Forensic Science International: Digital Investigation 34 (2020) 300999

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

# Digital forensic tools: Recent advances and enhancing the status quo

Tina Wu [a, *], Frank Breitinger [b], Stephen O'Shaughnessy [c]

[a] Department of Computer Science, University of Oxford, Parks Road, Oxford, UK
[b] Hilti Chair for Data and Application Security, Institute of Information Systems, University of Liechtenstein, Fürst-Franz-Josef-Strasse, 9490, Vaduz, Liechtenstein
[c] Department of Informatics, Technological University Dublin, Blanchardstown Campus, Dublin 15, Ireland

ARTICLE INFO

*Article history:*
Received 21 April 2020
Received in revised form
4 June 2020
Accepted 8 June 2020
Available online 14 August 2020

*Keywords:*
Digital forensic tools
Published software
Literature review
Open source software
Availability

ABSTRACT

Publications in the digital forensics domain frequently come with tools – a small piece of functional software. These tools are often released to the public for others to reproduce results or use them for their own purposes. However, there has been no study on the tools to understand better what is available and what is missing. For this paper we analyzed almost 800 articles from pertinent venues from 2014 to 2019 to answer the following three questions (1) what tools (i.e., in which domains of digital forensics): have been released; (2) are they still available, maintained, and documented; and (3) are there possibilities to enhance the status quo? We found 62 different tools which we categorized according to digital forensics subfields. Only 33 of these tools were found to be publicly available, the majority of these were not maintained after development. In order to enhance the status quo, one recommendation is a centralized repository specifically for tested tools. This will require tool researchers (developers) to spend more time on code documentation and preferably develop plugins instead of stand-alone tools.



## 1. Introduction

Compared to other domains, the digital forensics community has a very applied focus, meaning that we are not solving problems in theory but practically. Consequently, research endeavors frequently come with prototype implementations. For instance, Clark et al. (2017) analyzed the DJI Phantom III Drone. Their paper presented findings on the proprietary, encrypted file format including a reference implementation to automate the process. Additionally, tool development is encouraged by creating Digital Forensic Competitions, a prominent example of which is the Digital Forensics Research Workshop (DFRWS) Challenges: From 2005 onwards the DFRWS-conferences challenged researchers and practitioners with the goal of pushing the state-of-art into developing forensic tools. The idea is that each year a new challenge surrounding a different theme is set depending on what is voted the most popular topic, a scenario, guidelines and dataset is provided. The results from the DFRWS 2014–2019 challenges have been released and are summarized in Appendix B.

While these tools are essential to validate the proposed research, they can also provide benefits later for investigators in case they run into a similar problem. However, there have been no studies with respect to the diversity, availability or quality of the published tools; nor has there been a discussion with respect to challenges and problems. Consequently, this led us to the following research questions:

RQ1 What tools have been published? (categorization)
RQ2 Are tools freely available and/or maintained? (license, downloadable)
RQ3 Are tools applicable and useable? (assessment with respect to programming style, interfaces, code quality, tested, and documentation)

Knowing what has been released and various aspects of existing tools, this led us to a final question which will lead the discussion section:

RQ4 What are the current challenges in the area of forensic tool development?

This last question focuses on aspects that we think are currently missing with respect to tools themselves but also broader thoughts which we believe makes tool development more durable.

* Corresponding author.
*E-mail address:* Tina.Wu@cs.ox.ac.uk (T. Wu).

https://doi.org/10.1016/j.fsidi.2020.300999
2666-2817/

To answer these questions, we reviewed 799 research publications from various digital forensics journals and conferences between 2014 and 2019 and filtered out those presenting tools. For each tool-related paper, we answered (where possible) the aforementioned questions R[1–3]. In summary, our article provides the following three contributions:

- An updated version of the subfields in digital forensics based on prior work from Netherlands Register of Court Experts (NRGD) (2016) to better cluster (existing) tools.
- A list (overview) of recently published (in peer-reviewed venues) tools that can be used for various purposes.
- A discussion of challenges with published tools including some recommendations on how to enhance the status quo.

One may argue that technology (software and hardware) is changing rapidly and that tools are already outdated by the time they are published. Although this may be true in some cases, practitioners still encounter old hardware as well as software. For instance, Steven Watson (VTO Labs) said during his keynote at DFRWS EU 2019 that they still find old file systems on recently purchased devices such as Drones.

The rest of the paper is organized as follows: first, we discuss the limitations of the study followed by a Related work section. Next, we outline the survey methodology which starts by defining tool in Sec. 4. In Sec. 5 we present a summary of our findings followed by a clustering of the tools. Sec. 7 includes a critical discussion followed by the last section – the conclusion.

## 2. Limitations

Our work is based on a manual analysis of hundreds of research articles. Although we conducted several runs, there is the possibility for human errors. Given the sheer amount of publications, we also had to limit the scope and only consider the years 2014–2019 as well as seven different venues. Lastly, when validating the availability of a tool, it may have been moved/renamed which did not allow us to find it. However, we believe that this paper provides a solid overview of what tools are available and where to find them.

## 3. Related work

The study was inspired by Grajeda et al. (2017)[1], who published an article on the availability of datasets and what is missing. In their paper the authors analyzed various conference and journal publications from 2010 to 2015 with respect to the utilization of datasets and if they were released. They concluded that often datasets are used but most often they are not shared.

The usefulness and power of open source tools was discussed by Altheide and Carvey (2011) in their book 'Digital forensics with open source tools'. The authors discuss software which can be helpful during the forensic process including benefits of using open source software. However, given that the book was released in 2011 and does not focus on recent published research, the work differs from this article. Manson et al. (2007) underline the usefulness of open source products. In their work, they compared Sleuth Kit (open source software) to the commercial products EnCase and FTK. "The results indicated that the tools provided the same results with varying degrees of intricacy."

Carrier (2002) addresses digital forensic analysis tools and their use in a legal setting, stating that to enter scientific evidence into court, a tool must be reliable and relevant. Reliability is tested by the Daubert standard, taken from a rule of evidence regarding the admissibility of expert witness testimony (Daubert vs Merrell, 1993). This standard utilizes four guiding questions to assess tools[2]:

- Testing: Can and has the procedure been tested?
- Error Rate: Is there a known error rate of the procedure?
- Publication: Has the procedure been published and subject to peer review?
- Acceptance: Is the procedure generally accepted in the relevant scientific community?

Using the guidelines of the Daubert tests, Carrier demonstrates that open source tools may "meet the guideline requirements than closed source commercial tools" by publishing source code and data, which would allow the digital forensic community to examine and validate the procedures used to produce digital evidence for each tool. The reliability of tools is discussed further in Sec. 7.1.

With respect to digital forensic tools, we found two platforms providing lists of tools: `ForensicsWiki.org/wiki/Tools` and `toolcatalog.nist.gov`, where both platforms provide a combination of commercial and open source software and hardware tools. The former is a catalog that is categorized by their functions and then on the operating platform, vendor and version. The latter is part of the National Institute for Standards and Technology (NIST) Computer Forensic Tool Testing (CFTT) project, which allows searching a catalog by functionalities and then further refined by operating platform and various other technical parameters depending on the forensic tool. More details about these platforms are discussed in Sec. 5.1.

## 4. Methodology

While these aforementioned platforms are valuable to the community, they often focus on complete software solutions (e.g., from commercial vendors). In contrast, the goal of this paper is to identify software that has primarily been developed for research purposes and analyze any follow-up work as well as other aspects of these tools. As a first step and before reviewing the literature, the upcoming section defines what a tool is.

### 4.1. Definition of tool (software)

According to Lexico (2019), a tool is defined as a device or implementation used to carry out a particular function. Sammons (2015), on the other hand, has described tools as not only designed for a specific purpose but also for broader functionality.

For us, a tool is self-contained and provides a certain level of automation, i.e., user interaction is minimized, reduced and abstracted. For instance, a user should not have to manually find sector numbers for the tool to access the disk or translate virtual to physical addresses. Valid tools, for example, would include, but are not limited to, scripts that can be used to parse files, a code snipped that can reassemble (carve) files, or a program that helps visualizing information. Tools can be written in any programming language and are often developed by individuals or research groups. Lastly, a tool may use another program if it is automated. For instance, Autopsy is a tool supporting plugins which we also considered tools (they add additional functionality to Autopsy).

In contrast, we do not consider comprehensive software as a tool if it is a fully featured application such as EnCase or Cellebrite. This work will also ignore any articles that only explain procedures

[1] Note, Grajeda et al. (2017) was inspired by Abt and Baier (2014).

[2] Questions/itemization are copied from Carrier (2002).

or models without any reference implementation. Here are some examples: forensic process models, frameworks, methods, schemes and approaches. We also exclude algorithms as a tool unless it has been implemented and tested.

### 4.2. Collecting and analyzing articles

The first step focused on collecting articles from digital forensics conference proceedings and journal publications spanning the last six years (from 2014 to 2019). Our selection of venues was similar to Grajeda et al. (2017), however, we only focused on digital forensic platforms (venues are listed in Sec. 5) and ignored more traditional cybersecurity centered venues as our research was focused purely on digital forensic tools.

Next, we removed all articles that were not tool related as defined in Sec. 4.1. For the remaining tool-centered articles, we grouped them based on the subfields in digital forensics (details see Sec. 6) and tried to find information about: licensing, possible updates and availability. Lastly, we reviewed the program source code as discussed in Sec. 4.4. Additionally, to verify the tools we found listed in the articles, we carried out an automated search using a Python script that searches for the keyword 'tool/tools' in the PDF files.

### 4.3. Online searches

We also searched online for available digital forensic tool repositories using the following terms: 'digital forensic tool catalog' and 'digital forensic tool repository'. We focused our analysis on the first 50 results for each query. Once a forensic tool repository was identified, we collected the following information (when possible): author(s), the number of tools, how the tools have been categorized, whether the repository was being updated regularly and if it included any additional links to other websites related to digital forensics. Findings are presented in Sec. 5.1.

### 4.4. Code review

The code review process started with a brief manual review to see if it is commented (we assume that commented code has a better quality and allow other to make modifications). Additionally, the programs' source code was analyzed using an automated code review tool (Codacy[3]; see next paragraph). This is an important process in software development as it helps improve the quality of the software. Usually these tools analyze various aspects to check the quality (Ashfaq et al., 2019):

- Code complexity: Highly complex methods and classes that should be refactored
- Compatibility: Used mainly for front-end code, detects compatibility problems on different browser versions
- Errors: Code that may hide bugs and language keywords that should be used with caution.
- Security: Common security issues
- Code style: Code formatting and syntax problems. For example variable name style, enforce the use of brackets and quotation marks
- Documentation: Detects methods and classes that do not have the correct comment annotations
- Performance: Code that could have performance problems
- Unused code: Unused variables and methods

[3] www.codacy.com.

There are a diverse number of automated code analysis tools available, however the only one that met the aforementioned requirements was Codacy. Codacy is an open source tool that is integrated into Github and benefits from being transparent on the type of tools used during the code review process. For example, when finding security issues in Python code, it informs the user that it uses Bandit. An overall grade that ranges from A to F is given (with A being the highest) which makes it easier to assess the quality of each tool. In total, there are 33 tools where the source code for one tool was unavailable. The remaining 32 tools available on Github, Bitbucket and Gitlab were cloned or forked onto a new Github account. Tools available on private websites were manually uploaded to the new Github account. Codacy was then used to analyse each repository.

## 5. Results overview and availability of tools

This section examines the availability of the tools and whether they are maintained (see RQ2). Creating new tools can potentially be of great value to the digital forensics community. As tools become established, they encourage the growth of close-knit communities of developers providing consistently updated and patched or maintained code, bug-fixes and comprehensive documentation. We categorized the results from our tool search under the attributes of availability, code maintainability, documentation and licenses.

For this article, we reviewed 799 research publications where 62 (7%) included tools according to our definition. Almost 25% of all reviewed articles came from the Digital Forensics Research Workshops (US & EU) where 27 out of the 199 included tools; followed by Digital Investigation[4] (journal) where 20 out of 212 were found. Other venues had similar results: (a) International Conference on Digital Forensics & Cyber Crime (ICDF2C[5]) had 3 out of 44; (b) Association of Digital Forensics, Security & Law (ADFSL, Conference) contained 3 out of 84; (c) IFIP Working Group 11.9 on Digital Forensics had 5 out of 111; (d) International Workshop on Digital Forensics (WSDF) contained 1 out of 29; and (e) Journal of Digital Forensics, Security and Law (JDFSL) also had 3 out of 120. An overview of all identified tools can be found here: https://www.fbreitinger.de/wp-content/uploads/2020/05/ToolsTable.xlsx.

**Availability of tools.** On further examination of the 62 tools, we found that currently 53.2% (33/62) are available online (note, we did not contact the authors to see if they were willing to share the tool). The 33 available tools are listed in A. 78.7% (26/33) of the available tools were on public repositories (Github/25 and Bitbucket/1). Five of the tools were available on private/personal websites (e.g., downloadable from research group websites). The two remaining locations were Source Forge and Dropbox.

By implication, this means that 29 tools were not available where the most common (28) reason was unknown source. We found one article with a link to an empty Github which appears to have never been uploaded.

**Code Review.** Codacy was used to analyze the 32 tools and no issues were found relating to code complexity, compatibility, documentation or unused code. Overall, all the tools had issues in their source code with coding style, however this is subjective and parameters/rules can be adjusted. 24 of the tools had security issues, 14 had errors and 4 had performance issues. If major security issues were found in the code this could have an effect on the integrity of the tool. For example Codacy highlighted one tool that uses an insecure MD5 and SHA1 hash function, instead it is

[4] Volume 30 was the last volume considered for this research.

[5] We only considered articles between 2014 and 2018 as there is/was no ICDF2C 2019.

recommended that SHA512 be used (NIST, 2015).

To get a better understanding of the quality of the code, the tools were then scored from A to F where 6 tools scored an A; 20 tools scored a grade B; 5 scored a C and lastly 1 tool scored a D. The tool with the least issues was developed in C++ and had only 4 issues related to security and coding style. The tool with the most issues (and scored a D) was developed in C++, and had 1044 issues, these were coding style, errors, security and compatibility.

**Code maintainability.** We examined the 22 tools developed between 2014 and 2018 to determine whether they were being maintained after they had been developed/last modified, the 11 tools from 2019 were excluded due to being too recent. 72.7% (16/22) tools have not been maintained after development; for 2 we were unable to identify the date they were last modified; and only 4 have had recent updates.

**Programming language.** Tools have been developed in various programming languages, ranked in descending order of popularity; Python, C++, Java, C, Golang and JavaScript (note that some of tools were developed to be compatible with more than one programming language). Python was by far the most popular programming language and is frequently used for plugin-development for existing tools such as Volatility[6] or Autopsy.[7]

**Documentation.** 29 of 33 tools had some comments in their source code, which makes maintaining the code after publishing easier. However, the quality of those comments varied widely from being very helpful to of little help. Furthermore, 24 of the 33 tools provided documentation, while the majority included a description of the tool(20), some provided additional information including; usage commands(13), installation instructions (dependencies and requirements) (5), configuration(2), execution of the tool(1) and results from the output of the tool(1). There is no standard format for developers to follow when documenting their tools, however several sources have provided a recommended format. For example in Github a Readme file should be included with the following information: project name, description, table of contents, installation, usage, contributing, credits and license (Łyczywek, 2018).

**License.** Online repositories such as Github allow the licensing of source code so that they are open source (Github, 2019). However, only 11 out of 33 tools had licenses, this means that the tools without licenses would need to request permission from the copyright holder in order to redistribute the code which is cumbersome.

### 5.1. Tools from other sources

Overall, we identified five sources from online searches that provided repositories of forensic tools: Four were websites that provided links to a variety of tools from different categories; one source only focused on acquisition and analysis tools.

**Computer Forensic Tools & Techniques Catalogue** is a comprehensive list of forensic tools developed by NIST. As stated on the website, "the primary goal of the tool catalogue is to provide an easily searchable catalogue of forensic tools and techniques". It contains 37 different categories of forensic tools across all disciplines, e.g., disk imaging, live response, drone forensics or infotainment & vehicle forensics.

**Forensic Wiki** (https://www.forensicwiki.org/) contains about 110 either commercial or open source tools that have been grouped into 10 main categories based on their functionalities, e.g., disk imaging tools, memory imaging, memory analysis or network forensic. A category may have subcategories, e.g., disk imaging tools have been split based on Operating System (OS). Note, subcategories are different for each main category, e.g., file analysis has image analysis, software forensics, open source and file analysis tools as subdomains; there appears to be no universal method of categorizing the tools. Additionally, this website also includes links to websites hosting digital forensic challenges, various lists of topics and a mixed list of datasets.

**Awesome-Forensics** is a Github repository that provides a comprehensive list of forensic tools and was created by Jonas Plum. This repository was created in March 29$^{th}$, 2016 and as stated on the website provides a, "curated list of awesome free (mostly open source) forensic analysis tools and resources". This resource contains about 60 tools in 14 categories and was last updated on April 29$^{th}$, 2019. We found the repository also has a list of tools for various other domains including pentesting, malware analysis, hacking or honeypots.

**DFIR Training** lists 27 main categories with four categories being loosely categorized under "Forensic Utilities" and by OS (Linux, Mac, Misc and Windows) where Misc and Windows have further subcategories. We could not find any information on how often this repository is updated/last update: https://www.dfir.training/dfirtools/advanced-search.

**DF tools catalogue** (`dftoolscatalogue.eu`) was developed by the Institute of Legal Information Theory and Technique of the National Research of Council of Italy. This repository differentiates between acquisition and analysis digital forensic tools where it lists 464 acquisition tools and 1045 analysis tools that are clustered by forensic domains, e.g., network, mobile, malware etc. We could not find information on updates/last update.

## 6. Types of tools

To answer RQ1, this section focuses on published tools which allows practitioners to easily identify an appropriate tool for a given task. In order to cluster them, it is important to follow a digital forensics taxonomy. For example, Carrier (2003) introduced the notion of abstraction layers for digital forensics at multiple levels, identifying abstraction layers for physical media, media management, file system analysis, network analysis and memory analysis. He proposed that "the purpose of digital forensic analysis tools is to accurately present all data at a layer of abstraction and format that can be effectively used by an investigator to identify evidence".

More recently, Netherlands Register of Court Experts (NRGD) (2016) identified six subfields of digital forensics: (1) Computer, (2) Software, (3) Database, (4) Multimedia, (5) Device and (6) Network forensics; where software is defined as "uncovering potential evidence through examining software".

We feel the latter approach is more intuitive and easier to navigate,[8] so in this regard, we present a modified and extended version of the NRGD taxonomy as shown in Fig. 1.

In our updated classification taxonomy, we place databases under the software category, due to the lack of available database forensics tools. We also extend the taxonomy to include malware and memory forensics categories, since they both constitute distinct and significant realms within the scope of digital forensics. Universal tools (shown in red) are classified under computer forensics as we do not consider it to be a direct subfield of digital forensics. Additionally, the origin of these tools often started in traditional computer forensics (e.g., string search or timeline analysis). The subfields for each category (shown in gray) correspond to paragraphs in the upcoming subsections. Note, despite the

[6] https://www.volatilityfoundation.org/.
[7] https://www.autopsy.com/.
[8] Note: this is a personal preference and we do not say that one is better than the other.

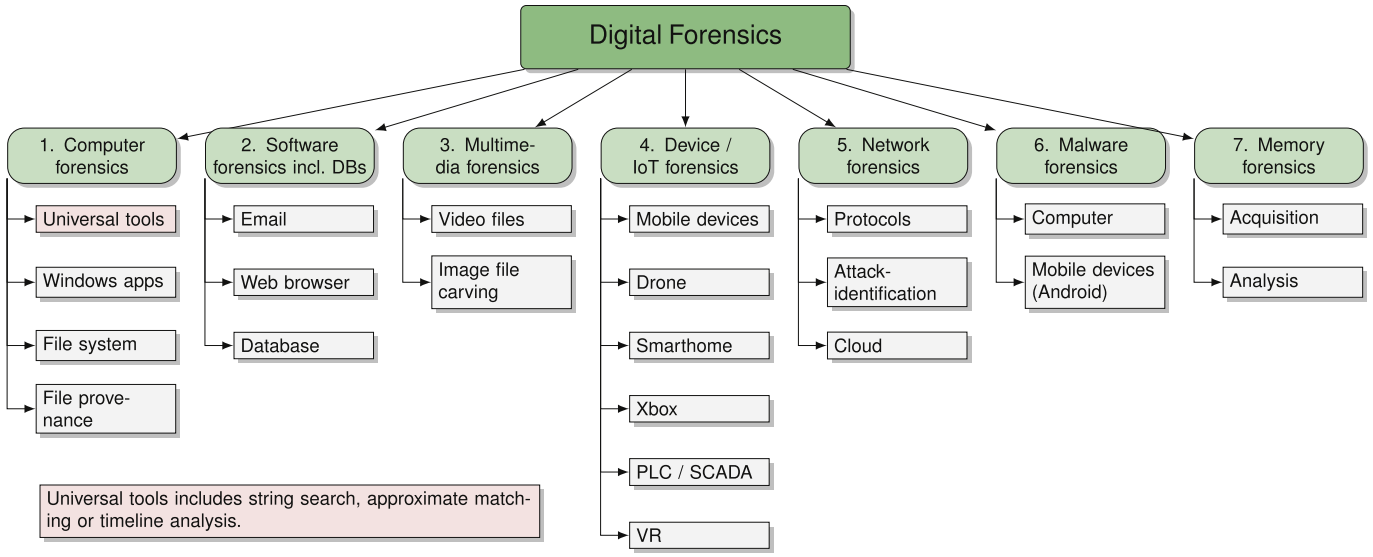


**Fig. 1.** An overview of Digital Forensics subfields which expands on the recommendations made by Netherlands Register of Court Experts (NRGD) (2016). Compared to the original version, we combined Software and Database forensics, renamed Device forensics to device/IoT forensics and added two new subfields: malware forensics and memory forensics. Additionally, we added some explicit examples (marked gray) compared the verbal description from the original document.

updated version, there are some tools that fit in several subfields. In these cases, we decided to place the work only in the subfield we identified as most appropriate.

One may argue that devices/IoT forensics could be placed into other sections as they typically contain well-known OSes and operating environments (e.g. Linux, BSD). However, we decided to have a separate subfield for this due to (a) they frequently also use proprietary file formats, (b) they are often in their own ecosystem involving Smartphone Apps, on-device information, cloud data and network communication that need to be correlated and (c) less common procedures like chip-off or JTAG forensics.

Although this paper is limited to tools developed from academic papers, other popular forensic tool suites should not be ignored. One source we identified is Forensic Control that lists over 100 open source forensic tools (last updated in Oct. 2019). Another source is GFI.com[9] which lists the top 20 forensic suites such as Volatility and The Sleuth Kit (Autopsy); tools can be easily extended via plugins to include new functionalities (last updated in 2019).

### 6.1. Computer forensics

This section discusses tools relating to various OSes as well as universal tools that can be used independently of the operating systems.

**Windows Apps.** In Windows 8, Microsoft introduced light-weight sand-boxed applications called “apps” to provide a full range of functionality on top of touch-enabled displays. Murtuza et al. (2015) created a tool called MetroExtractor to gather static and volatile forensic data from Windows apps but does not consider in-active hibernation files.

**File provenance.** Tools are available to extract the provenance of a file, but these require prior installation during the provenance generating events. Good (2017) presented the tool AutoProv which uses temporal artifacts to rebuild the file provenance of a Windows forensic image. However, there is the possibility a user can alter the provenance to incriminate another party.

[9] https://techtalk.gfi.com/top-20-free-digital-forensic-investigation-tools-for-sysadmins/.

**File system.** The Linux Ext4 file system saves metadata in the superblock or the group descriptor table. Previously this information was required to understand the correct structure of the file system in order to recover the data. Dewald and Seufert (2017) created a tool as a module within the Sleuthkit framework that does not require this information, instead uses a file carving method and metadata analysis to reconstruct the file.

Object IDs (OIDs) are created when a file is opened or saved by an application. Allocated files that have a OID can be used to reconstruct user activity, often multiple entries are created. The tool NTFSObjIDParser was developed to automatically parse these entries, however it is possible the OID can be manipulated using anti-forensic tools such as fsutil (Nordvik et al., 2019).

In the following we summarize tools that we classified as universal tools which means that they are general enough that they can (could) be used for various OSes.

**Triage.** Computer evidence can be quickly examined to assess the likelihood of acquiring artefacts using triage tools. Vidas et al. (2014) produced a tool called OpenLV for the purpose of quickly identifying relevant evidence, however the tools require administrator permissions. A second triage tool called Forensic2020 runs within a bootable environment and provides the ability to view results while the evidence is processed. In the evaluation they found the tool made slight changes to the file system (Baggili et al., 2014). The next tool was designed to allow an investigator to quickly view the devices that had recently synced with the seized devices. Hargreaves and Marshall (2019) proposed a plugin called Sync Triage which extracts details and provenance of the devices connected. Only a selected number of apps have been tested on this proof-of-concept, however the tool is compatible with a number of OSs, including Windows, iOS and Android. Advanced Automated Disk Investigation Toolkit (AUDIT) integrates commonly used command-line tools onto a Graphical User Interface (GUI). This was designed to support investigators with minimal technical knowledge searching the disk for evidence in files, emails and documents (Karabiyik and Aggarwal, 2016).

**Non-relevant data.** When a bitwise copy of a system is taken, non-relevant or sensitive data must be securely deleted from the image. Zoubek and Sack (2017) proposed a selective deletion method and implemented this as a plugin for the Digital Forensic

Framework (DFF). Testing of the tool was limited to the NTFS file system and required manual intervention from the investigator, hence the authors stated that all irrelevant data may not be identified.

**Timelines.** Timelines are used to reconstruct events and are an important step in an investigation. Log2Timeline is a command-line tool used to create timelines by combining several log files and events of a system. Timeline2GUI is based on Log2timeline but was created as a GUI tool to provide an easier interface for investigators (Debinski et al., 2019).

**Approximate Matching (AM).** The similarity between files and file fragments can be measured using Approximate Matching (AM) algorithms. Breitinger et al. (2014) presented mrsh-v2, an AM tool for detecting similarities at both file and fragment level. mrsh-v2 is an improvement on the original mrsh algorithm by Roussev et al. (2007). mrsh-net by Breitinger and Baggili (2014) is the network implementation of mrsh-v2 which has a single huge Bloom filter for the signature. Further work by Gupta and Breitinger (2015) created the tool mrsh-cf using a Cuckoo filter. mrsh-cf provides an improvement on the previous AM tools in terms of processing speed, memory footprint, compression rate and false positive rate.

**Litigation.** During the legal process all forms of documents related to the lawsuit including those stored in the cloud must be preserved. However, there is the possibility that documents can be manipulated by the parties involved. The authors developed the LItigation hold eNabled Cloud Storage (LINCS) prototype so that auditors can verify whether evidence has been manipulated (Zawoad et al., 2015).

**String search.** Search hit ranking algorithms can help investigators to search through retrieved evidence. The tool Sifter uses this method to rank the forensic importance of strings searches. It is capable of searching both allocated and unallocated space but have only validated their algorithms on a single synthetic case (Beebe and Liu, 2014).

**Evidence container.** Traditional evidence storage use vendor specific formats and lack formal specifications. Schatz (2019) proposed an open specified logical evidence container based on the original AFF4 evidence format (Cohen et al., 2009). It has the benefit of efficient storage of file content and is easily extended.

### 6.2. Software forensics including database forensics

As mentioned previously, “software forensics covers for example operating system forensics, application software forensics and digital forensic analysis tools” (Netherlands Register of Court Experts (NRGD), 2016).

**Email.** In total, three tools specific to the analysis of emails were found. (1) Armknecht and Dewald (2015) was implemented into the Autopsy framework as a plugin called Privacy Preserving Email Forensics (PPEF). The plugin encrypts the suspect's email account to protect their privacy. A keyword search is then performed and only ones that match are decrypted. The plugin works only if the keyword search provides an exact match. (2) InVEST (Koven et al., 2016) is used to discover evidence and information in large email datasets. This is a visual analytic tool that assists investigator in finding emails related to their case when the exact nature is unclear. (3) The last tool uses interactive graphical visualization and statistical information to identify patterns in emails. (Stadlinger and Dewald, 2017).

**Web browser.** Web storage is a client-side data storage containing web browser artefacts. Mendoza et al. (2015) built the prototype BrowStEx to parse both SQLite files and XML files. This provides a timeline for an investigator to search and present the web storage data based on the date/time frame. This prototype was only tested on Windows OS 64bit and modifications will be required for this tool to be compatible with another OS.

**Database forensics.** Research in this area has centered around relational databases. In all, we found five tools for recovering data from particular databases. Kim et al. (2016) tool recovered deleted records and tables from the Extensible Storage Engine (ESE) database but cannot recover deleted data with damaged record header. Database Image Content Explorer (DICE) recovers unallocated data from ten different Relational Database Management System (RDBMS) e.g. MySQL, Oracle (Wagner et al., 2016). Another recovery tool by Yoon and Lee (2018) focused on the MongoDB. The tool used metadata information, namespace file and the signature of deleted records to recover data. Meng and Baier (2019) developed a parser called Bring2lite to process deleted SQLite records. On evaluating the parser, they found it had slightly higher recovery rates than competing tools, although it does not work on encrypted databases. The last tool by Wagner et al. (2019) recovers data from multiple DBMSs then stores this in the Database Forensic File Format. The DF-Toolkit is then used to view and search the extracted data. Although investigators may struggle to interpret the data as some artefacts in plain-text alone may not mean anything.

### 6.3. Multimedia forensics

This section deals with tools used to recover and analyze video and images.

**Video file formats.** Gloe et al. (2014) developed parsers to extract file format structures and metadata from various cameras and mobile devices. They focused on the internal file structure of AVI and MP4 multimedia containers. Their tests did not comprise of multiple devices per camera model so they were unable to verify that this would be the same outcome for other devices of the same model.

**Image file carving.** Due to the increasing capacity of storage disks, file carving would be an ideal triage solution. The Decision Theoretic Carving (DECA) tool was developed to specifically carve JPEG files and uses decision theoretic analysis to consider the likely locations the data is stored (Gladyshev and James, 2017).

### 6.4. Device/IoT forensics

This subfield of device forensics is a broad topic, which gathers evidence from a range of small-scale devices such as mobile devices and smart home devices to larger-scale devices such as drones and games consoles.

**Mobile devices.** WhatsApp is the most widely used mobile messaging app. Capturing messages between the WhatsApp client and server can prove useful as part of an investigation since they can contain important forensic artefacts. The tool ConvertPDML was developed to decrypt and view the messages exchanged between the client and server (Karpisek et al., 2015). For the tool to function it required a rooted device and the password associated with the WhatsApp account.

The Forensic Evidence Acquisition and Analysis System (FEAAS) tool is used to recover data from the iPhone backup files such as date, time and changes created by multiple smart home devices. The results are presented in a report which can be used to establish whether the data was produced through voice command or the mobile app. This study focused on data acquired logically meaning the tool will only work if the data has not been deleted before examination (Dorai et al., 2018).

**Drone.** Due to their increasing popularity drones or unmanned aerial vehicles are being exploited by criminals and therefore are a potential source of evidence in an investigation. Clark et al. (2017) developed DRone Open source Parser (DROP) designed to parse DAT files from the DJI Phantom III drone and then correlate the DAT

files to TXT files. Using this tool, data can be correlated and linked to the user of a specific device based on the extracted metadata. This tool is limited to DAT and TXT files that are proprietary. The tool by Renduchintala et al. (2019) was used to analyze the log parameters of two drones; Yuneec Typhoon H and DJI Phantom 4. This tool is used to extract, examine flight information and convert files for 3D flight trajectory visualization. The tool is ineffective on drones with no logging capabilities or where storage of data is on the mobile application only.

**Smart home.** The increasing number of IoT devices present in smart homes creates new opportunities for obtaining evidence. We identified two tools that extracted user behavior artefacts from the cloud. The tools focused on a specific type of smart home device the Al Speaker ecosystem. The first tool is for the Amazon Echo and is called CIFT: Cloud-based IoT Forensic Toolkit, used the unofficial cloud API as a method of extraction. With the possibility of the cloud API being depreciated, this method of extraction can be seen as being unreliable. For example, Nest[10] have stopped new users from accessing their API. This study focused on four AI speaker ecosystems from different manufacturers released by South Korea. Jo et al. (2019) produced a tool to retrieve artefacts for one of the AI speakers (NAVER Clova), that identified commands the user had previously executed. The application can only retrieve this information with this unique access token which can only be obtained by intercepting the network traffic between the AI speaker and the cloud. The number of commands displayed is restricted to 20 requiring the investigator to continuously scroll to see other commands.

In a study of multiple smart home devices such as cameras, hub and an alarm system, evidence was extracted from multiple data sources. Servida and Casey (2019) developed three Autopsy plugins and one standalone tool to automate the extraction process. Two of the plugins could only be used to extract cloud credentials. The third, was used to extract cloud credentials, events and user actions. The standalone tool was used to download debug logs and parse events. These tools are the first to demonstrate that open source forensic tools can be customised for its use in IoT forensics.

**XBox.** In a study we found one tool the File System N-View (FSNView) developed to document metadata, using multiple storage system parsers on the same disk. The individual parsers then present their interpretation of the metadata in Digital Forensics XML, a storage language used to carry out differential analysis (Nelson et al., 2014).

**Programmable Logic Controller (PLC)/Supervisory Control and Data Acquisition (SCADA).** The PLC is one of the most important components on a SCADA system, used in various industries such as water, electrical etc., to automate many production processes. In this study we found three tools used to acquire and analyze data from individual manufacturers of PLCs. Yau (2015) tool called the Control Program Logic Change Detector (CPLCD) is used to detect two types of attacks; reprogramming of the PLC and alteration of the memory variables on the Siemens S7-1200 PLC. A further tool by Denton et al. (2017) was developed for the GE-SRTP network protocol PLC, that provides direct communication with the PLC to read the ladder logic program. The last tool called Cutter by Senthivel et al. (2017) is compatible with PLCs using the Programmable Controller Communication Commands (PCCC) protocol. The tool is capable of extracting updates, ladder logic program and configuration information.

**Virtual Reality (VR).** The rising popularity of VR devices on the consumer market make them a valuable source of digital evidence. To analyze the memory of a VR device, Casey et al. (2019) have produced a Volatility plugin to automate the extraction of memory artefacts such as location, state and class of devices.

### 6.5. Network forensics

This section summarizes a variety of different network forensic tools which includes protocols, wireless/wired security and cloud forensics.

In addition to network forensic tools, one publication by Vondráček et al. (2018) presented an offensive tool to raise awareness. Their tool Wifimitm "provides functionality for the automation of [Man-in-the-middle] MitM attacks in the wireless environment. The package combines several existing tools and attack strategies to bypass the wireless security mechanisms, such as WEP, WPA, and WPS."

**Protocols.** This study found three tools related to the network analysis for IP protocols. Gugelmann et al. (2015) built a visualization tool called Hviz, that structures, aggregates and correlates HTTP events between workstations. The evaluation of the tool identified that a day's network traffic required many hours of extracting the HTTP requests and responses. TLS Key EXtractor (TLSkex) by Taubmann et al. (2016) is used to detect and analyze threats from adversaries using TLS encryption to hide their attacks. This tool is used to extract the master key from a TLS connection to decrypt the TLS session. IPv6 is the most recent version of the Internet Protocol (IP), introduced to replace IPv4 which has vulnerabilities that can be exploited to create covert channels. Hansen et al. (2016) have developed a prototype that demonstrated IPv6 is also susceptible to covert channels. In this study testing was not carried out in a real-world environment.

**Attack-identification.** In all, we found three tools to analyze network attacks. Britt et al. (2015) presented the Simple Set Comparison tool that allows investigators to identify the target. The tool BotDAD uses DNS fingerprinting to identify bot infected machines on a network. Although the tool cannot detect a bot outside of the investigation time or if the IP address is changed (Singh et al., 2019). IoT botnets are rapidly increasing, hence Gannon and Warner (2017) have developed a Wireshark dissector to identify IoT DDoS C&C.

**Cloud.** In this study we found two tools for the recovery of evidence from consumer cloud storage services. The first tool called Synchronization Service Evidence Retrieval Tool (SSERT) remotely recovers evidence from the open source cloud service Syncthing and provides an audit log of actions. This tool is limited as it relies on the recovery of the public/private RSA key pairs and Device IDs from the suspects device (Quinn et al., 2015). Roussev et al. (2016) presented a proof-of-concept acquisition tool, Kumodd that can acquire historical data of document revisions from four major cloud storage providers. A limitation to this tool is that it requires the username and password of the account holder.

### 6.6. Malware forensics

This section includes tools used to perform the analysis of malware on computers and mobile devices (Android).

**Computer.** Our study found two tools that use binary analysis to analyze malware. The first tool uses compiler provenance information to understand how the malware binary is produced. Rahimian et al. (2015) built BinComp that automatically recovers compiler provenance of program's binaries using the syntax, structure and semantic features to capture compiler behavior. The second tool uses binary analysis to detect malware. Alrabaee et al. (2016) have designed the tool BinGold, that automatically recovers semantics of a binary code. The tool does require a compiler or the installation of additional tools. Results are unlikely to be accurate if the user has packed the binary or has used obfuscation techniques.

[10] https://www.home-assistant.io/blog/2019/05/08/nest-data-bye-bye/(last accessed 2019-10-10).

**Mobile devices (Android).** We found three tools for the acquisition and analysis of Android malware. AMExtractor (Yang et al., 2016) was created to acquire and analyze the volatile memory to detect malware e.g. rootkits. Although this tool will not work on Android devices running an ARM CPU. The next tool called AndroParse (Schmicker et al., 2019), contains a database of Android application (APK) features to speed up the analysis process. The final tool Fordroid analyzes APK by automatically identifying what, where and how sensitive information locally stored (Wei et al., 2018).

### 6.7. Memory forensics

Acquisition of the memory is used to gather a snapshot of the system in its current state, this is then analyzed with the appropriate tools. The acquisition and analysis tools is further divided based on the OS for which it has been developed.

**Acquisition.** In total, we found two tools that can be used to acquire memory. A tool that can acquire memory from Windows has been implemented as two plugins for Volatility and Rekall. It can extract ACPI tables from a memory image and scans for potential rootkits (Stüttgen et al., 2015). The next tool, Layout Expert, acquires memory from a Linux system, however knowledge of the memory layout is required (Socała and Cohen, 2016). To overcome this complication, the tool is used to predict the memory layout. This is implemented in Python due to having better accessibility to parsing libraries, but this limits the speed.

**Analysis.** In all, three tools were found to analyze memory for Linux OS: (1) System swap files are a back-up for the OS's virtual memory system. Both Linux and Mac OS compress the RAM to reduce the swapping of these files. Richard and Case (2014) developed four plug-ins that work within Volatility which is used to analyze the compressed RAM. (2) Roussev et al. (2014) presented the tool called sdkernel, which is used to identify the version of the OS kernel. This method does not require heavy reverse engineering but can only fingerprint kernel versions that are known. Lastly (3) Block and Dewald (2017) produced six Rekall plugins that automate the analysis of heap memory. Four of the plugins find the required information in the heap then places it into separate files. The remaining two plugins are used to analyze and extract command history for various applications. To fully analyze the heap memory, it is vital that the swap files are also analyzed, however this is not featured in the tool.

In this study, we found four tools used to analyze memory for the Windows OS: (1) Pool tag scanning is a process used in memory analysis to locate kernel objects, often hidden from the OS. The authors (Sylve et al., 2016) created a plugin to only scan the memory pool tags. (2) Uroz and Rodríguez (2019) designed a Volatility plugin called Winesap, which is used to analyze the Windows Auto Start Extendibility Points (ASEPs) for evidence of persistent malware. The tool is limited to registry ASEPs and cannot check the original ASEPs that triggered the execution of some programs. (3) The next tool looked at malware code injected into executable pages which can manipulate other processes or hide its existence. Block and Dewald (2019) have created a Rekall plugin called Ptenum to detect these hiding techniques. It is limited as it reports all executable pages whether malicious or not, producing a large amount of unwanted data. The final tool by Case et al. (2019) produced a Volatility plugin named Hooktracer, so that inexperienced investigators can analyze API hooks and quickly identify suspicious APIs.

## 7. Discussion of challenges and opportunities

While having a closer look at the last two research questions - RQ3 and RQ4 - we realized that both are closely related and thus we address those in the upcoming sections. A clearer separation of the two questions is provided in the section Conclusion and future work.

While freely available software/tools have many advantages, there are also challenges. As discussed by Tips4PC (2014) or Vadalasetty (2003), there are several risks of using free tools/software such as the lack of support, documentation and updates or safety of the software. Indeed, when analyzing the papers and tools, we realized that tools often were poorly commented or had no/little accompanying documentation. Another aspect pointed out by Tips4PC (2014) was advertising banners which we did not encounter. The last two reasons mentioned in this blog were quality of the user interface as well as developer loses interest where especially the latter one seems to be common: we noticed that although tools have been published, most of them were only used in their referenced articles. In other words (often) neither the authors themselves continued working the tools nor have other researches/practitioners utilized the tools. Of course, there are exceptions. The upcoming subsections present some ideas on how the community may be able to address these issues.

### 7.1. Reliability of tools

Much like crime scene forensics, digital forensics must follow a clear process of collecting, analyzing and reporting such that it can be deemed admissible in a court of law. Therefore, the tools used to collect digital evidence must be subject to stringent testing to verify their reliability to produce “forensically sound” digital evidence. Despite increasing reliance on digital forensics in criminal cases and hence the need for reliable tools, the only sustained effort towards standard tool testing is NIST's Computer Forensics Tool Testing Program.[11] Additionally, no specific international standard exists for digital forensic tools. Some countries have adopted the ISO 17025 standard, which was drafted for testing and calibration laboratories, but can be applied to digital forensics. In the UK, for example, the Forensic Science Regulator's code of practice stipulates that all digital forensic service providers must be ISO 17025 compliant.

Given the obvious need for digital forensics tools to produce accurate, repeatable and reproducible results, the question arises: Do research-based tools have a place in a digital forensics laboratory that is ISO 17025 accredited? While tools are not validated individually under ISO 17025, the standard sets out requirements for equipment and validation of methods and results.

**Equipment.** Sec. 6.4 of the ISO 17025 sets out the requirements for equipment and states that a laboratory shall have access to equipment including software “that is required for the correct performance of laboratory activities and that can influence the results.” To meet the requirements, the laboratory must:

- Verify equipment conforms to specified requirements before being used (Sec. 6.4.4)
- Ensure the equipment is capable of achieving accurate results (Sec. 6.4.5)
- Establish a calibration programme for equipment that is subject to review (Sec. 6.4.7)
- Ensure that any equipment giving questionable results or is defective be taken out of service (Sec. 6.4.9)

**Validation of methods and results.** Sec. 7.2.2 of ISO 17025 requires that the laboratory validate non-standard methods to the

[11] https://www.nist.gov/itl/ssd/software-quality-group/computer-forensics-tool-testing-program-cftt (last accessed 2019-10-10).

necessary extent to meet the needs of the given application or field of application. Validation requires that the laboratory implement robust testing methods through variation of controlled parameters, comparison of results achieved with other validated tools and inter-laboratory comparisons. Furthermore ISO 17025 Sec. 7.7 requires that laboratories have procedures for ensuring and monitoring the validity of their results.

It can be concluded that a laboratory may use research-based tools as long as they can ensure tools meet the requirements set out in the standard as outlined above for equipment and validation of methods and results.

Whilst the Daubert process, discussed in Sec. 3 is not an international standard, it may still be used to establish the reliability of a research tool to produce results that are admissible as legal evidence. To satisfy the requirements of the Daubert process, the tools from peer reviewed articles would need to be well documented, maintained and their development and testing processes, including code and data, clearly documented. While maintenance is difficult to enforce, documentation and testing could be ensured. Out of the 62 tools reviewed, while 46 have been evaluated within the articles, inconsistencies exist in their evaluations. Some tools tested provided a detailed evaluation on accuracy and reliability using multiple datasets/test sets e.g., the accuracy in detecting malware hiding techniques using a real-world malware sample. Whereas others tested on the performance and efficiency e.g., the time it took the tool to run on a memory dump and process artefacts. As stated by Garfinkel et al. (2009),"techniques developed and tested by one set of researchers cannot be verified by others since the different research groups use different data sets to test and evaluate their techniques".

With this in mind, verifying the reliability of the tools we reviewed is a non-trivial task. As a starting point, we searched the Computer Forensics Tool Testing catalog database,[12] but none of the tools found in the peer reviewed academic publications shown in Sec. 6 were found, yet this fact does not negate the tools suitability for use in digital forensic investigations.

Examining the tool catalog reveals it is somewhat limited, with just 198 tool reports published since the year 2000, which falls considerably short of the number of digital forensic tools in existence and in regular use by the digital forensic community. Furthermore, as Horsman (2019a) points out, NIST's tool testing criteria is "narrowly defined" in areas such as limited test data, use of specific tool versions and confinement to single image formats. It would appear that the lack of testing methods for tool reliability not only applies to tools developed for research but extends also to established tools. Globally accepted standards, overseen by a centralized governing body are evidently required to regulate the reliability of evidence produced from digital forensic tools.

### 7.2. Usability of tools in real world settings

When discussing challenges, we must also address the usability of research-based tools. Specifically, their use in real world digital forensics scenarios. We must be cognisant of the fact that the digital forensics landscape is ever changing, where advancements in technology lead to new or previously unseen digital artefacts encountered during a investigation. For this reason, no single digital forensics tool, can accomplish every task and so researchers often develop tools out of a necessity to bridge the gaps in missing functionalities or capabilities of such existing tools. Prime examples are RegRipper byCarvey (2011) or Bulk Extractor Garfinkel (2012) tools. These tools were borne from the researchers need to address the capability limitations of existing tools. Tools such as these have become almost "de facto" in the toolkit of digital forensics investigators. Horsman (2019b) points out that the impact of research on real world digital investigations is difficult to quantify as the current factors for evaluating research impact are primarily academic-focused, where there is a trade-off between the competing interests of the academic industry and that of the field which any work is aimed at. Evidently, a tools worth in real world digital investigations can only be realised if it is accepted by the digital forensics community. But how do we bridge this gap? As pointed out by Carrier and Spafford (2003), the digital forensics discipline serves several different communities, i.e., military, law enforcement, private sector, public sector and academia, all of which have operated in silos with little or no sharing of information or ideas. There is a definite need for a mindset of cooperation and openness, perhaps through initiatives where research-based tools are disseminated to industry actors who otherwise historically would not interact with academia. Some efforts have been made in this regard, an example of which is the Digital Forensics and Incident Response (DFIR) Review, which seeks to serve as a focal point for up-to-date community-reviewed applied research and testing in digital forensics and incident response (DFIR, 2020). Researchers can make submissions to the DFIR Review, where they are reviewed by practioners, academics and graduate students. The view of the DFIR Review is that the faster new knowledge can be produced, reviewed, and shared among the DFIR community, the better able the digital forensics industry will be able to cope with advances in device technology, digital artefacts and criminal activities. Despite such efforts however, we are still a long way from bridging the gap between academia and industry.

### 7.3. Centralized forensic tool repository

When doing our research, we found many tools/resources, but these were spread across the Internet on various platforms such as Github, personal websites or repositories (as listed in Sec. 5.1). Especially in the latter case, these repositories mostly contain commercial and open source tools, but do not include tools from peer-reviewed academic papers or those developed for the DFRWS challenges. Additionally, 'forensic catalogues' had different structures (subfields) and thus identical tools were found in different spots. On the other hand, publishers allow/ask for tools during submission. For instance, the IEEE-Dataport[13] "is a valuable and easily accessible data platform that enables users to store, search, access and manage data". Elsevier also encourages sharing research data[14] but positions itself a bit broader as research data is defined as: Raw or processed data files, software, code, models, algorithms, protocols, or methods. While we agree that this is necessary, tools will be even more spread across platforms making it harder to find them. In order to counteract, the community needs to agree on a standardized taxonomy which makes clustering tools easier. Furthermore, publishers and other platforms should agree on a single platform or a way to exchange information. Lastly, there should be a discussion if it should be mandatory to upload source code.

### 7.4. Increasing of reusability/maintainability

Most tools developed for research are done opportunistically. The tool is built to fit a specific purpose in a language that is chosen by the developer, that may not prove to be the best solution. The

[12] https://toolcatalog.nist.gov/(last accessed 2019-10-10).

[13] https://ieee-dataport.org/about-ieee-dataport (last accessed 2019-10-10).

[14] https://www.elsevier.com/authors/author-resources/research-data (last accessed 2019-10-10).

emphasis is not on robustness or maintainability, so the tool does not undergo sufficient testing or meet necessary standards of coding and documentation. Lack of good coding practices can lead to many issues, such as:

- Security: inconsistent, badly written code can introduce security and logic flaws that threaten the integrity of the tool.
- Reliability: If tools are not coded correctly, it can lead to inconsistencies in results from the tool. Known inconsistencies in tool outputs could cause evidence to be ruled inadmissible in a criminal case.
- Extensible: More often than not, tools are written for a standalone, single purpose, making updates or extensions inherently complex.
- Scalability: tools developed in confined conditions using constrained resources and data may not scale well

If open source tools are to be accepted as reliable by both the digital forensic community and courts of law, they must be developed using a clear methodology that is consistent and adheres to good coding practice. However, imposing good coding practices and standards globally may prove to be difficult in the realm of digital forensic tool development, especially in the case of tools built for academic research. There are solutions currently available to tool developers that will enable consistency in code structure, syntax and documentation and thus help support acceptance of the tool and ultimately its reliability. Tools such as linters and static code analyzers can greatly enhance code quality. Linters help identify general programming errors such as syntactic and stylistic errors as well as known coding bugs. Static code analyzers can identify similar errors in addition to searching for known vulnerabilities in the code, such as identifying where code is susceptible to SQL injection or cross-site scripting attacks. These tools are of course up to the discretion of the tool developer, so may never be considered.

One solution to increasing the maintainability of a tool is to develop it as a plugin for an already established tool. Generally, such software has good community support, so a useful plugin could easily achieve widespread acceptance. For example, Sleuthkit Autopsy is a widely accepted digital forensic tool that enables plugin additions and has a vibrant community repository on GitHub (Sleuthkit-Autopsy, 2019). Plugins have been developed for forensic tasks such as viewing forensic data, data extraction, approximate matching and reporting.

### 7.5. Development of IoT forensic tools

Our study also showed that the development of IoT forensic tools is still in its infancy; it has highlighted the gaps in IoT forensic tool development. While we identified multiple standalone tools and plugins that work with Autopsy, these are limited to specific manufacturers of smarthome devices. There is yet to be a generic tool compatible with multiple manufacturers or a comprehensive framework. Given the sheer amount of IoT devices, it is unclear if commercial tools alone can cover the market or if practitioners will require/depend on the research community to provide tools.

In particular, we found there needs to be more tools to extract traces left by IoT applications on mobile devices. Another challenging aspect is tools to acquire and analyse the memory of IoT devices. Lastly, we require an expansion of network forensics tools: IoT devices often use other protocols such as Bluetooth, Zigbee and Z-Wave and thus more development/research is required to process these protocols. It is also important to identify and triage IoT devices on the network, although there has yet to be a tool to automate this process. Analysing the communication protocols of an IoT devices network traffic can determine whether information is encrypted or in plaintext. This information can help identify different entry points and ways to obtain information from the devices.

## 8. Conclusion and future work

For this paper we reviewed almost 800 articles to study the digital forensic tools produced through research since 2014 from various venues. All identified tools were analyzed with respect to various criteria which allowed us to answer four research questions:

*RQ1: What tools have been published? (categorization).* As stated in the introduction, digital forensics is a discipline that often involves tool development. Those tools were highlighted in Sec. 6 where we identified 62 different tools which we categorized according to digital forensics subfields. These subfields were proposed by Netherlands Register of Court Experts (NRGD) (2016) and have been extended to cover the growing areas of digital forensics such as device/IoT forensics.
*RQ2: Are tools freely available and/or maintained? (license, downloadable).* Out of the 62 identified tools, unfortunately only 33 were found (still) to be publicly available. The majority ($\sim$ 80%) of these tools have been released on the Github with the potential to be used by the digital forensic community as viable tools. We closely examined whether these tools were maintained after development and found many were not. Regarding the quality of the comments within the source code we found 29/33 made comments, although the quality of these comments varied in how comprehensive they were. While 24 out of the 33 tools did provide documentation e.g. description of the tool, installation instructions. The level of detail in the documentation for each tool varied, as there is no standard format for developers to follow and present this information.
*RQ3: Are tools applicable and useable? (assessment with respect to programming style, interfaces, code quality, testing, and documentation).* We found that, of the tools available, a high percentage have not been maintained since their publish date, nor have sufficient documentation. The problem lies with the fact that in most cases, tool development is not the focus of the research, but rather a by-product. With a combined lack of coding standards, limited testing, disparate repository locations and poor documentation, it is unlikely these tools will ever be widely adopted by the digital forensic community. Commercial or established open source software remain the de facto “go to” tools for forensic investigators and thus have become accepted as reliable sources of evidence in criminal trials. However, with a distinct lack of proper tool testing standards across the board, it can be argued that the tools identified in our research may have still the potential to produce reliable evidence. In order for these tools to be recognized, we need to develop and adopt a set of guidelines and standards for developing and testing tools for quality, maintainability and ultimately reliability.
*RQ4: What are the current challenges in the area of forensic tool development?.* As suggested, a centralized repository specifically for tested tools is required if tools are to be exposed to the wider community and thus achieve widespread acceptance. We hope that tool researchers (developers) will spend some more time on code documentation and preferably develop plugins instead of stand-alone tools. This would increase re-usability.

Lastly, we found that there is underdevelopment of tools in the area of IoT forensics. Despite being a growing field, there are still many gaps in tool development to automate the process of identifying and triage of IoT devices and new extraction tools/methods of new traces created by the IoT mobile applications. In particular,

we found the need for network forensic tools to study the various communication channels used by IoT devices, knowing whether data is encrypted or in plaintext will help find ways to extract information from the devices.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

We wish to thank the UK EPSRC who have funded this research through a PhD studentship in Cyber Security. Special thanks go Matthew Vastarelli who started this project during his graduate studies at the University of New Haven.

## Appendix C. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.fsidi.2020.300999.

## Appendix A. Available tools

**Table A.1**
Available tools

| Year | Category | Tool Name | Source | Last Modified | P. Language | Licence |
|---|---|---|---|---|---|---|
| 2014 | Device incl. IoT forensics | FSNView | Github | 5 years | Python | No |
| 2014 | Software forensics incl. DBs | Mrsh_net | Private/personal website | 6 years | C/C++ | No |
| 2014 | Software forensics incl. DBs | Sifter | Github | 5 years | Python | No |
| 2014 | Software forensics incl. DBs | OpenLV | Github | 5 years | Java | GNU General Public License v2.0 |
| 2015 | Software forensics incl. DBs | Advanced forensic Ext4 inode carving (ADEIC) | Private/personal website | 1 year | C++ | No |
| 2015 | Memory forensics | Winpmem | | | | |
| Pmem | Github | N/A | Python | Apache License 2.0 | | |
| 2015 | Device incl. IoT forensics | ConvertPDML | Dropbox | 4 years | Python (2.7) | No |
| 2015 | Software forensics incl. DBs | Mrsh-cf | Private/personal website | 4 years | C/C++ | No |
| 2016 | Network forensics | Kumodd | Github | 4 years | Python | GNU General Public License v2.0 |
| 2016 | Computer forensics | Advanced Automated Disk Investigation Toolkit (AUDIT) | Source Forge | 3 years | Java | No |
| 2016 | Malware forensics | AMExtractor | Github | 4 years | C | No |
| 2017 | Software forensics incl. DBs | A Forensic Email Analysis Tool Using Dynamic Visualization | Private/personal website | 4 years | Python, JavaScript | No |
| 2017 | Device incl. IoT forensics | DRone Open source Parser (DROP) | Github | 3 years | Python | MIT License |
| 2017 | Software forensics incl. DBs | Digital Forensics Framework (DFF) plugin | Github | 2 years | Python/C++ | No |
| 2017 | Device incl. IoT forensics | GE-FANUC controller | Private/personal website | 7 months | C++ | No |
| 2017 | Multimedia forensics | DECA: a decision-theoretic carving program | Bitbucket | 1 year | C++ | No |
| 2017 | Memory forensics | Heapinfo<br>Heapdump<br>Heapsearch<br>Heaprefs | Github | 1 year | Python | No |
| 2017 | Computer forensics | AutoProv | Github | 1 year | Python | Apache License 2.0 |
| 2018 | Device incl. IoT forensics | AndroParse | Github | 1 year | Golang | GNU General Public License v3.0 |
| 2018 | Software forensics incl. DBs | MongoDB Deleted Data Recovery Tool | Github | – | Python (3.5) | No |
| 2018 | Device incl. IoT forensics | Forensic Evidence Acquisition and Analysis System (FEAAS) | Github | – | Python | No |
| 2018 | Network forensics | Wifimit | Github | – | Python | GNU General Public License v3.0 |
| 2019 | Software forensics incl. DBs | Timeline2GUI | Github | – | Python | No |
| 2019 | Device incl. IoT forensics | Arlo_autopsy<br>Ismartalarm_autopsy<br>Ismartalarm<br>Wink_db<br>Qbee_autopsy<br>Crypto_dec | Github | – | Python | GNU General Public License v3.0 |
| 2019 | Computer forensics | NTFSObjIDParser | Github | – | C++ | GNU General Public License v3.0 |

(*continued on next page*)

**Table A.1** (*continued*)

| Year | Category | Tool Name | Source | Last Modified | P. Language | Licence |
|---|---|---|---|---|---|---|
| 2019 | Memory forensics | Winesap | Github | – | Python | No |
| 2019 | Software forensics incl. DBs | Bring2lite | Github | – | | CC-BY-NC - Creative Commons license |
| 2019 | Memory forensics | Hooktracer | Github | – | | No |
| 2019 | Software forensics incl. DBs | AFF4 evidence container | Github | – | Python | Apache License 2.0 |
| 2019 | Memory forensics | Ptenum | Github | – | Python | No |
| 2019 | Memory forensics | Vivedump | Github | – | Python | No |
| 2019 | Device incl. IoT forensics | Digital Drone Forensic application | Github | – | Java 8(JDK 1.8), JavaFX 8 libraries | No |
| 2019 | Malware forensics | BotDAD | Github | – | Python | Apache License 2.0 |

## Appendix B. DFRWS Challenges

**Table B.2**
Tools developed from DFRWS challenges from 2014 to 2019

| Year | Challenge topic | Developer | Tool name | P. language | Last modified |
|---|---|---|---|---|---|
| 2014 | Android Malware | Society (2014) | Manal | N/A | 2015 |
| 2015 | GPU memory | Antonio & Davide (2015) | Gpuhost proc maps | | N/A |
| | | | Gpuhost dump map | | |
| | | Quates (2015) | Nullfinder | C++ | 2015 |
| 2016 | Software Defined Networking (SDN) | Bull et al. (2016) | Booze Allen Hamilton(BAH) | Python | N/A |
| | | | 2 Volatility patch Files find file.patch | | |
| | | | Recover filesystem.patch | | |
| 2017–2018 | IoT | Kost et al. (2017) | Google OnHub parser | Python | 2018 |
| | | Cho et al. (2017) | Google OnHub log parser | | |
| | | | JSON parser | | |
| | | | SQLite database extractor | | |
| | | | Timeline viewer | Python | 2018 |
| | | Gramajo et al. (2017) | OnHub parsing | Python | 2018 |
| | | | SmartThings | | |
| | | | Kodi | | |
| | | | Googlehangouts message | | |
| 2018–2019 | IoT | Hines et al. (2018) | IsmartAlarm android | | |
| | | | Android gmail plugin | | |
| | | Lee et al. (2018) | Alexa cift parser | Python | – |
| | | | IsmartAlarm dairy parser | | |
| | | | IsmartAlarm base station Server Stream Parser | | |
| | | | Nest video recovery | | |
| | | | Wink activity parser | | |

## References


Abt, S., Baier, H., 2014. Are we missing labels? a study of the availability of ground-truth in network security research. In: 2014 Third International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), pp. 40–55. https://doi.org/10.1109/BADGERS.2014.11.

Alrabaee, S., Wang, L., Debbabi, M., 2016. BinGold: towards robust binary analysis by extracting the semantics of binary code as semantic flow graphs (SFGs). Digit. Invest. 18, S11–S22. https://doi.org/10.1016/j.diin.2016.04.002. URL, 10.1016/j.diin.2016.04.002.

Altheide, C., Carvey, H.A., 2011. Digital Forensics with Open Source Tools. Syngress, Burlington, MA.

Antonio, V., Davide, B., 2015. GPU Malware Research. http://www.cs.uno.edu/~golden/Materials/gpumalware/dfrws-challenge-cquates.zip.

Armknecht, F., Dewald, A., 2015. Privacy-preserving email forensics. Digit. Invest. 14, S127–S136. https://doi.org/10.1016/j.diin.2015.05.003. URL, 10.1016/j.diin.2015.05.003.

Ashfaq, Q., Khan, R., Farooq, S., 2019. A comparative analysis of static code analysis tools that check java code adherence to java coding standards. In: 2019 2nd International Conference on Communication, Computing and Digital Systems (C-CODE), pp. 98–103.

Baggili, I., Marrington, A., Jafar, Y., 2014. Performance of a logical, five- phase, multithreaded, bootable triage tool. In: Peterson, G., Shenoi, S. (Eds.), Advances in Digital Forensics X. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 279–295.

Beebe, N.L., Liu, L., 2014. Ranking algorithms for digital forensic string search hits. Digit. Invest. 11, S124–S132. https://doi.org/10.1016/j.diin.2014.05.007. URL:, 10.1016/j.diin.2014.05.007.

Block, F., Dewald, A., 2017. Linux memory forensics: dissecting the user space process heap. Digit. Invest. 22, S66–S75. https://doi.org/10.1016/j.diin.2017.06.002. URL:, 10.1016/j.diin.2017.06.002.

Block, F., Dewald, A., 2019. Windows memory forensics: detecting (Un)Intentionally hidden injected code by examining page table entries. Digital investigation, 29, S3–S12. URL: https://doi.org/10.1016/j.diin.2019.04.008, 10.1016/j.diin.2019.04.008.

Breitinger, F., Baggili, I., 2014. File detection on network traffic using approximate matching. Digital Forensics, Security and Law (JDFSL) 9, 23–36.

Breitinger, F., Baier, H., White, D., 2014. On the database lookup problem of approximate matching. In: Proceedings of the Digital Forensic Research Conference, DFRWS 2014 EU, vol. 11, pp. S1–S9. https://doi.org/10.1016/j.diin.2014.03.001. URL, 10.1016/j.diin.2014.03.001.

Britt, J., Sprague, A., Warner, G., 2015. Phishing intelligence using the Simple set comparison tool. In: Annual Conference on Digital Forensics. Security and Law.

Bull, J., Arnott, W., Christou, C., Duquette, T., Ertekin, E., Lundberg, M., McAlister, M., Starkey, G., 2016. Sdn Forensics Challenge, 2016. URL: https://www.cmand.org/sdn/sdnf.html. Online; accessed 28-September-2019.

Carrier, B., 2002. Open Source Digital Forensics Tools: the Legal Argument. Technical Report stake.

Carrier, B., 2003. Defining digital forensic examination and analysis tool using abstraction layers. Int. J. Data Eng. 1.

Carrier, B., Spafford, E.H., 2003. Getting physical with the digital investigation process. Int. J. Digital Evidence 2 (2).

Carvey, H., 2011. Regripper. URL: https://www.dfir.training/dfirtools/tools/by-developer/harlan-carvey/25-harlan-carvey-regripper.

Case, A., Jalalzai, M.M., Firoz-Ul-Amin, M., Maggio, R.D., Ali-Gombe, A., Sun, M.,

Richard, G.G., 2019. HookTracer: a system for automated and accessible API hooks analysis. Digit. Invest. 29, S104–S112. https://doi.org/10.1016/j.diin.2019.04.011. URL:, 10.1016/j.diin.2019.04.011.
Casey, P., Lindsay-Decusati, R., Baggili, I., Breitinger, F., 2019. Inception: virtual space in memory space in real space – memory forensics of immersive virtual reality with the HTC vive. Digit. Invest. 29, S13–S21. https://doi.org/10.1016/j.diin.2019.04.007. URL:, 10.1016/j.diin.2019.04.007.
Cho, J., Kim, S., Kang, S., Kim, G., Kim, J., 2017. dfrws2017-challenge. https://github.com/dfrws/dfrws2017-challenge/tree/master/challenge-submissions/kookmin.university/Tools.
Clark, D.R., Meffert, C., Baggili, I., Breitinger, F., 2017. DROP (DRone open source parser) your drone: forensic analysis of the DJI Phantom III. Digital investigation. URL: https://dfrws.org/conferences/dfrws-usa-2017/sessions/drop-drone-open-source-parser-your-drone-forensic-analysis-dji, 10.1016/j.diin.2017.06.013, 22, s3-s14.
Cohen, M., Garfinkel, S., Schatz, B., 2009. Extending the advanced forensic format to accommodate multiple data sources, logical evidence, arbitrary information and forensic workflow. Digit. Invest. 6, S57–S68. https://doi.org/10.1016/j.diin.2009.06.010.
Debinski, M., Breitinger, F., Mohan, P., 2019. Timeline2GUI: a Log2Timeline CSV parser and training scenarios. Digit. Invest. 28, 34–43. URL: https://linkinghub.elsevier.com/retrieve/pii/S1742287618303232, 10.1016/j.diin.2018.12.004.
Denton, G., Karpisek, F., Breitinger, F., Baggili, I., 2017. Leveraging the SRTP protocol for over-the-network memory acquisition of a GE Fanuc Series 90-30. Digit. Invest. 22, S26–S38. URL: https://dfrws.org/conferences/dfrws-usa-2017/sessions/leveraging-srtp-protocol-over-network-memory-acquisition-ge, 10.1016/j.diin.2017.06.005.
Dewald, A., Seufert, S., 2017. Afeic: Advanced Forensic Ext4 Inode Carving. Digital Investigation, vol. 20, pp. S83–S91. URL: http://www.sciencedirect.com/science/article/pii/S1742287617300270. https://doi.org/10.1016/j.diin.2017.01.003. DFRWS 2017 Europe.
DFIR, 2020. Digital forensics and incicent response review. Online. URL: https://dfrws.org/dfir-review/.
Dorai, G., Baggili, I., Haven, W., 2018. I know what you did last summer : your smart home Internet of things and your iPhone forensically ratting you out. In: Proceedings of the 13th International Conference on Availability, Reliability and Security - ARES 2018 August, pp. 1–10. URL: http://dl.acm.org/citation.cfm?doid=3230833.3232814, 10.1145/3230833.3232814.
Gannon, M., Warner, G., 2017. An accidental discovery of IoT botnets and a method for investigating them with a custom lua dissector BOTNETS and A : method for. J. Digital Forensics, Security and Law. https://commons.erau.edu/adfsl/2017/papers/3.
Garfinkel, S., 2012. Bulk extractor. URL: https://github.com/simsong/bulk_extractor.
Garfinkel, S., Farrell, P., Roussev, V., Dinolt, G., 2009. Bringing science to digital forensics with standardized forensic corpora. Digit. Invest. 6, S2–S11. https://doi.org/10.1016/j.diin.2009.06.016.
Github, 2019. Licensing a repository. Online. URL: https://help.github.com/en/articles/licensing-a-repository.
Gladyshev, P., James, J.I., 2017. Decision-theoretic file carving. Digit. Invest. 22, 46–61. https://doi.org/10.1016/j.diin.2017.08.001. URL:, 10.1016/j.diin.2017.08.001.
Gloe, T., Fischer, A., Kirchner, M., 2014. Forensic analysis of video file formats. Digit. Invest. 11, S68–S76. https://doi.org/10.1016/j.diin.2014.03.009. URL:, 10.1016/j.diin.2014.03.009.
Good, R.A., 2017. AutoProv: an Automated File Provenance Collection Tool. Master's thesis Air Force Institute of Technology United States.
Grajeda, C., Breitinger, F., Baggili, I., 2017. Availability of datasets for digital forensics–and what is missing. Digit. Invest. 22, S94–S105.
Gramajo, M., Lewis, J., Sharo, R., Zwach, S., 2017. dfrws2017-challenge. https://github.com/dfrws/dfrws2017-challenge/tree/master/challenge-submissions/ssclant_dfrws_2018_challenge_submission.
Gugelmann, D., Gasser, F., Ager, B., Lenders, V., 2015. Hviz: HTTP(S) traffic aggregation and visualization for network forensics. Digit. Invest. 12, S1–S11. https://doi.org/10.1016/j.diin.2015.01.005. URL:, 10.1016/j.diin.2015.01.005.
Gupta, V., Breitinger, F., 2015. How cuckoo filter can improve existing approximate matching techniques. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, vol. 157, pp. 39–52. URL: http://link.springer.com/10.1007/978-3-319-25512-5{_}4, 10.1007/978-3-319-25512-5_4.
Hansen, R., Gino, L., Savio, D., 2016. Covert6: a tool to corroborate the existence of IPv6 covert channels. In: Annual Conference on Digital Forensics, Security and Law. URL: http://commons.erau.edu/adfsl/2016/tuesday/13.
Hargreaves, C., Marshall, A., 2019. SyncTriage: using synchronisation artefacts to optimise acquisition order. Digit. Invest. 28, S134–S140. https://doi.org/10.1016/j.diin.2019.01.022. URL:, 10.1016/j.diin.2019.01.022.
Hines, K., Lewis, J., Phillpott, N., Sharo, R., Zwach, S., 2018. dfrws2018-challenge. https://github.com/dfrws/dfrws2018-challenge/tree/master/challenge-submissions/spawar-niwclant.
Horsman, G., 2019. Tool testing and reliability issues in the field of digital forensics. Digit. Invest. 28, 163–175. https://doi.org/10.1016/j.diin.2019.01.009.
Horsman, G., 2019b. Raiders of the lost artefacts: championing the need for digital forensics research. Forensic Sci. Int.: Rep. 1, 100003. https://doi.org/10.1016/j.fsir.2019.100003, 2019.
Jo, W., Shin, Y., Kim, H., Yoo, D., Kim, D., Kang, C., Jin, J., Oh, J., Na, B., Shon, T., 2019. Digital forensic practices and methodologies for AI speaker ecosystems. Digit. Invest. 29, S80–S93. https://doi.org/10.1016/j.diin.2019.04.013. URL:, 10.1016/j.diin.2019.04.013.
Karabiyik, U., Aggarwal, S., 2016. Advanced automated disk investigation toolkit. In: Peterson, G., Shenoi, S. (Eds.), Advances in Digital Forensics XII. Springer International Publishing, Cham, pp. 379–396.
Karpisek, F., Baggili, I., Breitinger, F., 2015. WhatsApp network forensics: decrypting and understanding the WhatsApp call signaling messages. Digit. Invest. 15, 110–118. https://doi.org/10.1016/j.diin.2015.09.002. URL:, 10.1016/j.diin.2015.09.002.
Kim, J., Park, A., Lee, S., 2016. Recovery method of deleted records and tables from ESE database. Digit. Invest. 18, S118–S124. https://doi.org/10.1016/j.diin.2016.04.003. URL:, 10.1016/j.diin.2016.04.003.
Kost, D., Kouril, D., Kral, B., Nutar, I., 2017. dfrws2017-challenge. https://github.com/dfrws/dfrws2017-challenge/tree/master/challenge-submissions/masaryk.university.
Koven, J., Bertini, E., Dubois, L., Memon, N., 2016. InVEST: intelligent visual email search and triage. Digit. Invest. 18, S138–S148. https://doi.org/10.1016/j.diin.2016.04.008. URL:, 10.1016/j.diin.2016.04.008.
Lee, G., Choi, H., Kim, N., Jin, P., Park, S., Kim, S., Lee, S., Kim, S., Jin, S., 2018. dfrws2018-challenge. https://github.com/dfrws/dfrws2018-challenge/blob/master/challenge-submissions/tapioca.pearlo/TapiocaPearlo_Tools.zip.
Lexico, 2019. Tool | definition of tool by Lexico. URL: https://www.lexico.com/en/definition/tool.
Manson, D., Carlin, A., Ramos, S., Gyger, A., Kaufman, M., Treichelt, J., 2007. Is the open way a better way? digital forensics using open source tools. In: 2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07). IEEE, 266b–266b.
Mendoza, A., Kumar, A., Midcap, D., Cho, H., Varol, C., 2015. BrowStEx: a tool to aggregate browser storage artifacts for forensic analysis. Digit. Invest. 14, 63–75. https://doi.org/10.1016/j.diin.2015.08.001. URL:, 10.1016/j.diin.2015.08.001.
Meng, C., Baier, H., 2019. bring2lite: a structural concept and tool for forensic data analysis and recovery of deleted SQLite records. Digit. Invest. 29, S31–S41. https://doi.org/10.1016/j.diin.2019.04.017. URL:, 10.1016/j.diin.2019.04.017.
Murtuza, S., Verma, R., Govindaraj, J., Gupta, G., 2015. A tool for extracting static and volatile forensic artifacts of windows 8.x apps. In: Peterson, G., Shenoi, S. (Eds.), Advances in Digital Forensics XI. Springer International Publishing, Cham, pp. 305–320.
Nelson, A.J., Steggall, E.Q., Long, D.D., 2014. Cooperative mode: comparative storage metadata verification applied to the Xbox 360. Digit. Invest. 11, S46–S56. https://doi.org/10.1016/j.diin.2014.05.004. URL:, 10.1016/j.diin.2014.05.004.
Netherlands Register of Court Experts NRGD, 2016. Standards 008.0 Digital Forensics. Technical Report Netherlands Register of Court Experts. https://www.nrgd.nl/binaries/Standards\%20Digital\%20Forensics_tcm39-82994.pdf.
NIST, 2015. Nist policy on hash functions. Online. URL: https://csrc.nist.gov/Projects/Hash-Functions/NIST-Policy-on-Hash-Functions.
Nordvik, R., Toolan, F., Axelsson, S., 2019. Using the object ID index as an investigative approach for NTFS file systems. Digit. Invest. 28, S30–S39. https://doi.org/10.1016/j.diin.2019.01.013. URL:, 10.1016/j.diin.2019.01.013.
Quates, C., 2015. nullfinder. https://github.com/candicenonsense/nullfinder.
Quinn, C., Scanlon, M., Farina, J., Kechadi, M.T., 2015. Forensic analysis and remote evidence recovery from syncthing: an open source decentralised file synchronisation utility. Lecture Notes of the Ins. Comput. Sci. Social-Informatics and Telecommun. Eng, LNICST 157, 85–99. https://doi.org/10.1007/978-3-319-25512-5_7.
Rahimian, A., Shirani, P., Alrbaee, S., Wang, L., Debbabi, M., 2015. BinComp: a stratified approach to compiler provenance Attribution. Digit. Invest. 14, S146–S155. https://doi.org/10.1016/j.diin.2015.05.015. URL:, 10.1016/j.diin.2015.05.015.
Renduchintala, A., Jahan, F., Khanna, R., Javaid, A.Y., 2019. A comprehensive micro unmanned aerial vehicle (UAV/Drone) forensic framework. Digit. Invest. 30, 52–72. https://doi.org/10.1016/j.diin.2019.07.002. URL:, 10.1016/j.diin.2019.07.002.
Richard, G.G., Case, A., 2014. In lieu of swap: analyzing compressed RAM in Mac OS X and Linux. Digit. Invest. 11, S3–S12. https://doi.org/10.1016/j.diin.2014.05.011. URL:, 10.1016/j.diin.2014.05.011.
Roussev, V., Golden, R.G., Lodovico, M., 2007. Multi-resolution similarity hashing. Digit. Invest. 4, 105–113. https://doi.org/10.1016/j.diin.2007.06.011.
Roussev, V., Ahmed, I., Sires, T., 2014. Image-based kernel fingerprinting. Digit. Invest. 11 https://doi.org/10.1016/j.diin.2014.05.013.
Roussev, V., Barreto, A., Ahmed, I., 2016. API-based forensic acquisition of cloud drives. In: IFIP Advances in Information and Communication Technology, vol. 484, pp. 213–235. URL: https://arxiv.org/pdf/1603.06542.pdf%20http://arxiv.org/abs/1603.06542, 10.1007/978-3-319-46279-0_11. arXiv:arXiv:1011.1669vol. 3.
Sammons, J., 2015. Chapter 3 - Labs and tools. In: Sammons, J. (Ed.), The Basics of Digital Forensics, second ed. Syngress, Boston, pp. 31–46. URL:, second ed. http://www.sciencedirect.com/science/article/pii/B9780128016350000036. 10.1016/B978-0-12-801635-0.00003-6.
Schatz, B.L., 2019. AFF4-L: a scalable open logical evidence container. Digit. Invest. 29, S143–S149. https://doi.org/10.1016/j.diin.2019.04.016.
Schmicker, R., Breitinger, F., Baggili, I., 2019. Androparse - an android feature extraction framework and dataset. In: Breitinger, F., Baggili, I. (Eds.), Digital Forensics and Cyber Crime. Springer International Publishing, Cham, pp. 66–88.
Senthivel, S., Ahmed, I., Roussev, V., 2017. SCADA network forensics of the PCCC protocol. Digit. Invest. 22, S57–S65. URL: https://dfrws.org/conferences/dfrws-

usa-2017/sessions/scada-network-forensics-pccc-protocol, 10.1016/j.diin.2017.06.012.
Servida, F., Casey, E., 2019. IoT forensic challenges and opportunities for digital traces. Digit. Invest. 28, 22–29. https://doi.org/10.1016/j.diin.2019.01.012. https://www.sciencedirect.com/science/article/pii/S1742287619300222.
Singh, M., Singh, M., Kaur, S., 2019. Detecting bot-infected machines using DNS fingerprinting. Digit. Invest. 28, 14–33. https://doi.org/10.1016/j.diin.2018.12.005.
Sleuthkit-Autopsy, 2019. Sleuthkit Autopsy Tookit Plugin Repository. https://github.com/sleuthkit/autopsy_addon_modules.
Socała, A., Cohen, M., 2016. Automatic profile generation for live Linux Memory analysis. Digit. Invest. 16, S11–S24. https://doi.org/10.1016/j.diin.2016.01.004.
Society, S.T., 2014. SeoulTech/manal. Online. URL: https://github.com/SeoulTech/Manal.
Stadlinger, J., Dewald, A., 2017. A forensic email analysis tool using dynamic visualization. J. Digital Forensics, Security and Law 12. URL: http://commons.erau.edu/jdfsl/vol12/iss1/6/, 10.15394/jdfsl.2017.1413.
Stüttgen, J., Vömel, S., Denzel, M., 2015. Acquisition and analysis of compromised firmware using memory forensics. Digit. Invest. 12, S50–S60. https://doi.org/10.1016/j.diin.2015.01.010.
Sylve, J.T., Marziale, V., Richard, G.G., 2016. Pool tag quick scanning for windows memory analysis. Digit. Invest. 16, S25–S32. https://doi.org/10.1016/j.diin.2016.01.005. URL:, 10.1016/j.diin.2016.01.005.
Taubmann, B., Frädrich, C., Dusold, D., Reiser, H.P., 2016. TLSkex: harnessing virtual machine introspection for decrypting TLS communication. Digit. Invest. 16, S114–S123. https://doi.org/10.1016/j.diin.2016.01.014.
Tips4PC, 2014. 5 Dangers of Free Software. https://techtalk.pcmatic.com/2014/08/26/dangers-free-software/.
Uroz, D., Rodríguez, R.J., 2019. Characteristics and detectability of Windows auto-start extensibility points in memory forensics. Digit. Invest. 28, S95–S104. https://doi.org/10.1016/j.diin.2019.01.026. URL:, 10.1016/j.diin.2019.01.026.
Vadalasetty, S.R., 2003. Security Concerns in Using Open Source Software for Enterprise Requirements. SANS Institute.
Vidas, T., Kaplan, B., Geiger, M., 2014. OpenLV: empowering investigators and first-responders in the digital forensics process. Digit. Invest. 11, S45–S53. URL: https://www.sciencedirect.com/science/article/pii/S1742287614000115, 10.1016/J.DIIN.2014.03.006.
Vondráček, M., Pluskal, J., Ryšavỳ, O., 2018. Automated man-in-the-middle attack against wi-fi networks. J. Digital Forensics, Security and Law: JDFSL 13, 59–80.
vs Merrell, Daubert, 1993. Daubert v. merrell dow pharmaceuticals, inc. URL: https://supreme.justia.com/cases/federal/us/509/579/.
Wagner, J., Rasin, A., Grier, J., 2016. Database image content explorer: carving data that does not officially exist. Digit. Invest. 18, S97–S107. https://doi.org/10.1016/j.diin.2016.04.015. URL:, 10.1016/j.diin.2016.04.015.
Wagner, J., Rasin, A., Heart, K., Jacob, R., Grier, J., 2019. DB3F & DF-toolkit: the database forensic file format and the database forensic toolkit. Digit. Invest. 29, S42–S50. https://doi.org/10.1016/j.diin.2019.04.010. URL:, 10.1016/j.diin.2019.04.010.
Wei, F., Lin, X., Yang, K., Zhu, T., Chen, T., 2018. Automated forensic analysis of mobile applications on Android devices. Digit. Invest. 26, S59–S66. https://doi.org/10.1016/j.diin.2018.04.012. URL:, 10.1016/j.diin.2018.04.012.
Yang, H., Zhuge, J., Liu, H., Liu, W., 2016. A tool for volatile memory acquisition from android devices. In: Peterson, G., Shenoi, S. (Eds.), Advances in Digital Forensics XII. Springer International Publishing, Cham, pp. 365–378.
Yau, K., 2015. PLC forensics based on Control program logic change detection. JDFSL 10, 59–68. URL: http://ojs.jdfsl.org/index.php/jdfsl/article/view/349.
Yoon, J., Lee, S., 2018. A method and tool to recover data deleted from a MongoDB. Digit. Invest. 24, 106–120. URL: https://www.sciencedirect.com/science/article/pii/S1742287617302347, 10.1016/j.diin.2017.11.001.
Zawoad, S., Hasan, R., Grimes, J., 2015. LINCS: towards building a trustworthy litigation hold enabled cloud storage system. Digit. Invest. 14, S55–S67. https://doi.org/10.1016/j.diin.2015.05.014. URL:, 10.1016/j.diin.2015.05.014.
Zoubek, C., Sack, K., 2017. Selective deletion of non-relevant data. In: Digital Investigation, vol. 20, pp. S92–S98. URL: https://www.sciencedirect.com/science/article/pii/S1742287617300300, 10.1016/j.diin.2017.01.006.
Łyczywek, R., 2018. How to write a good readme for your github project? Online. URL: https://bulldogjob.com/news/449-how-to-write-a-good-readme-for-your-github-project.