling of CuFAs found across different data sources. We relied on both, a data source we created and scenario images that mock real digital forensic investigations. The results of our evaluation can be found in Section 4. We analyze each data source twice per the traversal level we specify, to compare the speed of our approach. During our evaluation phase, we used the same hardware for each test (See Apparatus in Table 2).

To evaluate Systematic Sampling, we built a disk image of known artifacts, which we placed on a clean installation of Microsoft Windows 10. The disk image contained (n=100) artifacts, known by AGP, and that were intentionally placed on the system for ForensicAF to find. We tested the ForensicAF’s ability to traverse up 0, 1, 2, and 3 levels from the path specified by AGP. Each test was run twice to ensure the validity of the results and to compare the speed of each run.

Table 2: Hardware Used for Evaluation of ForensicAF

Component	Manufacturer	Model	Details
OS	Microsoft	Windows	v10x64
CPU	Intel	i7-3770S	3.10GHz
RAM	Intel	DDR3	8GB
Disk	Intel	64GB	HDD

To evaluate Random Sampling, we employed data sources from DigitalCorpora[3] and Stevenson University’s 2016 Black T-Shirt Cyber Forensic Challenge[4] to analyze with ForensicAF. DigitalCorpora provides scenario based disk images, memory dumps, and network packages of user created cases. For all sources, we compared our results from ForensicAF to the provided solution guide(s). In our evaluation, we downloaded the entire list of CuFAs from AGP, and then used them in ForensicAF. We then manually explored the reported findings to explore if any artifacts of forensic relevance were returned. *It is important to note that these tests were ran without any apriori knowledge of artifacts that may be useful i.e. we did not purposefully create artifacts in AGP that were relevant to the examined scenarios.* We ran the plugin two times per data source, at traverse levels 0 and 2 to ensure the validity of the results and to compare the speed of each run.

4 RESULTS

We tested ForensicAF’s efficacy based on the evaluation outlined in Section 3. A series of Systematic and Random Sampling experiments were performed.

4.1 Systematic Sampling

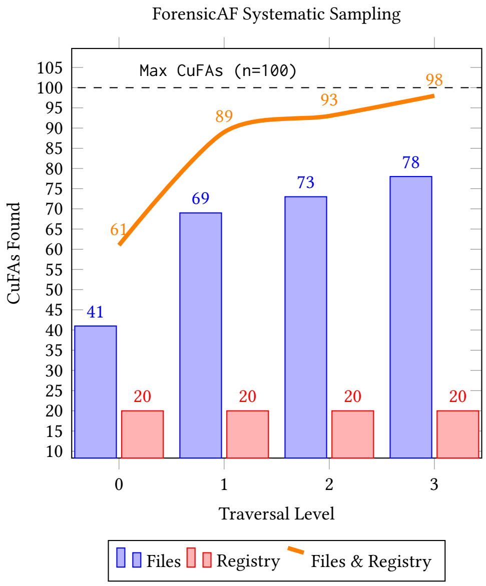


Figure 5: Autopsy results of ForensicAF using known manually placed CuFAs that are listed in AGP onto a data source

During our Systematic Sampling experiments, we installed Windows 10 on a `450GB` Hard Disk Drive (HDD) and placed *100 artifacts* on the device to test the accuracy of ForensicAF. We ran the experiment using different traversal levels.

In Figure 5, we present the findings of running ForensicAF at different traversal levels on a manually created data source. There were 100 CuFAs placed on the data source, as denoted by the dotted line at `y = 100`. As seen by the line *Files & Registry*, as we allowed ForensicAF to traverse up more directories from the path in AGP, we approached finding 100 CuFAs. It is noteworthy that the registry analysis portion of the plugin consistently found 20 results, as shown by *Registry* on the graph. Given the registry keys are in a set of predetermined files by Microsoft Windows, the plugin only has to search a specific location to look for registry CuFAs in a predictable fashion.

[3] DigitalCorpora (url: https://digitalcorpora.org)
[4] CyberWatch West (url: https://tinyurl.com/3stdjljj)

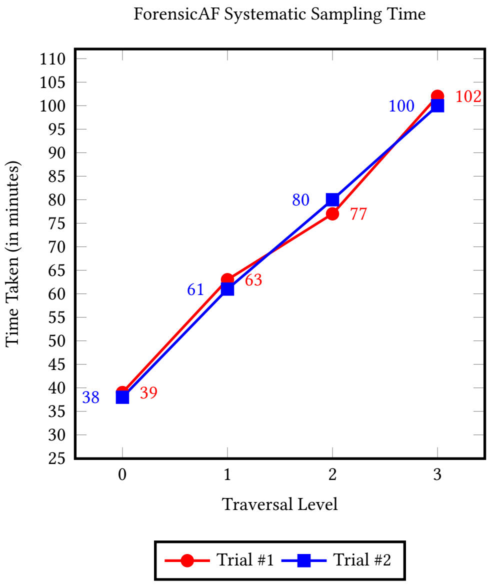


Figure 6: Time for ForensicAF Ingest Module to complete analysis on Systematic Sampling using AGP CuFAs

We compared the time taken for ForensicAF to analyze the 450GB data source at different traversal levels in Figure 6. At *Level 0*, the plugin takes an average of 38min, 30 seconds to analyze the data source. At *Level 3*, the plugin takes 1hr, 41min. As seen by the data in Figure 6, each trial completes in a similar time frame of ± 3min when the Ingest Module is ran on the same data source.

4.2 Random Sampling

Table 3: Data Sources used in Random Sampling

Data Source	Identifier	Disk Size	OS
M57-Jean	M57	3 GB	Windows XP
domexusers	DU	4.2 GB	Windows XP
Black T-Shirt	BT	10.6 GB	Windows 7
Lone Wolf	LW	15 GB	Windows 10

For the Random Sampling portion of our evaluation, we use DigitalCorpora and Stevenson University's data sources. The data sources vary in host OS and disk size. From DigitalCorpora, we use: "Lone Wolf", "M57-Jean", and "nps-2009-domexusers". From Stevenson University, we use the 2016 Black T-Shirt Challenge data source: "Black T-Shirt". The data source names, identifier we use on the graph, disk size, and OS can be found in Table 3. In Figure 7, we divide the graph into *Level 0* and *Level 2* based on the traversal level. The orange line denotes total artifacts found. Consistent with our Systematic Sampling, the same number of registry items are found regardless of the traversal level, however, at *Level 0*, no files are found in any of the data sources.

Unlike in Figure 6, we use a bar graph to display the timing of the Random Sampling in Figure 8. The Random Sampling Time in Figure 8 is an average of two trial runs, as listed in Table 4. We display this data in this format as there is no correlation between times for each data source. Our findings show that as the size of the data source increases, the time to run ForensicAF increases exponentially. When comparing LW to the 450GB data source used in our Systematic Sampling, LW at 15GB takes approximately 28min at *Level 2* to complete, while our manually created data source takes on average 78min, 30sec.

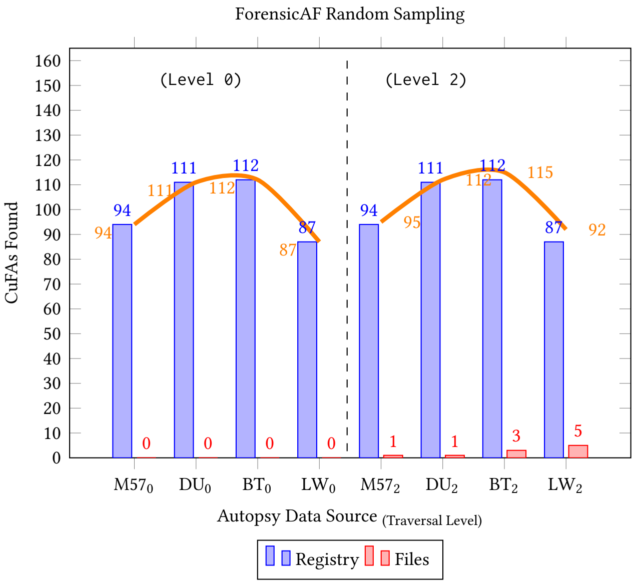


Figure 7: Autopsy results of using ForensicAF to find AGP CuFAs on a data source provided by DigitalCorpora

Table 4: Random Sampling Trial Timings (in seconds)

Identifier	$T1_0$ / $T1_2$	$T2_0$ / $T2_2$	AVG_0 / AVG_2
M57	73 / 158	79 / 172	76 / 165
DU	85 / 172	79 / 180	82 / 176
BT	461 / 904	471 / 922	466 / 913
LW	585 / 1676	593 / 1696	589 / 1686

As demonstrated across our evaluation, Registry Artifacts are consistently detected, while File Artifacts rely on the traversal level. In Section 5 we discuss the significance of the Random Sampling data source findings and review the different processing times when using ForensicAF.

5 DISCUSSION

Throughout our evaluation of ForensicAF, we learned that in each trial of Systematic and Random sampling we performed, regardless of the traversal level, the registry results were always consistent across the data source. Because registry look-ups depend on [8], we simply query the registry for keys in AGP and output the results to the report. The number of File Artifacts increases as the traversal level does, and if a file is not found in one location as expected by AGP, we traverse up a directory then search again for the file. This allows us to broaden our search from the expected location as many directories up as the user specifies when running the ForensicAF.

In the Systematic Sampling, we built a data source that had 100 expected CuFAs from AGP. In our results, at Level 3 traversal, we

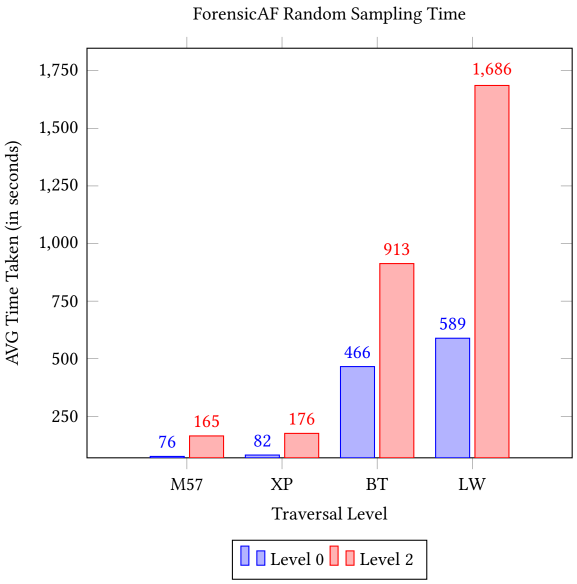


Figure 8: Time for ForensicAF Ingest Module to complete analysis on Random Sampling using AGP CuFAs

were only able to find 97 of these artifacts. When creating the data source, we purposely made anomalous artifacts, that should not be detected, to ensure the integrity of ForensicAF. These altered artifacts were not detected for a number of reasons:

- *file name was altered* - When we call `findFiles` in Algorithm 1, we search using '%' to find files that include the name but may have leading/trailing characters attached. If a file name is altered, ex: `log_out.txt` to `out_log.txt`, the plugin does not detect this as an artifact.
- *traversal level changed* - When adding artifacts to the data source, some artifacts were placed in different directories than intended to ensure the plugin could find them by increasing the traversal level.
  - Artifacts moved 1-3 levels up were found
  - Artifacts moved beyond 1-3 levels were ignored

During Random Sampling, the data sources provided by Digital-Corpora were selected at random. We did not have apriori knowledge if any artifacts on those data sources would be found, nor if they would support the investigation. Upon further review of the solution guide(s) for "Lone Wolf", "M57-Jean", and "Black T-Shirt": we find that ForensicAF locates artifacts, that per the solution guide, would assist in solving the case, as shown in Table 5.

Table 5: Random Sampling Relevant Results

	Identifier	Artifact Type	Details
1	M57	Registry	Messaging app info
2	M57	File	Recently used apps
3	BT	Registry	Computer usernames
4	BT	File	Recently used apps
5	BT	File	Contains browser history
6	LW	File	Recently used apps
7	LW	File	Contains browser history

Results 1 & 2 for *M57* include messaging information between two suspects, showcasing malicious plans between the suspects. Results 3, 4, and 5 for *BT* provides data about the users on the system and their computer usage. Results 6 & 7 for *LW* show recently used applications and browser history search results that pertain to the suspect's activities, providing information about incriminating search history. Each artifact listed in Table 5 directly relates to the solution guide(s) provided by the data source creators. This is significant, as the goal of ForensicAF is to aid in automating the analysis process by extracting forensically relevant artifacts.

As shown in Section 4, as a user selects to traverse higher, the time significantly grows for the plugin to run. Of course, hardware specifications may either improve or slow down ForensicAF.

6 RELATED WORK

With an increase of successful cyber-attacks threatening worldwide financial and personal security, digital forensics is at the forefront of incidence response. Evidence retrieved during an investigation is often large and can take days to fully analyze. The motivation behind ForensicAF is to aid practitioners in the analysis step and provide them with additional tooling at their disposal. Our work is within scope of the recent needs analysis findings explored in [29]. While we recognize that recent DF work is exploring the use of Artificial Intelligence (AI) and Machine Learning (ML) in the analysis of digital evidence [43], we deem it as out of this paper's scope, since our approach does not employ ML and AI techniques. This section explores work related to improving the forensics analysis process and highlights some past efforts in automation.

6.1 Forensic Software & Framework

Research by [24] suggested that many challenges in current forensic software stem from lack of standardization and standardized data formats. Further needs-analysis style surveys amongst practitioners, industry, and academia have agreed that standardization and formalization is a major weakness in digital forensic software [11, 35]. This challenge also extends to the development methodologies used to write these tools. Current forensic software uses a variety of different languages and libraries to function. Forensic tools are written in C, C++, Java, Perl, Python, and even proprietary languages such as Guidance Software's EnScript. These tools are capable of running on Microsoft Windows, Apple Macintosh, and Linux based OS.

Given the broad range of tools available that run on different hosts and developed in different languages, future forensic software should work towards creating frameworks that enable cross-language, cross-platform development. The paper [24] notes that frameworks that enable these features are the methods utilized by other types of development communities, while digital forensic tool developers have failed to follow suit. Widely used software that runs on a variety of hosts employ these techniques, such as: Apache2's module system[5]. Frameworks should include standardized processing models, cross-language API, and straightforward data marshaling. The authors in [24] call for frameworks tailored to DF that would allow plug-in file systems, processing of sectors, Internet Protocol (IP) packets, bytesteam "objects", timestamps,

[5] Apache2 Module System (url: https://tinyurl.com/1tubc38l)

email addresses, proper names, and so on. This, coupled with correlation subsystems, object-based hierarchical storage, and enhanced output subsystems could be used for interactive reports, visualization, and most importantly automated event systems.

In [11], the authors discuss unaddressed issues with forensic tool development such as volume and scalability, and standardization approaches. Efforts to examine and analyze existing analysis tools have also been explored [14]. A general consensus from these works is that analysis tools need to be further advanced and standardized, especially to deal with systems at scale. A callback model has been suggested to be the basis of plugins in digital forensic frameworks. The paper notes that [13]'s SleuthKit and [23]'s fiwalk.py provide limited callbacks but the APIs do not allow sufficient reporting and correlation.

Available frameworks for DF include: PyFlag [16], Open Computer Forensics Architecture (OCFA) [1], and Digital Forensics Framework (DFF) [21]. The popular "aaS" or "as a Service" model has made its way to DF as well, as seen in [45]. Work on DF frameworks for cloud computing has has also been explored [4, 19, 41, 42, 44]. However, these do not provide the necessary research tools that the experts need to generate goal-directed algorithms. They generate unnecessary and impertinent details during an investigation. The authors of [23] reaffirm that should a framework be developed that enables workflow automation, developers of DF software products will be able to focus more on algorithm enhancement, and less on specific DF details. They go on to note that commercial products such as NetIntercept [18], FTK [3], and Encase [2] could leverage these frameworks.

While tooling can be improved through standardization and frameworks, faster analysis still remains of paramount importance in digital forensic investigations. Time-lining has become an important consideration for forensic investigators as they aid in large investigations, as seen by tools like [34].

6.2 Faster Analysis

To improve forensic examination speeds, a number of research thrusts have been established in the DF domain, most notably:

- **Forensic triage:** Digital forensic investigations rely on digital triage which comes as both live and post-mortem triage. Authors in [32] explain that during live triage, rapid extraction of evidence from all available sources while post-mortem triage is conducted in laboratory settings with the goal of collecting evidence on a seized device. Forensic triage is an approach that helps prioritize digital evidence, and has been implemented in digital forensic tooling [7, 25]. The approach provides a solution for a case backlog problem, yet present tools have limitations in forensic use [24, 32, 38].
- **Stream-based disk forensics:** Byte-stream, another digital forensics processing model, processes the entire disk [40]. The method conducts a comprehensive search of a disk, however it requires significant Random-access memory (RAM) to create a file system hierarchy and determine file boundaries. Research finds that it enables the possibility to identify a tremendous amount of information without building any order because most forensic files will not be fragmented [22]. Stream-based disks forensics, such as byte-stream, are important for when involving HDD over Solid State Drive (SSD) storage mediums. It may be easier to computationally scan the entire storage medium to make the first pass for file-by-file recovery, followed by a second phase during which the undesignated fragments will be explored [24].
- **Stochastic analysis:** Stochastic analysis is another model for data discovery that relies on random sampling and processing of the designated portions of storage media. It is an efficient approach when considering speed, however, it may miss traces of data according to [24].
- **Hashing:** As the amount of data that needs analysis continues to exponentially grow, hash functions have become a viable approach to identifying known artifacts. The utilization of hash functions enables the identification of known files on a device. It allows the elimination of non-relevant data and retention of essential evidence, which expedites the forensic analysis process by only having to search by comparison [12, 28].
- **Whitelisting:** As outlined in [36], the increasing storage volumes have become a pressing problem during digital forensic investigations. One hurdle for an investigator to surpass during analysis is to avoid extracting known-good files that hold no importance to the investigation that are commonly found on many devices or vendor specific files (such as OS files). Whitelisting is a method of processing these known to be good files and comparing them against a well-known file collection to which the files seen as safe. Optimizing this technique, however, requires efficient matching of files, detection of the exact, near, and approximate matches [15].
- **Blacklisting:** Blacklisting is employed in digital forensic investigations as well. A common practice in cybersecurity is to block spam emails based on a list of known to be malicious senders. This same methodology can be applied to digital forensic investigations, very similar to whitelisting, but would allow the investigator to know there are files marked as bad entries on an image. This can reduce the amount of data needed to be analyzed, as these files would already be deemed noteworthy [33].

6.3 Artifact Databases & Curated Artifacts

Hashing, whitelisting, & blacklisting all rely on a known list of artifacts to compare potential important finds against. While each practitioner may have their own database, their set is limited to solely what their institution has found or has access to. This necessitates a centralized, crowd-sourced artifact database (repository) for practitioners to be able to employ for finding artifacts on storage media. The project [20] was an initial attempt at the concept of building a community-sourced artifact repository. Formerly, a standalone website, the project is now a GitHub repository that contains artifacts that date back to 2010. The goal on the original website, "to become a repository for useful information forensic examiners may need to reference during the course of their analysis." The website has since become obsolete, and only the GitHub is maintained, as explained in [26].

Our approach leverages published research from AGP, since it is vetted artifact database, and constructs ForensicAF. ForensicAF was implemented as an Autopsy plugin so that it can be used by field operatives and examiners [6].

7 CONCLUSION

In this paper, we presented ForensicAF, an approach implemented as an Autopsy DF *Ingest Module* plugin which uses CuFAs crowd-sourced by AGP. Our approach showed promising results in both systematic sampling and random sampling experiments. As AGP continues to vet more submitted CuFAs, ForensicAF will continue to become more useful by locating a larger variety of artifacts to search for. By using a crowd-sourced data source for plugins like ForensicAF, practitioners can benefit from knowledge generated from organizations around the world compiling known artifacts to look out for.

ForensicAF is not intended to be an end-all solution to DF. The purpose of ForensicAF is to provide the digital forensics community with additional tooling to help uncover artifacts of potential forensic relevance.

8 FUTURE WORK

ForensicAF may become more effective as additional CuFAs are created and validated in AGP. To aid in AGP gaining more CuFAs, a future iteration of ForensicAF to allow investigators to automatically upload interesting findings to AGP may be implemented to streamline the artifact submission process. This will allow ForensicAF to not only benefit from the existing CuFAs in AGP, but also provide crowd-sourced artifacts to assist other investigations.

Other future work may focus on ForensicAF's ability to utilize other sources of forensic artifacts such as ForensicArtifacts's [20] GitHub repository. Lastly, methods for locating artifacts relevant to specific filesystems, or operating systems, may be implemented to speed up ForensicAF's performance. Future work should also examine the utility of ForensicAF during real examinations, as our testing was limited to available scenarios for testing.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1900210. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] [n.d.]. Dutch National Police Agency. http://ocfa.sourceforge.net/. Accessed: 2010-12-12.

[2] [n.d.]. Encase Forensic. http://www.guidancesoftware.com/products/ef_index.asp. Accessed: 2007-12-12.

[3] [n.d.]. Forensic Toolkit (FTK). https://accessdata.com/products-services/forensic-toolkit-ftk. Accessed: 2021-02-04.

[4] Inikpi O Ademu, Chris O Imafidon, and David S Preston. 2011. A new approach of digital forensic model for digital forensic investigation. *Int. J. Adv. Comput. Sci. Appl* 2, 12 (2011), 175–178.

[5] Apache Foundation. [n.d.]. Class XSSFWorkbook. https://poi.apache.org/apidocs/dev/org/apache/poi/xssf/usermodel/XSSFWorkbook.html.

[6] Ibrahim Baggili and Frank Breitinger. 2015. Data Sources for Advancing Cyber Forensics: What the Social World Has to Offer. AAAI Spring Symposium Series. https://www.aaai.org/ocs/index.php/SSS/SSS15/paper/view/10227/10092

[7] Ibrahim Baggili, Andrew Marrington, and Yasser Jafar. 2014. Performance of a logical, five-phase, multithreaded, bootable triage tool. In *IFIP International Conference on Digital Forensics*. Springer, 279–295.

[8] Willi Ballenthin. 2014. Rejistry. https://github.com/williballenthin/Rejistry.

[9] Sean Barnum. 2012. Standardizing cyber threat intelligence information with the structured threat information expression (stix). *Mitre Corporation* 11 (2012), 1–22.

[10] Basis Technology. [n.d.]. Autopsy - Autopsy Forensic Browser Developer's Guide and API Reference. https://www.sleuthkit.org/autopsy/docs/api-docs/4.0/mod_dev_py_page.html. Accessed: 2020-02-06.

[11] Nicole Beebe. 2009. Digital forensic research: The good, the bad and the unaddressed. In *IFIP International conference on digital forensics*. Springer, 17–36.

[12] Frank Breitinger, Huajian Liu, Christian Winter, Harald Baier, Alexey Rybalchenko, and Martin Steinebach. 2013. Towards a process model for hash functions in digital forensics. In *International Conference on Digital Forensics and Cyber Crime*. Springer, 170–186.

[13] Brian Carrier. 2009. The Sleuth Kit and Autopsy: forensics tools for Linux and other Unixes, 2005. *URL http://www. sleuthkit. org* (2009).

[14] Brian Carrier et al. 2003. Defining digital forensic examination and analysis tools using abstraction layers. *International Journal of digital evidence* 1, 4 (2003), 1–12.

[15] Sudarshan S Chawathe. 2009. Effective whitelisting for filesystem forensics. In *2009 IEEE International Conference on Intelligence and Security Informatics*. IEEE, 131–136.

[16] MI Cohen. 2008. PyFlag–An advanced network forensic framework. *Digital investigation* 5 (2008), S112–S120.

[17] D. Compton, J. A. Hamilton, and Jr. 2011. An Examination of the Techniques and Implications of the Crowd-Sourced Collection of Forensic Data. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*. 892–895. https://doi.org/10.1109/PASSAT/SocialCom.2011.232

[18] Vicka Corey, Charles Peterman, Sybil Shearin, Michael S Greenberg, and James Van Bokkelen. 2002. Network forensics analysis. *IEEE Internet Computing* 6, 6 (2002), 60–66.

[19] Josiah Dykstra and Alan T Sherman. 2013. Design and implementation of FROST: Digital forensic tools for the OpenStack cloud computing platform. *Digital Investigation* 10 (2013), S87–S95.

[20] Forensic Artifacts. 2021. artifacts. https://github.com/ForensicArtifacts/artifacts.

[21] Baguelin Frederic, Jacob Solal, Mounier Jeremy, and Percot Francois. 2010. Digital forensics framework.

[22] Simson L Garfinkel. 2007. Carving contiguous and fragmented files with fast object validation. *digital investigation* 4 (2007), 2–12.

[23] Simson L Garfinkel. 2009. Automating disk forensic processing with SleuthKit, XML and Python. In *2009 Fourth International IEEE Workshop on Systematic Approaches to Digital Forensic Engineering*. IEEE, 73–84.

[24] Simson L Garfinkel. 2010. Digital forensics research: The next 10 years. *digital investigation* 7 (2010), S64–S73.

[25] Eric Gentry, Ryan McIntyre, Michael Soltys, and Frank Lyu. 2019. SEAKER: A tool for fast digital forensic triage. In *Future of Information and Communication Conference*. Springer, 1227–1243.

[26] Cinthya Grajeda, Laura Sanchez, Ibrahim Baggili, Devon Clark, and Frank Breitinger. 2018. Experience constructing the artifact genome project (agp): Managing the domain's knowledge one artifact at a time. *Digital Investigation* 26 (2018), S47–S58.

[27] Grand View Research. 2019. Digital Forensics Market Size is expected to grow to USD 6.95 billion by 2025. https://www.grandviewresearch.com/industry-analysis/digital-forensics-market. Accessed: 2021-02-02.

[28] Vikram S Harichandran, Frank Breitinger, and Ibrahim Baggili. 2016. Bytewise approximate matching: the good, the bad, and the unknown. *Journal of Digital Forensics, Security and Law* 11, 2 (2016), 4.

[29] Vikram S Harichandran, Frank Breitinger, Ibrahim Baggili, and Andrew Marrington. 2016. A cyber forensics needs analysis survey: Revisiting the domain's needs a decade later. *Computers & Security* 57 (2016), 1–13.

[30] Vikram S Harichandran, Daniel Walnycky, Ibrahim Baggili, and Frank Breitinger. 2016. Cufa: A more formal definition for digital forensic artifacts. *Digital Investigation* 18 (2016), S125–S137.

[31] Alastair Irons and Harjinder Singh Lallie. 2014. Digital forensics to intelligent forensics. *Future Internet* 6, 3 (2014), 584–596.

[32] Vacius Jusas, Darius Birvinskas, and Elvar Gahramanov. 2017. Methods and tools of digital triage in forensic context: Survey and future directions. *Symmetry* 9, 4 (2017), 49.

[33] Thomas Laurenson. 2017. *Automated Digital Forensic Triage: Rapid Detection of Anti-Forensic Tools*. Ph.D. Dissertation. University of Otago.

[34] log2timeline. 2021. Plaso. https://github.com/log2timeline/plaso.

[35] Laoise Luciano, Ibrahim Baggili, Mateusz Topor, Peter Casey, and Frank Breitinger. 2018. Digital forensics in the next five years. In *Proceedings of the 13th International*

[6] Autopsy (url: https://www.autopsy.com)

Conference on Availability, Reliability and Security. 1–14.

[36] Sebastian Neuner, Martin Schmiedecker, and Edgar Weippl. 2016. Effectiveness of file-based deduplication in digital forensics. *Security and Communication Networks* 9, 15 (2016), 2876–2885.

[37] Golden G Richard III and Vassil Roussev. 2005. Scalpel: A Frugal, High Performance File Carver.. In *DFRWS*. Citeseer.

[38] Marcus K Rogers, James Goldman, Rick Mislan, Timothy Wedge, and Steve Debrota. 2016. Paper Session II: Computer Forensics Field Triage Process Model. (2016).

[39] Marcus K Rogers and Kate Seigfried. 2004. The future of computer forensics: a needs analysis survey. *Computers & Security* 23, 1 (2004), 12–16.

[40] Vassil Roussev, Yixin Chen, Timothy Bourg, and Golden G Richard III. 2006. md5bloom: Forensic filesystem hashing revisited. *digital investigation* 3, 1 (2006), 82–90.

[41] Keyun Ruan, Ibrahim Baggili, Joe Carthy, and Tahar Kechadi. 2011. Survey on cloud forensics and critical criteria for cloud forensic capability: A preliminary analysis. (2011).

[42] Keyun Ruan, Joe Carthy, Tahar Kechadi, and Ibrahim Baggili. 2013. Cloud forensics definitions and critical criteria for cloud forensic capability: An overview of survey results. *Digital Investigation* 10, 1 (2013), 34–43.

[43] Laura Sanchez, Cinthya Grajeda, Ibrahim Baggili, and Cory Hall. 2019. A practitioner survey exploring the value of forensic tools, ai, filtering, & safer presentation for investigating child sexual abuse material (csam). *Digital Investigation* 29 (2019), S124–S142.

[44] George Sibiya, Hein S Venter, and Thomas Fogwill. 2012. Digital forensic framework for a cloud environment. (2012).

[45] Harm MA van Beek, Jeroen van den Bos, Abdul Boztas, EJ van Eijk, R Schramp, and M Ugen. 2020. Digital forensics as a service: Stepping up the game. *Forensic Science International: Digital Investigation* 35 (2020), 301021.