Advanced Automated Disk Investigation Toolkit

Umit Karabiyik, Sudhir Aggarwal

Chapter 20

ADVANCED AUTOMATED DISK INVESTIGATION TOOLKIT

Umit Karabiyik and Sudhir Aggarwal

Abstract Open source software tools designed for disk analysis play a critical role in digital forensic investigations. The tools typically are onerous to use and rely on expertise in investigative techniques and disk structures. Previous research presented the design and initial development of a toolkit that can be used as an automated assistant in forensic investigations. This chapter builds on the previous work and presents an advanced automated disk investigation toolkit (AUDIT) that leverages a dynamic knowledge base and database. AUDIT has new reporting and inference functionality. It facilitates the investigative process by handling core information technology expertise, including the choice and operational sequence of tools and their configurations. The ability of AUDIT to serve as an intelligent digital assistant is evaluated using a series of tests that compare it against standard benchmark disk images and examine the support it provides to human investigators.

Keywords: Digital forensics, disk investigation toolkit, expert systems

1. Introduction

Forensic investigations of disks are challenging because of the wide variety of available tools. Existing commercial and open source tools must be considered and new tools are constantly being released. Investigators are expected to know how to use and configure these tools and they are required to have a fair degree of information technology expertise. They must also have considerable knowledge about the technical details of each new disk type, filesystem and the locations on the disk where information could be hidden.

This chapter builds on previous work on tool development [10] and presents an advanced automated disk investigation toolkit (AUDIT) that has been substantially improved and that leverages a dynamic knowl-

edge base and database. AUDIT is designed to support the integration of open source digital forensic tools within the Java Expert System Shell (Jess) [4] in order to enhance disk forensic investigations. The most important internal design change is the ability of AUDIT to dynamically update the knowledge base and database components with information from the expert system component. This substantially extends system functionality. Another important addition is the reporting mechanism that describes the activities of the system, including the inferences involved in decision making, which are useful when explaining how and why certain forensic tools were invoked automatically by AUDIT.

The long-term goal of this research is to create an intelligent assistant that supports forensic investigations. The assumption is that an investigator may not have adequate technical knowledge about the tools that could be used and/or the disk images to be examined. The current version of AUDIT is designed to speed up and enhance disk examinations regardless of the investigator's knowledge and skill levels. AUDIT currently supports the examinations of disks for graphic files, documents, email files and addresses, and also more specialized examinations involving credit card information and social security numbers. AUDIT is designed to be extensible so that new functionality is easily integrated in the existing system.

Researchers caution that the automation of the digital forensic process should not be "dumbed down" by encouraging expert investigators to rely more heavily on automation than their own knowledge [9, 12]. An important goal for AUDIT is to ensure that this does not occur. Although AUDIT incorporates knowledge of tools and disk structures, the assumption is that the investigator is, in fact, skilled in the art of investigations. AUDIT speeds up the technical aspects of the investigative process as suggested in [9]. Experimental evidence obtained from the analysis of several test disk images is used to demonstrate AUDIT's ability to significantly support human investigators.

2. Related Work

Stallard and Levitt [15] proposed one of the earliest applications of expert systems in the area of digital forensics. They used an expert system with a decision tree to detect network anomalies when attackers attempt to clear traces in log files. Liao et al. [11] also used an expert system to analyze log files; however, their approach leveraged fuzzy logic.

The Open Computer Forensics Architecture (OCFA) [16] is an early example of automating the digital forensic process. OCFA modules work

independently on specific file types to extract file content, but the architecture is not designed to search and recover files from a given device.

The Digital Forensics Framework (DFF) is an open source digital investigation tool and development platform [2]. It is a good example of tool integration and collaboration that reduces the burden on investigators when they use task-specific tools. However, the framework still requires knowledge and skills related to integrated tools and disk structures.

Fiwalk [5] is the closest work to that described in this chapter. Fiwalk automates the processing of forensic data to assist users who desire to develop programs that can automatically process disk images. The main differences between Fiwalk and this work is that Fiwalk specifically works on filesystem data and does not incorporate any artificial intelligence techniques. Fiwalk does simplify the task of filesystem analysis, but it requires knowledge about the filesystem and its structure.

Hoelz et al. [8] have developed the MultiAgent Digital Investigation Toolkit (MADIK) to assist forensics experts. They use an artificial intelligence approach where each agent specializes in a different task such as hashing and keyword search. However, the work does not leverage knowledge about forensic tools to assist non-expert users.

Fizaine and Clarke [3] have proposed a crime-dependent automated search engine for digital forensics. This tool focuses on the early stage of an investigation and collects information about specific crimes by assuming that most crimes have similar patterns. However, their approach does not support automated tool integration and configuration.

Commercial tools such as FTK [1] and EnCase [7] also automate examination tasks in order to assist investigators. The tools expect users to be technically skilled and to take training courses in order to use the tools effectively. Furthermore, the tools do not explain the decision making process, leaving this to expert investigators. Additionally, the tools are closed source and are not extensible by users.

At this time, no researchers have specifically addressed the problem of assisting human examiners during the analysis phase of investigations using an expert system. Additionally, existing systems do not support a general open source tool integration process; instead, they only integrate task-specific modules to automate certain activities.

3. AUDIT

Figure 1 presents the architecture of the advanced automated disk investigation toolkit (AUDIT). AUDIT has three components: (i) database (of tasks and tools); (ii) knowledge base; and (iii) core engine (which

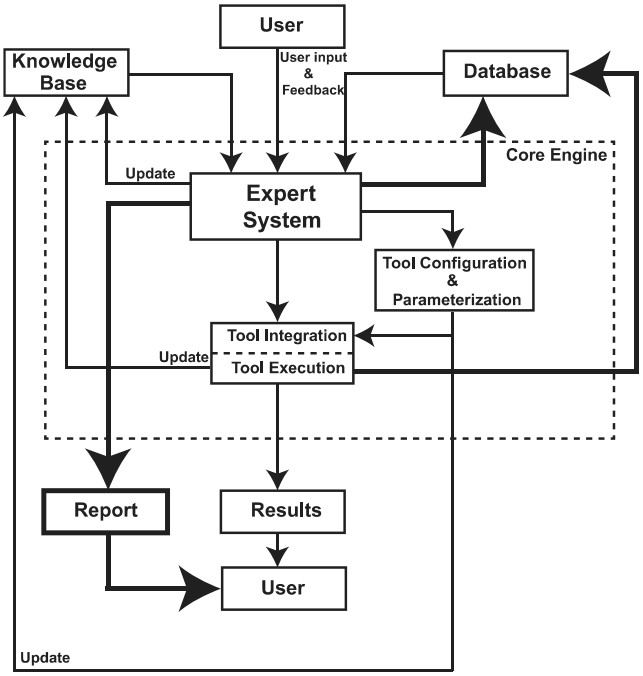

Figure 1. AUDIT architecture.

includes an expert system, tool integration component and configuration component). The elements in bold indicate major changes from the previous version of AUDIT. Bold boxes show new or substantially changed components. Bold connections show new update capabilities.

The new version of AUDIT can dynamically update its knowledge base (designed using Jess [4]) and its database. These are used by the expert system to configure and integrate the open source tools as needed. The dynamic knowledge base enables AUDIT to evolve during its execution by collecting more information about the target disk and investigation. For example, the database was initially designed to work on single partition disk images. However, this database is now modified based on the expertise in the knowledge base to enable AUDIT to analyze multi-partitioned disks.

3.1 Database

The AUDIT database maintains information about the tools that can be used with AUDIT and the investigative tasks that a typical investigator would perform (Figure 2). The database is designed using the MySQL relational database management system. The database main-

ident	toolname	task	para	p_in	input	p_o	output	p_conf	config	p1	p2	p3	S	F	R
☐ blklsUnalloc	blkls	recovering_unallo...	-A	N/A	?imageP...	>	?outputPath	N/A	N/A	N/A	N/A	N/A	0	0	0
☐ tskRecAlloc	tsk_recover	allocated_space_r...	-a	N/A	?imageP...	N/A	?outputPath	N/A	N/A	N/A	N/A	N/A	0	0	0
☐ tskRecBoth	tsk_recover	alloc/unalloc spac...	-e	N/A	?imageP...	N/A	?outputPath	N/A	N/A	N/A	N/A	N/A	0	0	0
☐ blklsSlack	blkls	recovering_slack_...	-s	N/A	?imageP...	>	?outputPath	N/A	N/A	N/A	N/A	N/A	0	0	0
☐ fileSystemType	fsstat	getting_filesytem...	-t -o	?imgO...	?imageP...	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0	0	0
☐ documentFilesFree...	photorec	document_files_se...	N/A	N/A	?imageP...	/d	?outputPath	/cmd	fileopt,	ever...	doc,e...	freesp...	0	0	0
☐ documentFilesPR	photorec	document_files_se...	N/A	N/A	?imageP...	/d	?outputPath	/cmd	fileopt,	ever...	doc,e...	whole...	0	0	0
☐ emailFilesPR	photorec	email_files_search	N/A	N/A	?imageP...	/d	?outputPath	/cmd	fileopt,	ever...	pst,e...	whole...	0	0	0
☐ emailsBulkExt	bulk_extra...	email_address_sea...	N/A	N/A	?imageP...	-o	?outputPath	-x	all	-e	email	N/A	0	0	0

Figure 2. Sample entries in the Tools table in the AUDIT database.

tains an entry for each tool that specifies configurations, input parameters and outputs for various tasks. Some of the configuration/parameter values are fixed whereas others are variable and can be defined during execution and, thus, changed dynamically.

For example, the `emailsBulkExt` entry specifies that the forensic tool `bulk_extractor` needs parameter `-x`. The `bulk_extractor` tool can use different scanners for different types of information such as email addresses and credit card numbers. Parameters `-x` in column `p_conf` and `all` in column `config` disable all scanners. Parameters `-e` in column `p1` and `email` in column `p2` enable a scanner for email address search.

In the previous version of AUDIT, the database table was static and could not be updated with knowledge collected during an investigation. In the new version, the database table is initially read by the expert system. The fields filled with keywords such as `?imagePath` are recognized by the expert system and changed as needed with the related values collected during the investigation. Note that empty fields are filled with `N/A` values to enable the expert system to recognize the empty fields correctly.

3.2 Knowledge Base

The AUDIT knowledge base is created using the rule-based programming functionality provided by the Java Expert System Shell (Jess). The knowledge base contains facts and rules, some of which are predefined and embedded in the system; others are created during an investigation. Most of the knowledge base is designed for forward chaining, which means that the rules are in the form of IF-THEN statements. However, backward chaining is used when creating inference reports that are discussed later in this chapter. Facts and rules can be added, deleted and modified as needed. In expert systems, facts can be quite simple such as "(John is male)" or more complex such as "(?person (gender ?gen)(age 25)(weight 180))."

```
(defrule update-knowledge-base-rule
(disk-image-path is ?imagePath)
(output-path is ?outputPath)
?toolInfo <- (tools (toolname ?tName)(input ?in)(output ?out))
=>
(if (eq ?in "?imagePath") then (modify ?toolInfo (input ?imagePath))
else (if (eq ?in "?outputPath")
then (modify ?toolInfo (input ?outputPath))))
(if (eq ?out "?outputPath")
then (modify ?toolInfo (output ?outputPath))))
```

Figure 3. Rule for updating the knowledge base.

The new version of AUDIT uses complex facts that were implemented in the previous version. These facts are typically modified frequently via update rules as shown in Figure 3.

```
(MAIN::tools (ident "emailsBulkExt")
(toolname "bulk_extractor")(task "email_address_search")
(input "?imagePath")(p_out "-o")(output "?outputPath")(p_conf "-x")
(config "all")(p1 "-e")(p2 "email"))

(MAIN::tools (ident "emailsBulkExt")
(toolname "bulk_extractor")(task "email_address_search")
(input "/home/utk/part5.img")(p_out "-o")(output "/home/utk/t2")
(p_conf "-x")(config "all")(p1 "-e")(p2 "email"))
```

Figure 4. Original fact before and after an update.

When a user enters a case, all the input and output values of the tool are changed. Figure 4 shows the facts present in the knowledge base before and after an update.

3.3 Core Engine

The core engine controls the execution of the system using the database, knowledge base and user input. The engine reads the tool specifications and investigative tasks from the database and creates new rules and facts as needed. It also links the investigative tasks and tools to the knowledge base and user input and feedback.

In the new version of AUDIT, the database is updated dynamically when new information is gathered by the expert system and after the tools are executed. The update process is performed by the core engine via updating rules. For example, Figure 5 shows the rule that updates the database when the input and output paths are entered by a user.

```
(defrule update-database-rule "updating database"
(disk-image-path is ?imagePath)
(output-path is ?outputPath)
(tools (toolname ?tName)(input ?in)(output ?out))
=>
(updateDatabase ?imagePath "?imagePath"
"UPDATE TOOLS SET input = ? WHERE input = ?")
(updateDatabase ?outputPath "?outputPath"
"UPDATE TOOLS SET output = ? WHERE output = ?"))
```

Figure 5. Rule for updating the database.

By doing so, previously-loaded incomplete data in the database (i.e., variables used with the ?<variableName> structure) become complete and are ready to be used by other rules. In the case of an update after a tool has terminated execution, AUDIT asks the user if the tool performed successfully or if it failed. Based on the feedback received from the user, related fields (S, F and R) for the tool are updated in the database. This information can be used to compare tools against each other; however, this feature is outside the scope of the research and is not discussed in this chapter.

3.4 Expert System

One of the important changes from the previous version of AUDIT is the migration of the expert system shell from CLIPS [14] to Jess [4]. Jess is written in Java and can access all Java APIs. The reasons for the migration were:

- **GUI User Interface:** CLIPS does not have a mechanism to allow users to create a GUI interface for expert system applications. Although extensions to CLIPS exist for this purpose, they are not maintained well, have limited access to CLIPS functionality and are infrequently used by the community.
- **External User Functions:** CLIPS and Jess support users in creating custom external functions. However, CLIPS requires recompilation to integrate functions whereas Jess provides direct access to an external function from a file that contains Jess constructs without recompilation.
- **Database support:** CLIPS does not have direct support for relational database management systems. However, Jess provides full support for most common relational database management systems, many of which are available in Java libraries.

```
(defrule find-email-addresses-BE
(search type is 4) ; email address search
(output-path is ?outputPath)
(disk-image-path is ?imagePath)
(tools (ident "emailsBulkExt")(toolname ?tName)
(task "emails_address_search")(params ?params)(p_in ?par_in)
(input ?input)(p_out ?par_out)(output ?output)(p_conf ?par_conf)
(config ?config)(p1 ?p1)(p2 ?p2)(p3 ?p3))
=>
(my-system-bulk-ex ?tName ?params ?par_in ?input ?par_out
(str-cat ?outputPath "/bulk_ext") ?par_conf ?config ?p1 ?p2 ?p3)
(assert (bulk-extractor is called))
)
```

Figure 6. Rule for email adddress searches.

AUDIT begins its execution by connecting to the database and reading all the data from it. The data is entered into a Jess template called `tools`. When a specific input, output and task are selected by a user, AUDIT starts running to collect certain information about the input disk (e.g., disk partitioning, disk volume type and disk physical sector size). All this information is also entered into two templates named `disk_layout` and `diskInfo` located in the knowledge base. The Jess templates maintain knowledge about tool usage and input disk information in the knowledge base.

AUDIT uses the data from the `task` column in Figure 2 to activate rules when a specific task needs to be performed using a particular tool. For instance, Figure 6 shows a rule that obtains relevant information from the database to use the `bulk_extractor tool` for email address searches. The line that begins with `(tools (ident ... )` represents the pattern in the `tools` template that is populated with information from the database and knowledge base. Note that `(toolname ?tName)` is a slot declared in the `tools` template and `?tName` is a variable name declared for this slot in order to store the tool name.

In Figure 6, it is specifically mentioned that the selected task is email address search. This is done by making the `(task "emails_address_search")` slot part of the pattern. In the event that multiple tools are available for the same type of task, another slot `(ident "emailsBulkExt")` is added to enable the rule to run only for a specific tool. The same rule can be used for other tools that perform the same task by specifying the `ident` slot (e.g., `(ident "emailsMultigrep")`). In fact, the rule can be extended by changing the object of `ident` to a variable, thus allowing multiple tools to run or choosing a specific

```
bulk_extractor /home/utk/part5.img -o /home/utk/t2/bulk_ext -x all -e email
```

Figure 7. Using `bulk_extractor` for email address search.

tool based on other knowledge. Figure 7 shows the Linux command that AUDIT executes for the selected tool after the rule in Figure 6 is activated.

The new capabilities of AUDIT are illustrated by its ability to implement examinations of multi-partition disks. In these scenarios, the knowledge base has to be changed dynamically because each partition in a disk has to be treated as an individual disk image. The new dynamic database and knowledge base capabilities enable this task to be performed in an efficient manner.

4. Reporting in AUDIT

Popular digital forensics tools (e.g., FTK and EnCase) generate investigation reports help explain the investigative findings using technical details. The reporting mechanism in AUDIT currently does not generate a full technical report; instead, it identifies all the procedures and tools used in the investigation. However, it also describes how and why the tools were used; this feature is not provided by other forensic tools.

AUDIT reports include detailed information about the input disk, the tools invoked and their usage, inference information about what caused a tool to run, and the disk layout in the case of multiple partitions. A report is created automatically and user feedback is added to the report after user interactions. Figure 8 shows a portion of the examination report generated after the analysis of a single partition disk.

Figure 9 shows the report related to one of the rules (slack-space-extraction-rule) that fired during the analysis. The firing facts (facts start with `fact-id`) explain the actions that were previously taken and why. Note that the `fact-id` number is actually the time the fact was (and, thus, the order in which facts were) added to the knowledge base.

The inference report informs the user that AUDIT has learned input, output and task information from facts `f-36`, `f-37` and `f-48`, respectively. The user can see that slack space carving starts only if the expert system knows that allocated space and unallocated space were previously analyzed as evidenced by facts `f-54` and `f-56`. The rule is fired when all the facts are true, including fact `f-10`. Note that this complex fact indicates that the tool `blkls` has been invoked. The execution order is seen clearly in Figure 8. Furthermore, `f-48` is added when the user

```
Single partition disk!
.
.
Volume Type = Unknown
Sector Size = 512
Disk (Physical) Block  Size = unknown
Path to the input file - >> /home/utk/ecitTest-3.raw
Path to the output directory - >> /home/utk/t1
Document file search will be performed!
**********************************************************
tsk_recover is running on the input disk image to extract
user created/deleted files! Command line below is used:
tsk_recover -a /home/utk/ecitTest-3.raw
/home/utk/t1/tsk_recover/allocated -o 0
**********************************************************
tsk_recover is running on the input disk image to extract
user created/deleted files! Command line below is used:
tsk_recover /home/utk/ecitTest-3.raw
/home/utk/t1/tsk_recover/unallocated -o 0
**********************************************************
blkls is running on the input disk image to extract
unconventional spaces! Command line below is used:
blkls -s /home/utk/ecitTest-3.raw -o 0 >
/home/utk/t1/slackSpace/slack.dd
**********************************************************
.
.
Feedback: Interesting data is not found so far.
```

Figure 8. Partial examination report for a single partition disk image.

```
Fired Rule Name : MAIN::slack-space-extraction-rule
Firing Facts : [Token: size=7;sortcode=14478322;negcnt=0
f-49 (MAIN::start slack space carving);
f-56 (MAIN::unallocated space analyzed);
f-54 (MAIN::allocated space analyzed);
f-36 (MAIN::disk-image-path is "/home/utk/part5.img");
f-37 (MAIN::output-path is "/home/utk/t1");
f-48 (MAIN::search type is 1);
f-10 (MAIN::tools (ident "blklsSlack") (toolname "blkls")
(task "recovering_slack_space") (params "-s") (p_in "N/A")
(input "/home/utk/part5.img") (p_out ">")
(output "/home/utk/t1") (p_conf "N/A") (config "N/A")
(p1 "N/A") (p2 "N/A") (p3 "N/A"));]
```

Figure 9. Partial inference report for slack space extraction.

selects "document search" and `f-49` is added because a document search analysis rule adds fact `f-49` to the knowledge base when it is fired.

Examination and inference reports are useful to expert and non-expert users. For example, expert users could use the inference order to discern the order in which the tools were invoked and parts of the disk were analyzed. They could then redo certain analyses (using information in the command lines reported by AUDIT) with other tools for purposes of verification. From the initial disk analysis by AUDIT, they would already have obtained substantial information about the disk with considerable time savings. On the other hand, non-expert users could, for example, obtain information about the use of a standard carving tool that could accelerate their learning.

5. Testing AUDIT

This section presents the results of tests conducted on AUDIT to explore and evaluate its capabilities. Note that even testing the data hiding process is non-trivial because few, if any, tools are available that support the hiding process.

The experiments involved two groups of tests. The first group of tests ran AUDIT on widely-used tool testing disk images from NIST [13] and the Digital Corpora [6]. The second group of tests compared AUDIT's performance against that of a fairly knowledgeable non-expert human investigator. Specifically, the investigator had moderate technical knowledge and some experience related to forensic investigations. He had good knowledge about open source tools and disk structures. The goal of the tests was to evaluate how AUDIT can support human investigators.

5.1 Experimental Setup

Disk images were created using ForGe [17], a forensics disk image generator. The first step involved setting up a "case," which involved generating a 1 GB disk image with a sector size of 512 bytes and a cluster size of eight sectors or 4 KB. The NTFS filesystem was used for the disk image because it is the only filesystem that is fully supported by ForGe. Note that ForGe does not allow the creation of multi-partition disk images.

The next step was to create a "trivial strategy" representing a directory tree containing files normally found in a filesystem. The directory tree included 31 directories named and structured to mimic a Windows operating system folder hierarchy. All the directories contained ten files, except for the root directory, which contained no files. Thus, each generated disk image had 300 "trivial" files that were not hidden.

Table 1. File extensions and numbers of files in the dataset.

Ext	Qty	Ext	Qty	Ext	Qty	Ext	Qty	Ext	Qty
pdf	257	xls	60	csv	17	log	5	sys	1
html	227	ppt	54	pst	9	png	3	tmp	1
jpg	104	xml	31	unk	7	text	3	dbase3	1
txt	84	gif	27	gz	7	kmz	2	rtf	1
doc	67	ps	22	swf	7	pps	2	kml	1

An initial dataset of 1,000 random unique files was obtained from the Govdocs1 digital corpus [6]. Table 1 lists the file extensions and numbers of corresponding files in the initial dataset.

Table 2. Sizes, types and numbers of horse images.

Size	Ext	Qty	Ext	Qty	Ext	Qty
> 4 KB	jpg	14	jpg	16	png	0
< 4 KB	jpg	11	jpg	5	png	4

Fifty horse images representing illegal images were added to the initial dataset. Table 2 lists the sizes, types and numbers of the horse images. The 300 trivial files were chosen randomly from the resulting set of 1,050 files.

The final step in generating a disk image using ForGe was to create a "secret strategy" representing files that were hidden in forensically-interesting ways. Three hiding methods were used in the tests: (i) placing a file in file slack space; (ii) placing a file in disk unallocated space; and (iii) deleting a file from the filesystem. The files to be hidden were chosen randomly from the set of 1,050 files. The original set of 1,050 files was divided into three subsets: (i) graphic files of horse images less than 4 KB (Type 1); (ii) Outlook email archive (pst) files (Type 2); and (iii) all the remaining files (Type 3). Files less than 4 KB were hidden in the file slack because larger files would not fit due to the fixed cluster size. The possibility of hiding a pst file in each disk image was considered in order to test AUDIT's new email search functionality.

ForGe imposes limitations on how files can be hidden. Specifically, it only allows a maximum of one file to be hidden on disk for each hiding method. Therefore, a file was chosen at random from each of the three subsets and was assigned a hiding method. A Type 1 file was assigned to the file slack hiding method only. Files chosen from the other two subsets were randomly assigned to the unallocated and deleted hiding methods.

Table 3. Number of files in each area on the disks.

	Allocated Space	**Deleted Space**	**Slack Space**	**Unallocated Space**
Disk 1	16	2	1	0
Disk 2	15	3	0	1
Disk 3	16	3	1	2
Disk 4	22	4	0	1
Disk 5	26	1	1	0

Next, it was determined randomly whether or not a hiding method would be used in a test. Thus, a minimum of zero and maximum of three hiding methods were present in each test.

A disk image contained very few hidden files. Since it was unlikely that files would contain credit card or social security numbers, some document files that included such numbers, email addresses and user names were manually hidden in deleted space or unallocated space. Table 3 shows the final numbers of forensically-interesting files contained in the five test cases (disks). Note that the human investigator was unaware of the numbers, types and locations of the hidden files.

5.2 Testing Regiment 1

The first testing regimen conducted experiments in two phases to evaluate AUDIT's performance on the five test disks. In the first phase, the investigator (expert) was asked to use his own skills and open source tools (e.g., SleuthKit and photorec), but not to use AUDIT. In the second phase, the investigator was asked to exclusively use AUDIT. The investigator analyzed the disks in order and was given instructions for each disk that are discussed below. Due to space limitations, only a subset of the results are presented.

For all the disks, the investigator was asked to find graphic and email files on the disks and report the locations of the files. For Disk 1, the exact numbers of graphic and email files were provided.

Table 4 shows the results related to finding graphic and email files. In general, the performance of the investigator was almost as good as that of the investigator using AUDIT. However, better performance was observed when the investigator used AUDIT on Disks 1 and 2. In the case of Disk 1, the investigator (without AUDIT) was unable to report the location of one horse image (located in slack space) because he found the image using a file carver that only handles sector-level information. When analyzing the remaining disks, the investigator (without AUDIT)

Table 4. Finding graphic and email files.

Disk		Graphic Files	Email Files	File Locations
1	Expert	17/17	2/2	×
	AUDIT	17/17	2/2	✓
2	Expert	15/17	2/2	✓
	AUDIT	17/17	2/2	✓
3	Expert	13/13	5/5	✓
	AUDIT	13/13	5/5	✓
4	Expert	21/21	3/3	✓
	AUDIT	21/21	3/3	✓
5	Expert	24/24	2/2	✓
	AUDIT	24/24	2/2	✓

correctly reported the locations because by then he had learned how slack space is analyzed using AUDIT.

In the case of Disk 2, the investigator missed two graphic (`png`) files because he did not extend his search to all graphic file types. In contrast, when the investigator used AUDIT, he found all the files and their locations.

The investigator was told that Disks 3 and 4 contained hidden document files. His task was to recover the files from the hidden locations and report the file types, numbers of files and their locations. In the case of Disk 3, the investigator was also asked to find files containing credit card numbers, social security numbers and email addresses to test AUDIT's search capabilities. In the case of Disk 4, the investigator was asked to find files containing email addresses. In the case of Disk 5, the investigator was asked to find social security numbers and email addresses.

Table 5. Hidden document files and their locations.

Disk		Qty	Type	Location	Qty	Type	Location
3	Expert	2	`pdf`	unallocated	2	`doc` and `xls`	deleted
	AUDIT	2	`pdf`	unallocated	2	`doc` and `xls`	deleted
4	Expert	1	`xls`	unallocated	1	`pdf`	deleted
	AUDIT	1	`xls`	unallocated	1	`pdf`	deleted

Table 5 shows that all the hidden files were found regardless of whether or not the investigator used AUDIT.

Table 6 compares the time required by the investigator to analyze each disk without and with AUDIT. Better results were obtained when the

Table 6. Analysis times with and without AUDIT.

	Disk 1	**Disk 2**	**Disk 3**	**Disk 4**	**Disk 5**
Expert	9m 26s	13m 45s	25m 10s	18m 45s	26m 13s
AUDIT	7m 55s	5m 33s	12m 8s	10m 16s	20m 51s

investigator used AUDIT. In the case of Disk 1 the investigator was not yet familiar with AUDIT's output, so the times were similar. However, for Disks 2, 3 and 4, the investigator generally took about half as much time when AUDIT was used. The time required when AUDIT was used on Disk 5 was surprising until it was determined that the investigator did not scan the AUDIT output of allocated files until very late.

Table 7. Sample benchmark disk images.

Disk Name	**Category**	**Target**	**Source**
`dfr-04-fat`	Deleted file recovery	36	NIST
`dfr-05-ntfs`	Deleted/fragmented file recovery	7	NIST
`L0_Documents`	Non-fragmented carving	7	NIST
`L0_Graphics`	Non-fragmented carving	6	NIST
`L1_Documents`	Fragmented file carving	7	NIST
`L1_Graphics`	Fragmented file carving	6	NIST
`nps-2010-emails`	Email address recovery	30	Digital Corpora

5.3 Testing Regimen 2

The second testing regimen evaluated the performance of AUDIT on benchmark disk images as well as on multi-partition disk images. Table 7 lists the sample benchmark disk images. AUDIT was applied to the first two disk images to recover several non-fragmented files with non-ASCII file names from `dfr-04-fat` and to recover several fragmented and deleted files from `dfr-05-ntfs`. All 36 deleted files, which were located across multiple partitions, were recovered from Disk 1. AUDIT also recovered all seven files from Disk 2. Although fragmented file recovery is typically a difficult task, it was not the case in this situation.

The file carving functionality of AUDIT was also tested. AUDIT was applied to the disk images `L0_Documents` and `L0_Graphics` in order to carve non-fragmented documents and graphic files, respectively. The filesystems in both disk images were corrupted, so no metadata information was available. Nevertheless, AUDIT was used to successfully carve all the files at the correct locations as in the disk creation reports [13].

The next two test cases evaluated the use of AUDIT when carving sequentially-fragmented documents and graphic files from the disk images, `L1_Documents` and `L1_Graphics`, respectively. AUDIT was used to successfully carve all the sequentially-fragmented files (seven documents and six graphic files). All the contents of the carved files were complete when compared against the disk image reports [13].

The last test case evaluated the email address search functionality of AUDIT. The test disk image `nps-2010-emails` contained 30 email addresses located in several file types, including documents and compressed files.

AUDIT invoked the `bulk_extractor` tool to find the email addresses. It retrieved all the email addresses in the narrative file `nps-2010-emails`, except for one email address (`plain_utf16@textedit.com`) with non-ASCII content. AUDIT also automatically recovered email addresses from documents; a visual check showed one of the addresses was in a `txt` file. This demonstrates the power of AUDIT's tool integration functionality.

6. Conclusions

AUDIT is a unique extensible tool that is designed to configure, integrate and use open source tools for disk investigations while leveraging expert system capabilities. The principal contribution of this research is the enhanced design and a new implementation of the AUDIT toolkit. AUDIT is the first tool to use an expert system to automate the technical aspects of disk investigations at an expert investigator level. The new implementation better automates several tasks using open source tools for general and specific search tasks. These include the determination of the level of knowledge of the investigator and the use of this information to make further choices during a disk investigation; use of domain-specific knowledge embedded in the knowledge base and database to configure and parameterize the appropriate digital forensic tools for execution; execution of the appropriate tools at the correct times; and execution of tools based on the dynamic knowledge obtained from previous executions. Thus, AUDIT supports the integrated use of digital forensic tools.

The second contribution of this research is that AUDIT now creates two reports for an investigator, an inference report and an examination report. The inference report specifies the logic of how and why certain tools were automatically used; none of the existing tools provides such information. The examination report details the findings related to the analysis.

The third contribution of this research is the significant testing that shows how AUDIT can support the investigative process by handling core information technology expertise, including the choice and operational sequence of tools and their proper configurations.

AUDIT is already a very effective digital forensic assistant. As the scope and functionality of AUDIT are extended and refined, it may well become an indispensable tool for digital forensic investigations. The latest version of AUDIT is available at `sourceforge.net/projects/audit-toolkit`.

Acknowledgement

The authors wish to thank Clayton Butler for his help with this research.

References

[1] AccessData, Forensic Toolkit (FTK), Lindon, Utah (`www.accessdata.com/solutions/digital-forensics/forensic-toolkit-ftk`), 2016.

[2] ArxSys, Digital Forensics Framework, Le Kremlin-Bicetre, France (`www.digital-forensic.org`), 2016.

[3] J. Fizaine and N. Clarke, A crime-dependent automated search engine for digital forensics, *Advances in Communications, Computing, Networks and Security*, vol. 10, pp. 73–87, 2013.

[4] E. Friedman-Hill, Jess, the Rule Engine for the Java Platform, Sandia National Laboratories, Livermore, California (`www.jessrules.com`), 2016.

[5] S. Garfinkel, Automating disk forensic processing with SleuthKit, XML and Python, *Proceedings of the Fourth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering*, pp. 73–84, 2009.

[6] S. Garfinkel, P. Farrell, V. Roussev and G. Dinolt, Bringing science to digital forensics with standardized forensic corpora, *Digital Investigation*, vol. 6(S), pp. S2–S11, 2009.

[7] Guidance Software, EnCase Forensic, Pasadena, California (`www.guidancesoftware.com/encase-forensic.htm`), 2016.

[8] B. Hoelz, C. Ralha and R. Geeverghese, Artificial intelligence applied to computer forensics, *Proceedings of the ACM Symposium on Applied Computing*, pp. 883–888, 2009.

[9] J. James and P. Gladyshev, Challenges with Automation in Digital Forensic Investigations, Digital Forensic Investigation Research Group, University College Dublin, Dublin, Ireland, 2013.

[10] U. Karabiyik and S. Aggarwal, AUDIT: Automated disk investigation toolkit, *Journal of Digital Forensics, Security and Law*, vol. 9(2), pp. 129–144, 2014.

[11] N. Liao, S. Tian and T. Wang, Network forensics based on fuzzy logic and expert system, *Computer Communications*, vol. 32(17), pp. 1881–1892, 2009.

[12] M. Meyers and M. Rogers, Computer forensics: The need for standardization and certification, *International Journal of Digital Evidence*, vol. 32(2), 2004.

[13] National Institute of Standards and Technology, The CFReDS Project, Gaithersburg, Maryland (`www.cfreds.nist.gov`), 2016.

[14] SourceForge, CLIPS: A Tool for Building Expert Systems (`www.clipsrules.sourceforge.net`), 2016.

[15] T. Stallard and K. Levitt, Automated analysis for digital forensic science: Semantic integrity checking, *Proceedings of the Nineteenth Annual Computer Security Applications Conference*, pp. 160–167, 2003.

[16] O. Vermaas, J. Simons and R. Meijer, Open computer forensic architecture a way to process terabytes of forensic disk images, in *Open Source Software for Digital Forensics*, E. Huebner and S. Zanero (Eds.), Springer, New York, pp. 45–67, 2010.

[17] H. Visti, ForGe: Forensic test image generator (`www.github.com/hannuvisti/forge`), 2015.