

# Back and Forth – On automatic Exposure of Origin and Dissemination of Files on Windows

SAMANTHA KLIER, University of the Bundeswehr Munich, RI CODE, Germany

JAN VARENKAMP, Technical University of Darmstadt, Germany

HARALD BAIER, University of the Bundeswehr Munich, RI CODE, Germany

The number of Child Sexual Abuse Material (CSAM) cases has increased dramatically in recent years. This leads to the need to automate various steps in digital forensic processing, especially for CSAM investigations. For instance if CSAM pictures are found on a device, the investigator aims at finding traces about the origin and possible further dissemination, respectively. In this paper we address this challenge with respect to the widespread Windows operating system. We model different common scenarios of system use by CSAM offenders in the scope of file inbound and outbound transfer channels. This gives us insights about digital traces in the Windows operating system and its applications to get knowledge about origin and possible destination of a file. We review available concepts and applications to support this issue. Furthermore we develop a recursive-based approach and provide a prototype as plugin for the open source application Autopsy. We call our prototype *AutoTrack*. Our evaluation against the different models of Windows system usage reveals that Autotrack is superior to existing solutions and provides support of an investigator to find digital traces about the origin and possible further dissemination of files. We publish our AutoTrack plugin and thus provide full reproducibility of our approach.

## ACM Reference Format:

Samantha Klier, Jan Varenkamp, and Harald Baier. 2023. Back and Forth – On automatic Exposure of Origin and Dissemination of Files on Windows. 1, 1 (May 2023), 18 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

## 1 INTRODUCTION

The number of CSAM cases is increasing dramatically in the recent years. The German Federal Criminal Police Office (BKA) registers a more than sixfold increase of CSAM cases for possession and dissemination in the past five years [2], as shown in Figure 1. Most CSAM cases originate from automated reports to the CyberTipline of CSAM uploads to Electronic Service Providers (ESPs), such as WhatsApp. In 2021, 79,701 CyberTipline reports were tracked to Germany alone, which is an almost unimaginable one CSAM upload per 1,000 inhabitants [15] that flood the digital forensic labs which already suffer from long case back logs [3].

Casey et al. [3] state that delays in processing evidence are harmful and will inevitably bog down the criminal justice system, giving offenders time to commit additional crimes and causing immeasurable damage to falsely accused individuals. Casey et al. [3] also argue that this does not restrict to be a technical issue and call for a change of mindset to examine as much data as necessary at an early stage. However, a change of mindset is not sufficient, it must also be supported by automated workflows in order to keep up with the rising number of cases and the increasing data volume.

---

Authors' addresses: Samantha Klier, University of the Bundeswehr Munich, RI CODE, Munich, Germany, [samantha.klier@unibw.de](mailto:samantha.klier@unibw.de); Jan Varenkamp, Technical University of Darmstadt, Darmstadt, Germany, [jan.varenkamp@stud.tu-darmstadt.de](mailto:jan.varenkamp@stud.tu-darmstadt.de); Harald Baier, University of the Bundeswehr Munich, RI CODE, Munich, Germany, [harald.baier@unibw.de](mailto:harald.baier@unibw.de).

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2023 Association for Computing Machinery.

XXXX-XXXX/2023/5-ART \$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

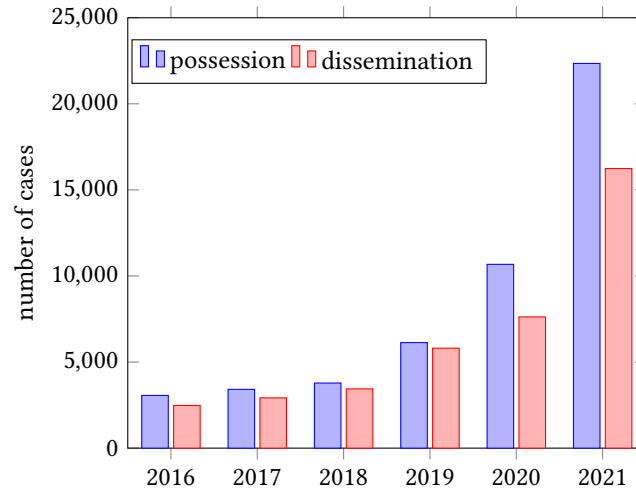


Fig. 1. Trend of possession and dissemination of CSAM cases, as registered by the BKA [2].

The automated extraction of pictures from a system under investigation is already a well-established standard procedure in the digital forensics field (e.g., due to known magic bytes in the common file type headers). Additionally, a system nowadays is automatically screened for known CSAM with the help of cryptographic and perceptual hash databases [12]. Additionally, the screening for yet unknown CSAM is increasingly supported by AI based approaches [12, 16]. As a result the detection of CSAM on a seized system is no longer a challenge.

Typically investigators are confronted with large-scale cases, which involve hundreds of thousands CSAM instances [3, 17] on a single system. Over and above to the detection of CSAM an investigator has to solve additional tasks, for instance the identification of CSAM, which has been produced by the suspect himself, its dissemination or its knowingly possession by the suspect, in order to prosecute.

In this paper we address this additional and important challenge with respect to the widespread Windows operating system and model different common scenarios of system use in the scope of file inbound and outbound transfer channels to answer the questions (i) where did the seized CSAM originate from and (ii) has it been disseminated by the suspect. The origin of the CSAM may reveal traces to victims and can help to stop ongoing sexual child abuse. But at least, it will reveal further suspects or yield information that allows the removal of the origin from the internet. On the other hand, the confirmation of dissemination by the suspect increases his penalty and leads to further suspects, too.

While tracing a file back to its origin or revealing dissemination is of use for other investigations, such as violation of confidentiality, we focus on CSAM investigations because the sheer number of these cases and the CSAM instances found is a huge challenge for Law Enforcement Agencies (LEAs) and of paramount importance to society. We furthermore point out that handling actual CSAM within a digital forensic investigation may lead to serious legal concerns and must be clarified in the first step with law enforcement.

With this paper we contribute by reviewing available concepts that address the given questions and their background (Section 2). Additionally, we systematically model common scenarios in the scope of file inbound and outbound channels of CSAM that include the Internet, the Dark Web, Peer-to-Peer and Instant Messaging. We present selection criteria to choose common applications for this task for later evaluation, i.e. we select Autopsy, Plaso and Magnet AXIOM (Section 3). Next, we present our concept of recursive-based search and its prototypical implementation as ingest module for the open source application Autopsy. Our prototype *AutoTrack*

fills the gap of file-trace tracking in available approaches (Section 4). Consequently, we evaluate the selected forensic applications and our prototype based on the modeled scenarios (Section 5) and reveal the advantages of AutoTrack compared to existing approaches. Finally, we conclude our paper (Section 7).

## 2 BACKGROUND AND RELATED WORK

We assume that a file inbound, any subsequent processing, and a file outbound leave a trace in the system that can be (partially) reconstructed using artefacts. By artefact we understand "Information or data created as a result of the use of an electronic device that shows past activity", as defined by the Scientific Working Group on Digital Evidence (SWGDE) [18]. These artefacts can be divided into operating system artefacts and application artefacts. For example, application artefacts can be SQLite databases, like from common web browsers, that contain logs of downloaded files [14]. Or proprietary binary files, like the known .met file of the eMule application, which contains not just downloaded but also shared files. Therefore, application artefacts do provide information of the inbound and outbound of files.

On the other hand, operating system artefacts can provide insight into the most recent internal processing of files, such as a list of transactions relating to files on the volume, including file creation, renaming and deletion, provided by the NTFS USN journal, as stated by [13]. Obviously, it depends on the specific application or operating system what information it stores and in what format. This is a constant challenge for digital forensics and leads to the development of artefact databases, as the Artifact Genome Project (AGP) that aims at a systematic approach for artefact knowledge management [7]. Although, recently a Forensic Artifact Finder (ForensicAF) for Autopsy was proposed by Balon et al. [1] the AGP is still not complete enough to be of use in real world cases. Therefore, the reason why artefacts were created in the first place is heterogeneous and may show only a small excerpt of the complete *file-trace* which is the set of each and only traces related to the processing of a particular file. This means that artefacts must be disassembled in the recorded past activities and subsequently reassembled to include the complete *file-trace*.

However, the explicit tracking of a file through a computer system received little attention so far in the forensics community. While Du et al. [6] extracted a so-called *file artefact timeline* for a machine learning approach to find unknown CSAM, they actually just searched for the name of a file in a Plaso timeline. Such timelines have become a staple of the digital forensics community [5] as the information extracted from artefacts can be easily reassembled based on their timestamps, as demonstrated by Guðjónsson [8] with the Plaso framework (formerly called Log2Timeline). Plaso is extendable and can parse and analyse roughly 140 different artefact types at the moment and is usable from within Autopsy.

As its key drawback, the created timeline is too complex to be manually screened for relevant activities. This issue was addressed by Hargreaves and Patterson [9] who propose the distinction of high-level and low-level events. For example, when a user connects a USB device (i.e. high-level event) this can result in several hundreds of low-level events on the timeline (a small excerpt is shown in Figure 2), while the investigator in the first step is only interested in "USB device connected". Therefore, a focus on high-level events reduces the complexity of the timeline and highlights the potentially interesting events.

However, the generation of such high-level events is problematic, as they have to be defined and maintained for an elusive amount of events, applications, operating systems and versions and suffer as usual from false-positives and false-negatives, respectively. The concept of high-level events has never been introduced to Plaso. Nevertheless, certain events, such as connecting a USB device, can be highlighted in colour when the results are imported into Microsoft Excel using a specific template<sup>1</sup>, which is a common way of working with Plaso timelines, as stated by Debinski et al. [5]. Figure 2 shows a small timeline excerpt of events related to connecting a USB device, with the most expressive events highlighted in blue.

<sup>1</sup><https://medium.com/dfclub/how-to-use-log2timeline-54377e24872a>, last accessed 2023-01-28

4 • Samantha Klier, Jan Varenkamp, and Harald Baier

Line	Timestamp	Source	Message
2082383	2022-10-10 08:45:20	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USBSTOR\Disk&Ven_Samsung&Prod_Type-C&Rev_1100\037622010001118;
2082382	2022-10-10 08:45:20	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USBSTOR\Disk&Ven_Samsung&Prod_Type-C&Rev_1100\037622010001118;
2082379	2022-10-10 08:45:20	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USBSTOR] Device type: Disk Display name: Samsung Type-C USB D
2082378	2022-10-10 08:45:20	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USBSTOR\Disk&Ven_Samsung&Prod_Type-C&Rev_1100\037622010001118;
2082376	2022-10-10 08:45:20	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USBSTOR\Disk&Ven_Samsung&Prod_Type-C&Rev_1100\037622010001118;
2082370	2022-10-10 08:45:20	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USBSTOR\Disk&Ven_Samsung&Prod_Type-C&Rev_1100\037622010001118;
2082369	2022-10-10 08:45:20	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USBSTOR\Disk&Ven_Samsung&Prod_Type-C&Rev_1100\037622010001118;
2082368	2022-10-10 08:45:20	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USBSTOR\Disk&Ven_Samsung&Prod_Type-C&Rev_1100\037622010001118;
2082367	2022-10-10 08:45:20	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Control\Class\{4d36e967-e325-11ce-bfcl-08002be10318}\0001] DriverDi
2082365	2022-10-10 08:45:20	EVT	[410 / 0x019a] Provider identifier: {9c205a39-1250-487d-abd7-e831c6290539} Source Name: Microsoft-Windows-Ki
2082364	2022-10-10 08:45:20	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USBSTOR] Device type: Disk Display name: Samsung Type-C USB D
2082362	2022-10-10 08:45:20	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USBSTOR\Disk&Ven_Samsung&Prod_Type-C&Rev_1100\037622010001118;
2082361	2022-10-10 08:45:20	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USBSTOR\Disk&Ven_Samsung&Prod_Type-C&Rev_1100\037622010001118;
2082355	2022-10-10 08:45:20	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USBSTOR\Disk&Ven_Samsung&Prod_Type-C&Rev_1100\037622010001118;
2082354	2022-10-10 08:45:20	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USBSTOR\Disk&Ven_Samsung&Prod_Type-C&Rev_1100] (empty)
2082350	2022-10-10 08:45:20	EVT	[112 / 0x0070] Provider identifier: {fcb06bb-6a2a-46e3-abaa-246cb4e508b2} Source Name: Microsoft-Windows-Di
2082322	2022-10-10 08:45:19	EVT	[4 / 0x0004] Provider identifier: {0bf2fb94-7b60-4b4d-9766-e82f658df540} Source Name: Microsoft-Windows-Kerr
2082321	2022-10-10 08:45:19	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USB\VID_090C&PID_1000\0376220100011185\Properties\{3464f7a4-2e
2082308	2022-10-10 08:45:19	EVT	[300 / 0x012c] Provider identifier: {fcb06bb-6a2a-46e3-abaa-246cb4e508b2} Source Name: Microsoft-Windows-Di
2082307	2022-10-10 08:45:19	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USB\VID_090C&PID_1000\0376220100011185\Properties\{3464f7a4-2e
2082306	2022-10-10 08:45:19	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USB\VID_090C&PID_1000\0376220100011185\Properties] (empty)
2082305	2022-10-10 08:45:19	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USB\VID_090C&PID_1000\0376220100011185\Device Parameters\Ceip]
2082304	2022-10-10 08:45:19	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USB\VID_090C&PID_1000\0376220100011185\Device Parameters] Enum
2082291	2022-10-10 08:45:19	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Control\DeviceClasses\{a5dcbf10-6530-11d2-901f-00c04fb951ed}\##7#U
2082282	2022-10-10 08:45:19	EVT	[400 / 0x0190] Provider identifier: {9c205a39-1250-487d-abd7-e831c6290539} Source Name: Microsoft-Windows-Ki
2082281	2022-10-10 08:45:19	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Services\USBSTOR] Type: Kernel Device Driver (0x1) Start: Manual (:
2082265	2022-10-10 08:45:19	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USB\VID_090C&PID_1000\0376220100011185\Device Parameters\5b3b
2082263	2022-10-10 08:45:19	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Control\Class\{36fc9e60-c465-11cf-8056-444553540000}] Class: [REG_
2082251	2022-10-10 08:45:19	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USB\VID_090C&PID_1000] (empty)
2082248	2022-10-10 08:45:19	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Control\DeviceContainers] (empty)
2082246	2022-10-10 08:45:19	EVT	[4 / 0x0004] Provider identifier: {0bf2fb94-7b60-4b4d-9766-e82f658df540} Source Name: Microsoft-Windows-Kerr
2082245	2022-10-10 08:45:19	REG	[HKEY_LOCAL_MACHINE\System\ControlSet001\Control\usbflags\090C10001100] (empty)

Fig. 2. Excerpt of events related to connecting a USB device, as extracted by Plaso. Most expressive events are highlighted in blue.

A more sophisticated way of highlighting certain events was proposed by Studiawan et al. [21], who use a deep learning approach for sentiment analysis of forensic timelines. They support an investigator by highlighting adverse events, like failed authentications, which is helpful for incident response but not applicable to our use case which is characterized by authorized user activities, e.g. there is no difference on system level between the download of CSAM via web browser or legal pictures.

By concept, a timeline depends solely on the existence and integrity of timestamps and, hence, is rather limited. A more resilient and holistic way to reassemble the extracted past activities from artefacts is through ontologies, which can store information based on logical predicates and introduce semantic context [11]. However, these ontologies must be pre-defined and the first well-supported ontology for the digital forensic community, namely CASE (an extension of the Unified Cyber Ontology (UCO)) which was proposed by Casey et al. [4], is still under development.

### 3 METHODOLOGY

Our goal is to provide an automated approach to support a digital forensic investigation to find traces about the origin or presumable dissemination of CSAM, respectively. We therefore start with the presentation of the common system usage for the inbound and outbound of CSAM. Based on the results of actual system usage we next derive our concept for the modeling of our scenarios, which serve as the basis to evaluate available software against our own approach.

### 3.1 Common CSAM system usage

Steel et al. [20] analysed the actual system usage of CSAM offenders and their technical behaviour. We summarise their key findings in Table 1. According to the left column in Table 1 the most common devices involved in CSAM-related crimes are desktop PCs and laptops. Consequently, we focus in our scenarios on systems running a Windows 10 operating system, which is the most widely used operating system for laptops and PCs [19]. More specifically, our scenarios, which model a common CSAM system usage, are implemented using a virtual machine<sup>2</sup> running a Windows 10 Home (x64, Version 21H2, Build 19044.2075) guest.

Device Type	Proportion	Channel	Proportion
Desktop PC	0.59	Peer-to-Peer	0.46
Laptop	0.58	Internet (browser)	0.22
Smartphone	0.27	Dark web	0.15
Other	0.13	Instant Messaging	0.12

Table 1. Proportion of device types used as means of crime and dissemination, as proposed by Steel et al. [20]. A offender may use several device types and channels, hence the proportion must not sum up to 1.00.

The right column in Table 1 reveals that peer-to-peer is by far the most common channel for inbound and outbound of CSAM. Further important dissemination channels are the general Internet (i.e. a common web browser), the Dark Web or instant messaging services. With respect to our scenarios we incorporate at least one representative for each dissemination channel, as shown in Table 2.

Channel	Application	Version	Source
Peer-to-Peer	eMule	0.50a	<a href="https://www.emule-project.com/">https://www.emule-project.com/</a>
Internet	Chrome	106.0.5249.119	<a href="https://www.google.com/chrome/">https://www.google.com/chrome/</a>
Internet	Firefox	105.0.3	<a href="https://www.mozilla.org/firefox/">https://www.mozilla.org/firefox/</a>
Internet	Edge	106.0.1370.42	<a href="https://www.microsoft.com/edge">https://www.microsoft.com/edge</a>
Dark Web	Tor Browser	11.5.2	<a href="https://www.torproject.org/download/">https://www.torproject.org/download/</a>
Instant Messaging	WhatsApp Web	as available in 2022-10	<a href="https://web.whatsapp.com/">https://web.whatsapp.com/</a>
Instant Messaging	WhatsApp Desktop	2.228.14.0	<a href="https://www.whatsapp.com/download/">https://www.whatsapp.com/download/</a>
Instant Messaging	Telegram Web	as available in 2022-10	<a href="https://web.telegram.org/">https://web.telegram.org/</a>
Instant Messaging	Telegram Desktop	4.2.4	<a href="https://desktop.telegram.org/">https://desktop.telegram.org/</a>

Table 2. Overview of included applications to our model of realistic system usage.

Additionally, we include a USB device in our scenarios to cover the case of self-produced CSAM that was copied from a camera, as this would be the worst case to miss in an investigation. Otherwise, any anti-forensic measures, like encryption or applications for artefact deletion are beyond the scope of this paper.

Please note, we did not use any CSAM for this research. For ethical and legal reasons, we use pictures of cats instead even when referring to CSAM in the scenario descriptions.

<sup>2</sup>based on Oracle VirtualBox (v6.1.38)

### 3.2 Concept for evaluation scenario models

In this section we explain our concept to model different scenarios of CSAM system usage, which we later use to evaluate different applications and approaches to extract the respective digital traces of CSAM inbound and outbound. Our actual evaluation in Section 5 makes use of our Windows 10 virtual machine to execute these usage scenarios.

First off, we execute very simple user activities to be able to understand the functions of an approach and as a baseline. In the next step, we model more realistic, but increasingly complex scenarios. The simple scenarios contain the following actions, executed with several files for each included application:

- (1) **inbound**: download (as offered by website/application or via save as), copy or move
- (2) **user actions**: copy, move, rename or (un)zip
- (3) **outbound**: upload or send, as offered by website/application

Our presentation of the scenarios uses a graph model that represents the processing steps performed in the system on the incriminating files. The symbols used in the graphs are shown in Table 3.






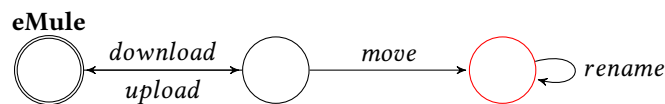
Symbol	Description
	Channel of dissemination (inbound & outbound).
	Directory in the local file system that contains CSAM at the moment of investigation.
	Directory in the local file system that contained the CSAM before the investigation.
	Activity related to the CSAM.
	Activity related to additional information of the CSAM, such as links.

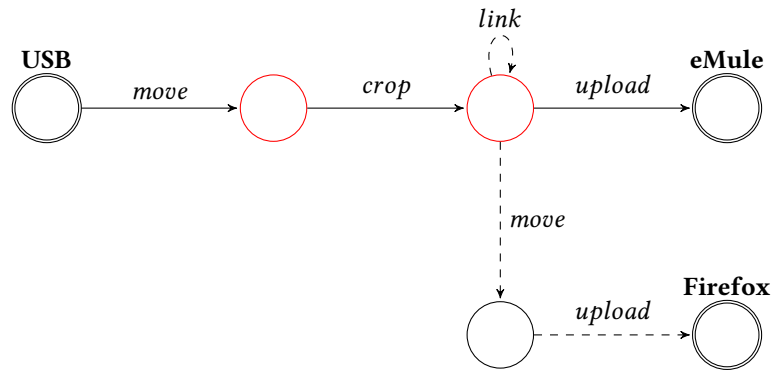
Table 3. Description of the used symbols in our graph model.

Our realistic user activity models are as follows:

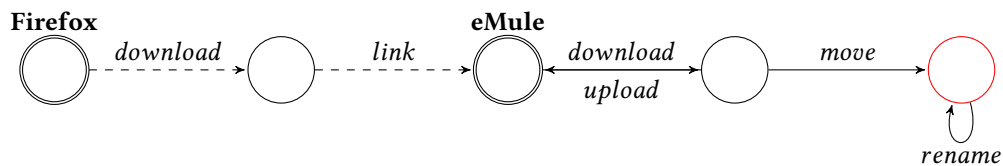
- (1) *The P2P User* downloads CSAM he found by the built-in search function of eMule. He moves and renames the files as soon as the download is complete. Thus, at the time of acquisition and investigation of the system, CSAM is only stored in one directory (red node), unrelated to eMule. However, there may be traces of downloading, moving or renaming (black, nodes and edges). The model graph is as follows:



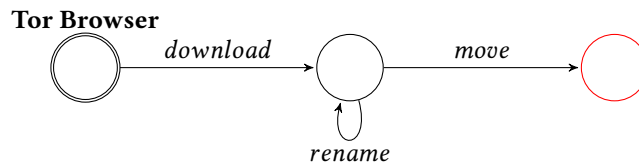
- (2) *The P2P Abuser* actually abuses children and documents it with images. This is self-produced CSAM, and in this scenario it enters the system via a USB device. The *P2P Abuser* crops the CSAM before it is uploaded via eMule. The *P2P Abuser* also creates an eMule link file to the CSAM that he uploads to the Internet. The model graph of this usage scenario is as follows:



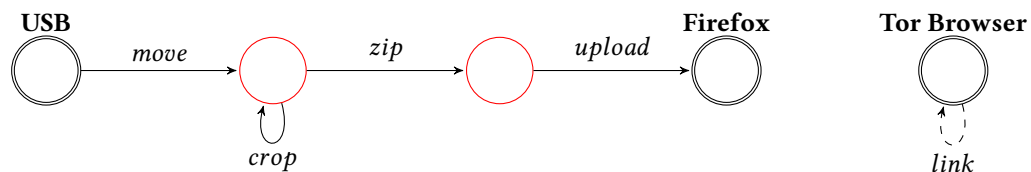
- (3) *The P2P Link User* downloads a link file from the Internet that points to CSAM available per eMule. As soon as the download via eMule is complete the CSAM is moved and renamed. The corresponding model graph looks as follows:



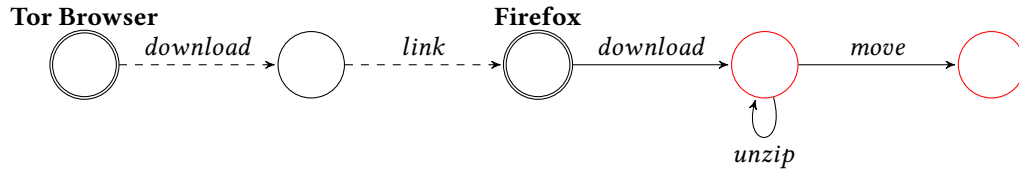
- (4) *The Dark Web User* downloads pictures from the Dark Web, renames and moves them. He makes use of the well-known Tor browser to access the Dark Web. The model graph of this scenario is as follows:



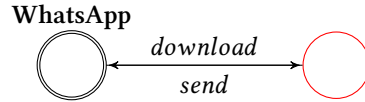
- (5) *The Dark Web Abuser* produces CSAM that enters the system per USB device. The CSAM is then cropped before being compressed into a ZIP archive that is uploaded to a file sharing host. A link to the ZIP archive is posted in the Dark Web via the Tor Browser. This scenario is visualized by the following model graph:



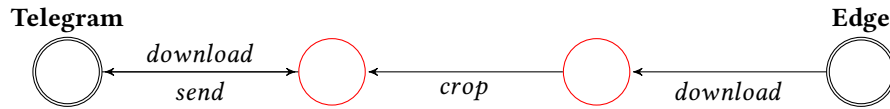
- (6) *The Dark Web Filesharing User* downloads a list of addresses of CSAM hosted by regular file sharing sites from the Dark Web. The CSAM is then downloaded from the surface web using Firefox, which is a way of mitigating the slow download speed of the Dark Web. The graph model of the *Dark Web Filesharing User* scenario looks as follows:



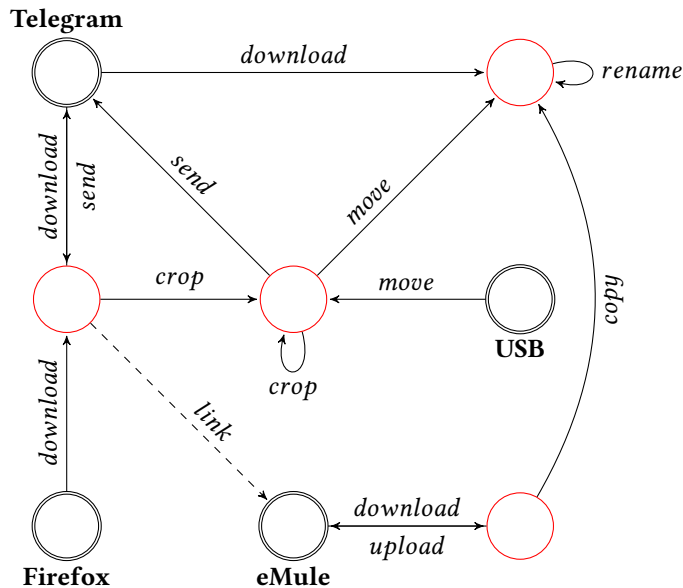
(7) *The WhatsApp* receives and sends pictures via WhatsApp Desktop. Its simple graph model is as follows:



(8) *The Telegramer* receives CSAM via Telegram Desktop as a member of a group. The Telegramer also downloads CSAM from the internet, crops it, and then sends it to all the group members. The corresponding graph model is:



(9) Finally we turn to a more complex scenario, which we call *The Ever More*. The Ever More is a member of a CSAM Telegram group and downloads CSAM and P2P links as such. On the other hand he sends self-produced CSAM via the Telegram Desktop client, too. Furthermore *The Ever More* also downloads CSAM from the Internet and uses eMule with link files for CSAM sharing. The graph model for this scenario is:





### 3.3 Selection criteria for applications

Now, we will present the selection criteria for forensic applications we will subsequently evaluate based on the proposed scenarios. The selection criteria are hierarchically deduced from our findings in Section 2. First off, any application to be considered must be able to parse artefacts (1), the more the better. Next, we consider whether and how the information is correlated (2). In Section 2 we showed that the correlation can be based on time, an ontology or by tracking a file. Finally, we consider the actual capability to extract a *file-trace*, either manually or automatically (3).

Our aim is to select the most powerful and community-accepted applications and to include one representative per approach. In all we choose the applications Magnet AXIOM<sup>3</sup>, Plaso<sup>4</sup> (with Timeline Explorer<sup>5</sup> frontend), X-Ways as well as the current version of Autopsy. Table 4 shows our evaluation of these applications against our selection criteria.

Criteria	AXIOM	Plaso	X-Ways	Autopsy
1 Parse artefacts	yes, 750+ <sup>6</sup>	yes, 140	yes	yes
2 Correlates artefacts	by ontology	by time	by time	no
3 Extracts <i>file-trace</i>	manual (file)	manual (filters)	manual (filters)	no
License	proprietary	open-source	proprietary	open-source

Table 4. Evaluation of forensic applications based on our selection criteria.

Apparently, the most powerful of these applications is Magnet AXIOM, as it apparently includes the most artefact parsers. Additionally, AXIOM is also the only application that correlates artefacts by using an ontology, to discover relations in evidence, as stated by Henseler and Hyde [10]. In order to use this feature (AXIOM refers to the ontology-based correlations as *Connections*) an investigator has to select a CSAM file and then opens the *Connections* view to extract the *file-trace*. Based on our selection criteria, especially due to the ontology-based correlation, we make use of AXIOM in our later scenario evaluation in Section 5.

In contrast, Plaso and X-Ways correlate the artefact information based on time stamps which means they create a timeline. These timelines can be filtered manually, e.g. by the file name, to expose a *file-trace*. As X-Ways and Plaso provide a similar functionality, we select Plaso for our later scenario evaluation in Section 5 due to its open-source licensing.

Unfortunately, none of the previous applications is capable of automatically extracting a *file-trace* or correlating artefacts by tracking a file. To bridge this gap, we provide a concept with improved file tracing in Section 4 and implement it as a prototype on base of Autopsy due to its open-source licensing and its ability to parse artefacts. Therefore, we select Autopsy for our scenario-based evaluation in Section 5 as a baseline for our own prototype.

## 4 CONCEPT OF RECURSIVE BACKWARD SEARCH AND ITS PROTOTYPE AUTOTRACK

In this section we present our concept of recursive backward search and provide some insights into our prototypical Autopsy ingest module *AutoTrack*<sup>7</sup>.

Figure 3 shows the general approach and major steps of our concept of file tracing backward search, which we provide as *AutoTrack* ingest module. First off, we extract metadata, namely the name, path, size, creation

<sup>3</sup>v2.10.0.13241, <https://www.magnetforensics.com/magnet-axiom/>

<sup>4</sup>plaso-20220930, <https://github.com/log2timeline/plaso/releases/tag/20220930>

<sup>5</sup>v2.0.0.0, <https://ericzimmerman.github.io/>

<sup>6</sup>As advertised by Magnet Forensics.

<sup>7</sup>Downloadable at: [https://cloud.digfor.code.unibw-muenchen.de/s/AutoTrack\\_PoC](https://cloud.digfor.code.unibw-muenchen.de/s/AutoTrack_PoC)

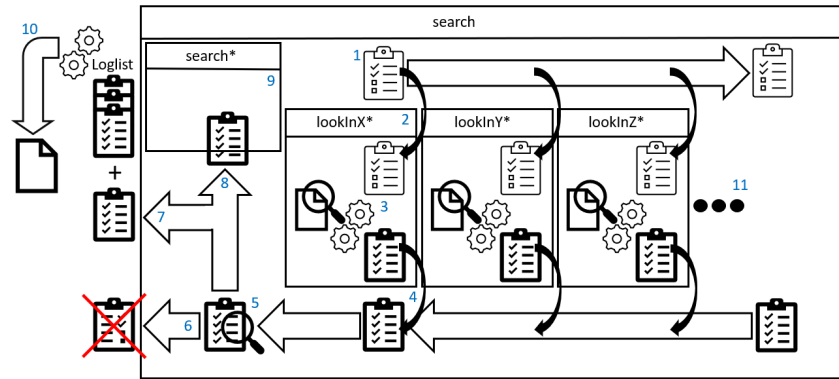


Fig. 3. Structure of the depth-first search of AutoTrack.

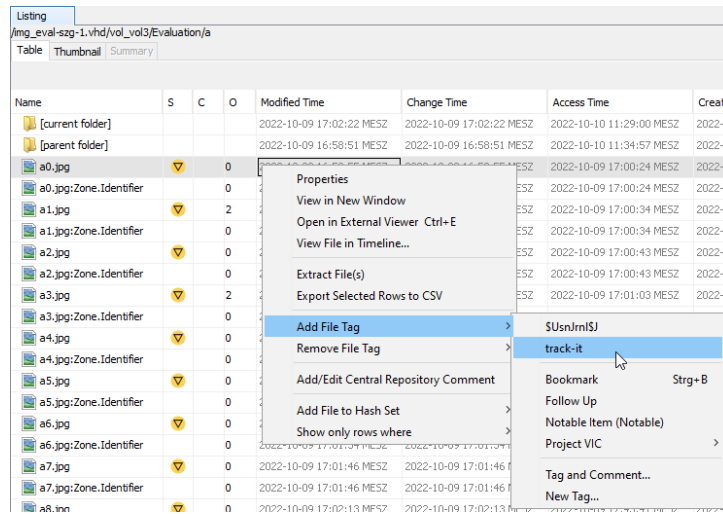


Fig. 4. AutoTrack integration in Autopsy’s context menu.

date, change date and cryptographic hashes, from the tagged files into a Loglist and start the recursive backward search (1). Firstly, the metadata is passed to one of the available artefact wrapper functions, we call *lookIn* (2). The underlying parser (3) disassembles their dedicated artefacts and yields alternative metadata (4) for the file. The alternative metadata is checked for duplicates (5) which are discarded (6) or otherwise added to the loglist (7) and immediately used for another call of the search, following a depth-first approach (8). The next wrapper function is called when the proceeding yields no new alternative metadata (9). Finally, from the Loglist containing all collected traces, AutoTrack generates a chronological HTML/JSON report (10) that is presented to the investigator.

We implement a prototype that tracks a file across artefacts and automatically extracts a *file-trace* based on a recursive backwards search. Our prototype AutoTrack consists of two Autopsy plugins. One plugin simply offers a *track-it* tag in the Result Viewer context menu of Autopsy, as shown in Figure 4.

The second plugin is a report module which tracks each as *track-it* tagged file by collecting relevant traces from dedicated artefact parsers and generates a JSON/HTML report. Therefore, an investigator first tags files that

should be tracked and then uses Autopsy’s *Generate Report* function. A sample report for a simple file download from the Internet is shown in Figure 5. Obviously, the exposed traces of AutoTrack merely depend on the quantity and quality of wrapped artefact parsers.

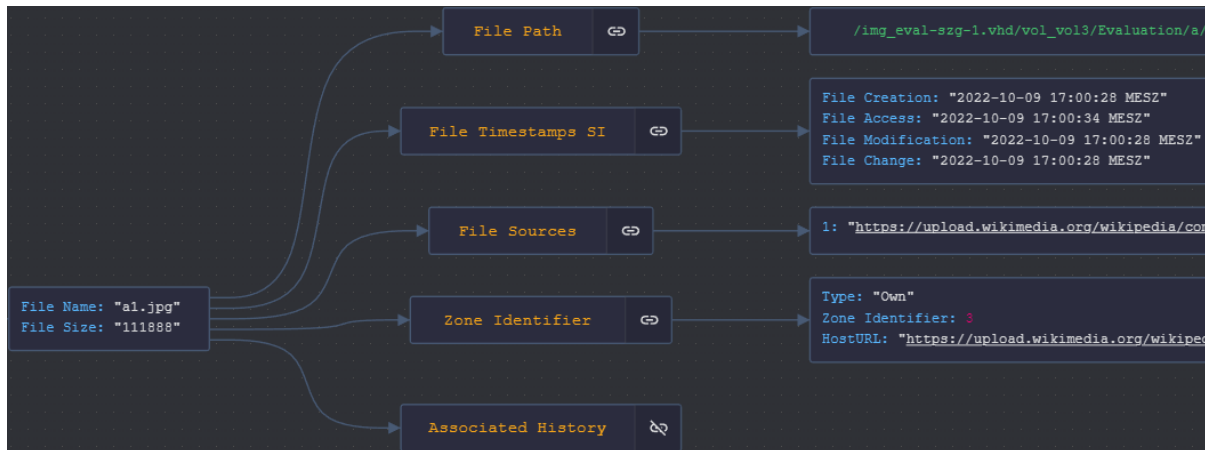


Fig. 5. Generated report of AutoTrack for a simple file download from the Internet. The associated history contains the detailed trace, including the used web browser.

In the current version of our prototype we include eight wrappers as shown in Table 5, however, AutoTrack can easily be extended by adding more wrappers as indicated as step (11) in Figure 3. At the moment, the external parsers, namely MFTECmd and DB4SQL, have to be initiated in a pre-processing step that can be shifted into AutoTrack for fully automated workflows in upcoming versions.

Wrapper	Input	Output	Artefact	Parser
NTFS activities	name, path	name, path	NTFS USN journal	MFTECmd <sup>8</sup>
ZIP archives	MD5 hash	name, path, size, timestamps, hashes	ZIP archive	Autopsy (Embedded File Extractor)
P2P links	AICH hash	name, path, size, timestamps, hashes	*.emulecollection	RHash <sup>9</sup>
P2P logs	AICH hash	channel of dissemination	known.met	RHash
Browser History	size, name	channel of dissemination	SQLite databases	DB4S <sup>10</sup>
Internet	name, path	possible channel of dissemination	Zone.Id	-
Tor Browser exec.	create time	possible channel of dissemination	FIREFOX.EXE-XXXXXXXXX.pf	Autopsy (Recent Activity)
USB device	change time	possible channel of dissemination	Microsoft-Windows-Partition %4Diagnostic.evtx	Autopsy (ParseEvtx)

Table 5. Overview of wrappers currently included in AutoTrack.

With just these eight wrappers, we are able to track file system activity, such as renaming or moving of a file based on the known file name and path and by parsing the NTFS USN journal. Additionally, we can trace the zipping of files based on Autopsy's standard functions by MD5 hashes. Furthermore, we can track files back to eMule by their AICH hashes which are recorded by eMule in its *known.met* and *\*.link* files. While the *known.met* proves that the recorded files have been shared, the *\*.link* files can reveal traces to communities sharing such links to CSAM. In order to trace files that have been disseminated by the Internet, we parse the browser histories and interpret the *Zone.ID* as provided by Autopsy. However, for files disseminated by a USB device or by the Tor Browser we can only give a hint instead of a robust *file-trace* to these channels. This means that we show that the Tor browser was executed or a USB device connected before a file was created on the system, which suggests that these channels are the origin of a file, especially if the *file-trace* shows no explicit origin.

## 5 EVALUATION

For the evaluation of the selected forensic applications we execute our scenarios group wise. First, we execute simple file inbounds from every included application, followed by the simple user activities, simple file outbounds and, finally, the realistic scenarios. After the execution of each group we create a snapshot of the virtual machine for the evaluation and reset the virtual machine to its initial state. However, the virtual machine was restarted once, in fact before the execution of our final scenario, we call *The Ever More*.

### 5.1 Qualitative evaluation of simple scenarios

We process the three snapshots that contain the simple scenarios with each forensic application and evaluate the results qualitatively, with the scheme shown in Table 6. Please note, we do not downgrade if additional non-relevant traces are shown to the investigator. This means that we evaluate the applications for their screening capabilities and not for proving, hence, an investigator always has to assess the results.

Rating	Description
++	<b>All</b> file-traces have been exposed and <b>automatically</b> extracted.
+	<b>Some</b> file-traces have been exposed and <b>automatically</b> extracted.
o	<b>All</b> file-traces have been exposed, but need to be extracted <b>manually</b> .
-	<b>No</b> file-traces have been exposed.

Table 6. Evaluation scheme for the simple scenarios.

Our evaluation for simple file in- and outbounds is shown in Table 7 which shows a striking difference in exposure of file inbounds in contrast to outbounds. This is no surprise, however, as all included web browsers despite the Tor browser record downloads, but not uploads, which results in corresponding ratings. Interestingly, Plaso did not expose all of the file traces that lead to an inbound from Firefox, presumably due to the different processing of files downloaded via a button ('Download') versus a context menu ('Save as...'). The file inbounds of WhatsApp Web appear to be regular browser downloads which are well supported, unlike the artefacts generated by the desktop application. In contrast, Telegram treats inbounds differently depending on whether they were sent as a document or a picture. While the received documents of the Web version appear to be regular browser downloads, the recompressed pictures could not be tracked by any application. The Desktop version also saves documents, but not pictures, to a Telegram Desktop directory where they are found by

<sup>8</sup><https://ericzimmerman.github.io/>

<sup>9</sup><https://github.com/rhash/RHash>

<sup>10</sup><https://sqlitebrowser.org/>

<b>Application</b>	<b>AXIOM</b>		<b>Plaso</b>		<b>Autopsy</b>		<b>AutoTrack</b>	
	in	out	in	out	in	out	in	out
eMule	++	++	-	-	-	-	++	++
Chrome	++	-	++	-	++	-	++	-
Firefox	++	-	+	-	++	-	++	-
Edge	++	-	++	-	++	-	++	-
Tor Browser	-	-	-	-	-	-	-	-
WhatsApp Web	++	-	++	-	++	-	++	-
WhatsApp Desktop	-	-	-	-	-	-	-	-
Telegram Web	+	-	+	-	+	-	+	-
Telegram Desktop	+	-	+	-	-	-	-	-
USB Device	-	-	-	-	-	-	-	-

Table 7. Evaluation results for a trivial file inbound.

AXIOM and Plaso. They solely utilize the file name which is an advantage in this case, but, in other cases leads to name collisions. On the other hand, eMule is the only channel of dissemination that is known to record in- and outbounds but only AXIOM and AutoTrack are able to expose these traces. Unfortunately, none of the forensic applications were able to expose a direct file-trace to the connected USB device.

Next, in Table 8 we show an overview of the traceability of simple user actions commonly executed after a file inbound. Obviously, Autopsy is not able to track user actions, while Plaso and AXIOM can track any user action that leaves the filename untouched. AXIOM also shows that the ZIP archive is stored on the same path as the incriminating files which allows to manually expose the file trace but is a very weak indicator. In contrast, AutoTrack can automatically expose any user actions in this scenario and does not solely depend on the file name.

<b>Application</b>	<b>AXIOM</b>	<b>Plaso</b>	<b>Autopsy</b>	<b>AutoTrack</b>
unzip	o	-	-	++
unzip & delete archive	-	-	-	++
move	++	++	-	++
copy	++	++	-	++
copy & delete originals	++	++	-	++
rename	-	-	-	++
crop	-	-	-	-

Table 8. Evaluation results for a trivial user actions.

## 5.2 Graph based evaluation of realistic scenarios

For evaluation purposes we extend our graph model, as shown in 9, to indicate which parts of the file trace could be followed. However, this does not mean that any particular activity, such as *move*, has been detected, it just means that the file can be tracked even though it has been moved.



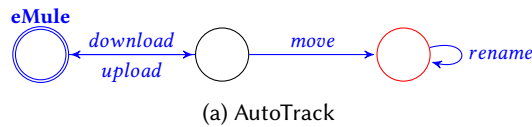
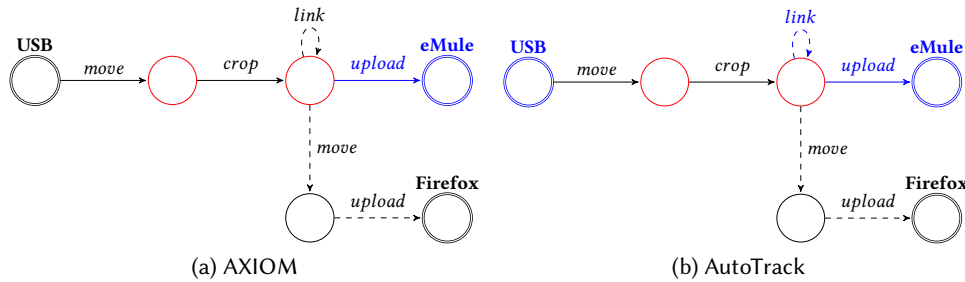
Symbol	Description
	Automatically exposed by the forensic application.
	Manual extraction for exposure necessary.

Table 9. Extension of our graph model for evaluation purposes.

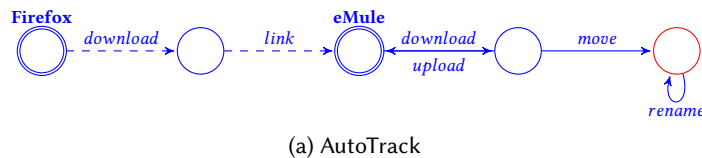
- (1) *The P2P User*: AXIOM, Plaso and Autopsy exposed no relevant traces for this scenario, in contrast, AutoTrack exposed that the files were shared with eMule. In this scenario, AXIOM could not expose eMule presumably due to the renaming. Even though, the moving and renaming was not traceable, as the NTFS USN Journal did not contain the dedicated information at the time of investigation, AutoTrack exposed eMule as the inbound and outbound channel based on AICH hashes. The graph based evaluation for AutoTrack looks like this:



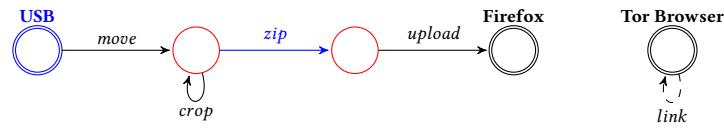
- (2) *The P2P Abuser*: Plaso and Autopsy exposed no relevant traces for this scenario. In contrast, AXIOM and AutoTrack exposed the upload of CSAM via eMule. Additionally, AutoTrack detected that the USB device was connected at the time the CSAM was created on the local file system. This means that AutoTrack detected the file outbound and suggests the USB device as possible source for the file inbound. The evaluation for AXIOM and AutoTrack is:



- (3) *The P2P Link User*: AXIOM, Plaso and Autopsy exposed no relevant traces for this scenario. AutoTrack is able to trace the CSAM back to its inbound channel and additionally detects the link file used, hence, offers additional information about the CSAM community for further investigations. The evaluation of AutoTrack is:

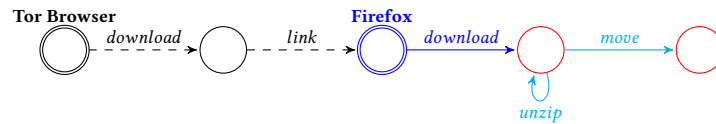


- (4) *The Dark Web User*: AXIOM, Plaso and Autopsy exposed no relevant traces for this scenario. AutoTrack exposed the execution of the Tor Browser in temporal relation to the creation of the CSAM files on the local filesystem and, therefore, suggest the correct inbound channel.
- (5) *The Dark Web Abuser*: AXIOM, Plaso and Autopsy showed no relevant traces for this scenario. AutoTrack detected that the USB device was connected at the time the CSAM was created on the local file system. The evaluation of AutoTrack is:

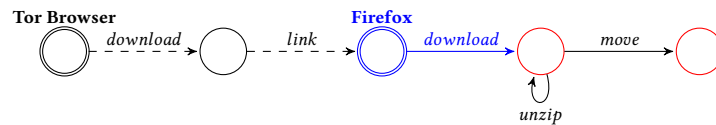


(a) AutoTrack

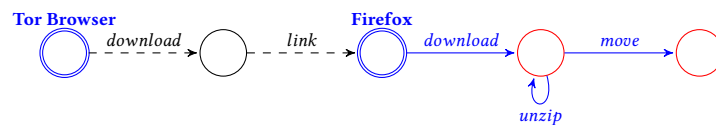
- (6) *The Dark Web Filesharing User*: All forensic applications exposed the download of the CSAM ZIP archive via Firefox while AutoTrack and AXIOM were also able to expose a trace between the ZIP archive and the unzipped/moved CSAM. However, with AXIOM an investigator must extract that trace manually which is only based on a partial file path match. The evaluation per application is:



(a) AXIOM

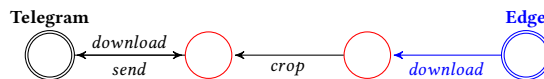


(b) Plaso & Autopsy



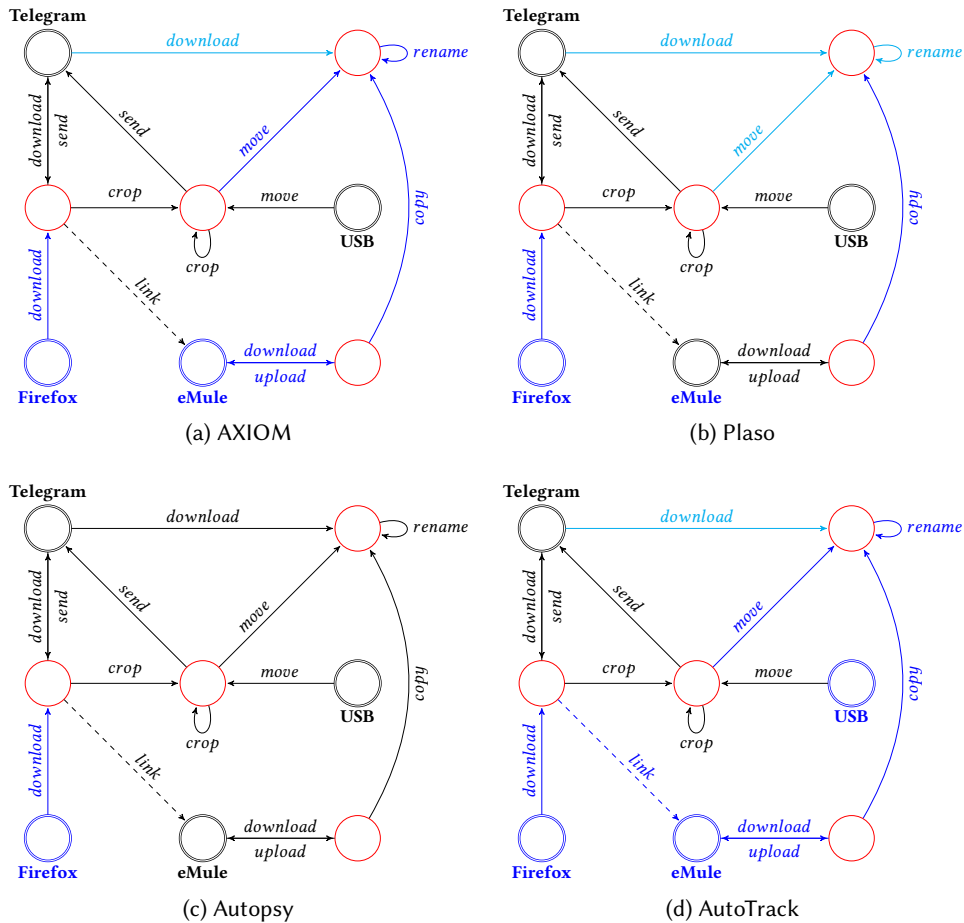
(c) AutoTrack

- (7) *The WhatsApp*: AXIOM, Plaso, Autopsy and AutoTrack showed no relevant traces for this scenario.
- (8) *The Telegram*: AXIOM, Plaso, Autopsy and AutoTrack exposed no traces to Telegram but the additional CSAM download via Edge has been exposed by every forensic applications. The evaluation of AutoTrack is:



(a) AutoTrack

- (9) *The Ever More*: Every forensic application was able to expose the download of incriminating files with Firefox. However, Autopsy does not expose any further trace. In contrast, AXIOM and AutoTrack additionally exposed automatically the inbound and outbound via eMule. The inbound via Telegram can be exposed based on file metadata which must be extracted manually. However, only AutoTrack suggested the USB device as possible source for the file inbound and shows the highest degree of automation. The evaluation per application is:



### 5.3 Overall assessment

In Section 5 we showed the results for the realistic scenarios which are more diverse than the results for the simple scenarios. Therefore, we summarize our findings in Table 10. In every realistic scenario, AutoTrack outperformed the other applications by simply combining and tracking available metadata. However, across all evaluated applications, in the simple and realistic scenarios alike, the only channel of dissemination that is reliably exposed is eMule due to artefacts that are explicit and well-known. On the other hand, the desktop applications of Telegram and WhatsApp may save data of file outbounds but are a blind-spot for every forensic application as their artefacts are not parsed. This is surprising, as AXIOM can parse Telegram’s database and



shows information from it in its *Media Explorer*, but does not use this information for the *Connections* feature. Furthermore, we were surprised that AXIOM apparently used only the file names for its *Connections* feature which is limiting and possibly misleading. Anyways, no application was able to track the processing of a picture, i.e. cropping, which indicates another blind spot.

	P2P User	P2P Abuser	P2P Link User	Dark Web User	Dark Web Abuser	Dark Web Filesharing User	WhatsApp	Telegramer	Ever More
<b>Superior</b>	AutoTrack	AutoTrack	AutoTrack	AutoTrack	AutoTrack	AutoTrack			AutoTrack
<b>Inbound</b>	yes	suggested	yes	suggested	suggested	yes	no	Telegram: no (Internet: yes)	eMule: yes Telegram: partial USB: suggested
<b>Outbound</b>	yes	yes	yes	-	no	-	no	no	eMule: yes Telegram: no
<b>Additional information</b>	-	-	yes	-	no	no	no	no	eMule: yes

Table 10. Overall assessment, including the superior application per scenario and whether inbound and outbound channels were detected.

## 6 LIMITATIONS

Despite the promising results, AutoTrack is only a proof of concept and very limited due to the fact that only eight parsers are implemented to address the given use case. While the file tracking approach is generally applicable to files on a system, additional parsers must be embedded to specifically address other use cases, such as confidentiality violations, or other operating systems. Even for our use case and the Windows system, the artifacts are not exhausted, e.g. the complete Registry is currently not used by AutoTrack. Furthermore, AutoTrack only aims to generate hypotheses efficiently, while the verification of the hypothesis is still the responsibility of the investigator. Additionally, AutoTrack is partially dependent on congruent timestamps (e.g., for connected USB devices, execution of the Tor Browser).

## 7 CONCLUSION AND FUTURE WORK

Starting from the extremely high and still increasing numbers of CSAM cases we searched the questions (i) where did a file originate from and (ii) has it been disseminated by the suspect for a post-mortem Windows system. Based on the actual technical behaviour of CSAM offenders we modeled scenarios that replicate this system usage. We then systematically selected three applications that address our investigative questions and showed that there is a gap between the investigators' needs, i.e. the automatic extraction of a *file-trace*, and the capabilities of common forensic applications. Consequently, we implemented an Autopsy Report Module named *AutoTrack* that provides an artefact correlation through file tracking and aims to fill this gap. In the ensuing evaluation, we show that the potential for automated exposure of possible origins and dissemination of files is far from being exhausted and that the exposure of origins is more feasible than the dissemination of a file. Although *AutoTrack* contains only eight parsers, we were able to achieve results superior to the other selected applications which is particularly interesting since our prototype is the only application that does not look at the whole system. We are confident that a backwards search, starting from a file with all known metadata to expose the dedicated *file-trace*, will be even more powerful when integrated into an application like AXIOM, which has a large pool of parsers and an ontology for reasoning at hand. While we concentrated on typical system usage of CSAM offenders, the proposed approach can also be used to track files for other reasons, such as violation of confidentiality.

## REFERENCES

- [1] Tyler Balon, Krikor Herlopian, Ibrahim Baggili, and Cinthya Grajeda-Mendez. Forensic artifact finder (forensicaf): An approach and tool for leveraging crowd-sourced curated forensic artifacts. In *The 16th International Conference on Availability, Reliability and Security*,

- ARES 2021, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450390514. doi: 10.1145/3465481.3470051. URL <https://doi.org/10.1145/3465481.3470051>.
- [2] Bundeskriminalamt. Vorstellung der Zahlen kindlicher Gewaltopfer -- Auswertung der Polizeilichen Kriminalstatistik 2021. <https://www.tagesschau.de/inland/bka-kriminalstatistik-101.pdf>, 2021.
- [3] Eoghan Casey, Monique Ferraro, and Lam Nguyen. Investigation delayed is justice denied: proposals for expediting forensic examinations of digital evidence. *Journal of forensic sciences*, 54(6):1353–1364, 2009.
- [4] Eoghan Casey, Sean Barnum, Ryan Griffith, Jonathan Snyder, Harm van Beek, and Alex Nelson. Advancing coordinated cyber-investigations and tool interoperability using a community developed specification language. *Digital investigation*, 22:14–45, 2017.
- [5] Mark Debinski, Frank Breitingner, and Parvathy Mohan. Timeline2gui: A log2timeline csv parser and training scenarios. *Digital Investigation*, 28:34–43, 2019.
- [6] Xiaoyu Du, Quan Le, and Mark Scanlon. Automated artefact relevancy determination from artefact metadata and associated timeline events. In *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pages 1–8. IEEE, 2020.
- [7] Cinthya Grajeda, Laura Sanchez, Ibrahim Baggili, Devon Clark, and Frank Breitingner. Experience constructing the artifact genome project (agg): Managing the domain’s knowledge one artifact at a time. *Digital Investigation*, 26:S47–S58, 2018. ISSN 1742-2876. doi: <https://doi.org/10.1016/j.diin.2018.04.021>. URL <https://www.sciencedirect.com/science/article/pii/S1742287618302007>.
- [8] Kristinn Guðjónsson. Mastering the super timeline with log2timeline. *SANS Institute*, 2010.
- [9] Christopher Hargreaves and Jonathan Patterson. An automated timeline reconstruction approach for digital forensic investigations. *Digital Investigation*, 9:S69–S79, 2012.
- [10] Hans Henseler and Jessica Hyde. Technology assisted analysis of timeline and connections in digital forensic investigations. In *LegalAIIA@ ICAIL*, pages 32–37, 2019.
- [11] Ian Horrocks. What are ontologies good for? *Evolution of semantic systems*, pages 175–188, 2013.
- [12] Hee-Eun Lee, Tatiana Ermakova, Vasilis Ververis, and Benjamin Fabian. Detecting child sexual abuse material: A comprehensive survey. *Forensic Science International: Digital Investigation*, 34:301022, 2020. ISSN 2666-2817. doi: <https://doi.org/10.1016/j.fsidi.2020.301022>. URL <https://www.sciencedirect.com/science/article/pii/S2666281720301554>.
- [13] Christopher Lees. Determining removal of forensic artefacts using the usn change journal. *Digital Investigation*, 10(4):300–310, 2013.
- [14] Apurva Nalawade, Smita Bharne, and Vanita Mane. Forensic analysis and evidence collection for web browser activity. In *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*, pages 518–522. IEEE, 2016.
- [15] National Center for Missing & Exploited Children (NCMEC). 2021 CyberTipline Reports by Country, 2021. <https://www.missingkids.org/gethelpnow/cybertipline/cybertiplinedata>, last accessed 2023-02-06.
- [16] Myeongsuk Pak and Sanghoon Kim. A review of deep learning in image recognition. In *2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT)*, pages 1–3, 2017. doi: 10.1109/CAIPT.2017.8320684.
- [17] Darren Quick and Kim-Kwang Raymond Choo. Impacts of increasing volume of digital forensic data: A survey and future research challenges. *Digital Investigation*, 11(4):273–294, 2014.
- [18] Scientific Working Group on Digital Evidence (SWGDE). SWGDE Digital & Multimedia Evidence Glossary 3.0, 2016.
- [19] StatCounter. Marktanteile der führenden betriebssysteme weltweit im januar 2023, January 2023. URL <https://de.statista.com/statistik/daten/studie/828610/umfrage/marktanteile-der-fuehrenden-betriebssystemversionen-weltweit/>.
- [20] Chad Steel, Emily Newman, Suzanne O’Rourke, and Ethel Quayle. Technical behaviours of child sexual exploitation material offenders. *Journal of Digital Forensics, Security and Law*, 17(1):2, 2022.
- [21] Hudan Studiawan, Ferdous Sohel, and Christian Payne. Sentiment analysis in a forensic timeline with deep learning. *IEEE Access*, 8: 60664–60675, 2020.