Digital Investigation 11 (2014) S45–S53

Contents lists available at ScienceDirect

Digital Investigation

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

OpenLV: Empowering investigators and first-responders in the digital forensics process

Timothy Vidas [a,b,*], Brian Kaplan [a], Matthew Geiger [b]

[a] Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA
[b] Dell SecureWorks, 1 Concourse Pkwy NE 500, Atlanta 30328, Georgia

Keywords:
Triage
Incident response
First-responder
Virtualization
Preliminary forensic examination

A B S T R A C T

The continuing decline in the cost-per-megabyte of hard disk storage has inevitably led to a ballooning volume of data that needs to be reviewed in digital investigations. The result: case backlogs that commonly stretch for months at forensic labs, and per-case processing that occupies days or weeks of analytical effort. Yet speed is critical in situations where delay may render the evidence useless or endanger personal safety, such as when a suspect may flee, a victim is at risk, criminal tactics or control infrastructure may change, etc. In these and other cases, investigators need tools to enable quick triage of computer evidence in order to answer urgent questions, maintain the pace of an investigation and assess the likelihood of acquiring pertinent information from the device.

This paper details the design and application of a tool, OpenLV, that not only meets the needs for speedy initial triage, but also can facilitate the review of digital evidence at later stages of investigation. With OpenLV, an investigator can quickly and safely interact with collected evidence, much as if they had sat down at the computer at the time the evidence was collected. Since OpenLV works without modifying the evidence, its use in triage does not preclude subsequent, in-depth forensic analysis. Unlike many popular forensics tools, OpenLV requires little training and facilitates a unprecedented level of interaction with the evidence.

Introduction

In today's increasingly connected world, criminal investigations are likely to entail a digital component at some stage of the process. Even an investigation of purely physical crimes, such as murder, commonly incorporate the analysis of digital evidence, ranging from cell phone records to the victim's email messages. Unfortunately, the personnel trained to perform forensic analysis of these digital artifacts are over-taxed and the influx of cases leads to backlogs. Yet timely action may be important to hold criminals accountable for their actions or to protect others from further harm.

Various forensics process models have been proposed since DFRWS in 2001 (Reith et al., 2002; Palmer, 2001; Carrier and Spafford, 2003; Beebe and Clark, 2005), but these generally assume that the entire, lengthy process is performed. A later stage common to most models is technical analysis, a stage that necessitates trained specialists and creates the backlog of work already noted. In reaction, the application of the medical field's concept of triage has been proposed in order to quickly assign degrees of importance and urgency to items (Rogers et al., 2006; Casey et al., 2009). With respect to digital forensics, triage typically refers to rapid analysis, possibly on-scene, of digital evidence, with steps to maintain the integrity of the evidence. Since the evidence is preserved, triage does not obviate later, extended analysis using a forensic model. Digital forensic triage can provide investigative leads in a timely manner so that they can be acted upon while still applicable.

* Corresponding author. Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA.
E-mail addresses: tvidas@cmu.edu (T. Vidas), bfkaplan@alumni.cmu.edu (B. Kaplan), matthew_geiger@dell.com (M. Geiger).

http://dx.doi.org/10.1016/j.diin.2014.03.006

In practice, first-responders are often trained to recognize potential digital evidence. However, the typically prescribed action for the responder is to collect the evidence (placing a hard drive in an anti-static bag, for example) or to secure the scene until trained personnel arrive to conduct the acquisition. In either case, the next venue for the digital media is the inbound queue of a forensics lab. This "find and forward" approach places a heavy burden on the lab and its trained personnel. In addition, under this model, the investigator is at the mercy of the lab, often waiting for results in order to further the investigation. A triage model that allows the first-responder or the investigator to generate leads, can not only facilitate faster investigation but could also inform and accelerate analysis by the trained lab technician. Triage does not supplant traditional forensics processes or tools, but can augment and enhance the investigative process.

The primary contribution of this paper is a description of a tool, OpenLV, designed and deployed over the past six years under the name "LiveView." OpenLV aims to meet the demand for an easy-to-use triage tool. As such, OpenLV's target audience is digital forensics practitioners, investigators, and first-responders, though OpenLV has also been used extensively in training and educational settings. OpenLV is a free, 100% GPL-licensed tool.[1] Over the past few years, LiveView has been downloaded hundreds of times per week since originally released. A 2008 survey indicated that 30% of universities and 22% of digital forensics practitioners use the tool in some way (Tu et al., 2012). In addition to incremental updates, such as supporting the use of forensic images of current versions of Windows, the most recent version of OpenLV notably adds support for analysts using Linux to run OpenLV, support for VirtualBox virtualization software, and the ability to handle Cached Domain Credentials (discussed in Section Windows passwords). After years of development, OpenLV is a mature product that not only addresses a digital forensics need, but does so while giving users options regarding host operating system and virtualization software.

The remainder of this paper has the following structure. We first discuss background material in Section Background. In Section OpenLV, we describe the design and usage of OpenLV. Section Windows passwords describes a particular feature of OpenLV, removing the obstacle of authentication in password-protected evidence. Then, in Section Limitations, we provide limitations to the current implementation of OpenLV. We discuss related work in Section Related work and future work in Section Future work. Finally, we conclude in Section Conclusion.

Background

Forensics is often divided into classes, Live and Traditional (or "dead"). Live forensics shares many concepts with incident response (Jones et al., 2006). The user interacts with a running computer in order to identify leads and determine the next investigative steps. Since interacting with the computer necessarily changes its state, purists often shun live forensics. However, the advent of purely memory-resident malware or the need to acquire in-use encryption keys offer little alternative to conducting live forensics (Vidas, 2007; Kaplan, 2008).

Conversely, traditional digital forensics often dictates the duplication of media prior to any other interaction (Jones et al., 2006). Some evidence collection procedures demand that running computers be unplugged from power in order to prevent changes to the hard disk during the shutdown process (Best practices for seizing electronic evidence, 2002). A duplicate copy of evidence is often called an *image*. A forensics image is made by copying data to a second physical hard drive or to one of many forensic files types. A dd or (raw) image is created by simply copying data blocks from the target device to a file. Other file types improve upon this simple copy strategy by improving redundancy, storing metadata, and reducing file size with compression. In addition to the dd/raw file type, popular file types include Guidance Software's proprietary E01 format and the open Advanced Forensics Format (AFF) (Garfinkel et al., 2006). When creating forensic images, the creator may choose to duplicate the entire disk, or some subset such as a disk partition.

In additional to the general digital forensics landscape that guided the creation of OpenLV, we also provide some foundation surrounding modern virtualization platforms. VMware produced one of the earliest virtualization products for personal computers and now maintains a leading line of commercial products. VMware offers free and commercial products targeting desktop users (as opposed to data centers) in the form of its Workstation, Player, Fusion and Server range of virtualization platforms. Competing products also exist in free and commercial forms, such as Microsoft's Virtual PC, Parallels' Desktop, and Oracle's VirtualBox. For brevity, we provide background on the underlying mechanics of VMware's implementation, but general principles hold for most of these desktop virtualization products.

Fundamentally, virtualization software allows for the emulation of general computing hardware, such as the CPU, graphics card, hard disk drive, etc, on a host computer system. In this way, one physical machine (the *host*) can be used to run multiple instances of various operating systems each within a *virtual machine* (VM). The VMs each run independently from other VMs and all external interaction via network or human interface devices is mediated by the virtualization software.

Structurally, on the host, virtual machines typically consist of two core components[2]: a virtual hardware specification and a data store. VMware's desktop products store the virtual machine specification in a plain-text `.vmx` file. This file dictates what hardware settings will available to the virtual machine. For example, as shown in Fig. 1, the `.vmx` file may specify the amount of RAM, if a virtual floppy disk drive is to be present, and BIOS settings.

Similarly, the data store specifications reside in a plain-text configuration file. This `vmdk` file specifies the type of virtual drive and information about its disk geometry. VMware products support different types of virtual disks. When

[1] OpenLV may be obtained at http://www.openlv.org/.

[2] There may be other files storing the "physical" memory of the VM, additional settings, a binary BIOS, snapshots, etc.

```
memsize="512"
floppy0.present = "FALSE"
bios.bootDelay = "5000"
```

Fig. 1. A small code snipped of a VMware .vmx file specifying 512 MB of RAM, no floppy disk drive attached, and a 5 s BIOS boot delay.

creating a VM, one can pre-allocate the space for a "FLAT" 20 GB virtual disk which results in a corresponding 20 GB file on the host. Alternately, one can create a "SPARSE" disk that will start much smaller on the host, but be presented as 20 GB in the VM. Fig. 2 shows a portion of a `vmdk` for a 5.4 GB disk (this example comes from the vmdk spec[3]). Different products also support different variations on this theme or even permit a physical disk to be dedicated to the VM.

OpenLV

An investigation may benefit from having professionals of varying experience working together toward the common goal. Different models of interaction have been suggested (Bem and Huebner, 2007; Yasinsac et al., 2003), often positing that less experienced users may be able to assist the investigative process, despite a lack of training or domain-specific knowledge. OpenLV was created with exactly this use-case in mind. In order to reduce the load on specialists, OpenLV must be easy to use by investigators with little digital forensics training. Likewise, the investigator must be able to work with the tool to quickly identify leads.

OpenLV is a tool written in Java that facilitates the loading of digital forensics images into a virtualization platform. The straightforward graphical user interface requires the user to configure only a handful of settings, many of which are pre-populated with reasonable settings. Fig. 3 shows the main interface, where the user can configure the amount of memory, system time, operating system, and the location of the input evidence and output virtualization files. Some settings, such as the size of the virtual hard disk, are inferred from the input evidence. Many of the system-specific settings are written to a new virtualization configuration file. Once configured, the user invokes the virtualization software and interacts with the evidence system as with any virtual machine (as shown in Fig. 4).

Virtual machine specification

Many of the user-configurable settings seen in Fig. 3 translate directly to the virtual machine specification. The settings shown in Fig. 3 would result in a `.vmx` file shown in Fig. 5. A `vmdk` file representing the single IDE hard disk is created in the user-specified output directory. VM memory is defined as 512 (MB) as specified (this setting was also pre-populated). Several time-related settings are also specified in order to prevent the virtualization software from syncing the VM clock with the host clock.

In Fig. 3 the user has specified "Windows XP" for the VM operating system. OpenLV presents a list of operating

[3] VMware's vmdk spec can be found at http://www.vmware.com/support/developer/vddk/vmdk_50_technote.pdf?src=vmdk.

```
createType="twoGbMaxExtentSparse"
RW 4192256 SPARSE "test-s001.vmdk"
RW 4192256 SPARSE "test-s002.vmdk"
RW 2101248 SPARSE "test-s003.vmdk"
ddb.adapterType = "ide"
ddb.geometry.sectors = "63"
ddb.geometry.heads = "16"
ddb.geometry.cylinders = "10402"
```

Fig. 2. A code snippet from a VMware .vmdk file specifying a 5.4 GB hard disk drive. This text file would be accompanied by a three other files test-s00*.vmdk, which together comprise the 5.4 GB of space the VM would use as an IDE hard disk. The twoGbMaxExtentSparse type creates files that are always under 2 GB to accommodate filesystem limitations on the host. This is a subset of an example in the vmdk specification.

systems supported by the installed virtualization platform (including all modern Windows versions), but also permits automatic operating system detection. If the user selects "Automatic," OpenLV will attempt to automatically determine Windows operating system types in the evidence by querying the `ProductName` value in the `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion` registry key.

Virtual disk creation

For dd/raw images, OpenLV creates a `vmdk` file with settings that map the areas of the image to that of a virtual disk. The partition table is read from the Master Boot Record (MBR) in the image and the appropriate calculations

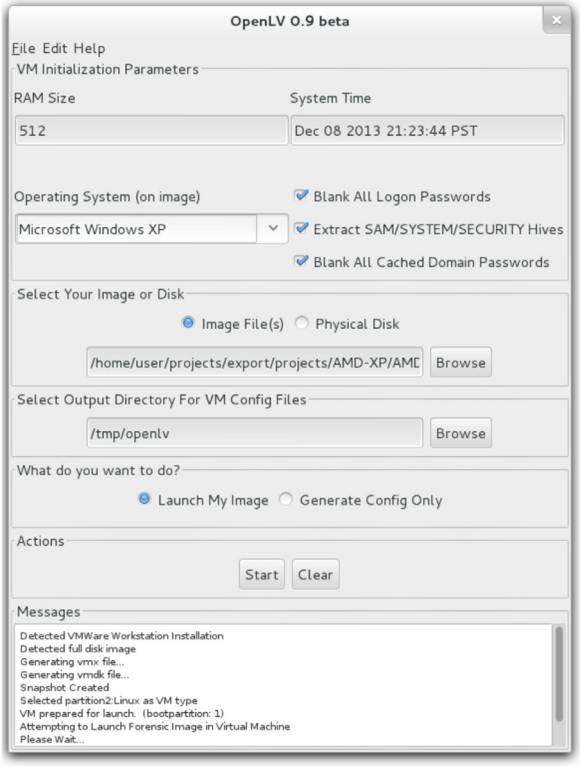

Fig. 3. The main interface for OpenLV. The user is able to boot a virtual machine from either forensic image file(s) or from a physical disk. Several virtual machine settings, such as the system time, are configurable.

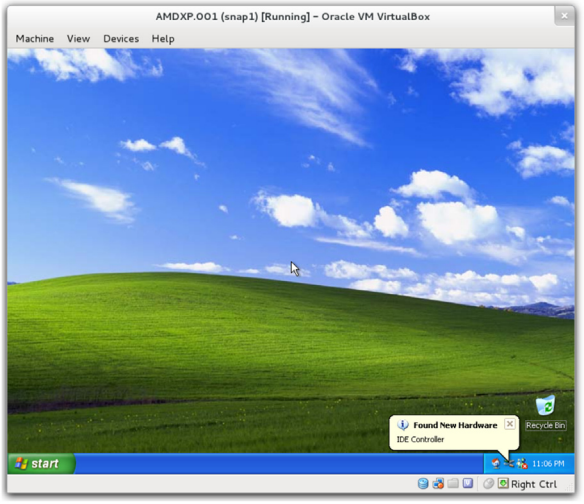

Fig. 4. The end result of using OpenLV with collected evidence is a virtual machine ready for user interaction. In this case the host machine is a Fedora 17 Linux host with VirtualBox virtualization software, and the collected image file is Windows XP.

are performed to populate sector, head, and cylinder information in the `vmdk`. The AMD-XP evidence specified in Fig. 3 is a "split image," meaning that when the evidence was collected, the responder elected to capture the evidence split into several files instead of one. This might be done to keep each file under a specific size to meet filesystem limitations, or to permit storage on DVD, etc. A set of dd/raw split images can be recombined into a single dd/raw image simply by concatenating all the split images into one. However, this would require twice the storage space on the host machine. Instead of requiring concatenation, OpenLV creates a `vmdk` mapping the split images together. The resulting virtual disk specification can be seen in Fig. 6.

```
config.version = "8"
virtualHW.version = "3"
floppy0.present = "FALSE"
displayName="AMDXP.001"

ide0:0.present = "TRUE"
ide0:0.fileName = "/tmp/openlv/AMDXP.001.
    vmdk"
ide0:0.deviceType = "disk"
ide0:0.mode = "persistent"
ide1:0.present = "TRUE"
ide1:0.fileName = "auto detect"
ide1:0.deviceType = "cdrom-raw"

memsize="512"
rtc.starttime="1386537824"
tools.syncTime="FALSE"
time.syncronized.continue="FALSE"
time.syncronized.restore="FALSE"
time.syncronized.resume.disk="FALSE"
time.syncronized.resume.memory="FALSE"
time.syncronized.shrink="FALSE"
time.syncronized.tools.startup="FALSE"
isolation.tools.setOption.disable="TRUE"
guestOS="WindowsXP"
```

Fig. 5. The resulting .vmx from the settings specified in Fig. 3. In addition to setting the time, options are set to prevent the virtualization software from subsequently updating the time.

The `monolithicFlat` type specifies a pre-allocated disk. This way the virtualization software will utilize the dd/raw forensic image as if it was a flat virtual disk type created for a virtual machine. The lines beginning with RW, denote the hard disk extents. The RW indicates that the data areas are readable and writable, the numeric values reflect data area (in sectors), the `AMDXP.00?` are the split images the user specified, and the 0 is the offset into the image that data begins. The data areas must be marked as writable so that the virtualization software will permit the guest OS to perform writes; mitigating alteration of evidence will be discussed in Section Booting the VM.

Partition images are handled similarly, but a partition does not contain enough information for a virtualization platform to boot the operating system. For this reason, a basic MBR compatible with the operating system is added to a virtual disk mapped to the partition image. Some Microsoft Windows operating systems require the serial number to be present in the MBR, for these systems, the MBR is further modified to include this value as specified by the first four bytes of the `HKLM\SYSTEM\MountedDevices` registry key.

For physical devices, the `vmdk` is prepared slightly differently, specifying a physical disk instead of a virtual disk; virtual machines can be backed by a real, physical disk. For these types of images the local drives attached to the system are queried (e.g. `\\.\PhysicalDrive`) including IDE, USB, Firewire, etc. devices. When the user selects a physical drive from the drop-down selection menu, OpenLV creates a physical disk `vmdk` file specifying the appropriate attached device.

For other types of images, OpenLV relies upon third party software to interpret the image. Commercial software such as Mount Image Pro, FTK Imager, or Guidance Software's Physical Disk Emulator are capable of interpreting many types of images. Using this software, images that are not in dd/raw format can be mounted in such a way that they appear to the host system as physical devices. In this way, OpenLV can use these images to create virtual machines just as OpenLV uses physical devices.

```
version=1
CID=fffffffe
parentCID=ffffffff
createType="monolithicFlat"

RW 3072000 FLAT "AMDXP.001" 0
RW 3072000 FLAT "AMDXP.002" 0
RW 3072000 FLAT "AMDXP.003" 0
RW 787456  FLAT 'AMDXP.004" 0

ddb.adapterType = "ide"
ddb.geometry.sectors = "63"
ddb.geometry.heads = "254"
ddb.geometry.cylinders = "625"
ddb.virtualHWVersion = "3"
```

Fig. 6. The resulting .vmx from the settings specified in Fig. 3. This particular image is a "split image" consisting of four dd/raw files.

Booting the VM

Regardless of evidence type, once the virtual machine settings are fully configured a VM snapshot is created. The creation of a snapshot prevents the virtualization software from attempting to perform disk writes to the evidence data, highly desirable behavior for a forensic tool. As the user interacts with the evidence, disk writes are re-directed to a differential snapshot file by the virtualization software. This way the VM executes without error even if the logical files are marked read-only in host filesystem, or if a physical disk is connected via a hardware write-blocker.

If an Intel-compatible driver is not already present in the image, OpenLV immediately, prior to boot, takes advantage of this snapshot to install a Intel-compatible disk driver. Without this driver, booting the virtual machine in an emulated Intel BX440 environment (as VMWare does) would result in the infamous "blue screen of death." Since the virtual machine is not yet executing, the standard driver installation process cannot be followed. Instead the appropriate driver is extracted from an existing cab file found in the image, or a surrogate is copied into the virtual hard disk drive. Then several registry keys are added to the `HKEY_LOCAL_MACHINE` hive so that the driver is loaded with the `CurrentControlSet`.

In addition to the evidence-preserving snapshot, several design decisions were made with digital forensics in mind. For example, the resultant virtual machine has no network interface configured. This precludes communications to or from the machine with remote systems on the Internet, and the machine is not able to interact with any external service. The user may later add a host-only or even Internet-reaching network connection using standard configuration tools. Similarly, when specifying a disk image as input, OpenLV will check the file system to see if the image files are marked as read-only, and if not, prompts the user with an offer to set this attribute. Setting image files to read-only is generally a good practice for forensic data, and adds a layer of protection to collected evidence.

OpenLV allows the examiner to specify the date and internal clock time the virtual system will start with, and configures the virtualization platform to refrain from updating the time. This feature has a range of applications, from preserving the functionality of time-limited software to avoiding time-triggered operations that could affect system state that the examiner wants to preserve.

Use cases

The flexibility to work with a variety of image formats means OpenLV can be used throughout the investigative process. Even users with specialized forensics training may use OpenLV for a variety of reasons including speed, the ability to use live-response tools or access to software installed in the evidence system.

Field users can interact with computers on-scene to obtain investigative leads without the risk of modifying potential evidence. Similarly, field users can use OpenLV to aid in determining if the computer warrants further analysis. Without a safe way to analyze media on-site, all available digital media may be sent wholesale to a forensics lab, further contributing to the lab backlog. When assessing computers on-site, OpenLV's capability to use physical disks to create VMs allows responders to connect hard disk drives to a hardware write-blocker and safely boot a virtualized system for review. Since these physical drives are original evidence (not copied), the use of a hardware write-blocker is recommended to preserve the candidate evidence.

Field users may have some training or possess basic PC administration skills. These users may use OpenLV to interact with candidate evidence using incident response or administration tools and techniques.

Lab users can quickly assess the state of the machine at collection time. The lab user will often have access to specialized digital forensics tools capable of listing the files present on the desktop, displaying the desktop background image, and enumerating the most recently used documents. However, using OpenLV, all of this information is immediately available to the user after the virtual machine starts.

Encountering uncommon software and data formats can drastically slow an investigation. Consider special accounting or computer-aided design software. Such software is expensive and may use file types that analysis tools cannot readily interpret. Worse, file types may be readable only by versions of software that are no longer be available, and the vendors may be uncooperative or even defunct. By using OpenLV, the software that was already installed on the acquired computer can be used to open and view files found in the image.

Forensics professionals presenting evidence in courtroom settings can also use OpenLV to illustrate findings from examined computers in a more natural, easily grasped way for a non-technical audience. In the courtroom, the user can easily demonstrate interaction, repeatedly if necessary, including the use of any specialized software residing on the system.

OpenLV's flexibility and ease of use has led to its application in a variety of unanticipated settings, from data recovery to gaining access to encrypted drives (Butler, 2009; Geiger, 2008).

Windows passwords

Computer passwords impede interaction with the evidence, the primary goal of OpenLV. So, OpenLV takes steps to remove this barrier. Windows local user account passwords can be blanked, effectively making these accounts password-less for the examiner using OpenLV. For machines that are joined to a Windows Domain, the cached domain credentials can be blanked, similarly removing login obstacles. Still, it may be useful in the investigation to recover the user's password, either because the credentials may be reused elsewhere or because the password used is important in itself. For this reason, OpenLV can export the original, cryptographically-protected credentials for use in password cracking software. The remainder of this section details the exportation and blanking of these Windows credentials.

Local passwords

Local passwords are stored in a hashed form in the Security Accounts Manager (SAM) registry file. Combined with information from the SYSTEM registry hive, these hashes can be extracted, and the accounts' passwords can be set to

```
0x8, 0x5, 0x4, 0x2, 0xb, 0x9, 0xd, 0x3, 0x0, 0x6, 0x1, 0xc, 0xe, 0xa, 0xf, 0x7
```

Fig. 7. Static descrambling key used to derive the system bootkey, required as part of password decryption. This key is the same for all instances of the Windows operating system.

blank in the VM snapshot. Here we describe the steps necessary to extract and blank local passwords with a system that is "protected" with a bootkey (also called syskey).

The bootkey is retrieved from the SYSTEM hive. The bootkey is stored across several registry keys each under `\\Control\\Lsa` in the current ControlSet: `JD`, `Skwe1`, `GBG`, and `Data`. For each of these keys, data is stored as 16 UTF-16LE bytes. From a practical standpoint, this means that the hexadecimal representation of the bytes are stored one character at a time and are separated by 0. For example, the value 0x157b is stored in the registry as '1','0','5','0','7','0','b'.

Once the values are obtained from the various SYSTEM keys, the resulting data must then be descrambled using a static key, shown in Fig. 7. The nth byte of the bootkey is derived by using the nth byte of the static key as an index into the bytes retrieved from the registry. Therefore, the first byte of the bootkey will always be the 8th byte of the recovered data (the first byte of the static key is 0x8).

With the bootkey value calculated, the hashed bootkey can be retrieved from the SAM and decrypted. The MD5 hashed bootkey is 32 bytes of encrypted data residing 0x80 bytes into `\\SAM\\Domain\\Account\\F`. To decrypt, OpenLV first extracts 16 bytes from 0x70 bytes into the same registry value. The MD5 update function[4] is used to transform the hash using these 16 bytes, then updated again with the bytes represented in Fig. 8(keyA), then with the bootkey recovered previously, and finally the bytes represented in Fig. 8(keyB). The resulting MD5 is then used as an RC4 key to decrypt the hashed bootkey.

Each local user's password is cleared in the SAM file found in the image. For each user, the user's RID is determined from registry settings. This RID is used to update the `\\SAM\\Domains\\Account\\Users\\RID\\V` key for each user. Setting the LM (offset 0xa0) and NT (offset 0xac) structures to 0 indicates that these users have no password.

The hashed bootkey previously recovered is used to decrypt each password hash in the SAM for export and possible password cracking. The encrypted LM and NT passwords are DES encrypted as a function of the users RID. Key1 is comprised of bytes 0–3 of the RID, then bytes 0–2 of the RID (again). Key2 is comprised of byte 3 of the RID, followed by bytes 0–2 and 0–2 again. The first hash block is decrypted with the DES algorithm under Key1, and the second hash is decrypted with DES under Key2. For a syskey protected machine, an RC4 key is derived from a magic string ("LMPASSWORD" for LM hashes, "NTPASSWORD" for NT hashes). This RC4 key is used to decrypt the data prior to the two DES decryptions.

The software used to perform the above decryption is lengthy and complex. Pseudo-code detailing local password recovery is shown in the Appendix A.

[4] MD5 is a block-based hashing algorithm. The *update* function is used to continue an existing hash calculation with a new block. The *digest* function is used to complete a hash calculation.

Domain passwords

For machines that are part of a domain, authentication is not performed against the local SAM. Instead, authentication is performed against a network server known as a Domain Controller. When authenticating against a Domain Controller, Windows caches the domain credentials for up to 10 of the most recent users to authenticate from the machine.

The mechanics OpenLV implements for clearing cached domain credentials are similar to those described for the local accounts. The registry hives that Windows uses to store these credentials are different, as is the overall algorithm. However, the same basic hashing and cryptography primitives are employed as is the methodology of storing components in various registry locations.

Unlike the local SAM described above, the Local Security Authority (LSA) secrets for cached domain credentials are stored in the SECURITY hive. In particular, the LSA secret named `NL$KM` is retrieved and then decrypted using the aforementioned bootkey, and a Policy Secret Encryption Key (PSEK). The PSEK is obtained from `\\Policy\\PolSecretEncryptionKey\\@`. An MD5 is created with system bootkey, then it is MD5 updated 1000 times with the PSEK. The resulting MD5 is used to create an RC4 key to decrypt the final value of the LSA key. The LSA key is used to DES decrypt the `NL$KM` secret. A Java code section showing the algorithm can be seen in Fig. 9.

Blanking a cached domain credential is not as simple as setting the value to 0. Instead the password is set to an MD4 hash derived from the empty string, updated with the digest of the string (in this case empty), and finally the username. Involving the username in this calculation, essentially a salt, requires OpenLV to calculate a "blank password" hash for each user.

Limitations

OpenLV's origins are fairly Windows-centric. Early versions of OpenLV only ran on a Windows host, and only with VMware Server or Workstation. Similarly, automatic operating system detection and password blanking only worked on Windows evidence. Even then, certain versions of Windows evidence presented additional complications. For example, local Windows password blanking was limited for Windows NT and cached credentials may only work on Windows NT4, 2000, XP and 2003.

The current version of OpenLV runs on Linux and Windows hosts. Even though the core of OpenLV is written in platform-independent Java, OpenLV interacts with the installed virtualization software in system-specific ways. OpenLV supports a variety of VMware products and Oracle's VirtualBox, each on both host platforms.

As discussed above, OpenLV only natively supports physical devices and dd/raw image types. For other forensic image types, such as Guidance Software's E01 format, additional software is needed to interpret the image type.

```
keyA = !@#$%^&*()qwertyUIOPAzxcvbnmQQQQQQQQQQQQ)(*@&%
keyB = 0123456789012345678901234567890123456789
```

Fig. 8. Static strings used to update an MD5 hash derived from the SAM, specific to a particular user. This hash is updated using data from the registry, keyA, the system bootkey, and keyB, in that order.

OpenLV is still compatible with these types as long as the third party software, such as MountImage Pro, presents the image data as a physical device usable by OpenLV.

In many cases, OpenLV must be run with Administrator or root permissions. For instance, on a Windows host, the PhysicalDevice objects cannot be enumerated without Administrative permission. Likewise, the mechanism for performing registry edits requires Administrative permissions. Special cases, such as secondary data disks and multi-boot systems are not handled as well as the typical case: A Windows machine with a single partition.

While not a limitation specific to OpenLV, it is worth noting that when using OpenLV in a form of first-response that rebooting a suspect machine will result in a loss of volatile information. For this reason, responders should create a formal response plan prior to arriving on-scene. Such a plan may well include some form of memory capture early in the response process.

Related work

Rogers et al. (2006) proposed a field triage process model for digital forensics. The authors observe that in some investigations time is of the essence, and quickly developing leads may be of utmost importance. The triage concept, borrowed from the medical field, is applied to digital forensics as a "process in which things are ranked in terms of importance or priority." OpenLV can be used in such a model to inform the decisions on what evidence deserves analytical priority.

Bern and Huebner (2007) suggest employing two teams, each possessing different skills and tools. These two teams work in tandem and the authors suggest that this may lead to faster results, though no concrete evidence is given. Similarly, Yasinsac et al. (2003) documented the concept of having professionals with various educational backgrounds

```
md5 = MessageDigest.getInstance("MD5");
md5.update(bootKey);

//extract bytes 60-76
byte[] pseks = bSlice(PSEK, 60, 0x10);
for (int i = 0; i < 1000; i++) {
  md5.update(pseks);
}
rc4.setKey( md5.digest() );

//decrypt 48 bytes starting at 12
rc4.decrypt(PSEK, 12, PSEK, 12, 0x30);

//extract bytes 28-44
lsakey = bSlice(PSEK, 28, 0x10);
```

Fig. 9. Code section from OpenLV used to obtain the Local Security Authority (LSA) key, derived from the system bootkey and the Policy Secret Encryption Key (PSEK). This key can be used to decrypt LSA secrets found in the registry.

work collaboratively on a digital forensic investigations. Four classes of professional are detailed each possessing familiarity, understanding, or deep knowledge in sets of digital forensics knowledge. Those with little familiarity and only general understanding of forensics processes can benefit from tools like OpenLV, which enable them to contribute more to the investigative process.

Huebner et al. (2007) suggest recreating the environment being investigated as closely as possible. The authors admit that the accuracy of this recreation is impossible to measure, and use OpenLV software as an example of a step in the right direction. The authors also mention that the investigation of a recreated environment using a tool like OpenLV could be performed in parallel with traditional tools.

Penhallurick (2005) outlined a workflow and manual process for creating virtual machines from forensic disk images, including a technique for resolving virtual hardware conflicts.

OpenLV software has been incorporated into various research projects. For example, OpenLV has been cited in books about open source digital forensics (Altheide and Carvey, 2011), used for malware forensics (Aquilina et al., 2008), and employed in digital forensics education (Pollitt et al., 2008; Hay et al., 2008). In 2007, Hargreaves and Chivers use OpenLV to investigate issues a new operating system may present to digital investigations. Lillard and Garrison (2010) even advocate the use of dd/raw images specifically for compatibility with OpenLV.

Mrdovic suggests combining static and dynamic digital forensics analysis techniques (Mrdovic et al., 2009). Occasionally there is a desire to execute software from with a forensic image. This presents obvious problems to the integrity of evidence and in the technical ability to execute the software. OpenLV is mentioned as a method to permit this type of interaction with the evidence.

Future work

For six years, OpenLV only supported Windows as a host platform. With new support for a host operating system unencumbered by commercial licensing restrictions, OpenLV can be bundled as part of a "live CD." Specifically, support for VirtualBox and Linux enables OpenLV to be bundled as part of a bootable CD that can be used for triage. Such a CD, combined with a sufficient RAM disk or a USB thumb drive to which the virtual disk writes can be directed, could allow inspection of a target machine without physically removing the hard disk drive.

OpenLV currently supports Windows and Linux host platforms. Some may prefer OSX as an analysis host, and future support is planned. Similarly, several features are clearly oriented toward Windows systems, notably the password-blanking features that only work on Windows images. These features could be expanded to handle platforms such as Linux.

Many users employ formats such as AFF or E01 as a standard practice. In order to use OpenLV with these formats, third party tools must be used to emulate a physical device. If OpenLV had built-in support for these formats, OpenLV would be easier for these users to incorporate into their workflow.

Conclusion

OpenLV permits rapid triage of computer systems in the field in a fashion that does not degrade their forensic value. Empowering field personnel with no specialized forensic training to review digital artifacts without delay maintains the velocity of time-sensitive investigations.

The flexibility and usability of OpenLV lends to unanticipated use-cases. OpenLV enables easy presentation of evidence in the courtroom and has found a home in digital forensics education.

OpenLV may also help alleviate critical resource constraints in investigations that involve the review of digital evidence. By enabling investigators to review evidentiary computer systems in a natural fashion, OpenLV can accelerate the discovery of relevant data. The alternative is, too often, an iterative process that requires investigators to wait for information to be produced by forensic personnel in response to their requests. In this context, OpenLV complements, rather than replaces, traditional forensic analysis.

Appendix A. Local Windows account password blanking.

```
scramblekey = 0x8,0x5,0x4,0x2,0xb,0x9,0xd,0x3,0x0,0x6,0x1,0xc,0xe,0xa,0xf,0x7

for each VALUE in "JD" "Skew1" "Data" "GBG"
  scrambled = UTF16LE.decode(SYSTEM\\CurrentControlSet\\Control\\Lsa\\VALUE)

  //descramble bootkey
  for(i = 0; i < bootkey.length; i++)
    bootkey[i] = scrambled[scramblekey[i]]

//16 bytes starting at offset 0x70
fKey = bSlice(SAM\\Domain\\Account\\F,0x70,16)
F80 = bSlice(SAM\\Domain\\Account\\F,0x80,32)
keyA = !@#$%^&*()qwertyUIOPAzxcvbnmQQQQQQQQQQQQ)(*@&%
keyB = 01234567890123456789012345678901234567890123456789

md5 = new MD5 instance
md5.update(fKey)
md5.update(keyA)
md5.update(bootKey)
md5.update(keyB)

rc4.setKey( md5.digest() )

//hBootKey is decrypted in place
rc4.decrypt(hBootKey)

for each RID in SAM\\Domain\Account\Users
  for each StringValue in "LMPASSWORD" "NTPASSWORD"
    // ..\Users\RIDvalue\\V
    passHash = RID\\V
    des1[7] = RID[0-3], RID[1-2]
    des2[7] = RID[3], RID[0-2], RID[0-2]

    md5b = new MD5 instance
    md5b.update(bBootKey)
    md5b.update(RID)
    md5b.update(StringValue)

    rc4.setKey(md5b.digest())
    rc4.decypt(passHash)

    des1.decrypt(passHash)
    des2.decrypt(passHash)

    //export for cracking
    export(pashHash)

    //set the V value to zero
    pashHash = 0
```

Fig. A.10. Pseudo-code demonstrating the local account password extraction and blanking for a system with bootkey (syskey) enabled.

References

Altheide C, Carvey H. Digital forensics with open source tools: using open source platform tools for performing computer forensics on target systems: Windows, Mac, Linux, Unix, etc. Syngress; 2011.

Aquilina J, Casey E, Malin C. Malware forensics: investigating and analyzing malicious code. Syngress Media Incorporated; 2008.

Beebe N, Clark J. A hierarchical, objectives-based framework for the digital investigations process. Digit Investig 2005;2:147–67.

Bem D, Huebner E. Computer forensic analysis in a virtual environment. Int J Digital Evid 2007;6:1–13.

Best practices for seizing electronic evidence. United States Secret Service; 2002. Ed. 2.

Butler JM. Decrypting a pointsec encrypted drive using live view, VMWare, and helix. URL: http://computer-forensics.sans.org/blog/2009/09/11/decrypting-a-pointsec-encrypted-drive-using-live-view-vmware-and-helix; 2009.

Carrier B, Spafford E. Getting physical with the digital investigation process. Int J Digital Evid; 2003:1–20.

Casey E, Ferraro M, Nguyen L. Investigation delayed is justice denied: proposals for expediting forensic examinations of digital evidence. J Forensic Sci 2009;54:1353–64.

Garfinkel S, Malan D, Dubec K-A, Stevens C, Pham C. Advanced forensic format: an open extensible format for disk imaging. In: Advances in Digital Forensics II. Springer; 2006. pp. 13–27.

Geiger M. How to build your own password cracker with a disassembler and a little VM magic. URL, http://conference.hitb.org/hitbsecconf2008kl/materials/D2T2%20-%20Matthew%20Geiger%20-%20How%20to%20Build%20Your%20Own%20Password%20Cracker%20and%20Disassembler.pdf; 2008.

Hargreaves C, Chivers H. Potential impacts of windows vista on digital investigations. In: Proceedings of 2nd Advances in Computer Security and Forensics, Liverpool, UK; 2007.

Hay B, Dodge R, Nance K. Using virtualization to create and deploy computer security lab exercises. In: Proceedings of The IFIP TC 11 23rd IISC. Springer; 2008. pp. 621–35.

Huebner E, Bem D, Bem O. Computer forensics – past, present and future. Inf Secur Tech Rep 2007;8:32–46.

Jones K, Bejtlich R, Rose C. Real digital forensics: computer security and incident response. Addison-Wesley; 2006.

Kaplan B. Ram is key: extracting disk encryption keys from volatile memory (Master's thesis). Carnegie Mellon University; 2008.

Lillard T, Garrison C. Digital forensics for network, Internet, and cloud computing: a forensic evidence guide for moving targets and data. Syngress Publishing; 2010.

Mrdovic S, Huseinovic A, Zajko E. Combining static and live digital forensic analysis in virtual environment. In: ICAT. IEEE; 2009. pp. 1–6.

Palmer G. A road map for digital forensics research – report from the first Digital Forensics Research Workshop (DFRWS); 2001. Utica, New York.

Penhallurick MA. Methodologies for the use of VMware to boot cloned/mounted subject hard disk images. Digit Investig; 2005:209–22.

Pollitt M, Nance K, Hay B, Dodge R, Craiger P, Burke P, et al. Virtualization and digital forensics: a research and education agenda. J Digital Forensic Pract 2008;2:62–73.

Reith M, Carr C, Gunsch G. An examination of digital forensic models. Int J Digital Evid 2002;1:1–12.

Rogers M, Goldman J, Mislan R, Wedge T, Debrota S. Computer forensics field triage process model. In: Conference on Digital Forensics Security and Law; 2006. pp. 27–40.

Tu M, Cronin K, Xu D, Wira S. On the development of digital forensics curriculum; 2012.

Vidas T. The acquisition and analysis of random access memory. J Digital Forensic Pract 2007;1:315–23.

Yasinsac A, Erbacher R, Marks D, Pollitt M, Sommer P. Computer forensics education. Secur Priv IEEE 2003;1:15–23.