# Uncertainty in Live Forensics

Antonio Savoldi, Paolo Gubian, Isao Echizen

Chapter 12

# UNCERTAINTY IN LIVE FORENSICS

Antonio Savoldi, Paolo Gubian and Isao Echizen

**Abstract** The goal of live digital forensics is to collect crucial evidence that cannot be acquired under the well-known paradigm of post-mortem analysis. Volatile information in computer memory is ephemeral by definition and can be altered as a consequence of the live forensic approach. Every running tool on an investigated system leaves artifacts and changes the system state. This paper focuses on the understanding and measurement of the uncertainty related to the important and emerging paradigm of live forensic investigations. It also presents some practical examples related to the evaluation of uncertainty.

**Keywords:** Live forensics, collection tools, measurement of uncertainty

## 1. Introduction

When performing live forensics, a practitioner is expected to run tools, toolkits and/or custom live distributions on a powered computer system. This is especially the case when a timely triage or mission-critical analysis is required [19]. Indeed, apart from the well-known post-mortem paradigm, which has been used for many years, live forensics is emerging as the new standard for dealing with large, mission-critical computer systems [19].

The goal of live forensics is to collect evidentiary data from computer memory in order to define the state of the computer system at the time of an incident. Naturally, the act of collecting data from a live system causes changes to the volatile memory [10]; consequently, it is important that the forensic practitioner documents and explains the possible artifacts in the resulting digital evidence. For example, running a popular tool such as `dd` [5] from a removable media device alters volatile data as soon as the program is loaded in main memory. Another example is the Helix tool [4], which may create or modify files and registry entries on

the system being investigated. Similarly, a remote forensic tool necessarily establishes a network connection, executes instructions in memory and makes other alterations to the system.

From a forensic standpoint, the acquisition process should modify the original system memory state as little as possible, and all changes should be documented and assessed in the context of the final analytical results. However, it is controversial to measure how much of the volatile memory is modified by a collection tool whose goal is to acquire the full volatile memory content. Moreover, it is difficult (if not impossible) to establish the extent of the "perturbation" caused by a running process on volatile memory. As a consequence, it is necessary to design a sound and reliable methodology that measures the volatile memory changes caused by a forensic tool or toolkit running on a live system.

This paper describes a methodology for measuring the uncertainty in live forensics. The application of the methodology is illustrated using collection tools targeted for live Windows and Linux systems.

## 2. Background

Uncertainty is a term that is used in subtly different ways in fields such as philosophy, physics, statistics, economics, finance, insurance, psychology, sociology, engineering and information science. It applies to predictions of future events, to physical measurements already made, or to the unknown. Although the term is used in a variety of ways in public discourse, specialists in decision theory, statistics and other quantitative fields have defined uncertainty and risk more specifically. Hubbard [6, 9] defines uncertainty and the related concept of risk as follows:

- **Uncertainty:** Uncertainty is the lack of certainty. It is a state of having limited knowledge where it is impossible to exactly describe the existing state or future outcome or more than one possible outcome.
- **Measurement of Uncertainty:** The uncertainty of a set of possible states or outcomes is measured by assigning probabilities to each possible state or outcome; this also includes the use of a probability density function for continuous variables.
- **Risk:** Risk is a state of uncertainty where some possible outcomes have an undesired effect or significant loss.
- **Measurement of Risk:** Risk is measured in terms of a set of uncertainties where some possible outcomes are losses and in terms of the magnitudes of the losses; this also includes the use of loss functions for continuous variables.

From an engineering perspective, uncertainty is characterized as Type A or Type B, depending on the method used to evaluate it. A Type A method evaluates uncertainty using a statistical analysis of a series of observations. On the other hand, a Type B method evaluates uncertainty by a means other than the statistical analysis of a series of observations. These two components of the uncertainty can be quantified by the statistically estimated standard deviation, $u_i$ and $u_j$, for Type A and Type B methods, respectively [14].

To illustrate Type A uncertainty, consider an input quantity $X_i$ whose value is estimated from $n$ independent observations $X_{i,k}$ of $X_i$ obtained under the same measurement conditions. In this case, the input estimate $x_i$ is usually the sample mean given by:

$$x_i = \overline{X_i} = \frac{1}{n}\sum_{k=1}^{n} X_{i,k}$$

The standard uncertainty $u(x_i)$ associated with $x_i$ is the estimated standard deviation of the mean:

$$u_{x_i} = \left(\frac{1}{n(n-1)}\sum_{k=1}^{n}(X_{i,k} - \overline{X_i})^2\right)^{1/2}$$

A Type B evaluation of standard uncertainty is usually based on a scientific judgment using all the relevant information that is available. This may include: (i) previous measurement data; (ii) experience with, or general knowledge of, the behavior and property of relevant materials and instruments; (iii) manufacturer specifications; (iv) data provided in calibration and other reports; and (v) uncertainties assigned to reference data taken from handbooks.

Type B uncertainty is often quite difficult to measure. As a consequence, it is useful to establish an upper-bound measurement of the uncertainty when performing a live forensic analysis that involves the execution of one or more tools on a target system that affects the live memory.

## 3. Measuring Uncertainty in Digital Forensics

Casey [3] proposed that the uncertainty for network-related evidence be calculated by estimating probability distributions. Given a series of real measurements (e.g., clock offsets from a network of computers), it is possible to estimate the real distribution of data. If a normal or Gaussian data distribution is observed, an accuracy measurement for the data set under investigation can be obtained. The precision of the

measurement can be described using the mean $\mu$ and standard deviation $\sigma$. Although, this statistical method is a relatively straightforward way to estimate the uncertainty, it is not necessarily applicable or accurate and it strongly depends on the kind of data that is available.

In order to measure how much a running forensic tool affects system memory, it is necessary to define a methodology that quantifies how much the volatile memory changes. This is closely related to the concept of a "footprint," which measures the amount of volatile memory that a program uses or references when running. The footprint includes all active memory regions such as code, static data sections (initialized and uninitialized), and heap and stack areas. In addition, it includes the memory required to hold data structures such as symbol tables, constant tables, debugging structures and open files that the program requires while executing and that are loaded at least once during its execution [15]. In the case of Microsoft Windows, it is not possible to determine the footprint by examining the working memory (heap memory) using the Windows Task Manager (WTM). For example, WTM is unable to properly quantify all the kernel space memory that is affected by a tool like `dd`. In addition, effects such as the paging of less used memory pages to disk may occur when running the collection tool.

A promising approach is to sample the volatile memory (RAM contents) before and after a forensic tool runs and to repeat the procedure several times to account for statistical variations of the measurements. The collection tools used would depend on the specific operating system running on the live machine. Tools for Windows systems include `dd` [5], `mdd` [12], FastDump (`fd`) [7], `win32dd` [20] and Memoryze [11]. Tools for Linux platforms include GNU `dd` [16] and `dc3dd` [17]. These collection tools also probe the running live system. Ideally, such probes should not affect the volatile memory. Unfortunately, the available collection tools modify the volatile content to a greater or lesser extent.

The binary difference between memory snapshots (samples) collected by a tool (e.g., `dd`) at different times is a good estimator of the uncertainty introduced by the collection tool. Also, by taking a series of memory snapshots and measuring the statistical variation using the sample average ($\mu_{dmp}$) and standard deviation ($\sigma_{dmp}$) of the different memory pages between consecutive memory snapshots, it is possible to define a more robust measurement of the uncertainty introduced by the tool. The uncertainty may be quantified as $U = \mu_{dmp} \pm \sigma_{dmp}$, where the sample average is related to the different memory pages between consecutive snapshots, and the sample standard deviation is related to the statistical error of the measurement. Figure 1 illustrates the timeline of an experiment required to evaluate the uncertainty of a collection tool.

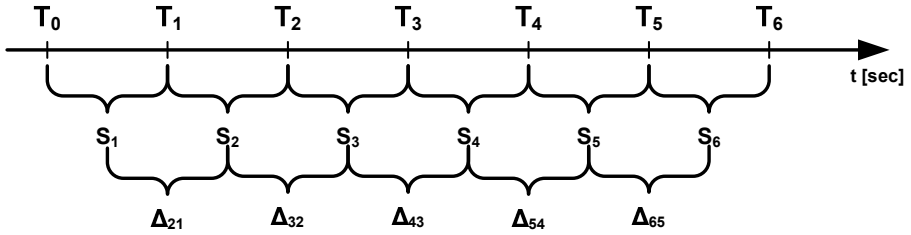


*Figure 1.* Timeline for evaluating the uncertainty of a tool.

The sampling procedure may be repeated as many times as possible in order to account for normal statistical variations.

Our definition of uncertainty requires the measurement of the real memory variations caused by a tool; in our case, the collection tool that is used as a probe to evaluate the uncertainty caused by other forensic tools. In this way, it is possible to show the extent of the "perturbation" caused by an imaging tool. Moreover, the practical effect of this variation can impact the integrity of the memory snapshot, which should not be altered according to digital forensic science principles.

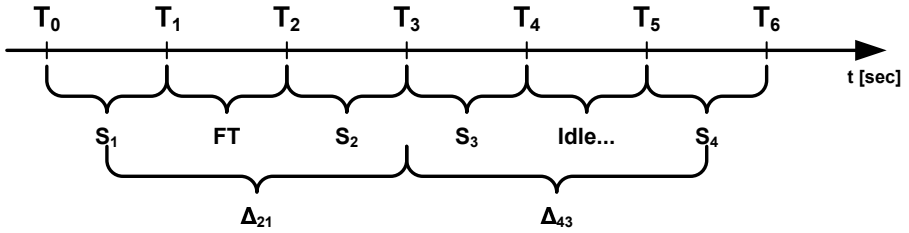


*Figure 2.* Timeline for evaluating the uncertainty of a live forensic tool.

This approach considers the ordinary uncertainty of the measurement tool (i.e., collection tool) as if it were the only tool running on the system. This assumption implies that the uncertainty of the probing tool should be determined only by the probing process (e.g., `dd`, `fd`, `mdd`, `win32dd` or Memoryze), without considering other processes (e.g., background processes) on the live system being examined. The resolution of the probing tool defines an upper bound on the precision of the measurement procedure. In fact, the probing tool itself affects the measurement procedure, and it is desired to have the best possible precision. On the other hand, in order to evaluate the uncertainty of a live forensic tool (e.g., Windows Forensic Toolchest [13]), it would be necessary to follow the procedure illustrated in Figure 2. The memory variation caused by

the running toolkit is measured as the binary difference between the two snapshots $S_1$ and $S_2$, which is denoted as $\Delta_{21}$. This represents an upper bound limit on the uncertainty. The lower bound is determined by the binary difference between $S_3$ and $S_4$ (say $\Delta_{43}$), which takes into account the memory variations while the system is in an idle state (i.e., only background processes are running) for the period of time it takes to run the forensic toolkit. As a consequence, the uncertainty corresponding to the forensic toolkit is in the range $\Delta_{43} \leq U_{tot} \leq \Delta_{21}$. This approach may be used on live systems where it is not possible to acquire the memory by halting the system (as in virtual machines [1]). In fact, the ideal procedure would be to halt the system, take the memory snapshot using an atomic operation, run the forensic tool, and finally collect an additional memory snapshot after halting the system. At this time, such a procedure can be conducted only by using virtual technology.

## 4. Experimental Setup

Our experiments used several collection tools for two major operating systems, Windows XP SP3 and Linux Suse 11.1 (Kernel 2.6.27.7-9). The tools used on the Windows system were `dd`, `fd`, `mdd`, `win32dd` and Memoryze. The tools used on the Linux system were GNU `dd` [16] and `dc3dd` [17].

- `dd` **(version 1.0.0.1035)**
  - Usage: `dd.exe if=\\.\PhysicalMemory of=dump.bin conv=noerror`
- `fd` **(version 1.3.0)**
  - Usage: `fd.exe [-nodriver] dump.bin`
  - If no options are specified, the memory snapshot is collected by installing the `fd` driver. The `[-nodriver]` option causes the snapshot to be collected using a Windows API.
- `mdd`
  - Usage: `mdd.exe -o dump.bin`
- `win32dd` **(version 1.2.2.20090608)**
  - Usage: `win32dd.exe -l 0 -r dump.bin`
  - The `-r` option produces a raw binary image of the memory content. The `-l 0` option accesses the `\\Device\\PhysicalMemory` device. The `-l 1` option uses `Windows Kernel API MmMapIoSpace()`.

- **Memoryze**
  - Usage: `MemoryDD.bat`
  - The `-offset` option specifies the offset into physical memory. The `-size` option specifies the size of physical memory to acquire. The `-output` specifies the directory where the results are written (default "Audits").
- **GNU `dd`**
  - Usage: `dd if=/dev/mem of=/mnt/sdb1/dump.bin conv=noerror`
- **dc3dd**
  - Usage: `dc3dd if=/dev/mem of=/mnt/sdb1/dump.bin conv=noerror`

## 5. Experimental Results

This section describes the results of evaluating the uncertainty of collection tools in Windows XP SP3 and Linux (OpenSuse 11.1) systems. As described in Section 2 and in our previous work [18], the evaluations involve the analysis of the real footprints of running tools.

### 5.1 Windows System

This section presents the results obtained for five collection tools that were run on a Dell Optiplex 330 platform (1 GB RAM) under Windows XP SP3. Table 1 presents the uncertainty for the five collection tools; note that `fd` was run under two operational modes `fd1` and `fd2`. Ten memory snapshots (S = 10) of size MP were collected using each tool/ operational mode (MP denotes the number of collected 4 KB memory pages). Each snapshot was copied to an external USB 2.0 drive. $\mu_{dmp}$ denotes the sample mean of the different (4 KB) memory pages and $\sigma_{dmp}$ denotes the sample standard deviation for consecutive memory snapshots. The number of 4 KB memory pages, size in MB and percentage are provided for $\mu_{dmp}$ and $\sigma_{dmp}$. $T_{coll}$ lists the average collection times and $U_{tot}$ lists the uncertainty measurement, which is expressed as a percentage of the modified volatile memory. For example, $U = 65.5\% \pm 0.15\%$ quantifies the uncertainty of the memory snapshot collected with the `dd` tool. The result is quite surprising in that more than 65% of the total memory is affected during the collection process.

Table 1 also shows that while the uncertainty for `win32dd` ($U_{\texttt{win32dd}} = 9.6\% \pm 0.42\%$) is the lowest among the tested tools, the corresponding

*Table 1.* Tool uncertainty for a Windows system.

| Tool | S [n°] | MP [n°] | $\mu_{dmp}$ [n°] | [MB] | [%] | $\sigma_{dmp}$ [n°] | [MB] | [%] | $T_{coll}$ [sec] | $U_{tot}$ [%] |
|---|---|---|---|---|---|---|---|---|---|---|
| `dd` | 10 | 259,417 | 169,835 | 663.41 | 65.5 | 389 | 1.52 | 0.15 | 55 | $65.5 \pm 0.15$ |
| `fd1` | 10 | 259,287 | 168,811 | 659.42 | 65.1 | 346 | 1.35 | 0.13 | 55 | $65.1 \pm 0.13$ |
| `fd2` | 10 | 259,287 | 167,865 | 655.72 | 64.7 | 1181 | 4.61 | 0.45 | 56 | $64.7 \pm 0.45$ |
| `mdd` | 10 | 259,287 | 167,945 | 656.03 | 64.8 | 360 | 1.41 | 0.14 | 42 | $64.8 \pm 0.14$ |
| `win32dd` | 10 | 259,428 | 249,51 | 97.46 | 9.6 | 1093 | 4.27 | 0.42 | 447 | $9.6 \pm 0.42$ |
| Memoryze | 10 | 259,417 | 161,672 | 631.53 | 62.3 | 406 | 1.59 | 0.16 | 60 | $62.3 \pm 0.16$ |

time ($T_{coll}$) is approximately eight times higher. This is not the ideal case because the memory snapshot should be collected as an atomic operation, as fast as possible and with minimum integrity leaks. Indeed, the results show that a trade-off exists between uncertainty and speed.

*Table 2.* Tool uncertainty for a virtual Windows system.

| Tool | S [n°] | MP [n°] | $\mu_{dmp}$ [n°] | [MB] | [%] | $\sigma_{dmp}$ [n°] | [MB] | [%] | $T_{coll}$ [sec] | $U_{tot}$ [%] |
|---|---|---|---|---|---|---|---|---|---|---|
| GNU `dd` | 10 | 262,143 | 189,762 | 741.25 | 72.4 | 4,689 | 18.32 | 1.79 | 93 | $72.4 \pm 1.79$ |
| `fd1` | 10 | 262,010 | 187,930 | 734.10 | 71.7 | 270 | 1.05 | 0.10 | 90 | $71.7 \pm 0.10$ |
| `fd2` | 10 | 262,010 | 186,482 | 728.44 | 71.2 | 958 | 3.74 | 0.36 | 92 | $71.2 \pm 0.36$ |
| `mdd` | 10 | 262,010 | 238,482 | 931.57 | 90.1 | 100 | 0.39 | 0.038 | 74 | $90.1 \pm 0.038$ |
| `win32dd` | 10 | 262,144 | 22,589 | 88.24 | 8.6 | 714 | 2.79 | 0.27 | 2,059 | $8.6 \pm 0.27$ |
| Memoryze | 10 | 262,143 | 199,832 | 780.64 | 76.2 | 146 | 0.57 | 0.056 | 99 | $76.2 \pm 0.056$ |

Table 2 presents the results obtained for a virtual Windows XP SP3 based system (1 GB RAM). Note that the virtual system was customized in the same manner as the real Windows system (Table 1) in order to provide an almost identical testing environment. The snapshots were collected using the same tools and copied to an external USB 2.0 hard disk. The uncertainty values shown in Table 2 are on the average 10% higher for the `dd`, `fd1`, `fd2` and Memoryze tools compared with the results in Table 1. Also, the uncertainty for the `mdd` tool is 30% higher and the collection time has increased by 30%. Interestingly, the uncertainty for `win32dd` has remained almost constant, whereas its collection time has increased about 8.5 times to 2,059 seconds.

Clearly, an ideal probing tool should have the fastest collection time but should affect the memory snapshot as little as possible. However, the results show that this may not be appropriate in the case of a virtual system, where it would be more precise to sample the memory from outside the system.

*Table 3.* Tool uncertainty for a virtual Linux system (no direct I/O).

| Tool | S [n°] | MP [n°] | $\mu_{dmp}$ [n°] | [MB] | [%] | $\sigma_{dmp}$ [n°] | [MB] | [%] | $T_{coll}$ [sec] | $U_{tot}$ [%] |
|---|---|---|---|---|---|---|---|---|---|---|
| GNU `dd` | 10 | 131,072 | 32,032 | 125.25 | 24.4 | 969 | 3.78 | 0.74 | 25 | $24.4 \pm 0.74$ |
| `dc3dd` | 10 | 131,072 | 33,134 | 129.43 | 25.3 | 1123 | 4.38 | 0.10 | 28 | $25.3 \pm 0.10$ |

## 5.2 Linux System

This section presents the experimental results obtained for two collection tools that were run on a virtual Linux system (Suse Linux 11.1; Kernel 2.6.27.7-9) equipped with 512 MB RAM and a 10 GB hard disk. The two tools used were GNU `dd` and `dc3dd`. An interesting feature of the Linux OS is that it bypasses the memory cache system [2]. This makes it possible to transfer a memory snapshot with direct I/O communication, resulting in better memory coherence of the snapshot (i.e., lower uncertainty than when the page cache system is used). This feature is not available for the Windows OS.

Table 3 summarizes the experimental results for the Linux platform. All the memory snapshots were copied to an external USB 2.0 hard drive without direct I/O. The uncertainty values for the GNU `dd` and `dc3dd` tools are $U_{GNU\texttt{dd}} = 24.4\% \pm 0.74\%$ and $U_{\texttt{dc3dd}} = 25.3\% \pm 0.10\%$, respectively. Note that the uncertainty values and collection times for the two tools are very similar.

*Table 4.* Tool uncertainty for a virtual Linux system (direct I/O).

| Tool | S [n°] | MP [n°] | $\mu_{dmp}$ [n°] | [MB] | [%] | $\sigma_{dmp}$ [n°] | [MB] | [%] | $T_{coll}$ [sec] | $U_{tot}$ [%] |
|---|---|---|---|---|---|---|---|---|---|---|
| GNU `dd` | 10 | 131,072 | 5,950 | 23.24 | 4.53 | 121 | 0.47 | 0.09 | 35 | $4.53 \pm 0.09$ |
| `dc3dd` | 10 | 131,072 | 5,035 | 19.67 | 3.84 | 104 | 0.41 | 0.08 | 33 | $3.84 \pm 0.08$ |

Table 4 shows the results obtained for the virtual Linux system with direct I/O communications (USB transfer with `DIRECT_IO` and user mode buffer size of 50 KB). This was accomplished via the `oflag=direct` parameter using the command:

```
dd if=/dev/mem of=/media/SIGMA/img1/dump01.dd oflag=direct bs=50K
```

The parameter `bs` sets the size of the buffer in user mode, which avoids the use of the Linux kernel page cache [2]. The snapshots were copied to the external USB storage device with an average time of 35 seconds (average transfer speed of 15 MB/s). The estimated footprints of the

*Table 5.* GNU `dd` tool uncertainty for different buffer sizes.

| Uncertainty (U) [%] | Buffer Size [KB] | Collection Time [sec] | Transfer Speed [MB/sec] |
|---|---|---|---|
| $6.9 \pm 0.09$ | 5 | 235 | 2.3 |
| $5.3 \pm 0.11$ | 10 | 107 | 5.0 |
| $6.2 \pm 0.06$ | 15 | 93 | 5.8 |
| $4.2 \pm 0.07$ | 30 | 60 | 8.9 |
| $3.9 \pm 0.09$ | 60 | 35 | 15.1 |
| $3.7 \pm 0.08$ | 100 | 33 | 16.0 |
| $4.6 \pm 0.13$ | 500 | 29 | 18.0 |
| $4.1 \pm 0.14$ | 1,024 | 28 | 19.2 |

GNU `dd` and `dc3dd` collection tools are $U_{GNU\texttt{dd}} = 4.53\% \pm 0.09\%$ and $U_{\texttt{dc3dd}} = 3.84\% \pm 0.08\%$, respectively. Note that the uncertainty for the two collection tools is reduced by almost five times. Avoiding the page cache also provides better memory coherence.

It is important to be aware of how much the user mode buffer size influences the uncertainty. For this purpose, a set of statistically meaningful, albeit small, memory snapshots were collected with direct I/O communications using different user mode buffer sizes. Table 5 presents the uncertainty values for the GNU `dd` tool (based on five snapshots) for buffer sizes ranging from 5 KB to 1,024 KB. Note that the `bs` parameter specifies equal input and output buffers for the reading and writing phases for the GNU `dd` and `dc3dd` tools. The lowest uncertainty value is obtained with the buffer set to 100 KB ($U_{GNU\texttt{dd}} = 3.7\% \pm 0.08\%$) with a collection time of 33 seconds and transfer speed of 16.0 MB/s.

In another experiment, the effect of LAN communications on memory snapshots was determined by transferring the snapshots to a remote Windows system via a network link. This was accomplished using the command:

```
dd if=/dev/mem | nc 192.168.1.12 10000
```

*Table 6.* Tool uncertainty with LAN communications.

| Tool | S [n°] | MP [n°] | $\mu_{dmp}$ [n°] | $\mu_{dmp}$ [MB] | $\mu_{dmp}$ [%] | $\sigma_{dmp}$ [n°] | $\sigma_{dmp}$ [MB] | $\sigma_{dmp}$ [%] | $T_{coll}$ [sec] | $U_{tot}$ [%] |
|---|---|---|---|---|---|---|---|---|---|---|
| GNU `dd` | 10 | 131,072 | 6,331 | 24.73 | 4.83 | 393 | 1.53 | 0.30 | 40 | $4.83 \pm 0.30$ |
| `dc3dd` | 10 | 131,072 | 6,034 | 23.57 | 4.60 | 248 | 0.97 | 0.19 | 39 | $4.60 \pm 0.19$ |

Table 6 shows the uncertainty values in the case of LAN communications: $U_{GNU\texttt{dd}} = 4.83\% \pm 0.30\%$ and $U_{\texttt{dc3dd}} = 4.60\% \pm 0.19\%$. The

*Table 7.* Tool uncertainty under heavy RAM load without direct I/O.

| Tool | S [n°] | MP [n°] | $\mu_{dmp}$ [n°] | $\mu_{dmp}$ [MB] | $\mu_{dmp}$ [%] | $\sigma_{dmp}$ [n°] | $\sigma_{dmp}$ [MB] | $\sigma_{dmp}$ [%] | $T_{coll}$ [sec] | $U_{tot}$ [%] |
|---|---|---|---|---|---|---|---|---|---|---|
| GNU `dd` | 10 | 131,072 | 121,134 | 437.18 | 92.4 | 133 | 0.52 | 0.10 | 25 | $92.4 \pm 0.10$ |
| `dc3dd` | 10 | 131,072 | 118,036 | 461.08 | 90.0 | 248 | 0.97 | 0.19 | 26 | $90.0 \pm 0.19$ |

memory snapshots were copied to the host system using `netcat` with an average time of 40 seconds. The results in Table 6 demonstrate that the uncertainty is comparable with that obtained for a virtual system with direct I/O.

Experiments were also conducted to verify how much the volatile memory load (i.e., percentage of volatile memory being used by running processes) impacts collection by the GNU `dd` tool. This can occur when malware consumes large amounts of RAM, possibly impacting the volatile memory collection.

To conduct the experiments, a custom script was created to progressively allocate memory in 1 MB blocks during the collection process. Initially, all the memory snapshots were copied without direct I/O communications; direct I/O was set in a subsequent series of experiments. This enabled us to verify the net effect of direct communications on the memory snapshot uncertainty.

Table 7 shows the results under heavy RAM loads without direct I/O communications. The uncertainty for the GNU `dd` tool is $U_{GNU\texttt{dd}} = 92.4 \pm 0.10\%$ or $U_{GNU\texttt{dd}} = 437.18 \pm 0.52$ MB. Also, the memory snapshots were copied to the host system via the USB 2.0 link with an average time of 25 seconds. Similar results were obtained for the `dc3dd` tool.

*Table 8.* Tool uncertainty under heavy RAM load with direct I/O.

| Tool | S [n°] | MP [n°] | $\mu_{dmp}$ [n°] | $\mu_{dmp}$ [MB] | $\mu_{dmp}$ [%] | $\sigma_{dmp}$ [n°] | $\sigma_{dmp}$ [MB] | $\sigma_{dmp}$ [%] | $T_{coll}$ [sec] | $U_{tot}$ [%] |
|---|---|---|---|---|---|---|---|---|---|---|
| GNU `dd` | 10 | 131,072 | 82,652 | 322.9 | 63.06 | 534 | 2.09 | 0.41 | 31 | $63.06 \pm 0.41$ |
| `dc3dd` | 10 | 131,072 | 82,302 | 321.5 | 62.80 | 456 | 1.78 | 0.34 | 30 | $62.80 \pm 0.34$ |

Table 8 shows the experimental results under heavy RAM loads with direct I/O communications. The uncertainty values for the GNU `dd` tool is $U_{GNU\texttt{dd}} = 63.06 \pm 0.41\%$ and $U_{GNU\texttt{dd}} = 322.9 \pm 2.09$ MB. The memory snapshots were copied to the host system via USB in an average time of 31 seconds. Similar results were obtained for the `dc3dd` tool.

Note that the difference between the uncertainty values in the two cases is 29.4%, which is due to the Linux kernel page cache. This result is not surprising because we have already determined the net effect of the page cache. As a consequence, it is necessary to use the direct I/O communications mode when performing a live memory acquisition on a Linux system.

## 5.3 Practical Consequences of Uncertainty

The experimental results show that the ordinary uncertainty for the collection tools applied to a Windows system is high, especially when using a USB connection to transfer memory snapshots. The notable exception is the `win32dd` tool, which affects the volatile memory less than the other tools. Also, memory dumps are more coherent (i.e., have lower uncertainty) when a LAN connection is used to transfer snapshots.

The net effect of a high uncertainty in the memory snapshots is that some memory pages (even processes) can be swapped out to a dedicated disk partition or a specific set of files depending on the operating system [2]. In some situations, especially when many processes that require plenty of RAM are running, there could be an "out of memory" event, which forces the operating system to kill a running process, usually the last one that was started. Such a critical situation can produce an evidence "leak" or even prevent the collection tool from running.

Increased usage of the page cache in Windows and Linux systems can also cause process pages to be swapped out. If this occurs during a digital forensic investigation, it is necessary to copy the volatile memory (RAM contents) as well as the swap area in order to recover all the process pages. In the case of Windows [8, 18], it is possible to recover almost all the pages related to a running process, even if some pages have been swapped out.

In a Linux system, it is not clear if some processes are deleted or altered when a collection tool runs with direct I/O. Nevertheless, it is preferable to use the direct I/O mode in digital forensic investigations because the volatile memory undergoes less changes.

# 6. Conclusions

Every digital forensic practitioner should be aware of how much the volatile memory is affected when a forensic tool is used in a live investigation. In particular, it is important to know and possibly predict the extent of the memory perturbation or uncertainty. Experimental results demonstrate that our methodology for evaluating the uncertainty of collection tools is simple and effective, and is applicable to Windows and

Linux systems. The methodology may also be used to evaluate the extent to which a generic forensic tool or toolkit perturbs volatile memory. Although the measurements are limited by the uncertainty of the probing tool, it is still possible to obtain a range of uncertainty for a forensic procedure that affects the volatile memory more than the probing tool itself.

## References

[1] D. Bem, Computer forensic analysis in a virtual environment, *International Journal of Digital Evidence*, vol. 6(2), 2007.

[2] D. Bovet and M. Cesati, *Understanding the Linux Kernel*, O'Reilly, Sebastopol, California, 2006.

[3] E. Casey, Error, uncertainty and loss in digital evidence, *International Journal of Digital Evidence*, vol. 1(2), 2002.

[4] e-fense, Helix3 Enterprise, Washington, DC (www.e-fense.com/helix), 2009.

[5] G. Garner, Forensic Acquisition Utilities (www.gmgsystemsinc.com/fau), 2009.

[6] J. Halpern, *Reasoning about Uncertainty*, MIT Press, Cambridge, Massachusetts, 2005.

[7] HBGary, FastDump Pro, Sacramento, California (www.hbgary.com/products-services/fastdump-pro).

[8] J. Kornblum, Using every part of the buffalo in Windows memory analysis, *Digital Investigation*, vol. 4(1), pp. 24–29, 2007.

[9] D. Lindley, *Understanding Uncertainty*, John Wiley, Hoboken, New Jersey, 2006.

[10] C. Malin, E. Casey and J. Aquilina, *Malware Forensics: Investigating and Analyzing Malicious Code*, Syngress, Burlington, Massachusetts, 2008.

[11] Mandiant, Memoryze, Washington, DC (www.mandiant.com/software/memoryze.htm).

[12] ManTech, Memory DD, Vienna, Virginia (cybersolutions.mantech.com/products.htm).

[13] M. McDougal, Windows Forensic Toolchest (WFT) (www.foolmoon.net/security/wft), 2005.

[14] National Institute of Standards and Technology, The NIST Reference on Constants, Units and Uncertainty, Gaithersburg, Maryland, 2006.

[15] M. Oliveira, R. Redin, L. Carro, L. da Cunha Lamb and F. Wagner, Software quality metrics and their impact on embedded software, *Proceedings of the Fifth International Workshop on Model-Based Methodologies for Pervasive and Embedded Software*, pp. 68–77, 2008.

[16] P. Rubin, D. MacKenzie and S. Kemp, GNU `dd` (www.gnu.org/software/coreutils).

[17] P. Rubin, D. MacKenzie, S. Kemp, J. Kornblum and A. Medico, `dc3dd` (dc3dd.sourceforge.net).

[18] A. Savoldi and P. Gubian, Blurriness in live forensics: An introduction, *Proceedings of the Third International Conference on Information Security and Assurance*, pp. 119–126, 2009.

[19] A. Savoldi and P. Gubian, Volatile memory collection and analysis for Windows mission-critical computer systems, *International Journal of Digital Crime and Forensics*, vol. 1(3), pp. 42–61, 2009.

[20] M. Suiche, `win32dd` (win32dd.msuiche.net).