Automating the Flow of Data Between Digital Forensic Tools Using Apache NiFi

Xiaoyu Du(✉), Francis N. Nwebonyi, and Pavel Gladyshev

Digital Forensics Investigation Research Laboratory, School of Computer Science, University College Dublin, Dublin, Ireland
{xiaoyu.du,francis.nwebonyi,pavel.gladyshev}@ucd.ie
http://dfire.ucd.ie/

Abstract. In digital forensics, sources of digital evidence range from computer disk drives, memories, mobile phones, network dumps, and all kinds of IoT devices, etc. Therefore, different tools are required for digital evidence collection and analysis from various sources. Even though each tool works automatically, data from one tool to another often need to be prepared manually. This paper introduces a NiFi-based solution that enables automatically moving data between digital forensic tools, reducing manual work in practice. A DataFlow designed in NiFi can monitor and fetch the input data, pre-processing the data and run digital forensic tools for data analytics. Besides, NiFi can also be used for remote evidence acquisition and data sharing between law enforcement agencies (LEAs). This paper also presents a couple of use cases of using NiFi for digital evidence processing: they are 1) file carving, 2) NSRL (National Software Reference Library) hash lookup, 3) categorising files by MIME type, and 4) IoT logs parsing.

Keywords: Digital Forensics · Automated Digital Evidence Processing · Apache NiFi

1 Introduction

The focus of digital forensics gradually has changed from recovering data from storage devices to acquiring and analysing data from mobile devices, tablets, and smart devices. Challenges in digital forensics are widely discussed by researchers in digital forensic field [6,18]. With the prevalence of technology in modern life, the ever-increasing volume of data leads to data collection and analysis taking a longer time. On the other hand, the diverse data formats require various tools for processing, and data from novel devices in new format may not have tools that fit.

With the diverse sources of digital evidence from novel applications and IoT devices raise challenges in digital forensics. For each of types of device like the wearable devices and smart homes, researchers in the field have been working

S. Goel et al. (Eds.): ICDF2C 2022, LNICST 508, pp. 435–452, 2023.
https://doi.org/10.1007/978-3-031-36574-4_26

on analysing what user data are there and how are they preserved [1]. Many forensic tools have been developed to process the various forensic targets. From digital forensic practitioners' perspective, there is a trend that the number of digital forensic tools needed is continually increasing.

Another challenge is digital forensic tools become outdated quickly because, in the age of rapid change in technology, prevalent devices and applications changed over time. For application forensics, different tools are needed as popular applications changes over time. Skype was a popular tool for online calls, research on Skype forensic in 2012 [4]. Since the global pandemic of COVID-19, the video conferencing application Zoom has experienced a surge in its user. Zoom application forensics has become significant as well in 2021 [19]. There are many other tools, TicTok [7], Discord [22], and many more to come in the future.

Currently, in an investigation, to collection and analysis of evidence with multiple tools is very common. Practitioners have to wait for a tool to complete the data collection, and extraction, and then manually prepare the data for analysis. Moreover, as mentioned, due to the ever-increasing volume of data, the processing time takes longer. Even though there are integrated commercial tools, however, in most situations, problems in an investigation can not be solved with a single tool.

This research work explores using Apache NiFi to manage data movement between digital forensic tools. Apache NiFi[1] is a tool for data movement between systems. The NiFi-based solution for digital forensics can automated fetch data from filesystem and database, and run digital forensic tools for data extraction, hash searching and so on. in addition, leveraging NiFi in digital forensics allows to include more analysis modules from excellent open-source projects, like Text Summaries and Sentiment Analysis, which are not forensic tools can also be useful in an investigation. Thereby, only is the cost of developing specialised digital forensics tools saved, but new techniques can be used for digital forensics more quickly.

2 Backgrounds

2.1 Digital Forensic Challenges

Exponential growth in the volume of digital evidence is faced by LEAs around the world. Digital Forensics Laboratories (DFLs) started to face backlogs of 6 months to 1 year as early as 2009 [5], backlogs continue to grow [13], it commonly reached one to two years and even four years in extreme cases in 2016 [18]. It was reported in 2022, police have digital backlogs, 45 police forces across the UK found that a total of 21,022 devices were waiting for examination.

With the constant emergence of new and diverse devices with varied OSs, diverse types of devices and formats of data, there are much more sources of digital evidence. The nature of investigations has changed from simply extracting data, and recovering deleted files to mobile devices forensics, and real-time

[1] https://nifi.apache.org/.

network forensics. Digital evidence could be from various devices, and different tools are required for: disk forensics, cloud forensics, memory forensics, and mobile forensics. However, most new innovations take time to develop and can add significant costs to the investigator's toolkit. There is a trend that the number and variety of analytic tools continue to grow [13].

Researchers in the field of digital forensics have proposed solutions aimed at a faster processing, such as data deduplication [9,28], data reduction [25], triage [10,12,20], and other advanced methods. However, there is a gap between research and practice. Additional research is required to address the real-world relevance of the proposed methods and implement practical infrastructural enhancements [21]. The implementation integrates automated collection and examination, heterogeneous evidence process, data visualisation, multi-device evidence and timeline resolution can significantly improve the efficiency of the digital forensic process.

2.2 Digital Forensics: Tools and Techniques

There are many tools/toolkits available for different tasks in investigations [15], such as Autopsy, Encase, Volatility, etc. There are also various techniques such as blockchain [27], machine learning and deep learning [8], which have been constantly employed by researchers in the field to improve the digital forensics process.

Case investigations are various, the evidence collection can be different accordingly. For example, there are different ways of acquisition. Full disk image acquisition preserves all data from a storage device, files from the file system, slack space, and unallocated space which contains deleted data. The forensically sound disk image collection usually takes a very long time for the collection of large data volume. An approach aimed at a faster acquisition has been proposed, named selective acquisition. Selective acquisition only collects data (types of files) which are possibly more relevant to the investigation [26]. *sifting collectors* is an approach proposed in 2015 [11], which images only those regions of a disk with expected forensic value.

Investigators choose the method according to the actual situation. Forensically soundness of the disk image is important when the analysis result is used to prove guilt or innocence in court, the reproducibility of the analysis result is important in this case. Selective imaging should be applied when the case is time-critical, about the life and death of a victim.

In digital forensic examination and analysis, file carving tools are usually applied to the discovery and reconstruction of files in case the system metadata is corrupted, missing, or otherwise unreliable [2]. MIME (multipurpose internet mail extensions) information is usually used in digital forensics for recognising real file types. Because the file extension is for the proper identification of programs that can open and display the correct information, it is easy to be change manually by users. MIME represents file types for messages and files sent over the Internet and is detected using a magic number inside the file [23]. In digital

forensics, using the MIME type instead of the file extension to recognise the file type is more reliable.

Hashing is another technique usually used in digital forensics. The hash value is a digital fingerprint of a file, which is used to recognise a file in digital forensics for detecting known illegal files or filtering known benign files. There are forensic corpus data reduction methods proposed by Joseph et al. [16] in 2019, this research proposes an efficient methodology for forensic investigations, that eliminated 49.4 million uninteresting files. NSRL hash databases are freely available over the internet for faster analysis by eliminating tedious files. The NSRL lookup tool developed by Rob Hansen of RedJack Security LLC allows users to extend the hash library in use.

AI-based techniques provide another way for automated analysis in digital forensics [14]. For user files, pictures are important sources of digital evidence. There are a number of tools that can assist in image analysis, such as object detection [24], face detection, age estimation [3]. For text data, different tools could be used determined by the sources (documents, instant messages, and emails) and objectives (clustering, classification, translation).

2.3 Use of Apache NiFi in Research

Apache NiFi is not a digital forensics tool, actually, it is for data management and movement between software systems. It was used and evaluated by some researchers in their work. In a paper by Kim et al. [17], Apache NiFi was applied to deal with different types of spatial information (Geo-IoT sensor, SNS information, unstructured spatial information, etc.). This research reports that NiFi has the advantages: 1) easy to develop DataFlow, 2) real-time data transmission is provided, 3) resource sharing through powerful resource and authority management, 4) data history management, 5) communication between several NiFi systems is available.

2.4 Summary

Digital forensic investigation is complicated as the diverse types of cases, and the investigation focuses can vary. One single tool usually can only solve part of the problem in an investigation. Multiple tools are needed for data collection, extraction and analysis from various devices. It is important for digital forensic tools to accommodate the continued appearance of new devices and formats of data. Besides, removing the research-practice gap is very important for combating the digital forensic backlogs.

3 Methodology

This section presents the what and how NiFi assists to solve challenges and problems in digital forensics.

3.1 Overview

NiFi can be used for processing digital evidence data in many different ways. Firstly, NiFi can be used for data transmission in the digital investigation between investigation phases, different digital forensic tools, as well as different servers. NiFi allows for automatically detecting and fetching data from a system through the configured NiFi processors. Either the data is in the local machine or a remote server, there are corresponding processors for it.

Figure 1 shows that running NiFi in investigators' computers to automatically access services from servers and getting the analysis result back from the server. All these operations can be defined to perform automatically through NiFi: 1) a server preserves data collected, investigators share the collected data in the data storage server; 2) a server runs machine learning models to assist analysis, and investigators send data for analysis to the server and get the analysis result back to local machine; 3) save illegal files found in an investigation into a known file database for sharing between LEAs.

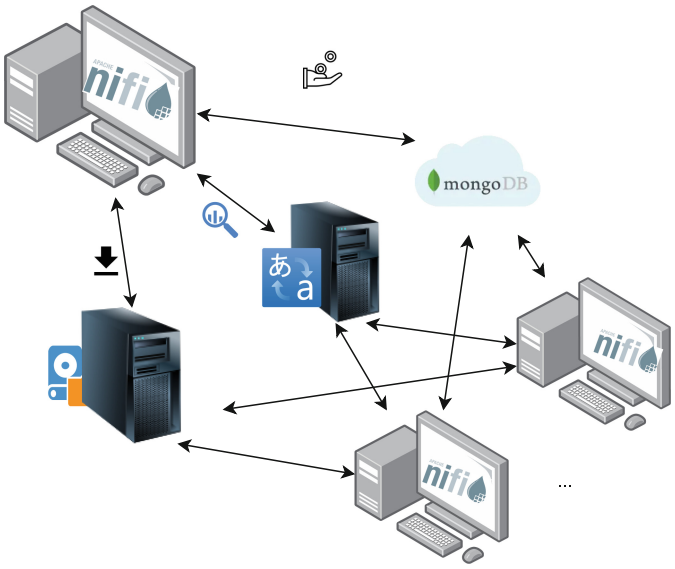

Fig. 1. Use NiFi in Digital Forensics for Data Transmission: NiFi allows to get data/services from different servers and to preserve the result of data processing and analysis of cloud databases, such as MongoDB.

NiFi can also assist to move data between tools in different investigation phases. Figure 2 shows evidence processing in each phase of digital forensics, i.e., collection, extraction, and analysis. NiFi is used to move data from one tool to the next and finally get the processing result to investigators. Even though there are so many tools available, it is not all of them are required for every case

investigation. A NiFi DataFlow can be designed to only include the tools which are required by the investigation.

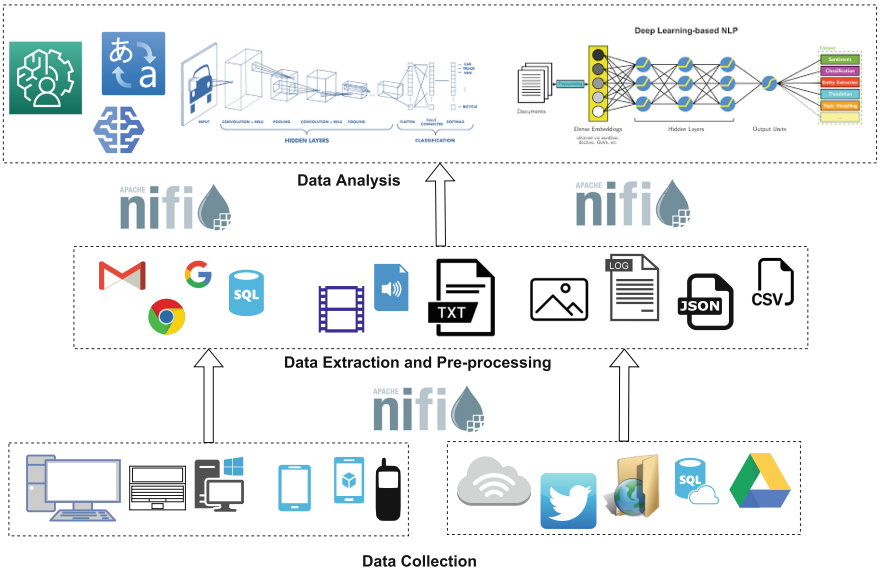

Fig. 2. Use Apache NiFi to automate flow of data between digital forensic tools - 1) run data collection tools within NiFi processors, 2) send the collected data to data extraction tools, 3) send the extracted data to analysis tools.

There are also NiFi processors which can be configured to execute command-line tools and scripts on data for processing and analysing. Thereby, it is convenient to employ open-source tools, and machine learning models (shown in the Fig. 2) for forensic analysis in the NiFi-based digital forensic process.

3.2 Design a NiFi DataFlow for Digital Evidence Data Processing

Apache NiFi is a DataFlow system based on the concepts of flow-based programming. It provides a web-based user interface for design, control, feedback, and monitoring of DataFlows.

A NiFi DataFlow can consist of one or more components. NiFi processor is the most used component in NiFi, various processors are designed to fetch, process and preserve data. When a file is fetched by a processor, the file data in NiFi is in a form called FlowFile. The processors are connected together to process data step by step.

– **Processor**: There are a number of NiFi processors that can fetch data from different sources, and put data to a specified location in a local or remote computer. It is also allowed users to develop a customised NiFi processor. The customised NiFi processor can also be shared with other users.

- **FlowFile**: The FlowFile represents a single piece of data in NiFi. A FlowFile is made up of two components: FlowFile Attributes and FlowFile Content. All kinds of files can be ingested by NiFi processor, disk images, pictures, audios, videos, various log files from systems and applications, as well as file fragments. A FlowFile is created when a file is ingested to the DataFlow. These data can be monitored by a NiFi processor, either they are saved in a folder of the local machine or a remote server, as well as in a cloud data lake.
- **DataFlow**: A NiFi DataFlow consists components for getting data, processing data and preserving data. The process of designing the DataFlow determines how the data will be processed. A DataFlow can be preserved as an XML file for sharing and reusing.

When start the processors in the DataFlow, all operations, fetching data from the source location, transition data, processing data by each digital forensic tool, load data to the destination location, automatically happen.

3.3 NiFi Processors for Auto-Detecting and Fetching Data

There are NiFi processors which can fetch data from different sources locally and remotely. Accordingly, there are processors for load data to destination locations in a local or remote system.

- *GetFile*: Creates FlowFiles from files in a directory.
- *FetchSFTP*: Fetches the content of a file from a remote SFTP server and overwrites the contents of an incoming FlowFile with the content of the remote file.
- *GetMongo*: Creates FlowFiles from documents in MongoDB loaded by a user-specified query.

3.4 NiFi Processors for Auto-Examine and Analysis of Data

There are some other NiFi processors that can be used for data processing for structured data such as JSON, XML, CSV, and records from a SQL/NoSQL database.

- *SplitRecord*: Splits up an input FlowFile that is in a record-oriented data format into multiple smaller FlowFiles
- *QueryRecord*: Evaluates one or more SQL queries against the contents of a FlowFile. The result of the SQL query then becomes the content of the output FlowFile. This can be used, for example, for field-specific filtering, transformation, and row-level filtering. Columns can be renamed, simple calculations and aggregations performed, etc.

It is also possible to run digital forensic tools in a NiFi processor. The tool is configured in a NiFi processor and it starts the process of data automatically when the FlowFile goes down to the processor.

ExecuteStreamCommand is the processor that can be configured to executes an external command. It works on the data of a FlowFile and creates a new FlowFile with the results of the command. For example, the command-line tool *foremost* can be run in this NiFi processor for carving files from a disk image. The output folder can be specified in the processor configuration. It is also possible to run a python script in the NiFi processor.

3.5 Benefits of Using NiFi in Digital Forensics

Even though using script language such as Python could be developed to do the same work as NiFi does - to automatically start to run a digital forensic tool and process data as needed, NiFi provides a web-based user interface and is very easy to use; users only need to drag processors to the canvas and configure the properties.

Figure 3 shows an example of NiFi DataFlow, it monitors a folder in a local machine and fetches data when files are put in this folder the processor fetches it to the DatabFlow, and then data is processed by each processor; the processing includes file carving from a disk image, unzip a file in zip format, know hash library look up, categorising files as their MIME type.

Benefits of using NiFi in digital forensics listed as follows:

- **Easy to use** - After designed and started the DataFlow, users simply need to put the data into the source folder and will get the result in the destination folder.
- **Automate the flow of data between tools** - It allows to automatically send the processing result of one tool to another tool for further analysis, which saves time for manual processing.
- **Remote acquisition and data sharing** - NiFi processors can be configured for dynamic get and transmit data either a remote server or a cloud database. Automate the data movement can be achieved and sharing data between LEAs.
- **Verify the result of a tool** - It can be designed to send the same data to different tools for the same analysis purpose, and compare the analysis results to verify the reliability of the result.

3.6 Summary

This section outlined the key terminologies in the NiFi framework and presented how NiFi can be applied to fit into the digital forensic process model. NiFi can fetch and load data not only from the specified file folder on the local machine, but as well from a remote machine, or cloud database. It is also possible to use NiFi for data processing, splitting, or querying records for structure data.

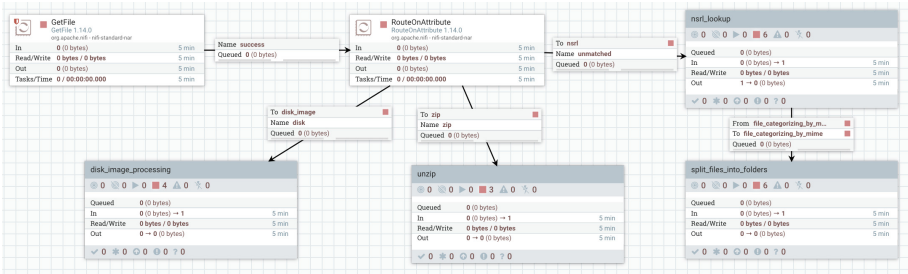

Fig. 3. An implemented NiFi DataFlow for digital forensic data ingestion and preprocessing: The *GetFile* processor monitors the source folder, and fetches the data put in it; the *RouteonAttribute* processor routes data to different processor groups based on the file type; there are 4 processor groups for unzipping files, carving files from disk images, (the extracted and craved file data are put back to the source folder), single files are compared to NSRL hash library and then categorised as their mime type put to different folders for further analysis.

4 Use Cases

In this section, use cases are presented, showing how NiFi processors are used to performing digital forensic tasks. As it is shown in Fig. 3, firstly, get data from a folder and check the file type. Files then are processed by different process groups based on their types. In the following subsections, process groups for file carving, file system data extraction, NSRL hash lookup and categorising files by MIME types will be introduced.

4.1 File Carving

As it is shown in Fig. 4, a file caving process group consists of 3 NiFi processors. Firstly, the *FetchFile* processor fetches data to the DataFlow. The processor *ReplaceText* replaces the file contents with the filename, because the command-line tool *foremost* is running inside the processor *ExecuteStreamCommand* needs the full path of the file as the input. And then, the *ExecuteStreamCommand* processor executes an external command on the contents of a FlowFile. The file carving tool *Foremost* can be configured to be executed in the processor, so as when a disk image file is in the DataFlow, data can be carved and then put into a specified folder for further processing.

Another argument required by *foremost* is an output folder. In the configuration, to set the output as the source folder, as a result, the carved files will be fetched again and recognised as single files, and then flow to other branches for other analysis operations.

4.2 Data Extraction from Disk Images

Another way that extracting data from disk images based on partition and file system information is also implemented. It works through running a Python

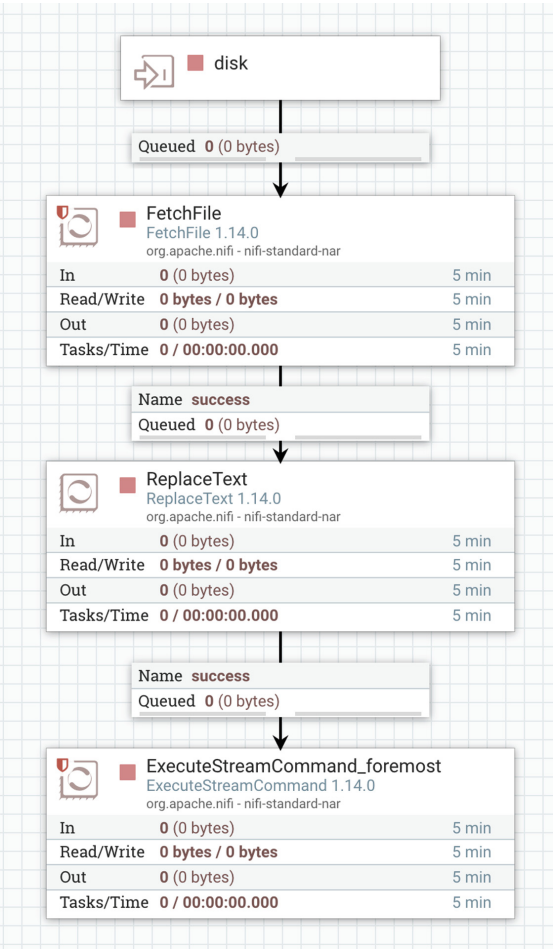

Fig. 4. File Carving from Disk images

script in the processor *ExecuteStreamCommand*. The processor can extract files and data from disk images through configure it to run a Python script.

Figure 5 shows the configuration of the processor for running the script. The Value "pytsk_recursive_extraction.py" for the Property "Command Arguments" specifies the name script file. The python script accesses partitions of an image and extracts file data and metadata from file systems.

The metadata extracted from the disk image is preserved as a CSV file in the result folder for further analysis. The files extracted from the disk image are saved into the source folder, same as the file carving process, the extracted files then are fetched by the DataFlow and processed by other analysis process groups.

Required field

Property	Value
Command Arguments Strategy	Command Arguments Property
Command Arguments	pytsk_recursive_extraction.py
Command Path	**python3**
Ignore STDIN	false
Working Directory	/app
Argument Delimiter	;
Output Destination Attribute	No value set
Max Attribute Length	256

Fig. 5. Configure the *ExecuteStreamCommand* Processor: this configuration sets the processor to run the python script for data extraction from disk image

4.3 NSRL Hash Lookup

NSRL lookup tool detects known files. It allows automated filtering out the known benign files and potentially can be applied for eliminating redundant images and videos found across multiple devices. Hash lookup reduces the amount of data an investigator must examine manually.

A process group is created for detecting files by their hash values. NSRL hash sets cover the most common OS installation and application packages and can aid the investigation of data from computer systems. To implement the NSRL filter function, two tools namely *nsrlsvr* and *nsrllookup* are used in this NiFi DataFlow.

As is shown in Fig. 6, the file data is hashed by the processor *HashContent*. Afterwards, the md5 of the file is generated and added to the FlowFile metadata. Other hash algorithms are also available through configuring the property of *HashContent* processor.

The *HashContent* Processor calculates a hash value for the Content of a FlowFile. And then, run the tool *nsrllookup*[2] in the *ExecuteStreamCommand* processor to check if a file existing in the NSRL library. If the file exists, the hash value is returned and if not, no return value.

The file data then is flow to different processors. Files are filtered out directly if they are known by the NSRL library. For files which are not found in the NSRL library, they are moved to the next process group for categorising based on the MEMI type.

4.4 Categorising Files by MIME Type

The *IdentifyMimeType* processor is for identifying the MIME type of a file. As it is shown in Fig. 7, files' MIME are firstly recognised by the IdentifyMimeType

[2] http://rjhansen.github.io/nsrllookup/.

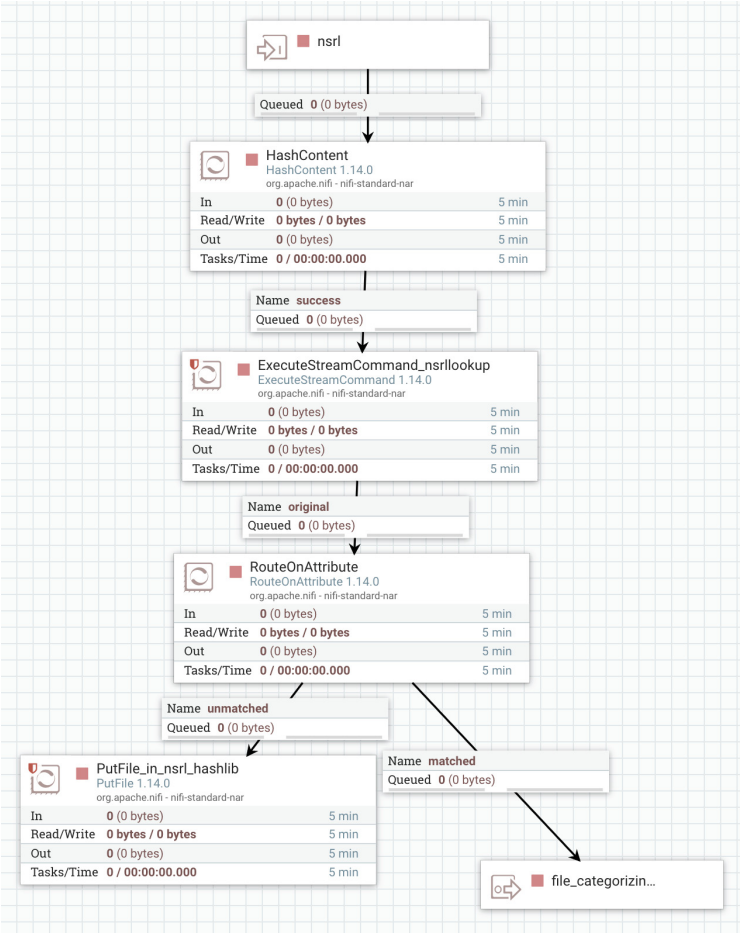

Fig. 6. Example DataFlow: NSRL Lookup

processor and then route to different branches by the RouteonAttribute processor. Finally, as different file types, files are saved to different folders by PutFile processors.

4.5 IoT Sources Ingestion

In the IoT (Internet of Things) domain, interoperability is a problem because different solutions often come with different vendors and file formats, among other differences. To make the advanced processing of IoT data more efficient, it is important to have them in a unified format. Apache NiFi can be used here to unify different file formats from different IoT sources into a common format, to help with further processing or investigation.

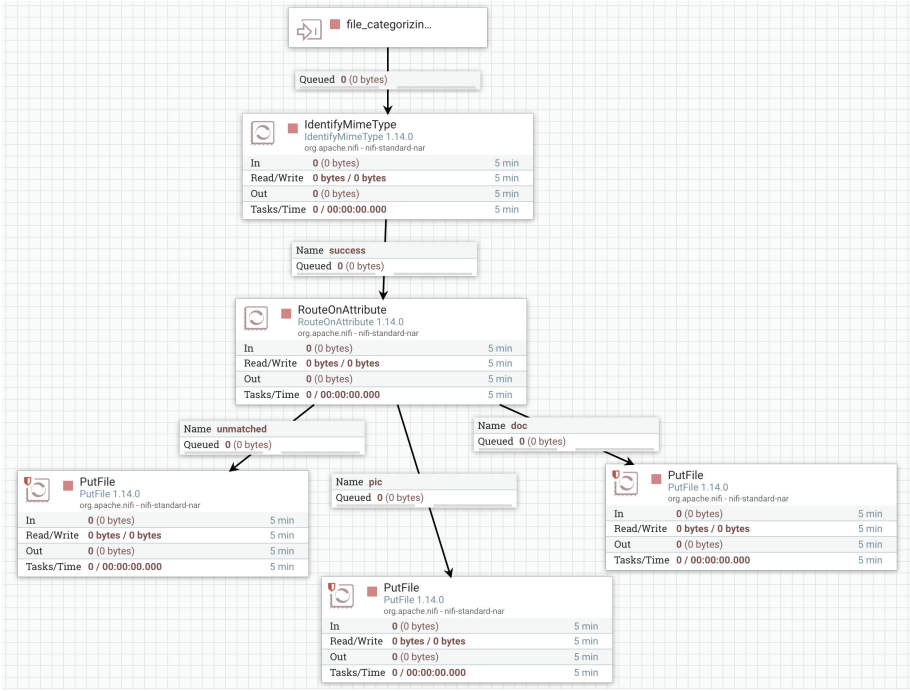

Fig. 7. Example DataFlow: Categorising File by MIME type

To demonstrate this, a DataFlow was built (see Fig. 8) to take in a number of file formats, including Avro, Parquet, and CSV, and convert them to JSON. Several NiFi processors, many of which have been explained, were put together and configured to make this possible. The *ConvertRecord* processor is perhaps worth a mention here. It has properties such as Record Reader and Record Writer which can be configured for reading incoming data and writing out records respectively. Each of these properties can be leveraged to allow the processor to support different file formats. It has some out-of-the-box properties, as well as scripted instances for reading, parsing, and generating records from (and to) FlowFiles, through some of its properties like ScriptedReader and ScriptedRecordSetWriter.

4.6 Summary

This section presented the development of NiFi DataFlow in use for digital forensics tasks, i.e., file carving, NSRL lookup, and MIME categorising. The output data of this DataFlow (files in the destination folders) could be sent to machine learning or deep learning models for further analysis. These output data then can be preserved in a database for further in-depth analysis.

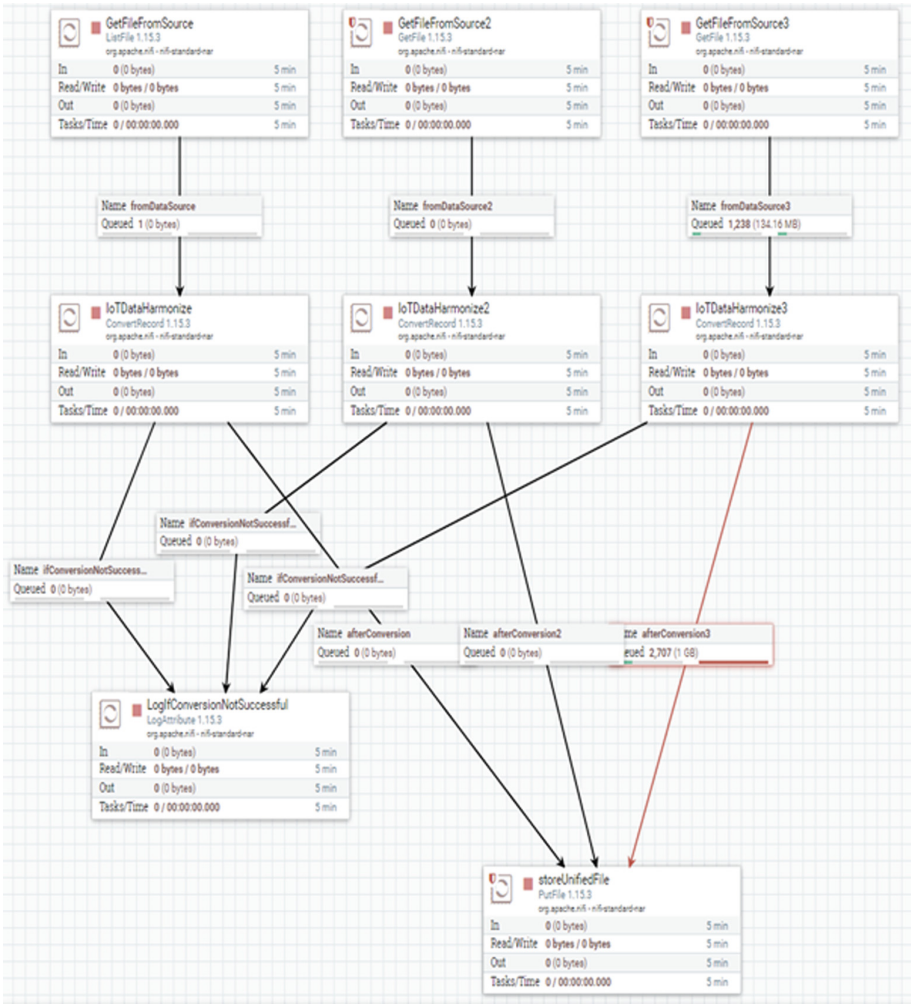

Fig. 8. Harmonizing Several Data Formats (CSV, Avro, and Parquet) into JSON.

5 Experiments and Results

5.1 Test Data

The most common file types in digital forensics are prepared for Experiment to test and verify functions of the NiFi DataFlow. The categories of files for testing are:

- **Raw disk images**, which are generated from VirtualBox, with windows operating system installed.
- **Zip files**, which contain various numbers of files and folders in them;

– **Single Files**, which are documents, Pictures, etc.
– **Data in different formats**, which are JSON, CSV, etc.

5.2 Experiment Environment Setup

Experiments presented in this paper are run in a Docker container, the docker image is from docker hub[3]. Forensic tools (foremost, pytsk, nsrlsvr, nsrllookup) are installed to the docker image.

5.3 Experiments and Results

Firstly, the user needs to start the processors in the web-based user interface. Secondly, the user put test data in the source folder. NiFi fetches the data to the DataFlow and processes it based on the file types of data.

1. **When the input is a zip file:**
   (a) The file was detected in the source folder.
   (b) The file was pulled into the DataFlow.
   (c) The file was recognised as a zip file.
   (d) The file was Route to the unzip process group.
   (e) The file was unzipped by the UnpackContent processor.
   (f) The unpacked files are in the DataFlow.
   (g) The unpacked files (single files) are put back to the source folder.
2. **When the input is a raw disk image file:**
   (a) The file was detected in the source folder.
   (b) The file was pulled into the DataFlow.
   (c) The file was recognised as a raw disk image file.
   (d) The file was Route to the disk image data extraction processor.
   (e) Files on the disk images were extracted by pytsk.
   (f) The extracted files are saved in the source folder.
3. **When the input is a single file** (document, picture, etc.):
   (a) The file was detected in the source folder.
   (b) The file was pulled into the DataFlow.
   (c) The file was a single file and then was Route to NSRL hash Detection process group.
      i. If the file's hash has existed in the hash sets, the file was benign and does not require further processing.
      ii. If the file's hash has not existed in the hash sets, the file was unknown.
   (d) Unknown files are flowing to the next process group - mime type file filter.
   (e) Files are filtered by mime type and put into different destination folders.

After these processing steps, the content of the FlowFiles finally is written to the local file system in the location specified by the *PutFile* processor. If the destination/output folder does not exist, NiFi can dynamically create a folder in the location as specified.

[3] https://hub.docker.com/r/apache/nifi.

6 Conclusion

This paper proposed a NiFi-based solution that contributes to the automation of the digital forensics process and reduces manual work in practice. It automatically fetched evidence data, start a digital forensic tool to process (examine or analyse) the data, and load the data to the destination location. Moreover, use cases on processing disk images and documents are presented which preliminary proved the effeteness of the proposed solution. Potentially, NiFi can be applied to process more diverse sources/formats of evidence data.

Another advantage of using NiFi is its browser-based user interface for design, control, feedback, and monitoring enables rapid development and iterative testing. As a result, it is convenient to use it for integrating up-to-date tools and techniques into digital forensic practice.

6.1 Future Work

This research explored using NiFi to run digital forensic tools for digital evidence data processing. Next, further research related to the NiFi-based digital forensics solution will be explored including:

- **Test on larger data sets**: In this paper, the test focuses on verifying the result correctness of the NiFi DataFlow. In the next stage, larger data sets consist of more logs, multimedia files, and disk images to quantify the data processing effectiveness of the approach.
- **NiFi logs analysis**: There are logs generated as the NiFi runs, what data are in the flow and what operations are conducted are recorded by the NiFi logs. To analyse these logs and figure out what are there can be useful for documentation.
- **Multiple sources correlation analysis**: To develop and run multiple NiFi DataFlows for processing data from the same case but different devices and integrate the analysis results from these DataFlows for further in-depth manual analysis.
- **Report generation**: It is also possible to visualise the analysis result and generate a report for non-technical end-users.

Acknowledgment. This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 883596. The content of the publication herein is the sole responsibility of the publishers and it does not necessarily represent the views expressed by the European Commission or its services.

References

1. Alabdulsalam, S., Schaefer, K., Kechadi, T., Le-Khac, N.-A.: Internet of things forensics – challenges and a case study. In: DigitalForensics 2018. IAICT, vol. 532, pp. 35–48. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99277-8_3

2. Ali, R.R., Mohamad, K.M., Jamel, S., Khalid, S.K.A.: A review of digital forensics methods for JPEG file carving. J. Theor. Appl. Inf. Technol. **96**(17), 5841–5856 (2018)
3. Anda, F., Lillis, D., Le-Khac, N.A., Scanlon, M.: Evaluating automated facial age estimation techniques for digital forensics. In: 2018 IEEE Security and Privacy Workshops (SPW), pp. 129–139. IEEE (2018)
4. Azab, A., Watters, P., Layton, R.: Characterising network traffic for skype forensics. In: 2012 Third Cybercrime and Trustworthy Computing Workshop, pp. 19–27. IEEE (2012)
5. Casey, E., Ferraro, M., Nguyen, L.: Investigation delayed is justice denied: proposals for expediting forensic examinations of digital evidence. J. Forensic Sci. **54**(6), 1353–1364 (2009)
6. Caviglione, L., Wendzel, S., Mazurczyk, W.: The future of digital forensics: challenges and the road ahead. IEEE Secur. Priv. **15**(6), 12–17 (2017)
7. Domingues, P., Nogueira, R., Francisco, J.C., Frade, M.: Analyzing TikTok from a digital forensics perspective. J. Wirel. Mob. Netw. Ubiquit. Comput. Dependable Appl. (JoWUA) **12**(3), 87–115 (2021)
8. Du, X., et al.: SoK: exploring the state of the art and the future potential of artificial intelligence in digital forensic investigation. In: Proceedings of the 15th International Conference on Availability, Reliability and Security, pp. 1–10 (2020)
9. Du, X., Ledwith, P., Scanlon, M.: Deduplicated disk image evidence acquisition and forensically-sound reconstruction. In: 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), pp. 1674–1679. IEEE (2018)
10. Gentry, E., Soltys, M.: SEAKER: a mobile digital forensics triage device. Procedia Comput. Sci. **159**, 1652–1661 (2019)
11. Grier, J., Richard, G.G., III.: Rapid forensic imaging of large disks with sifting collectors. Digit. Investig. **14**, S34–S44 (2015)
12. Horsman, G., Laing, C., Vickers, P.: A case-based reasoning method for locating evidence during digital forensic device triage. Decis. Support Syst. **61**, 69–78 (2014)
13. Hosmer, C.: Python Forensics: A Workbench for Inventing and Sharing Digital Forensic Technology. Elsevier, Amsterdam (2014)
14. Jarrett, A., Choo, K.K.R.: The impact of automation and artificial intelligence on digital forensics. Wiley Interdis. Rev. Forensic Sci. **3**(6), e1418 (2021)
15. Javed, A.R., Ahmed, W., Alazab, M., Jalil, Z., Kifayat, K., Gadekallu, T.R.: A comprehensive survey on computer forensics: state-of-the-art, tools, techniques, challenges, and future directions. IEEE Access **10**, 11065-11089 (2022)
16. Joseph, P., Norman, J.: Forensic corpus data reduction techniques for faster analysis by eliminating tedious files. Inf. Sec. J. A Glob. Perspect. **28**(4–5), 136–147 (2019)
17. Kim, S.S., Lee, W.R., Go, J.H.: A study on utilization of spatial information in heterogeneous system based on apache NiFi. In: 2019 International Conference on Information and Communication Technology Convergence (ICTC), pp. 1117–1119. IEEE (2019)
18. Lillis, D., Becker, B., O'Sullivan, T., Scanlon, M.: Current challenges and future research areas for digital forensic investigation. arXiv preprint arXiv:1604.03850 (2016)
19. Mahr, A., Cichon, M., Mateo, S., Grajeda, C., Baggili, I.: Zooming into the pandemic! a forensic analysis of the zoom application. Forensic Sci. Int. Digit. Invest. **36**, 301107 (2021)

20. Mislan, R.P., Casey, E., Kessler, G.C.: The growing need for on-scene triage of mobile devices. Digit. Investig. **6**(3–4), 112–124 (2010)
21. Montasari, R., Hill, R., Parkinson, S., Peltola, P., Hosseinian-Far, A., Daneshkhah, A.: Digital forensics: challenges and opportunities for future studies. Int. J. Organ. Collective Intell. (IJOCI) **10**(2), 37–53 (2020)
22. Motyliński, M., MacDermott, Á., Iqbal, F., Hussain, M., Aleem, S.: Digital forensic acquisition and analysis of discord applications. In: 2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI), pp. 1–7. IEEE (2020)
23. Parveen, A., Khan, Z.H., Ahmad, S.N.: Classification and evaluation of digital forensic tools. Telkomnika **18**(6), 3096–3106 (2020)
24. Qadir, S., Noor, B.: Applications of machine learning in digital forensics. In: 2021 International Conference on Digital Futures and Transformative Technologies (ICoDT2), pp. 1–8. IEEE (2021)
25. Quick, D., Choo, K.-K.R.: Big forensic data reduction: digital forensic images and electronic evidence. Clust. Comput. **19**(2), 723–740 (2016). https://doi.org/10.1007/s10586-016-0553-1
26. Quick, D., Choo, K.K.R.: Big Digital Forensic Data: Volume 1: Data Reduction Framework and Selective Imaging. Springer, Berlin (2018)
27. Ryu, J.H., Sharma, P.K., Jo, J.H., Park, J.H.: A blockchain-based decentralized efficient investigation framework for IoT digital forensics. J. Supercomput. **75**(8), 4372–4387 (2019)
28. Scanlon, M.: Battling the digital forensic backlog through data deduplication. In: 2016 Sixth International Conference on Innovative Computing Technology (INTECH), pp. 10–14. IEEE (2016)