Forensic Science International: Digital Investigation 38 (2021) 301264

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

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

Dataset construction challenges for digital forensics

Graeme Horsman [a, *], James R. Lyle [b]

[a] Teesside University, United Kingdom
[b] National Institute of Standards and Technology, United States

ARTICLE INFO

Article history:
Received 15 March 2021
Received in revised form
14 July 2021
Accepted 14 July 2021
Available online xxx

Keywords:
Digital forensics
Datasets
Testing
Tool-testing

ABSTRACT

As the digital forensic field develops, taking steps towards ensuring a level of reliability in the processes implemented by its practitioners, emphasis on the need for effective testing has increased. In order to test, test datasets are required, but creating these is not a straightforward task. A poorly constructed and documented test dataset undermines any testing which has taken place using it, eroding the reliability of any subsequent test results. In essence, given the time, effort and knowledge required to generate datasets, the field must guide those carrying out this task to ensure that it is done right at the first instance without wasting resources. Yet, there are currently few standards and best practices defined for dataset creation in digital forensics. This work defines three categories of dataset which typically exist in digital forensic - tool/process evaluation datasets, actions datasets and scenario-based datasets, where the minimum requirements for their creation are outlined and discussed to support those creating them and to help ensure that where datasets are created, they offer maximum value to the field.

1. Introduction

It would be wrong to tread over already very worn ground and to restate the words of those over the past 15 years who have championed the need for knowledge and data sharing within the digital forensics (DF) field (for those interested, see an example set of works which include Bruschi et al., 2004; Biros et al., 2007; Kahvedžić and Kechadi, 2009; Huang et al., 2010; Horsman et al., 2014). Instead, this work will proceed straight to the point. The sharing of discovered knowledge is one area where this discipline must continue to improve, with attitudes towards community sharing of knowledge beginning to change (Horsman, 2019). Of particular need is the generation of appropriate 'datasets' for communal and/or organisational use in relation to training, method/tool evaluation and knowledge discovery and confirmation (Garfinkel et al., 2009; Grajeda et al., 2017), and this area will remain the focus of this work. Recent formalised pushes to enhance the validation and verification of the tools and methods used in DF investigations (Arshad et al., 2018; Horsman, 2019b; Nance et al., 2019; Tulley, 2020) have arguably exposed the limited number of resources in the form of appropriately designed datasets which can support those in this field to carry out this task (Park, 2018). If the DF discipline is to increase its tool evaluation activities, dataset generation is arguably pivotal to achieving this (Luciano et al., 2018).

A dataset is defined 'as a collection of related, discrete items that have different meanings depending on the scenario and was utilized for some kind of experiment or analysis' by Grajeda et al. (2017 p.95) with over 350 known datasets formally recorded (see https://datasets.fbreitinger.de/datasets/ for a dataset reference repository). Consideration should also be given to available datasets which were not created for the purpose of testing in DF, but may have gone on to be used in such a way (perhaps the best example may be the ENRON email dataset). In the context of DF, datasets typically are utilised in the following three areas (National Institute of Standards and Technology, 2019).

1. For training: Datasets can be generated for the purpose of training, both investigative training and tool-driven training. Often datasets simulate mock scenarios in order to provide a practitioner with the 'experience' of undertaking casework in order to assess their ability to identify, examine and interpret any actionable 'mock-evidential' information. Training datasets offer value in the form of practitioner skills-development and assessment.
2. Tool/process evaluation: Datasets can be designed to simulate or facilitate the running of investigative procedures, process or methodology in order to demonstrate compliance to a standard,

* Corresponding author.
E-mail address: graeme.horsman@googlemail.com (G. Horsman).

https://doi.org/10.1016/j.fsidi.2021.301264
2666-2817/

consistency of behaviour, or, to assess completeness and accuracy. Datasets designed for this purpose support wider field initiatives related to quality assurance and management. Perhaps the most difficult of the four contexts is the generation of data designed to facilitate the evaluation of a DF tool or tool function. Such datasets may also be referred to as tool validation/verification datasets, but due to the context-intricacies associated with each term (discussed further in Section 2.2), we opt for the term 'tool/process evaluation' to avoid any potential terminology confusion.

3. Data exploration and reverse engineering (research & development): These datasets attempt to correlate user interactions with a piece of software or application to digital traces left behind by these acts. As a result, these datasets provide a resource for understanding how certain software/application behaves on a device and to support a practitioner's interpretation of any digital traces left by usage.

Datasets for DF are an invaluable resource, supporting practice development and improvement within the field, providing they are appropriate for the purpose for which they were designed. The need to create and share datasets is by no means a new concept where the benefits are clear, however, the requirement to do this in a fit-for-purpose way, is. Whilst sounding critical, it is arguably naive to suggest that all efforts to create and share datasets are beneficial, where substandard datasets can be detrimental to those who use them and impart any form of reliance on the results of their subsequent use. This is particularly concerning if poorly constructed datasets are utlised for tool validation purposes. As a relatively small field, DF does not possess the resources to waste via the construction of datasets which offer little value - we need to get it right in as many cases as possible. The field must critically assess what form datasets must maintain if they are to have the greatest value for their given purpose, and then encourage this format and design. In essence, a minimum standard for the construction of datasets is suggested as a means of ensuring that those who dedicate time and effort to creating them, do so in an effective way.

In regard to the creation and sharing of datasets, practitioners must be able to both use the knowledge contained in them, and validate it if they are to serve a purpose for developing practices in DF and the three contexts noted above. In order to facilitate this, the contents of a dataset must be accurately and comprehensively constructed and described, where this task is far from straightforward. Accurate construction involves determining the purpose of the dataset, followed by identifying the type and structure of the data it must contain in order to fulfill its purpose. This alone offers little value if it isn't supplemented by an accompanying description of the dataset.

Datasets which are not well described can only serve as informative in terms of value, offering little evaluative use. Taking a critical view, it is difficult to ascertain the value of a dataset for an evaluative context where there is no way to assess reliably what it contains. Using the example of a shared dataset which contains a set number of 'evidential files' to be identified and processed, if the dataset lacks sufficient documentation describing said evidential files, then those using it are unable to attribute any level of reliability or meaning to any process they have used to investigate it of inferences made about the files. Such datasets allow practitioners to 'take a look' at the data, but the value of this process is low in terms of training, development and validation. In essence, poorly described datasets are unusable in many contexts where an analogy is drawn to a black-box testing system where the tester has no knowledge of the inputs; in essentially, a broken system. This problem is compounded by the fact that often, test datasets are produced once and are not editable, emphasising the need to ensure that contemporaneous documentation is maintained throughout the dataset's production.

This work examines the challenges of dataset generation in DF and offers an analysis of the requirements needed to ensure those creating datasets do so in a fit-for-purpose way, bringing maximum value to the DF field. Three dataset 'types' are identified and defined - tool/process evaluation datasets, actions datasets and scenario-based datasets, where the minimum requirements for their creation are outlined and discussed.

2. The purpose of datasets

Regardless as to which of the three purposes of datasets notes in Section 1, their implementation is often with the intention to 'test' a process, either the practitioner within a training scenario, a tool or methodology in regards to any claimed functionality or a hypothesis in relation to a believed function of a piece of software or application. Whilst the purpose of 'testing' may seem at first glance a very straightforward concept to define, in reality the 'purpose' is often a multifaceted concept-driven largely by the domain in which any testing resides. At a high-level, testing often sets out to establish whether something (the 'tested entity') performs in the way in which it was intended to. In a simplified scenario, the test may evaluate the performance of the tested entity objectively against a specification of required performance (Meyer, 2008) - in essence, did the thing tested do what it was meant to do? On completion, for any test to be effective in its implementation, it must be able to determine the difference between correct and incorrect behaviour. If ambiguity exists as to the 'correctness' of performance, then issues exist with regards to the test construction and/or implementation. Mili and Tchier (2015) state that the success of a test depends on the quality of test data. If weaknesses exist in the quality of any test dataset used in DF, then the reliability of any results derived from its use is eroded.

Whilst it may seem a controversial question to ask, it is one which needs addressing - why may a practitioner and/or their organisation opt to test their investigatory processes and tools using a dataset? There is no single answer to this, in fact multiple motivations may exist.

1. Because they are told to test: Whilst it is argued that the value of testing tools, processes and hypotheses is clear, it may not be considered necessary by all those in the DF field. To explain further, attitudes may exist which believe that the burden of testing does not lie with the practitioner, but the vendor of the tools they use, or the creator of the methodologies they implement. This is perhaps a controversial stance to take as frequently vendors state within their end-user license agreement (EULA) that the burden of testing lies with the user and errors may exist (Horsman, 2018).
2. Because they believe they should: Those who recognize the value of testing and the benefits it can bring with regards to quality assurance and reliability may opt to test because they believe they should test as part of their role.
3. To achieve accreditation: There are those who opt to test because it is a requirement of an accreditation process, and without documented test protocols accreditation cannot be obtained. Whilst this is not a negative thing, it is important that testing is properly developed and implemented within the spirit in which it is designed. Token efforts to test can be detrimental if they instill false confidence in the process tested.
4. To detect error & prevent error: Testing can be implemented in order to detect and prevent errors from occurring in the investigatory process. As a product of this process, testers are able to

assess their trust in a process and confidence in its sustained usage.
5. To confirm functionality: Thorough testing allows the functionality of a process/tool to be verified, and to confirm understanding as to what is occurring when the process/tool is implemented.
6. Risk mitigation: Testing is a form of risk mitigation, and therefore effective testing lowers the chance of critical errors going undetected in the investigatory process.
7. Familiarity: In addition, running a set of test cases with known results helps the tool user know what to expect from a tool and not be surprised when the tool exhibits unexpected behavior.

The importance of robust datasets in DF for performance evaluation due to any one of the seven contexts noted above cannot be overstated.

2.1. Existing literature

Where datasets in DF are discussed in DF literature, their coverage is often as a supplementary presence to a core research theme being addressed. In essence, any dataset which accompanies a piece of research is often seen as supplementary in nature. The problem this may create is that whilst the research itself may be valid, it may be difficult to evaluate if the dataset is poor in terms of construction, where details of the dataset 'build process' should be considered equal in importance to the primary findings of the research contribution. There is coverage of datasets for DF in academic literature, where to acknowledge but a few, datasets exist for image/multimedia forensics (Dang-Nguyen et al., 2015; Al-Sanjary et al., 2016; Shullani et al., 2017), Microsoft Windows registry files (Park, 2018), botnets (Koroniotis et al., 2019), facial recognition (Al-Kawaz et al., 2018) and age estimation (Anda et al., 2018). In addition, Garfinkel's (2012) Digital Corpora offers a range of scenario-based datasets. Yet there remains limited commentary on dataset creation best practices, with Park (2018) offering one of the only debates. Arguably one of the most prominent initiatives currently in operation where dataset generation for testing occurs lies with that of the National Institute of Standards and Technology's Computer Forensics Tool Testing Program (CFTT), where it is necessary to consider their practices here.

2.2. The National Institute of Standards and Technology approach - 'a commentary'

The CFReDS (Computer Forensic Reference Data Sets) project at National Institute of Standards and Technology's (NIST) is a repository of digital storage device images. Investigators can use CFReDS in several ways including validating the software tools used in their investigations, equipment check out, training investigators, and practice using forensic tools. Some images are produced by NIST, often from the CFTT (tool testing) project, and some are contributed by other organizations. The Computer Forensics Tool Testing Program (CFTT) project at the National Institute of Standards and Technology's (NIST) (2020) The CFTT project posted its first document for public comment, a specification for Disk Imaging including requirements and test assertions in March 2001. CFTT submitted the first forensic tool test report, Red Hat GNU fileutils 4.0.36 dd, based on the final version of the specification to the National Institute of Justice for publication in August of 2002.

The CFTT project approached forensic tool testing with the Conformance Testing Model, often used to certify that a product conforms to some standard. The conformance testing model verifies that a product performs according to its specified standards. For example, the Common Criteria for Information Technology Security Evaluation follows the conformance testing model to certify that security software meets formal security standards. Because there were no published standards for forensic tools, CFTT took on the task of writing specifications for the tool functions they were tasked with testing. The tool specification included definitions of the function to be tested, a list of requirements the tool should meet, a list of test assertions to specify conformance to requirements and a set of test cases to be run. CFTT also created software to create test data, test data sets, software to evaluate test case results and procedures to follow when executing test cases. A formal test plan, test report and code review were published for the test support software used with the disk imaging tool tests. No anomalies were found. Due to the limited resources available to CFTT, later support tools were created and tested without production of a formal test report.

Before CFTT creates test data sets, CFTT first needs a tool function specification and a test plan with test cases:

1. With the help of law enforcement representatives that advise the CFTT project, a forensic tool function is identified for testing along with a list of candidate software or hardware tools.
2. CFTT examines the selected tools and produces two lists of tool features offered: core features that are offered by all tools and optional features that are offered by some tools. For example, all imaging tools can acquire an entire hard drive, but only some tools allow imaging of a single partition (an optional feature). Of course, as with any generalization there will be occasional exceptions that have to be handled.
3. For each feature a list of requirements is created to specify what the feature is supposed to do.
4. A list of parameters that could impact tool behavior is created to help specify test cases. These may be tool settings that didn't fit as a tool feature or run time environment factors such as type of file system to be examined.
5. Test cases are created by examination of test assertions and test parameters to create test run scenarios that trigger a subset of test assertions and test parameters. The test parameters and test assertions guide specification of tool settings. The test assertions specify the expected inputs and outputs of the test case. Additional test cases are created until all test assertions and test parameters have been addressed.
6. If a test assertion cannot fail in a detectable way, then something is wrong with the specification and needs to be corrected.

After we have a set of test-cases we are ready to create test data. In fact, as we are writing test assertions, we should be developing a plan for how each assertion can fail. Testing is all about getting a tool to fail. The more unique opportunities a tool is given to fail, the more confidence in the correctness of tool results we have when used for a real investigation.

CFTT uses two data set creation approaches: static and 'on-the-fly'. The static data sets are created in such a way that it is convenient to make a disk image of the data and then the disk image can be imported into a forensic tool for examination. Creation of a test data set is often a combination of scripts and custom tools. The on-the-fly data sets are usually for some type of function that interacts directly with a device. Each test case has a set of procedures for preparing a device or test image for the test. In addition, there may be a set of custom tools to help evaluate the result for both test data creation approaches.

The on-the-fly testing usually follows this protocol:

1. Populate a device with test data designed to reveal anomalies. Using custom tools to set-up the device.
2. Run the tool under test, possibly with a custom test harness.

3. Examine the results, using custom tools to help evaluate the results. If the test case does not modify the device (e.g., hard drive), the device can be reused for testing another tool. Precautions are taken to back up the device in case it is modified during running the test case and needs to be restored.

Tool functions at CFTT that use the on-the-fly approach include disk imaging, write blocking, forensic media preparation (drive wiping) and mobile device testing. Procedures for setting up test devices are posted on the CFTT web site along with a description of notable features of the data setup. For example, for disk imaging each sector of the device is given unique content that includes the LBA address of the sector. This allows easy diagnoses of a common issue of misplaced sectors if a tool places an imaged sector in the wrong location in the image or places a given sector in the image more than once.

Sometimes writing the procedures to follow were made challenging because an unusual condition would require additional steps to finish the procedure. For example, when testing disk wiping, for a tool that allows using the built-in security erase command you need to ensure that (1) the disk drive supports the security erase command and (2) the test computer BIOS does not disable the security feature set.

The static test data sets are usually provided as a set of small disk images for testing each tool function. The tool functions that use a static data set are file carving for graphic and video files, Meta-data based deleted file recovery and string searching.

For some test data sets additional tools need to be available to evaluate the test results. For example, it is often suggested to hash (MD5 or SHA-1) the result of deleted file carving or meta-data based deleted file recovery to see if the file is correctly recovered. However, this only gives a yes or no answer and does not measure if the recovered file is a total failure or a near miss. Rather than use an all or nothing measure it is more useful to measure the quality of the recovered files. For file carving we used two measures. First a visual evaluation to see if the returned file can be viewed. Second an examination of the data returned to see how much of the original file is returned, how much data is omitted and how much is not from the original file. Both measures were often revealing of important aspects of the tool that just one measure did not show. For example, one tool for one graphic file format, the returned result was not displayable, but an examination of the returned data file revealed that the tool returned all the image except for the last block of data.

For some data sets, no evaluation tools are needed. For example, each string search test case has a list of the string instances that should be returned. Evaluating each test case is just a matter of comparing the list of expected hits to the actual hits returned. For the string search test cases several test assertions are tested at the same time. For example, the string "DireWolf" appears 15 times in the test data set. Each instance of the target string is followed by a unique ID number so that an examination of a hit context confirms the actual instance returned. Some of the combined test assertions that can be tested:

- Find a string in an active file for each of 7 file systems (FAT, ExFat, NTFS, ext4, HFS + ignore case, HFS + case sensitive, & APFS).
- Find a string in a deleted file for each of 7 file systems.
- Find a string in unallocated space.

In order to give opportunities for testing to fail we also include the strings "WOLF" (all caps), "Wolf" (mixed case), "wolf" (all lower case) and "WereWolf". These strings support several searching test assertions with enough close strings for likely errors:

- Search for "wolf" with match case might fail by hitting "WOLF" or "Wolf" or "DireWolf."
- Search for "Wolf" as a whole word might fail by returning "WereWolf."

For testing UNICODE (UTF-8, UTF-16-BE & UTF-16-LE) 43 string instances are needed. CFTT also considered types of character sets and decided to include Latin based character sets with diacritic marks: Spanish, French, German, and Italian; A non-Latin character set: Russian; a right-to-left presentation: Arabic; distinct Asian character sets: Chinese, Korean, and Japanese Kana. There were many other possibilities, but this covers most character set forms likely to be found.

The general discussion in the forensic community about tool testing needs to be careful about using the terms "validation" and "verification" from software engineering. Forensic techniques should be validated, i.e., shown to be fit for purpose. CFTT avoids both terms and call what they do "tool testing" because there is a subtle distinction between the two activities and what each activity implies. Validation is a demonstration of fitness for purpose, but testing is not necessarily an effective way to demonstrate this. For validation you want to show that the algorithm you pick does what you need to be done. If you are not careful you will just test that the implementation is correct, but you also need to show that you have picked the right algorithm. If I construct test sets of files and try implementations of each algorithm, I might detect flawed implementations of an algorithm, but it is challenging to construct test data to reveal "fitness for purpose." Determination of an algorithm's fitness depends on the properties we want the algorithm to possess not if some implementation is flawed or not.

For example, consider selecting an algorithm for detecting if a digital object has changed (to verify image file integrity) there are several candidates, e.g., CRC16, CRC32, MD4, MD5, SHA-1, SHA-2. The CRC algorithms have been used for decades to check if a block of data has been transmitted without an error and was used in early imaging tools to verify image integrity. The CRC is fit for detecting changes caused by random noise, however a malicious actor can easily modify a file in such a way that the CRC does not change (it is trivial to generate a hash collision). Some additional requirements are needed for a hash algorithm to be fit for purpose:

- Can be computed quickly.
- Collision resistance, i.e., requires an unreasonable amount of computation to find a hash collision.
- Original message cannot be recovered.
- Any change to the original brings about changes in the hash output value.

CRC does not meet all these criteria, MD5 and SHA-1 were considered to meet these criteria until hash collision production algorithms were created. SHA-2 has been validated to meet these requirements and doesn't need to be revalidated.

What CFTT does is not validation testing (fitness of purpose, i.e., building the right tool) but rather verification testing (is the tool built right, doing what the tool maker intended). CFTT tests for correct implementation because it is not always clear what the tool should do. CFTT creates a Requirements Specification but considers it as descriptive of what the available tools have implemented rather than a prescriptive specification of what they should do. Specifying what should be done needs to come from a broad

community consensus.

Some considerations for constructing the CFReDS data sets are the following:

- It is often suggested that "real-world" data sets should be used. This has several advantages:
- The data set is like what the forensic tool would encounter in investigative use.
- The data set includes a large amount of noise, i.e., data that is not relevant to the investigation, that the tool must show that it can deal with.
- The actions of a computer user in the real-world are in a random order and produce a variety of layouts so that the data set may include a situation that would cause the tool to fail.
- A "real-world" data set also has disadvantages:
- Data set ground truth is difficult to determine. The large amount of noise in the data is one factor in the difficulty.
- Significant effort is required to obtain enough data sets so that there is coverage of all features included in the test plan.
- Creating the data set takes significant effort.
- Executing the test plan is time consuming when invoking the tool under test on several large image files.
- The data sets are intended for sharing over the internet and large image files take significant time to download.
- Small data sets seem to be able to address the disadvantages.
- It is easy to create data sets with known ground truth.
- Data set can be focused on the features included in the test plan.
- Not as much effort is required to create the data sets since effort can be focused on creating the features needed for the test plan.
- Small data sets take much less time for a tool to scan and analyze.
- Small data sets are quicker to down load.

The above considerations are just a few of many. Using some real-world data sets might reveal additional anomalies, but the small ones that are in use have been effective at revealing significant anomalies in many tools.

3. Types of test data

This work posits that in general, there are three types of dataset in circulation for use in the field of DF - tool/process evaluation datasets, actions datasets and scenario-based datasets (see Fig. 1). Each dataset type maintains their own challenges in terms of construction and documentation in order to be fit for the purpose for which it is designed, where it is important to note that these challenges must also be considered against the sheer volume of resources and time required to create datasets. Whilst appropriately constructed and content-rich datasets are of value to DF, emphasis must be primarily placed upon the first criteria, where inappropriately designed datasets offer little value to the field, regardless of their content. As a result, this work not only describes the role of each of the three aforementioned dataset-types, but also defines a minimum 'construction standard', to ensure that those creating datasets do so in a way which offers maximum benefit to those utilising this content.

3.1. Tool/process evaluation datasets

Tool/process evaluation datasets are created for the purpose of exhausting the functionality of a specific tool, or part of it (arguably more feasible to achieve), in order to assess whether it is operating correctly (Lyle, 2002). It is important to note that often tool/process evaluation datasets are not designed to assess a tool's general performance, but its performance within the limited set of situations explicitly tested for, where the more independent situations tested, the more confidence in the tool may be implied. Therefore tool/process evaluation datasets must be designed to reveal errors, should they exist where data may be manually constructed in ways designed to engage the perceived limits of function in order to expose incorrect behaviors. In the case of tool-suites, often this type of dataset will be focused upon a specific element of a tool's capability, such as a specific algorithm for parsing or data recovery (see for example approaches taken by the NIST's (2020) CFTT), due to both the degree of difficulty and time needed to create them. Producing tool/process evaluation datasets typically consists of the following three step process:

1. Determining the functionality of a tool to be tested and establishing the scope of its operation. It is important to consult vendor documentation in order to assess 'what' a function is designed to do and the remit of its operation. Doing so prevents the construction of test-cases which sit outside the scope of the tool's functionality, ultimately offering no value in terms of tool-evaluation and may even seek to be of detriment by mistakenly attributing poor-functionality to a tool. Described by Software Testing fundamentals (2020) 'a test case is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly'. This process may be obstructed if vendor documentation is vague with regards to the exact remit of a tool function, leaving those designing the test data to carry out this scoping exercise manually.
2. Design test-cases which are structured in a way which engages the necessary aspects of the functionality subject to testing. In some cases, multiple test-cases will be required to exhaust any given tool functionality, leading to a single dataset containing multiple test cases which may be capable of more comprehensive testing either a single or multiple tool functions.
3. Document the contents of the dataset and its test-cases thoroughly, ensuring that those who utilise the dataset understand how to use it, and ultimately how to interpret any outcomes of its use (see Fig. 2). This facilitates the purpose of verifying software functionality, where this can only be achieved by those testing a specific tool functionality if they know of the exact

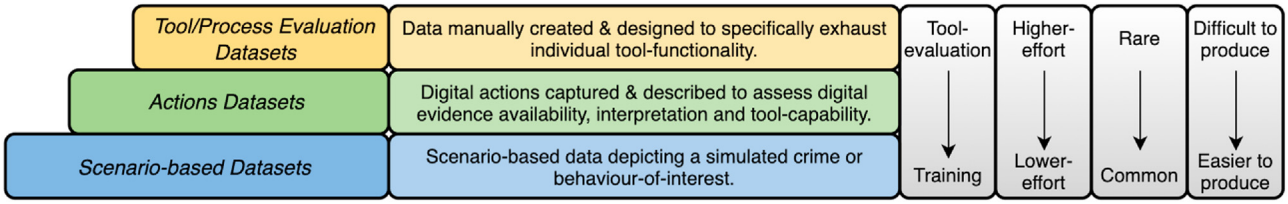

Fig. 1. Tool/process evaluation, actions datasets and scenario-based datasets.

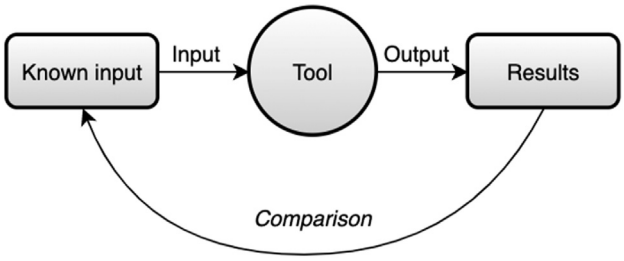

Fig. 2. Tool/process evaluation dataset usage for result comparison.

specification required from a tool 'response' in order to identify whether it is correct or incorrect.

Ideally, tool/process evaluation datasets should not contain any unknown variables or data, and must be fully described to ensure that responses given by a tool under test conditions can be mapped to a known set of variables used to trigger such a response. Any uncertainty within the dataset undermines the validation process, where use of the dataset cannot lead to reliable test results. However, it is also acknowledged that achieving this is difficult, where in some cases it may not be either feasible or possible, therefore guidance regarding the limiting of 'unknown variables' must be to do so as far as practically possible.

It is argued that there are 9 elements which must form part of any tool/process evaluation dataset, drawing reference to the NIST's (2020) CFTT, arguably the currently biggest independent DF tool testing program in operation. Whilst these criteria may seem burdensome, they reflect the role of a tool/process evaluation dataset and the issues which can result from the use of poor test-data in this context.

1. A known, documented structure of the dataset. If the dataset consists of data designed to invoke multiple test-cases, each test-case and its associated data must be described, where the number of test-cases contained in the dataset should be stated. This will often require data mapped at both a physical-level where the offsets of key data structures must be documented, along with the structures themselves, and at a logical-level if file-based content forms part of the dataset. Tool/process evaluation dataset producers should consider the inclusion of a 'table of contents' structure within the dataset itself which contains a formal specification of each test-case content and how to invoke each test-case.
2. A description of the methodology used to create the test dataset if data has been manually created. Where custom built automated processes have been engaged, access to these methods should be provided so that the method itself can be evaluated and implemented in future dataset creation/development. This also allows any challenges to evaluative results to be thoroughly investigated to ensure that errors at the dataset-generation level can be ruled out. Contemporaneous records of the dataset construction should be maintained and distributed with the dataset allowing the dataset creation methodology to be examined and evaluated by peers.
3. A record of the expected 'outputs' for each test-case contained within the dataset - a comprehensive description of what should be returned by a tested function if it is performing 100% correctly. This allows the target software function to be evaluated in terms of meeting the requirements of the test or not and a 'level' of correct functionality to be established as a test may result in partial compliance. Documentation of the expected outputs of a test-case may also influence iterations of future testing of a given functionality.
4. A statement of scope for each individual test-case, ensuring that those using the dataset understand its remit. This prevents any misinterpretation of test results or misuse of the dataset. The need for specific requirements to be defined before testing takes place is suggested by the Forensic Science Regulator in England and Wales (2016).

'Validation is a demonstration of fitness for purpose. The first step is to define what the requirements are in terms of inputs, effects, constraints and desired outputs. Validations that skip this step may miss the key quality issues. Unfocused testing can lead to amassing of data that may or may not increase understanding or give confidence in the method' (Forensic Science Regulator, 2016).

Whilst defining the scope of test-cases does not prevent the dataset from being used in other test-capacities, it does ensure that any results given from testing outside of a test-cases remit from being incorrectly relied upon.

5. A description of which tool functionality is intended to be engaged by each of the test-cases in the dataset. Generally, there are three categories:
   a. Algorithm testing associated with a specific process.
   b. Testing of the tool's ability to correctly display information.
   c. Testing of the tool's configuration and navigation.

This include both a generic description of the function (for example, 'file carving algorithm') and if test data is targeted towards a specific software-type, the tool's function should also be named (for example, the name of the function which may be vendor-specific and importantly the configuration and settings to be used). If a test case is designed to test the function of a tool which requires a specific configuration to be defined prior to testing, then these settings must be explained. This prevents a correctly performing tool from failing a test-case due to a misconfiguration.

6. A duplication requirement. In order to assess consistency regarding test performance, a tool/process evaluation dataset should contain data which can assess a tool's performance in multiple instances for the same functionality. This helps to distinguish tool performance which is compliant through luck, and tools which provide consistently compliant results which can be relied upon. Dataset creators must document how duplicate tests have been implemented within a dataset, and how duplication is evidenced. It is important to stress that duplication does not mean creating multiple identical tests, but that the function tested is tested in a way which engages the function in the same fundamental way.
7. File lists (logical content):- Where test-cases within a dataset contain file-structures which form part of any functionality test, accompanying documentation should record at a minimum:
   a. The number of relevant files included in any test-case data.
   b. The physical and logical location of each file.
   c. Each file's size.
   d. Each file's type (with accompanying signature if the file is non-standard).
   e. A description of each file's content. A logical file extraction is recommended, providing a copy of each file as a reference.
   f. Each file's hash value using an appropriate hashing algorithm.
   g. All available metadata regarding the file including filename, and available filesystem metadata. File metadata

should be captured as part of a logical extraction noted above in point 'e'.

8. Data descriptions (physical content):- Where test-cases contain non-file-based structures (for example, string information used for keyword search algorithm testing) the dataset should maintain accompanying documentation which records at a minimum:
   a. The physical location of data.
   b. The type of data encoding (American Standard Code for Information Interchange (ASCII), Unicode etc.).
   c. The size of the data (string length etc.).
9. A dataset integrity mechanism (hash) which confirms the integrity of a dataset or highlights tampering.
10. Details of an appropriate evaluation metric to be used when interpreting any results obtained whilst using the dataset for tool testing.

3.1.1. Test-case annotation

Where a dataset contains multiple test-cases, documentation describing the dataset content should maintain annotations for each test-case, a common practice for software testing (Bentley et al., 2005). Whilst multiple generic test-case templates exist online which can be adopted for the purposes of DF test-case, this work suggests any test-case annotation should at a minimum contain details regarding the following six areas:

1. A test-case identifier which allows the test-case to be directly referenced (discussed below in Section 3.1.2).
2. The location and dimension of test-case data in the dataset. In regards to complex tool/process evaluation datasets, compartmentalising test-cases particularly where the dataset takes the form of a binary image is a difficult task. Therefore, it is important to identify what region or content of a dataset forms a specific test-case to prevent 'test-case bleed' (where data from separate test cases may become indistinguishable from one another making it difficult to determine digital 'cause and effect' under test conditions).
3. A description of the:
   a. Name of the tool and function tested.
   b. Type of functionality tested - algorithm, navigation or display of information.
   c. The purpose of the test-case (e.g., to determine function 'X's' ability to recover files of type 'Y' from unallocated space).
4. How any function tested must be configured prior to running the test-case. Where a test case is specific to or aimed at a single tool vendor, explicit instructions regarding tool-configuration should be defined, including the use of the language used by the vendor in relation to their tool's setup, menu labels and functionality names.
5. Directions for invoking the test-case, in essence, those test-actions which must be carried out in order to engage the test case. This point is a natural follow-on from point four above, where once configured, those carrying out testing must then correctly engage the function to be tested.
6. A complete set of expected results, allowing post-test results to be benchmarked and evaluated.

3.1.2. Dataset chain of custody

Given the importance of tool evaluation in DF, there is a need for the creation and maintenance of a chain of custody with regards to the datasets used as part of evaluative testing. Doing so supports the reliability of any subsequently claimed test results as these can be attributed to a dataset used as part of testing. It also promotes the circulation of identifiable and traceable datasets in the DF field. To achieve this, it is argued that there is a need for a formalised method for tagging datasets which allow both the dataset itself, and all test cases it contains, to be identified. Currently datasets are typically identified by their host destination or creator, where if datasets become detached or if local copies are made, attribution may be difficult. To combat this, the need for an embedded dataset identifier is offered.

A dataset Identifier: At any one given period in time, multiple different datasets may exist which each contain data that can be used to test the same function of a DF tool - determining which dataset has been used for any given testing carried out is important for the purposes of reliability and transparency. Given that test dataset creation is unregulated, where contributions are made by individual practitioners, academics and government organisations it is arguably important that the DF field moves towards the use of mechanisms to identify, track and trace test datasets and their use in any formal evaluative studies. Ensuring that test datasets maintain an 'identifier', and that such an identifier remains with the dataset for the duration of its existence allows those carrying out evaluative testing to report their results in a way which is transparent to those in DF. To support this, a dataset identifier should be centrally referenced to ensure that datasets can be monitored and accessed by all in the field (see Fig. 3). This also allows any subsequent issues with any evaluative testing which has used a specific dataset to be evaluated in greater detail allowing dataset content to be evaluated and if necessary warnings be attached to a dataset if known issues exist. As the field seeks to increase the rigour of evaluative methods used upon its tools, it is argued that this should start with ensuring greater control over the datasets in circulation.

The ability to identify a dataset used during testing also supports the 'transparency of testing' process as it allows individuals and tool vendors to first, describe the data they have used to evaluate their tool (or tool-functionality), and second, it allows third parties to evaluate the test process by being able to identify the test dataset and repeat the test.

1. A unique ID (<creator initials>_<date of creation>_<dataset name>) which can be used to reference the dataset.
2. The name of the dataset creator(s).
3. Contact details of the dataset creator(s).
4. Primary hosted location of the dataset.
5. Dataset information - points 1 to 9 noted in Section 3.1.

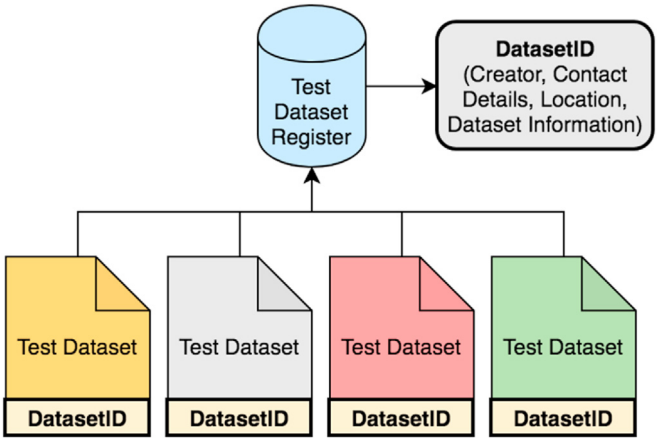

Fig. 3. Dataset Identifiers. It is proposed that a dataset identifier should contain the following metadata:

a. A test-case identifier:- As a dataset may contain multiple test-cases, each test case should be identifiable and formally referenceable. It is proposed that as part of the dataset information each test-case maintains a reference tag - < creator initials>_<date of creation>_<dataset name>_<*Numeric TestCaseID>).

A central dataset register: The maintenance of a 'central dataset register' where dataset creators can record the dataset identifier of their dataset is argued beneficial for dataset governance. Whilst the dataset register primarily contains dataset metadata, supporting those in DF to access appropriate datasets, it should also serve as a historical record of validation usage, noting those who have implemented successful evaluative tests with particular datasets. Importantly, the register should also record concerns raised regarding datasets and fully document any issues which are known to exist with any given dataset.

3.1.3. Being realistic

Testing DF tools, as with any software testing in general, provides for a difficult process. The issue lies with the very nature of the task and the fact that any number of potential test case scenarios may exist, each with the potential to present an error. In reality, it is doubtful as to whether a sufficient number of test cases can be designed, implemented and analysed in order to fully test any given software (Meyer, 2008), even more unlikely in the case of large DF tool-suites. As a result, Kumar and Syed, (2010, p.52) note that when those evaluating software should 'perform tests efficiently and effectively, within budgetary and scheduling limitations'. Perhaps the key takeaway point lies with the need to be realistic when testing in the DF field, where it is arguably better to ascertain and test core functionality with sufficient robustness as opposed to attempting to tackle a tool's entire functionality and doing this to a sub-standard. This is acknowledged by Mil and Tchier (2015, p.141) who state the following.

Given an input space S (which is assumed to be so large that it is impractical to test the program on all elements of S), choose a small subset T of S such that we can achieve the goal of the test by executing the candidate program on T rather than on S . Clearly, the requirement that T must satisfy depends on the goal of the test. Mil and Tchier (2015, p.141)

To that end, there are some points which have to be accepted by the field when venturing into tool testing.

1. It is unlikely that any DF tool can be tested and evaluated in its entirety, so it is important that practitioners do not blindly set about this task. Complete evaluation should be seen as a goal, but considering an already limited set of resources for evaluation, strategic testing should be implemented.
2. Arguably, datasets should be created for testing focusing on high-risk and high-usage functions. Doing so has the greatest potential for wider quality assurance to be achieved in the short-term. This position reflects the difficulties associated with dataset creation and the time and effort needed to complete this task correctly. Those creating datasets are faced with having to decide which functions of a tool to focus a test, leaving others untested (where in the case of off-the-shelf tools, relying on the vendor to have tested the tool). This situation is far from ideal, but a current representation of the DF field's position.
3. The use of datasets and correctly implemented testing increase confidence in DF tools, but they cannot guarantee functionality 'correctness' in a lot of cases. It is unlikely that any dataset can completely exhaust a tool's functionality to the point of allowing an end-user to establish that a tool is functioning 100% correctly.
4. Tools which have not had their functionality robustly tested through the use of an independently and correctly created dataset do, and, are likely to remain in circulation for the foreseeable future.

Regardless of these issues, the DF field must look towards tool/process evaluation dataset creation, when done correctly, as a mention of ascertaining and maintaining the reliability of digital evidence produced by its tools. Examples of tool/process evaluation dataset attempts include those by Carrier (2010).

3.2. Actions datasets

'Actions datasets' consist of scripted digital behaviours which are then acted-out on a device, seeding test data (if data regarding a specific function is actually maintained by the device/software being tested). Data from this device is then subsequently extracted (either partially or fully) and made available as a dataset (coined here as an 'actions dataset'). When actions datasets are examined, any outputs/discovered data must be evaluated against the actions which were carried out during the dataset construction process, making reference to the 'script of digital actions' which should have been contemporaneously maintained and made available by the dataset creator (see Fig. 4). There are two fundamental components of actions datasets, a 'script of digital actions' which describes what the dataset creator has done in terms of digital-interactions on a device, and the data itself created by those digital actions - which is extracted to form the dataset.

Actions datasets usually serve three purposes:

1. To assess data availability in a given scenario: In terms of data availability, actions datasets contain any digital traces left behind following a series of actions on a device. This may involve interactions with a specific piece of software or application, or general device usage leading to operating system traces. Therefore a test creator may seek to script actions for a specific piece of software or application which may replicate potential suspect actions which may be assessed as likely to occur if a suspect has engaged with such software/application (consider a grooming case, where a suspect may have utilised communication App 'X'. Understanding the functionality of 'X' when used on a device is core to being able to then interpret digital traces of 'X's' usage in a real forensic case). Actions datasets allow practitioners to ascertain what digital traces

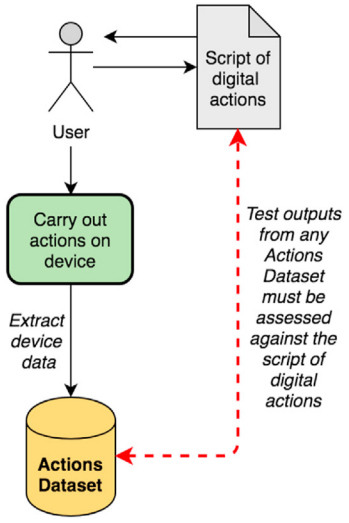

Fig. 4. Actions dataset creation process.

occur following specific actions and usage in a given scenario involving a piece of software, application or device, where this data occurs and what form this data may take, helping to reverse-engineer the digital behaviour of any specific software/application/device at a data-level.
2. Evidence interpretation: Actions datasets content combined with the script of digital actions support practitioners to interpret digital evidence traits, helping to tie digital traits to digital actions. Correlations between scripted actions and resident data can be used to determine how a specific piece of software or system behaves following certain actions and to support practitioners to interpret trace-data.
3. Tool capability, not validation: Actions datasets allow for an assessment of a tool or function's capability to access, parse and display data. Whilst actions datasets can support tool/function validation, the detail required in the description of data needed to achieve this, means it is unlikely. This can be seen due to the differences in requirements for tool/process evaluation dataset documentation discussed in Section 3.1.1 and the minimum requirements for actions datasets discussed below in Section 3.2.1 where the burden is less. Yet, actions datasets will serve a partial purpose in regards to determining a tool/functions ability to access and process data, but will not validate it's correctness in doing so.

It is important to stress that the value of actions datasets lies with the script of digital actions describing the actions of the dataset creator. Datasets which do not have a script of digital actions (or maintain a script which is poorly constructed, in terms of limited information describing the actions) are of little value to any user of the dataset as they cannot evaluate or interpret any outputs generated by any tool/function with any large degree of confidence. Examples of actions datasets can be seen at the Digital Corpora (Garfinkel, 2020).

Arguably, the value of actions datasets lies less with their use in tool-testing and more with their ability to help practitioners to understand how specific software behaves when it is interacted with through data reverse engineering.

3.2.1. Script of digital actions

The script of digital actions is a key component of actions datasets as it describes any potential test cases which are present within the dataset. Any specific actions dataset may contain digital interactions with a number of software/applications, for example consider a test mobile device where multiple applications have been installed and interacted with to create one actions dataset, essentially containing test-cases for each application used.

The script is an important validator for the dataset as, without it, the dataset has limited value and meaning to those who use it as they are unable to attribute what actions caused the presence of digital traces in the dataset and ultimately interpret these. When developing a script of digital actions, consideration must be given to the purpose that the dataset intends to serve and in particular, the target application/software which is being interacted with. If the goal of the actions dataset is to inspect the functionality of a single piece of software or single application, then the dataset must be both comprehensively documented and exhaust all relevant functions of the software. To do this, the actions which are used to generate the dataset must be wide-ranging enough to engage all of the relevant functions of the software/application.

Exhaustive data: In relation to the term exhaustive, this means that the data contained in the dataset should aim to engage all relevant functions of the software it is targeting. Ensuring the dataset is exhaustive first involves establishing what a target software/application is capable of. This means determining what the purpose of an application is (which in most cases, this will be obvious), but also all of the sub-functions it is capable of in terms of achieving its goal. For example, a communication application may allow communication in many media, including text, audio and video, but also contain valuable metadata regarding any account(s) that is synched with the application. Achieving an exhaustive dataset requires mapping what an application/piece of software is capable of and establishing all of its functions. It is suggested that a functionality map denoting the full capability of any software/application should be made (see Fig. 5 for an example of a functionality map of the Instagram application).

Following the establishment of a functionality map, there is a need to determine those actions required to engage all of the functions of the application/software and ensure that sufficient test actions have been carried out to engage all functions. Functionality mapping helps to establish which parts of an application/software's functionality have been engaged with and therefore form part of an actions dataset. Importantly, it also helps to determine what functions have not been engaged by digital actions, preventing any misbelief that digital traces are not left following specific usage, if the dataset creator has never actually engaged that particular function.

Comprehensive documentation: Actions datasets require comprehensive documentation describing the setup and interaction with any software/application to allow a user of the dataset to be able to interpret any results. This is the script of digital actions documentation and at a minimum, the following is required:

1. A dataset identifier (see discussion in Section 3.1.2).
2. Name and version number of the application/software being interacted with for the purpose of the dataset's creation. Where multiple applications/software have been included in the dataset a list of this information is required. In addition, the software/application creator's information should be captured along with where the software/application can be (and has been) acquired from.
3. Description of the physical device and setup where the dataset comes from, including device name, make and model.
4. Description of the extraction method and software used to create the dataset including version numbers and any bespoke protocols implemented (consider jailbreaking/rooting in mobile devices etc.). This is important in cases where a dataset originates from a mobile device where multiple extraction types and tools exist which can impact the amount of data available in a subsequent dataset.
5. Account information and credentials used with the application/software if this content is required to use it. Attention should be paid to each service provider's terms and conditions for acceptable use. Account information and credentials should be unique for each service, and identifiable and attributable to only one service to avoid confusion when interpreting digital traces.
6. Any test data used in the application/software (files or string data) must be recorded. This content must be both unique (attributable to the user, and nothing else) so that it can be distinguished during any testing, and contemporaneously recorded. In the case of files used, copies should be maintained separately and hash values of files should be taken and recorded prior to the file entering the test setup to ensure that it can be identified in any subsequent dataset usage and that any changes to the file by the application/software can be determined.
7. All test actions carried out must be contemporaneously recorded. This includes the order of key press and names of menus/functions engaged with as shown to the user by the application/software. Whilst textual methods may be favoured, consideration should also be given to the use of external screen

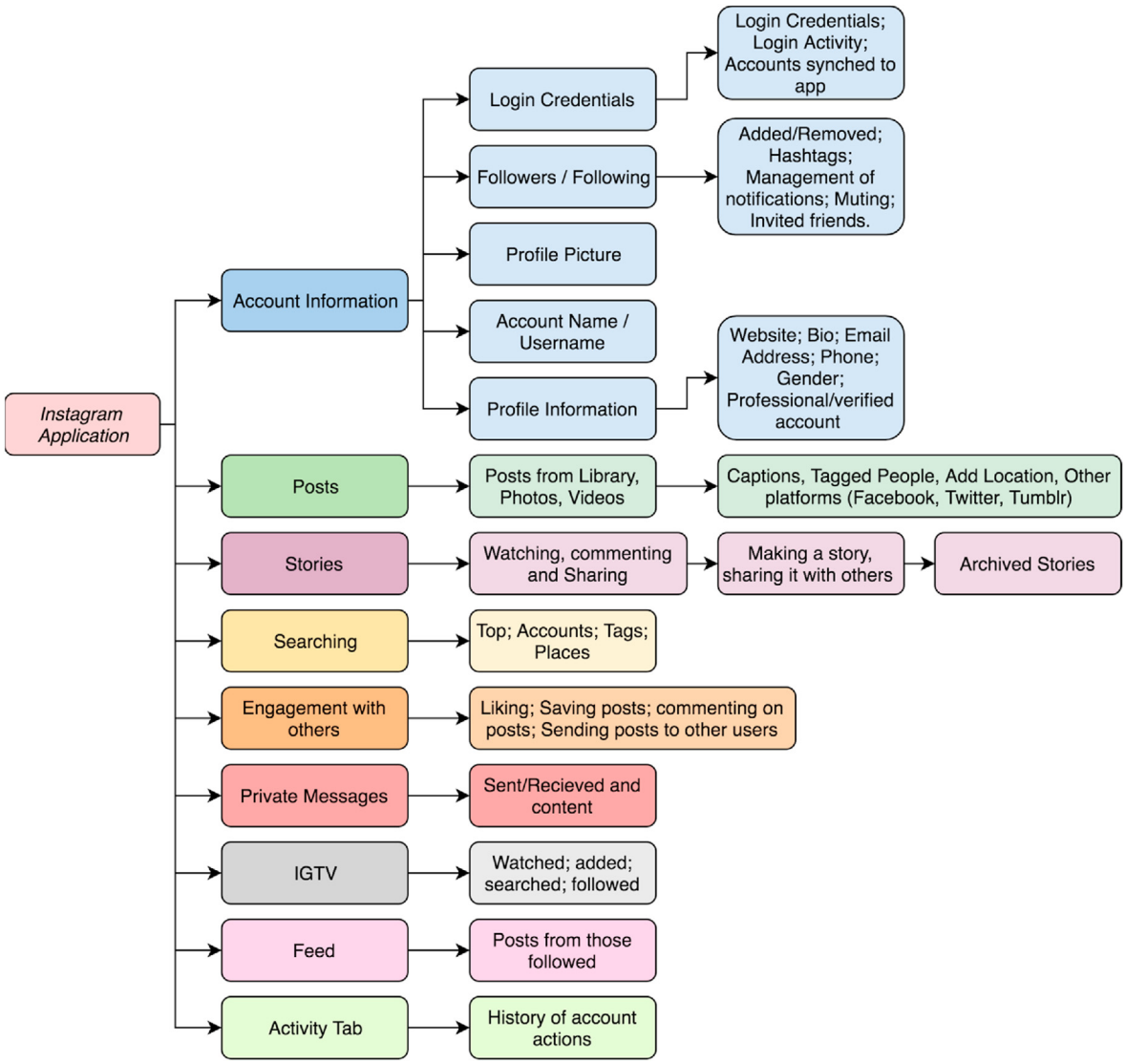

Fig. 5. An example functionality map of the Instagram application.

recordings (camera on tripod etc.) which require less interaction and effort by the dataset creator.

To note - Each function should be engaged multiple times in an effort to duplicate digital traces following engagement with an application/software functionality to allow for testing to assess the consistency of behaviour. Each engagement of a function should be achieved with distinguishable data in order to separate iterations of the test.

8. Time and date information associated with every action carried out should be recorded. In some instances, time and date information may be a priority method of correlating test actions with digital traces left by the application/software. If screen recording approaches are taken, consider the use of a separate and visible clock (accurately configured with a noted time zone) next to a device so captured records continuously show the time and date of actions. Where text-based contemporaneous records are kept, consider the use of software which can time-stamp created notes (with time zone information).

The value of actions datasets hinges on the accompanying script of digital actions and arguably this is where the main effort exists. The creation time and effort for actions datasets is arguably less than tool/process evaluation datasets as the data within the dataset is not validated in terms of presence and structure - this is a task which forms part of the purpose of the datasets creation. Instead, creators must concentrate on ensuring any test actions used to create the dataset are exhaustive and methodical whilst recorded in sufficient detail.

3.2.2. Be aware

Creating actions datasets is not a straightforward process where the creator is not always guaranteed to end up with a dataset which they expected to create. There are issues which exist when creating actions datasets and the following serve as points of caution for dataset creators to be aware of:

1. Are digital traits left in the dataset?: Given that actions dataset involve the use of scripted interactions in order to engage a software/application's functions which may potentially leave behind a digital trace on a device, it is not always possible to determine if an action has actually left a trace which is then captured in the extracted dataset. Consider the creation of an

actions dataset which focuses on producing data regarding a new mobile application ('X') and its functionality.

Even where all functions of X are interacted with, there is no guarantee that all actions result in a trace being left (for example, some functions may interact with data stored server-side, which never reaches the physical device's storage media). Emphasis here is placed upon the maintenance of a detailed script of digital actions and repeated tests, the likes of which may demonstrate the consistent absence of a specific digital trace-type. However, added complexity is introduced if the method for extracting the dataset does not capture all available data. Devices which only permit limited data extractions (for example, mobile devices extracted logically), create uncertainty as to whether a test action has left a trace on a device, particularly if it is not captured within the extracted dataset, as it may reside on a non-extracted region of a device. This scenario is difficult to address, therefore datasets created through limited extraction methods should be treated with caution. In some cases this may simply be unavoidable as an extraction method may be limited, but still be state-of-the-art for a given test device. The result of which is always likely to mean a level of ambiguity will exist for those using the dataset to reverse engineer the data left by the application's use.

2. Compartmentalising test cases: Due to the nature of the way in which this dataset-type is created it is difficult to compartmentalise test-cases within the dataset. As a result, it may be challenging to determine which user actions should be attributed to a specific digital trace, emphasising the need for contemporaneous records of user actions to be kept, complete with timestamp information denoting when actions occurred, as time-comparisons may be a key method of attribution. The use of unique and attributable test data where possible (for example, instant messages sent use identifiable unique strings) as part of any given test actions is also important.

3.3. Scenario-based datasets

Scenario-based datasets are often generated to simulate a particular offence, offence-setup, or potential investigative narrative. These datasets aim to immerse the practitioner in a realistic case-work experience and may be designed as training material both for the practitioner's personal development or as part of specific training for a vendor tool or training course. Scenario-based datasets are unlikely to focus on exhausting a specific application/software's functionality or evaluating a tool, but to imitate real-world use of a device within the context of a given offence. Therefore, datasets of this type are likely to have a variety of application, software or system usage which each alone is not in exhaustive depth (in terms of the software's functionality), but at a surface level of interaction detail. However, where a scenario-based dataset attempts to incorporate, for example, intentions of tool evaluation, then those requirements noted in Section 3.1 must be adhered to. This would essentially lead to the creation of a hybrid scenario-based tool/process evaluation datasets, arguably requiring significant effort to create. As a result, we distinguish the two dataset-types and assume that most scenario-based dataset creators will focus on the realism and scenario narrative.

One of the greatest challenges here lies with the scale of dataset creation, as often creating a scenario requires interacting with a range of system functionalities in order to replicate real usage. As a result, effort is likely spent in developing a realistic plot which is representative of the types of digital data a practitioner is likely to encounter. This may be described in some cases as a 'long and thin' approach, where dataset actions are created across a range of applications/software but no actions focus on fully exhausting the entire functionality of any given application/software being interacted with.

Whilst arguably scenario-based datasets are the easiest to 'technically produce' there are still requirements which must be met in regards to dataset production if they are to serve their purpose. As a minimum, the following information is suggested as a requirement.

1. A dataset identifier (see discussion in Section 3.1.2).
2. A scenario-brief which outlines the scope of the scenario-based dataset and any offence which it attempts to replicate. In the case of criminal offence replication, the brief should as a minimum outline the legal elements of the replicated offence in relation to the jurisdiction in which it was created/aimed at, allowing a dataset user to develop an investigation strategy. This is important as datasets may be used internationally, and therefore what may be classed as an evidential trait within the remit of a scenario may differ between geographical regions. Supplying contextual information/background intelligence regarding the scenario may also be considered as often scenario-based datasets are designed to test a user's forensic inference of digital traces, a task not possible without scenario-context.
3. A description of the technical setup utilised to create the scenario (software/hardware utilised etc.) and any key configurations deployed which may be important for the practitioner to know whilst conducting their examination and interpretation of the scenario.
4. An acknowledgement of which stages of the DF investigative process the scenario is designed to engage (i.e. acquisition, screening, interpretation etc.)
5. A list of all actions carried out on the device/system where the dataset originates from. These actions should have time and dates of their occurrence recorded.
6. All files 'expected' to be in the dataset as a result of any test actions. It is assumed in many cases, a complete list of 'evidence' may not be deemed possible as datasets of this type may be too wide in scope to contemporaneously record everything.

Where time and resources permit, a scenario-based dataset should look to expand documentation to include:

1. A list of all files of 'evidential value' included in the dataset in relation to the scenario-brief, including:
   a. File location (or physical offset for unallocated data).
   b. File names.
   c. File content description or retain a copy of the file.
   d. File hash value.
2. An interpretation of what each file/piece of data means in relation to the given scenario-brief, supporting scenario reconstruction whilst allowing a dataset user to evaluate their performance and evidence interpretation attempts.

*To note:- It should be noted that we are not intending to underestimate the effort required to generate high-quality scenario-based datasets, but it is unlikely that creators will invest in the level of documentation required for tool/process evaluation datasets when operating at a scenario level. Instead, we expect focus to be maintained on attempting to achieve scenario-realism as opposed to the fine-grained mapping of dataset contents and metadata. Yet, obviously where a creator chooses to do so, the time, resources and effort needed to achieve this will be vast.

4. Some challenges

Whilst section 3 has set out three different types of dataset for use in DF, it is also necessary to outline some of the wider challenges which exist around dataset creation, use and engagement. As a starting point, dataset realism will be considered, where datasets must not just ensure that they contain the correct structure and type or information, but also considerations of 'transferability into the real world' must be given. Here we refer to the need to ensure that datasets are not just created properly but in some cases it is also necessary that they are reflective of those scenarios which are likely to be encountered in real investigative scenarios, so far as is feasible to do so. This means that when seeding data, this process should be robust enough to ensure a suitable diversity and depth of data is achieved. Concerns over dataset scalability for testing purposes may be raised and therefore depending on the function and use of any dataset, this should be considered an important factor when creating it (Garfinkel, 2010). Datasets which are not representative of the real world in terms of data volume and type may not be suitable for purposes such as tool evaluation as they may fail to expose the tool to enough relevant data-scenarios in order to evaluate its performance. Further, where scenario-based datasets are being generated, the dataset creator should consider how 'realism' is to be achieved with regards to the data contained in the dataset, where data seeded for example over the course of a few days is unlikely to reflect the true complexity and volume of data which is apparent in real world cases. Conversely, creating a data-rich dataset with months of activity is a burdensome task which requires prior planning and the dedication of resources to carry out this task - in some cases this will be unachievable. The generation of scalable, realistic datasets is a difficult task, however one which should be aimed for so far as practically possible.

In addition, the creation and sharing of datasets also poses a challenge due to issues surrounding the sharing of data that may be subject to legal restriction or, data governance and protection. In some cases this concern may be known in regards to dataset data, preventing a breach from occurring, but in others accidental data-sharing in breach of regulations may occur if the creator is not aware of their obligations in regards to the dataset content. This is a real risk which must be considered by the dataset creator and they must evaluate the contents of their dataset and any restrictions which may exist regarding the distribution of data within it. In some cases it may be possible to place appropriate caveats on dataset content, or restrict access to it (particularly if the data is restricted/confidential), however in many cases, dataset creators should consider the implications of placing any data within their dataset prior to doing so. This issue, when combined with any attempts to instil realism in a dataset may be difficult to overcome where appropriate legal advice with regards to data usage is advisable.

Encouraging engagement with dataset creation and sharing also required within the DF field. While dataset creation can be a resource intensive task, those in DF should be encouraged to think of datasets as an important resource, and where feasible to do so dataset creation should be undertaken in conjunction with any research or testing conducted. Arguably the field of DF cannot have too many datasets - providing they are constructed effectively and therefore the more of these resources which are available, the more benefit which can arguably be gained by those seeking to test and evaluate processes and procedures. Those conducting research should consider the production of a dataset as part of their exploratory work a natural part of their research and development process. The creation and dissemination of good datasets in DF benefits all of those operating within it, arguably increasing the ability of the field to develop quality control and assurance measures through testing and evaluation.

5. Final thoughts

This work has identified and defined the three categories of dataset which typically exist in digital forensics - tool/process evaluation datasets, actions datasets and scenario-based datasets. With DF narratives increasingly encouraging testing and validation (Marshall and Paige, 2018; Casey, 2019; Horsman, 2019b; Page et al., 2019; Hughes and Karabiyik, 2020), whether it be methods, tools or the practitioner ability, doing this requires fit-for-purpose datasets. To support the development of such datasets, we define a set of minimum requirements for each dataset type in the hope of supporting dataset creators.

It is important to note whilst three dataset-types have been defined, we do not wholly discourage dataset sets which do not fit exactly into these categories, only express the need for caution. To explain, consider where an unexpected opportunity occurs for a practitioner to capture data depicting an event or scenario which they encounter, but have not had the opportunity to document it fully or control the condition by which the data was created. This data may be considered an informal type of dataset and 'opportunistic data', where its accompanying description may be minimal. It is not suggested that such a dataset should never be shared, there is some value in terms of initially 'taking a look' at the data and the believed context from where the dataset was taken. This may be informative in nature and therefore influence further formal dataset creation. What is important is that such datasets are acknowledged as lacking in formal construction and documentation and therefore those that encounter them must be aware of the limitation of their use.

References

Al-Kawaz, H., Clarke, N., Furnell, S., Li, F., Alruban, A., 2018. June. Advanced facial recognition for digital forensics. Proceedings of the 17th European Conference on Information Warfare and Security. ECCWS, pp. 11–19.

Al-Sanjary, O.I., Ahmed, A.A., Sulong, G., 2016. Development of a video tampering dataset for forensic investigation. Forensic Sci. Int. 266, 565–572.

Anda, F., Lillis, D., Le-Khac, N.A., Scanlon, M., 2018. May. Evaluating automated facial age estimation techniques for digital forensics. 2018 IEEE Security and Privacy Workshops (SPW). IEEE, pp. 129–139.

Arshad, H., Jantan, A.B., Abiodun, O.I., 2018. Digital forensics: review of issues in scientific validation of digital evidence. J. Inf. Process. Syst. 14 (2).

Bentley, J.E., Bank, W., Charlotte, N.C., 2005. April. Software testing fundamentals—concepts, roles, and terminology. Proceedings of SAS Conference, pp. 1–12.

Biros, D.P., Weiser, M., Witfield, J., 2007. March. Managing digital forensic knowledge an applied approach. Australian Digital Forensics Conference, p. 11.

Bruschi, D., Monga, M., Martignoni, L., 2004. August. How to reuse knowledge about forensic investigations. Digital Forensics Research Workshop. Linthicum, Maryland.

Carrier, Brian, 2010. Digital Forensics Tool Testing Images. Available at:. Accessed. http://dftt.sourceforge.net/. (Accessed 6 August 2020).

Casey, E., 2019. The chequered past and risky future of digital forensics. Aust. J. Forensic Sci. 51 (6), 649–664.

Dang-Nguyen, D.T., Pasquini, C., Conotter, V., Boato, G., 2015. March. Raise: a raw images dataset for digital image forensics. Proceedings of the 6th ACM Multimedia Systems Conference, pp. 219–224.

Forensic Science Regulator, 2016. Guidance method validation in digital forensics FSR-G-218. Available at: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/528123/FSR_Method_Validation_in_Digital_Forensics_FSR-G-218_Issue_1.pdf. (Accessed 26 July 2020).

Garfinkel, S.L., 2010. Digital forensics research: the next 10 years. Digit. Invest. 7, S64–S73.

Garfinkel, S., 2012. Lessons learned writing digital forensics tools and managing a 30TB digital evidence corpus. Digit. Invest. 9, S80–S89.

Garfinkel, Simson., 2020. 'Cell Phones' Available at:Accessed. https://digitalcorpora.org/corpora/cell-phones. (Accessed 25 July 2020).

Garfinkel, S., Farrell, P., Roussev, V., Dinolt, G., 2009. Bringing science to digital forensics with standardized forensic corpora. Digit. Invest. 6, S2–S11.

Grajeda, C., Breitinger, F., Baggili, I., 2017. Availability of datasets for digital

forensics—And what is missing. Digit. Invest. 22, S94—S105.
Horsman, G., 2018. "I couldn't find it your honour, it mustn't be there!"—Tool errors, tool limitations and user error in digital forensics. Sci. Justice 58 (6), 433—440.
Horsman, G., 2019. Raiders of the lost artefacts: championing the need for digital forensics research. Forensic Sci. Int.: Reports 1, 100003.
Horsman, G., 2019b. Tool testing and reliability issues in the field of digital forensics. Digit. Invest. 28, 163—175.
Horsman, G., Laing, C., Vickers, P., 2014. A case-based reasoning method for locating evidence during digital forensic device triage. Decis. Support Syst. 61, 69—78.
Huang, J., Yasinsac, A., Hayes, P.J., 2010. May. Knowledge sharing and reuse in digital forensics. 2010 Fifth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering. IEEE, pp. 73—78.
Hughes, N., Karabiyik, U., 2020. Towards Reliable Digital Forensics Investigations through Measurement Science. Wiley Interdisciplinary Reviews: Forensic Science, p. e1367.
Kahvedžić, D., Kechadi, T., 2009. DIALOG: a framework for modeling, analysis and reuse of digital forensic knowledge. Digit. Invest. 6, S23—S33.
Koroniotis, N., Moustafa, N., Sitnikova, E., Turnbull, B., 2019. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: bot-iot dataset. Future Generat. Comput. Syst. 100, 779—796.
Kumar, P., Syed, K., 2010. Software testing—goals, principles, and limitations. https://www.ijesat.org/. Volumes/2011_Vol_01_Iss_01/IJESAT_2011_01_01_, 12.
Luciano, L., Baggili, I., Topor, M., Casey, P., Breitinger, F., 2018. August. Digital forensics in the next five years. Proceedings of the 13th International Conference on Availability. Reliability and Security, pp. 1—14.
Lyle, J.R., 2002. NIST CFTT: testing disk imaging tools. Proceedings of Second Digital Forensic Research Workshop, 2002.
Marshall, A.M., Paige, R., 2018. Requirements in digital forensics method definition: observations from a UK study. Digit. Invest. 27, 23—29.
Meyer, B., 2008. Seven principles of software testing. Computer 41 (8), 99—101.
Mili, A., Tchier, F., 2015. Software Testing: Concepts and Operations. John Wiley & Sons.
Nance, K., Nestler, V., Bishop, M., 2019. Use my digital forensics tool... It's shiny! J. Int. Technol.Inf. Manag. 28 (3), 91—100.
National Institute of Standards and Technology, 2019. The CFReDS Project. Available at. Accessed. https://www.cfreds.nist.gov/. (Accessed 25 July 2020).
National Institute of Standards and Technology, 2020. Computer forensics tool testing program (CFTT). Available at, Accessed. https://www.nist.gov/itl/ssd/software-quality-group/computer-forensics-tool-testing-program-cftt. (Accessed 25 July 2020).
Page, H., Horsman, G., Sarna, A., Foster, J., 2019. A review of quality procedures in the UK forensic sciences: what can the field of digital forensics learn? Sci. Justice 59 (1), 83—92.
Park, J., 2018. TREDE and VMPOP: cultivating multi-purpose datasets for digital forensics—A Windows registry corpus as an example. Digit. Invest. 26, 3—18.
Shullani, D., Fontani, M., Iuliani, M., Al Shaya, O., Piva, A., 2017. VISION: a video and image dataset for source identification. EURASIP J. Inf. Secur. (1), 15, 2017.
Software Testing fundamentals, 2020. 'Test Case' Available at:, Accessed: http://softwaretestingfundamentals.com/test-case/. (Accessed 25 July 2020).
Tulley, Gillian, 2020. Codes of practice and conduct for forensic science providers and practitioners in the criminal Justice system. Available at:Accessed: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/880708/Codes_of_Practice_and_Conduct_-_Issue_5.pdf. (Accessed 26 July 2020).