

sequence or simply by pulling a power plug. The second approach prevents possible execution of scripts set up to remove evidence in a case of the shutdown. It might also result in inconsistent disk state and data in write cache unsynchronized. In both cases of halting the target system its dynamic state is inevitably lost. Volatile memory content is not preserved, except for possible data paged to disk. Other relevant dynamic data like a process list, open network ports, established network connections and installed kernel modules cannot be examined with static analysis. Since all above items could be relevant for the investigation being conducted, static analysis gives incomplete evidence.

Encryption makes access to data stored in encrypted files or volumes much harder if not impossible. Encryption keys, used during system operation for access to encrypted data, are inaccessible once the system is powered off.

There is also a question if creation of an image for analysis is practical or even possible. For terabytes of disk data imaging can take many hours. Imaging of large RAID arrays, NAS, SANs is extremely difficult. The time and effort needed for analysis increases with the disk size. Since manual analysis is not feasible, automatic analysis is used. For terabytes of data even automatic analysis takes time [4] and requires a distributed analysis approach [5].

An often neglected drawback of static analysis is an inconvenience for a regular user. A system cannot be used while forensic copies of media are made. For certain systems that require high availability this could completely prevent static analysis.

An alternative to static analysis, or to be more precise complementary approach, is live analysis. In this case evidence is collected while the system is running. Live analysis resolves some of the issues of static analysis. On the other hand there are some open questions for live analysis. The most important issue is that with live analysis any action the analyst executes causes irreversible change of the investigated system state [6]. The change of the system state is against accepted principles in forensics that evidence should not be altered and might cause examination results not to be verifiable and repeatable [7]. Limitations imposed on acceptability of evidence collected by live analysis depend on the applicable legal system.

There are some other issues with live analysis. Investigator might not have appropriate level of access to the investigated system. In a case of a compromised system its integrity is questionable. An attacker might have modified the system in a way that prevents detection of attack and modifications. This is especially true if the investigated system's user interface is used. Even using known-good binaries from CD/DVD or other trusted media might not reveal the true facts since many of the forensic tools rely on the data provided by the kernel or the file system that might have been tampered with.

There are some alternatives to static and live analysis. Two that are relevant to this paper will be explained in more detail in the next section. The two are: static analysis using virtualization and offline analysis of volatile memory.

III. VIRTUAL MACHINES FOR STATIC ANALYSIS

Static analysis of hard disks is activity with a number of constraints. A basic requirement that no changes can be made to the original hard disk might significantly limit investigation efforts. Practice is to obtain forensic image of the original hard disk. There is a requirement for the image to remain unchanged to be acceptable as evidence in a court of law. Some information from the hard disk or its image cannot be obtained by passive reading of the data from the disk. Sometimes an operating system and/or programs from the disk need to be started in order to get insight into what is on the disk and what it was used for. Starting the program or the operating system from the disk unavoidably includes writing to the disk. This writing renders the disk unusable as evidence.

Usage of virtual machines was proposed for an analysis phase of a digital forensic investigation [8]. The idea is to create a virtual machine from an image to be investigated. Since the virtual machine simulates only some basic hardware components the image cannot be immediately booted in the virtual machine. There are many changes to the original environment required to enable the image to boot in the VM environment, and once the system is booted new data will be written to the original image thus modifying it. Such a changed image would be immediately challenged in a court of law as flawed.

This issue was addressed by CERT. They created Live View [9]. Live View is a forensics tool that creates a VMware virtual machine out of a raw disk image or a physical disk. This allows the forensic examiner to boot up the image or the disk and gain an interactive, user-level perspective of the environment, all without modifying the underlying image or disk. All changes made to the disk are written to a separate file. This enables continuation of analysis from where it stopped or a restart from the original state of the image or the disk.

IV. VOLATILE MEMORY ANALYSIS

A recent addition to live analysis is an idea to make a dump of a volatile memory for offline analysis. Volatile memory analysis shows promise in that the only source of evidence is a physical memory dump. An investigator can then build the case by analyzing the memory dump in an isolated environment that is non-obtrusive to the evidence. This approach addresses some of the issues with live analysis. It limits impact to the compromised system, the analysis is repeatable and it is possible to ask new questions later. Also, offline volatile memory analysis does not rely on operating system of possibly compromised machine. This enables detection of processes hidden by installed rootkit or a similar tool [10].

The first idea was to use a special pre installed hardware that can copy memory to an external storage device without modifying its contents [11]. The need for the special hardware that must be fitted to the system being protected before anything bad happens makes this approach impracticable in general case, but it had started a lot of research dedicated to analysis of memory dumps. a number of papers is devoted to Windows memory analysis [12][13][14][15]. A method for recovering files mapped in a memory and to link mapped file information process data is subject of [16]. Different tools for

memory analysis are proposed like: FATKit [17], BodySnatcher [18], Volatools [19] and FACE [20].

The biggest issue with the memory dump analysis was a lack of tools. This is because with every release of a new operating system, the physical memory structure changes. In spite of difficulties some tools are being developed. The first ones had only basic functionality [21][22]. Current tools [23][24] are still script based but offer functionality similar to the live analysis. They allow an investigator to interrogate an image in a style similar to that used during a live response.

The most recent development in volatile memory analysis is usage of hibernation. Most current computers systems and OS have power management features that save the state of the computer while the processor and devices (hard drive, monitor, etc.) are disabled to conserve power. This is also known as suspending to disk feature. If the system state and memory are copied to disk using the power management features of the computer, then this method may be more reliable than the software solutions for creation of memory dump [11]. System state and memory are usually saved in a file that contains all the physical memory saved by the OS and aims to be restored by the user the next time the computer is powered on.

Live forensics analysis is used on a physical memory dump to recover information from a targeted machine. One of the main problems is to obtain a readable physical memory dump, hibernation is an efficient way to save and load physical memory. Hibernation analysis has notable advantages. System activity is totally frozen, therefore coherent data is acquired and no software tool is able to block the analysis. The system is left perfectly functional after analysis, with no side effects.

Usage of hibernation was mentioned for the first time when volatile memory investigation was suggested [11]. It was not really used until recently when a tool for Windows hibernation file analysis was presented [25].

V. PROPOSED METHOD

This paper proposes combination of static and live analysis. The combination should provide more insight into current state of the system being examined and better understanding of the events that led to it.

A hard disk image can be booted in VMware using Live View. The memory dump obtained before the system was powered off can be used to restore the system booted in virtual environment to the state it was in when the memory dump was created or as close to it as possible. The memory dump contains data on the processes that were running on the system. It also contains times when the processes were started. This enables manual start of the programs that were not started during boot process. The programs can be started in the same order as they were started on the investigated system. This includes hidden programs like rootkits that would not be visible in live analysis. Open files and open network ports with processes that opened them are also stored in the volatile memory dump. With this information investigator can try to open the same files and network ports. Furthermore, sometimes even encryption keys can be recovered from the memory dump

[19], but this is out of the scope of this paper and will not be tested.

Since Live View enables booting of an image in VMware while keeping the image file intact, an investigator can use trial and error approach trying to restore and understand the events that took place on the system. What-if analysis can be performed with the goal to understand possible development scenarios. The best thing is that investigators actions are repeatable and can be presented in a court of law.

The proposed approach to live analysis is to simply put the system to hibernation. This is a fast way to preserve volatile memory and system state with no need for additional tools and no change of system state. After system goes to hibernation, hard disk image can be created for further investigation. The image contains hibernation file that in turn contains volatile memory content.

As a proof of concept, test system was created and was taken through the proposed steps of acquisition and analysis. Testing results are presented in the next section.

VI. TESTING

System to be investigated was Windows XP SP3 installed on 8 GB NTFS partition with 256 MB of RAM. The hard disk partition and RAM are small compared to the current standards. They are set small to enable a faster manipulation but big enough to show the procedure. A user with administrative privileges was logged on the system. Some sample programs were started on the system. One file was opened for editing in Notepad. Netcat tool was started from a command prompt and set to listen for incoming connections on port 80. TrueCrypt 6.1a [26] was used to create a new volume as an encrypted file container. One file on this encrypted volume was opened for editing in Notepad.

At this point the system was hibernated. The hard disk was taken out of the computer. An image of 8 GB partition in a raw (dd) format was created using dd tool. This image was stored on the investigators desktop computer and set read only. MD5 hash value was calculated using Microsoft File Checksum Integrity Verifier. The value was stored for later comparison.

Live View 0.7b was started and given some basic data about the image and the environment it should be booted in. It was instructed to provide the same amount of RAM, 256 MB, in the virtual machine. Also, information that the image is Windows XP was supplied. Live View created necessary files in the selected directory. Then Live View started VMware Server 1.0.8 and pointed it to virtual machine files it created from the image.

VMware server started boot up process. It realized that the system was hibernated and tried to wake it up. Coming out of hibernation failed, which was expected due to hardware difference between the original system and virtual environment. Windows XP offered to delete restoration data and boot up system without retrieving hibernated system state. It was accepted and the system started. The programs that were manually started on the original system before hibernation were not running. The same goes for the files manually opened

on the original system. There was a message about new hardware being found, but it was ignored at this point. The system in virtual machine was powered off. It was done for two reasons. One was to test if the image has changed. Image, being read only, has not changed which was confirmed calculating its MD5 hash again. The other reason for shutdown was to extract hibernation file from the image.

Partition image was mounted, read only, using Mount Image Pro v3. File hiberfil.sys was copied from the mounted image to the file system. The copy was set read only and its MD5 hash value calculated. Using SandManSHELL utility [27] memory dump was created from the hiberfil.sys. Again the file was set to be read only and its MD5 hash was calculated. Although SandMan provides some basic utilities for extracting data from hibernation file they are rather crude. At this time it was concluded that it is easier and less error prone to use more mature tools for memory dump analysis, like Volatility framework [23].

Before analysis of the memory dump partition image was booted again using Live View. Live View offered choice to continue working on the modified virtual machine or start over from the image file. The option to continue working was selected. The system booted up without messages and without hibernated data. Now, there was a running picture of investigated system that could be compared with a picture of the system preserved in hibernation file.

Using Volatility it was possible to read various data from the memory dump. List of running processes with times when they were started was generated (Fig. 1). The list was compared with the list of running processes in virtual machine (Fig. 2). The difference was in the processes that were manually started on the original investigated system. Volatility list of DLLs loaded for each process (Fig. 3) show the command line, e.g. path, which was used to start the process.

For instance, the following lines show that Notepad was started with an empty file:

```
notepad.exe pid: 768
Command line : "C:\WINDOWS\system32\notepad.exe"
```

Name	Pid	Ppid	Thds	Hnds	Time
System	4	0	57	288	Thu Jan 01 00:00:00 1970
smss.exe	388	4	3	19	Fri Jun 19 08:28:45 2009
csrss.exe	608	388	13	403	Fri Jun 19 08:28:48 2009
winlogon.exe	636	388	20	645	Fri Jun 19 08:28:49 2009
services.exe	688	636	16	345	Fri Jun 19 08:28:50 2009
lsass.exe	700	636	21	351	Fri Jun 19 08:28:50 2009
svchost.exe	860	688	17	195	Fri Jun 19 08:28:51 2009
svchost.exe	936	688	11	247	Fri Jun 19 08:28:51 2009
svchost.exe	1028	688	93	1653	Fri Jun 19 08:28:52 2009
svchost.exe	1076	688	6	82	Fri Jun 19 08:28:52 2009
svchost.exe	1128	688	12	172	Fri Jun 19 08:28:52 2009
explorer.exe	1512	1420	11	482	Fri Jun 19 08:28:53 2009
spoolsv.exe	1548	688	10	105	Fri Jun 19 08:28:54 2009
ctfmon.exe	1856	1512	1	71	Fri Jun 19 08:28:59 2009
svchost.exe	1884	688	5	107	Fri Jun 19 08:29:01 2009
alg.exe	1836	688	6	105	Fri Jun 19 08:29:06 2009
wscntfy.exe	1840	1028	1	39	Fri Jun 19 08:29:06 2009
notepad.exe	768	1512	1	43	Fri Jun 19 08:31:16 2009
cmd.exe	960	1512	1	33	Fri Jun 19 08:32:37 2009
TrueCrypt.exe	1428	1512	2	136	Mon Jun 22 09:11:24 2009
nc.exe	240	960	3	47	wed Jun 24 09:19:41 2009
cmd.exe	1492	240	1	36	wed Jun 24 09:20:58 2009
notepad.exe	172	1512	1	49	wed Jun 24 09:22:21 2009

Figure 1. List of running processes from memory dump

Image Name	User Name	CPU	Mem Usage
System Idle Process	SYSTEM	98	16 K
System	SYSTEM	00	212 K
wscntfy.exe	student	00	2,356 K
smss.exe	SYSTEM	00	400 K
taskmgr.exe	student	02	4,684 K
csrss.exe	SYSTEM	00	3,024 K
winlogon.exe	SYSTEM	00	5,244 K
services.exe	SYSTEM	00	3,252 K
lsass.exe	SYSTEM	00	900 K
svchost.exe	SYSTEM	00	4,780 K
svchost.exe	NETWORK SERVICE	00	4,104 K
svchost.exe	SYSTEM	00	17,960 K
svchost.exe	NETWORK SERVICE	00	2,808 K
svchost.exe	LOCAL SERVICE	00	3,784 K
spoolsv.exe	SYSTEM	00	4,540 K
explorer.exe	student	00	14,064 K
svchost.exe	LOCAL SERVICE	00	3,688 K
ctfmon.exe	student	00	3,160 K
alg.exe	LOCAL SERVICE	00	3,508 K

Figure 2. List of running processes in virtual machine

Other interesting things could have been noticed from this list.

```
TrueCrypt.exe pid: 1428
Command line : "D:\TC\TrueCrypt.exe"
```

Above lines show that TrueCrypt was running on the system and that encryption was probably being used. Also, TrueCrypt was not started from the system disk but from some other volume. Since there is only one partition on the system it must have been a removable disk. This was exactly the case on the original investigated system.

```
notepad.exe pid: 768
Command line : "C:\WINDOWS\system32\notepad.exe"
Base 0x1000000 Size 0x14000 Path C:\WINDOWS\system32\notepad.exe
...
TrueCrypt.exe pid: 1428
Command line : "D:\TC\TrueCrypt.exe"
Base 0x4000000 Size 0x186000 Path D:\TC\TrueCrypt.exe
...
nc.exe pid: 240
Command line : nc -l -p 80 -t -e cmd.exe
Base 0x4000000 Size 0x10000 Path C:\Documents and Settings\student\Desktop\nc111nt\nc.exe
...
notepad.exe pid: 172
Command line : "C:\WINDOWS\system32\notepad.exe" G:\Secret_document.txt.txt
Base 0x1000000 Size 0x14000 Path C:\WINDOWS\system32\notepad.exe
...
```

Figure 3. Parts of list of DLLs loaded for each processes

PID	Port	Proto	Create Time	Offset
1028	123	17	wed Jun 24 09:17:48 2009	0x009596a0
1128	1900	17	wed Jun 24 09:17:48 2009	0x01117bf8
4	139	6	wed Jun 24 09:17:48 2009	0x0113ce98
1128	1900	17	wed Jun 24 09:17:48 2009	0x04d122a8
936	135	6	Fri Jun 19 08:28:52 2009	0x0503fc20
1028	123	17	wed Jun 24 09:17:48 2009	0x05124c28
4	445	6	Fri Jun 19 08:28:45 2009	0x051ec368
4	138	17	wed Jun 24 09:17:48 2009	0x0533d590
4	445	17	Fri Jun 19 08:28:45 2009	0x054e9e98
700	500	17	Fri Jun 19 08:29:02 2009	0x05abd780
700	0	255	Fri Jun 19 08:29:02 2009	0x05c77930
700	4500	17	Fri Jun 19 08:29:02 2009	0x05c77e00
4	1030	6	Fri Jun 19 08:29:39 2009	0x08849b48
240	80	6	wed Jun 24 09:19:41 2009	0x0889de98
1636	1028	6	Fri Jun 19 08:29:06 2009	0x0a641e18
4	137	17	wed Jun 24 09:17:48 2009	0x0d9d2690

Figure 4. List of open sockets

Another two lines point to a different event:

```
nc.exe pid: 240
Command line : nc -l -p 80 -t -e cmd.exe
```

Netcat was started to listen on port 80 and set to spawn the shell when it gets connected to by a client. It could be confirmed from the Volatility scan of open sockets (Fig. 4):

PID	Port	Proto	Create Time	Offset
240	80	6	Wed Jun 24 09:19:41 2009	0x0889de98

It can also be noticed, from Volatility list of running processes that Netcat has indeed received a connection and has spawned a shell:

Name	Pid	PPid	Thds	Hnds	Time
nc.exe	240	960	3	47	Wed Jun 24 09:19:41 2009
cmd.exe	1492	240	1	36	Wed Jun 24 09:20:58 2009

Unfortunately, Volatility list of open connections was empty. It could have been expected since going to hibernation all open network connections had to be closed. This is something to keep in mind if using hibernation to preserve system state. It seems that running `netstat` and saving its output to a file should be a step before hibernation. In this way list of active connections of investigated system could be preserved.

Another interesting thing was noticed in the list of loaded DLLs. One instance of Notepad was started on an existing file:

```
notepad.exe pid: 172
Command line : "C:\WINDOWS\system32\NOTEPAD.EXE"
G:\Secret_document.txt.txt
```

File is located on a volume named G. Using Windows Sysinternals utility Strings [28] memory dump was searched for references to "G:\ ". Results were saved to a file. This file, which provides (memory offset: string) mappings was used as an input to Volatility function strings. The output of this function is process that corresponds to these mappings (Fig. 5). Several similar entries that connect volume G:\ with TrueCrypt process (1428) were found.

```
131967640 [1428:128a98 ] G:\
```

Address	Process	String
125371230	[1512:2dcd35e]	G:\
131967640	[1428:128a98]	G:\
133206102	[1512:2e0203e]	G:\
146973888	[172:763e14c0]	G:\
147373784	[1428:127ed8]	G:\
148738261	[1512:1c620d5]	G:\
2996730	[kernel:802db9fa kernel:e213d9fa]	tc_container.tc
14046283	[kernel:ca88044b]	tc_container.tc
31472756	[1428:3b4c74]	?C:\Documents and Settings\student\Desktop\tc_container.tc
31472892	[1428:3b4cfc]	C:\Documents and Settings\student\Desktop\tc_container.tc
31472954	[1428:3b4d44]	tings\student\Desktop\tc_container.tc
31473000	[1428:3b4d7c]	Settings\student\Desktop\tc_container.tc
31473076	[1428:3b4db4]	Settings\student\Desktop\tc_container.tc
31473132	[1428:3b4dec]	\Documents and Settings\student\Desktop\tc_container.tc
33998672	[1428:17a750]	C:\Documents and Settings\student\Desktop\tc_container.tc

Figure 5. Part of memory offset:string mappings

Using the same utilities connections between TrueCrypt process (1428) and a file on a desktop `tc_container.tc` were found in the memory dump.

```
31472892 [1428:3b4cfc] C:\Documents and Settings\student\Desktop\tc_container.tc
```

The file `tc_container.tc` represents TrueCrypt encrypted file container. Next step would be to search the memory dump file for cached passwords and encryption keys as it was suggested in [19]. As it was previously stated decryption using memory content is out of scope of this paper and was not tried.

Above processes, found in the volatile memory copy from the original system, were started in the virtual machine. Similarly, the opened files found in the memory copy were open with the same applications in the virtual machine. No new information was obtained but the virtual machine was brought close to the state the original system was in before hibernation. If need be, this would enable further analysis of relationships among processes and files on the system.

After testing was over the virtual machine was stopped. All relevant files: the image of the original hard disk partition, `hiberfil.sys` and the memory dump created from it were checked for changes. Hash values for all three of them had the same values as for the original files. The system state has not changed, meaning that evidence was not altered. Also, the investigation process and steps taken are verifiable and repeatable. Accepted principles in forensics were not violated.

VII. CONCLUSION AND FUTURE WORK

The proposed combination of static and live digital forensic analysis in a virtual environment offers new possibilities. Investigators can now play with a system image and volatile memory data trying to figure out the exact state of the original investigated system and sequence of events that led to it. All advantages of static analysis are still there. In addition, data usually available only during live analysis is on hand. Even process hidden from live analysis, like rootkits, are visible thanks to volatile memory copy. Virtual environment enables booting the image up for interactive investigation similar to live analysis. The best thing is that the original image is preserved and unchanged and that analysis is repeatable. This should make evidence obtained by using this approach acceptable in a court of law. It has to be repeated that this is

still research and proposed approach should be thoroughly tested before real world usage.

There are some ideas on future usage and improvements. One usage could be to put a virtual copy of the system online with the same IP address and open network ports acting as a honeypot. Idea is to attract the same attackers that compromised the machine for the first time trying to locate them more precisely.

It would be interesting to establish with authority what are the changes made to the system in preparation for hibernation apart from closing network connection. Future work will also be directed towards trying to enable return from hibernation in virtual environment. This could be achieved by adjusting hibernation file for different hardware environment offered by virtual machine. SandMan project started work on writing to hibernation file. Another approach to this issue could be to copy data from hibernation file to snapshot file of virtual machine. Both files contain picture of the memory plus some additional info. It should be possible to convert one format to the other.

REFERENCES

- [1] M.A. Caloyannides, "Forensics Is So Yesterday," *IEEE Security and Privacy*, vol. 7, Mar. 2009, pp. 18-25.
- [2] B.D. Carrier, "Digital Forensics Works," *IEEE Security and Privacy*, vol. 7, Mar. 2009, pp. 26-29.
- [3] B. Hay, M. Bishop, and K. Nance, "Live Analysis: Progress and Challenges," *IEEE Security and Privacy*, vol. 7, Mar. 2009, pp. 30-37.
- [4] V. Roussev and G.G. Richard III, "Breaking the performance wall: The case for distributed digital forensics," *Proceedings of the 2004 Digital Forensics Research Workshop*, 2004, pp. 75-82.
- [5] I.I.I. Golden G. Richard and V. Roussev, "Next-generation digital forensics," *Commun. ACM*, vol. 49, 2006, pp. 76-80.
- [6] F. Adelstein, "Live forensics: diagnosing your system without killing it first," *Commun. ACM*, vol. 49, 2006, pp. 63-66.
- [7] M.M. Pollitt, "Principles, practices, and procedures: an approach to standards in computer forensics," *Second International Conference on Computer Evidence*, 1995, pp. 10-15.
- [8] D. Bem and E. Huebner, "Computer forensic analysis in a virtual environment," *International Journal of Digital Evidence*, vol. 6, 2007.
- [9] CERT, "Live View," <http://liveview.sourceforge.net/>, Jun. 2009.
- [10] C. Waits, J.A. Akinyele, R. Nolan, and L. Rogers, *Computer Forensics: Results of Live Response Inquiry vs. Memory Image Analysis*, CERT, 2008.
- [11] B.D. Carrier and J. Grand, "A hardware-based memory acquisition procedure for digital investigations," *Digital Investigation*, vol. 1, 2004, pp. 50-60.
- [12] A. Schuster, "Searching for processes and threads in Microsoft Windows memory dumps," *Digital Investigation*, vol. 3, Sep. 2006, pp. 10-16.
- [13] A. Schuster, "Pool allocations as an information source in Windows memory forensics," *International conference on IT-incident management and IT-forensics*, 2006, pp. 104-115.
- [14] J.D. Kornblum, "Using every part of the buffalo in Windows memory analysis," *Digital Investigation*, vol. 4, Mar. 2007, pp. 24-29.
- [15] A. Schuster, "The impact of Microsoft Windows pool allocation strategies on memory forensics," *Digital Investigation*, vol. 5, 2008, pp. 58-64.
- [16] R.B. Van Baar, W. Alink, and A.R. Van Ballegooij, "Forensic memory analysis: Files mapped in memory," *Digital Investigation*, vol. 5, 2008, pp. 52-57.
- [17] N.L. Petroni, Jr, A. Walters, T. Fraser, and W.A. Arbaugh, "FATKit: A framework for the extraction and analysis of digital forensic data from volatile system memory," *Digital Investigation*, vol. 3, Dec. 2006, pp. 197-210.
- [18] B. Shatz, "BodySnatcher: Towards reliable volatile memory acquisition by software," *Digital Investigation*, vol. 4, Sep. 2007, pp. 126-134.
- [19] A. Walters, N.L. Petroni Jr, and I. Komoku, "Volatools: integrating volatile memory forensics into the digital investigation process," *Black Hat DC*, vol. 2007, 2007.
- [20] A. Case, A. Cristina, L. Marziale, G.G. Richard, and V. Roussev, "FACE: Automated digital evidence discovery and correlation," *Digital Investigation*, vol. 5, 2008, pp. 65-75.
- [21] C. Betz, "memparser," 2005, <http://www.dfrws.org/2005/challenge/memparser.shtml>, (accessed Jul. 2009)
- [22] G.J. Garner, "kntlist," 2005, <http://www.dfrws.org/2005/challenge/kntlist.shtml>, (accessed Jul. 2009)
- [23] Volatile Systems, "The Volatility Framework: Volatile memory artifact extraction utility framework," <https://www.volatilitysystems.com/default/volatility/>, (accessed Jun. 2009)
- [24] A. Schuster, "Memory analysis: "PTFinder Version 0.3.05"," http://computer.forensikblog.de/en/2007/11/pfinder_0_3_05.html, (accessed Jul. 2009)
- [25] N. Ruff and M. Suiche, "Enter Sandman (why you should never go to sleep)," *PacSec applied security conference*, 2007.
- [26] T.C. Foundation, "TrueCrypt-Free Open-Source On-the-fly Encryption," <http://www.truecrypt.org/>, (accessed Jul. 2009)
- [27] M. Suiche, "SandManSHELL Project," <http://www.msuiche.net/hibrshell/>, (accessed Jul. 2009)
- [28] M. Russinovich, "Strings - Windows Sysinternals," <http://technet.microsoft.com/en-us/sysinternals/bb897439.aspx>, (accessed Jul. 2009)