JOHNS HOPKINS
APPLIED PHYSICS LABORATORY

11100 Johns Hopkins Road • Laurel, Maryland 20723-6099

May 2019

A Practical Guide to Use of Opal Drives

Release 1.0

Prepared by: D. C. Challener, The Johns Hopkins University Applied Physics Laboratory

This page intentionally left blank.

ABSTRACT

Opal drives are widely deployed media that are a class of self-encrypting drives (SEDs). Based on a specification from the Trusted Computing Group (TCG), such drives have extended characteristics beyond merely being self-encrypting. They have the ability to create multiple independent regions each having individual encryption keys, and read/write controls.

For the most part, those characteristics are not used, for a variety of reasons. The specification is hard to read; it is not easy for end users to create software to exploit the characteristics. Closed-source software generally does not allow for the optional use of the extended characteristics. The open-source software that does allow for manipulation of extended characteristics is not well documented. Lastly, the interaction of operating systems with various architectures that might be created using these characteristics is not widely understood.

This document was written to address the last of these issues, with notes regarding the usage of open-source software that will make it easier to use. Most probably, commercial software will someday exist to produce easy-to-use applications of these additional characteristics as well. This document explores several examples of solutions that are enabled by these new facilities.

This page intentionally left blank.

CONTENTS

CONTENTS (Continued)

CONTENTS (Continued)

This page intentionally left blank.

FIGURES

TABLES

1. INTRODUCTION

The Opal Drive Specification was released by the Trusted Computing Group (TCG) in 2005. It has thus far only been used by the industry for self-encrypting drives (SEDs), as a faster replacement for utilities such as BitLocker, and for making it easy to destroy data on a disk before disposal. Although it works well for these applications, these are the "tip of the iceberg" as far as utility of an Opal drive is concerned.

Unfortunately, the specification is somewhat hard to read, and the implementation is somewhat cumbersome to work with. Through a lot of trial and error, and a lot of help from collaborators, the Johns Hopkins University Applied Physics Laboratory (JHU/APL) team has managed to get a number of things done with these drives. This document is intended to share that information with the community so that new applications will start appearing.

This document does not require any form of deep programming experience. Although the team found it useful to create a few programs to speed up the experimentation, the only tool that is really necessary to use the Opal drive is a version of opaltool called *opaltoolc*, which is available from SOURCEFORGE at http://sourceforge.net/projects/opaltest/files/, or a different program, *sedutil-cli*, which the Drive Trust Alliance has made available at https://github.com/Drive-Trust-Alliance/sedutil/wiki/Executable-Distributions. Both of these projects are open source and both have relatively benign license requirements.

This document explains how an Opal drive works, why someone would want to use one, and how to make use of one. Therefore, it can be used by a home hobbyist, an information technology (IT) manager, or an IT employee who is trying to implement direction from an IT manager. Sections 1 and 2, provide an introduction about the need for Opal drives, which includes the history of how the standard and drives were developed. Section 3 covers the basic architectural components, and Section 4 provides some major use cases which these Opal architectural components support. Section 5 transitions to more detailed information, including developer tools and Opal CLI commands that can be used to perform configuration, key establishment, and other operations on Opal drives. Section 6 includes initial design and development guidance on several options available, and Chapters 7 through 11 provide some Opal command details supporting usage and establishment of partitions, boot loader options, support for read-only operating systems, along with other info that's invaluable for developers.

This page intentionally left blank.

2. HISTORY OF OPAL DRIVES

In the 1990s, engineers at Seagate and IBM began considering how enhancing hard drives could increase their utility. Government rules on the sanitation of drives for re-use were onerous, taking hours to implement. Used and recycled drives were increasingly turning up with confidential information on them. Malware and viruses were starting to be exploited. Hard disk passwords were being broken relatively easily by companies offering that as a service. An "in-line" device was being produced that could be placed between a drive and controller that would encrypt data going in and decrypt data coming out, although it relied on the relatively insecure Data Encryption Standard (DES) algorithm.

Seagate started working internally on producing SEDs, with full support of IBM Corp. marketing, who believed they could sell them. TCG then got involved in creating a standard for such drives; it was called the Opal storage specification.

The Opal storage specification is not limited to hard disks or even to spinning media. It applies to any kind of storage and does not limit itself to a single use case. Hard disks contain a fairly sophisticated processor, a relatively large amount of Dynamic Random Access Memory (DRAM), and an enormous amount of non-volatile memory. As a result, sophisticated operations could be provided by a disk, and some of that was exploited.

Hard disks do not have file systems; that is provided by an operating system (OS). Instead, hard disks refer to locations as logical block addresses [earlier common hardware/software (CHS) or cylinder, head, sector]. This means that whatever the hard disk does is independent of the file system.

The design produced by the Opal committee was clever. The drive would always be encrypting its contents, from the day it was manufactured, using an internal key. The drive would also have a default administrator password that was used to derive a key, which in turn was used to decrypt a data blob that contained the actual data encrypting key. The drive knew its default administrator password; therefore, when it arrived from the factory, it could easily determine its internal key and begin encrypting all data going in and decrypting all data going out without the need for any user input.

Furthermore, the end user could change this data blob with the output of a random number generator by using either his/her own password or a password printed on the drive itself. This would return the drive to a vanilla state and inherently leave the data on the drive irrecoverably encrypted.

However, the drive had many additional capabilities. It could be split into ranges—contiguous, non-overlapping sets of Logical Block Addresses (LBAs)—which could be individually controlled. Upon being created, each range is assigned its own encryption key, and data going into the range will be encrypted with that key and data coming out will be decrypted with that key. Each range has four bits associated with it: WriteLockEnabled, ReadLockEnabled, WriteLocked and ReadLocked. The last two are set "on" after a power cycle, but have no effect

unless the corresponding bit of the first two is also set. Table 2-1 shows the various combinations of these bits and their effect.

In the table, the row outlined in bold represents the drive as it is shipped. At this point, it only has one region (the global region), which consists of the entire drive. When it is powered on, the ReadLocked and WriteLocked bits are set but have no effect because the ReadLockEnabled and WriteLockEnabled bits have not been set.

Table 2-1 Write Enable and Read Enable Bits

WriteLockEnabled	WriteLocked	ReadLockEnabled	ReadLocked	Effect
CLEAR	CLEAR	CLEAR	CLEAR	Region is both readable and writable.
CLEAR	CLEAR	CLEAR	Set	Region is both readable and writable.
CLEAR	CLEAR	Set	CLEAR	Region is both readable and writable, but on the next power cycle, ReadLocked will be set.
CLEAR	CLEAR	Set	Set	Region is writeable but not readable.
CLEAR	Set	CLEAR	CLEAR	Region is both readable and writable.
CLEAR	**Set**	**CLEAR**	**Set**	**Region is both readable and writable.**
CLEAR	Set	Set	CLEAR	Region is both readable and writable, but on the next power cycle, ReadLocked will be set.
CLEAR	Set	Set	Set	Region is writeable, but not readable
Set	CLEAR	CLEAR	CLEAR	Region is both readable and writable, but on the next power cycle, WriteLocked will be set
Set	CLEAR	CLEAR	Set	Region is both readable and writable, but on the next power cycle, WriteLocked will be set
Set	CLEAR	Set	CLEAR	Region is both readable and writable, but on the next power cycle both ReadLocked and WriteLocked will be set.
Set	CLEAR	Set	Set	Region is writeable but not readable, but on the next power cycle WriteLocked will be set.

Table 2-1 Write Enable and Read Enable Bits (Continued)

WriteLockEnabled	WriteLocked	ReadLockEnabled	ReadLocked	Effect
Set	Set	CLEAR	CLEAR	Region is readable but not writeable.
Set	Set	CLEAR	Set	Region is readable.
Set	Set	Set	CLEAR	Region is readable, but on the next power cycle, ReadLocked will be set.
Set	Set	Set	Set	Region is neither readable nor writeable, and the drive does not have the key necessary to recover the encryption key.

Legend:

Represents the states of regions that are used for confidentiality. This state is when it is in use.

Represents regions (e.g., partition tables) that have been write locked for resiliency. These rows cannot be overwritten by malware but are still readable.

Represents the states of regions that are used for confidentiality. This is when the drive is first powered on.

This page intentionally left blank.

3. BASIC ARCHITECTURE OF AN OPAL DRIVE

To understand how to use an Opal drive, the user first must understand a few basic concepts.

3.1 Shadow Master Boot Record

The poorly named Shadow Master Boot Record (MBR) is a small (usually 128 MB) region of the drive that the drive controller handles in a different way from the rest of the hard disk. As shipped, the Shadow MBR is not visible on the drive at all, except through special Opal Advanced Technology Attachment (ATA) commands. With those commands, it can be written to (with administrator privilege) or read (without any privilege at all).

When enabled, the drive reroutes any query made to the first 128 MB of the drive to look instead at the equivalent sector of the Shadow MBR. In this case, the normally visible 128 MB of the drive (called the Normal MBR) is not visible.

When the Shadow MBR is enabled, it is generally readable using regular hard disk calls, but is read-only (i.e., write protected) as far as those calls are concerned. However, a "Done" bit can be set ON, which will make the Shadow MBR then disappear as long as the hard disk does not lose power. Should the Shadow MBR lose power, the Done bit will be reset to OFF, and the Shadow MBR will again replace the Normal MBR when read requests are made, and write requests will simply fail.

Why such a weird device? The Shadow MBR was originally made to provide a means of running software that would allow asking the end user for authorization to decrypt the rest of the drive and then execute a soft reboot into the Normal MBR. By making the region read-only, it became difficult for malware to insert itself into the authorization process and compromise the credentials being offered. This offered a solution to the question: If the drive is fully encrypted, where do we put the software asking the user for authentication to decrypt the drive? As a result, the Shadow MBR has special properties that must be worked around.

3.2 Ranges

Any contiguous set of LBAs (except those in the Shadow MBR) can be identified as a "range" on the drive, as long as it does not overlap another range. Although ranges do not have to be configured to match a Partition (which also has to be a contiguous set of LBAs), it is generally a good idea for them to be matched.

There are 16 ranges available. The first range, the global range (or range 0) consists of all LBAs not in the Shadow MBR or other ranges. The other 15 ranges can be set by the end user. When a range is created, a new encryption key is associated with that range, and all data going into or out of the range will start to be encrypted or decrypted with that key. This has the immediate effect of making the data that used to be reported by the drive as being in that range, disappear. However, if the range is deleted, the LBAs that used to be associated with it return to using the

key associated with the global range, and data that used to be there reappear, assuming they were not overwritten.[1]

All ranges can be set to be Read Protect Enabled or Write Protect Enabled. Setting these bits enables another bit (Read Protect or Write Protect) to then be effective in enforcing the range to be readable or write protected. This unusual double bit was created to allow for the Read Protect and Write Protect bits to automatically be set during a power cycle of the drive. That is, if a range is Write Protect Enabled, then upon a power cycle, the range is read-only. The end user may unlock the range for writing either by turning the Write Protect Enabled bit off (which is effective from that point on) or by turning the Write Protect bit off (which is effective until the next power cycle.) The Read Protect Enabled bit works similarly, as does the Shadow MBR Enabled bit.

The encryption key that is used to encrypt a range is stored in non-volatile memory in the Opal drive itself unless two things are true. If the range is both read-protected and write-protected (they are both enabled and also 'set'[2]), a password, together with random data stored in non-volatile memory in the Opal drive, is used to recreate the encryption key on demand. As shipped, the password used to perform this operation is called the Manufacturer's Secure Identifier (MSID), and it can be retrieved from the drive by asking for it with an Opal-specified ATA command. After creation, the encryption key is never stored in the non-volatile memory of the drive; rather, it is stored in the volatile memory in the drive. To make the drive secure, the end user must change the password that was initialized to the MSID to something else.

3.3 Security Providers

To future-proof the design of the Opal specification, the creators envisioned that someday Opal drives might have capabilities other than range manipulation. They might, for example, contain a cryptographic module. Therefore, the range control commands are grouped into a "Locking" security provider (SP). To control the various SPs, an "Admin" SP was created.

The userid that administers the Admin SP is called either the Security Identifier (SID) or the Physical Security Identifier (PSID). Their initial passwords are the MSID and the PSID, respectively (discussed further later in this document).

The userids that administers the Locking SP are called admin1, admin2, admin3, or admin4. The initial password of admin1 is the MSID. The other userids are not available until they are enabled using one of the admin userids and passwords.

This document describes precisely which userid the end user must employ in various commands to the Opal drive.

[1] A similar thing happens if a range is contracted or expanded.

[2] See the last row in Table 2-1.

3.4 Resetting the Drive

If the drive gets hung up and needs to be reset, and you have forgotten the correct admin password to do that, it is always possible to set the drive into a vanilla state by executing a reset command together with the PSID. The PSID is a 32 alphanumeric password that is physically printed on the drive itself.

Once this command is sent, the random data stored on the drive are "spun" so that they cannot be reproduced, and the Locking SP is set disabled. All ranges are reset, and sometimes the Admin SP is also set disabled. The Admin SP's SID password is reset to the MSID, as is the Locking SP's admin1 password. All other userids in the Locking SP are disabled. It is then in an as-shipped condition, except that the base encryption key will be different.

3.5 Users

There are a number of differences between admin and user passwords. Admin passwords have the authority to change any admin or user passwords. User passwords can only be changed by the user. Admin passwords can change the Shadow MBR bit and the Shadow MBR Done bit. Users can only change those only if they have been given that authority by an Admin. Only Admin passwords can change the Read Protect Enable and Write Protect Enable bits of a range. Admin passwords can change the Read Protect and Write Protect bits of a range. Users can only do that if they have been specifically enabled for a particular range.

The SP userids are fixed by the specification and cannot be changed. The Admin SP has two userids: the SID, which has its password initially set to the MSID (which the drive will reveal upon request), and the PSID, which is set to a number that is usually typed on the physical drive. The SID has all authority, and the PSID only has authority to reset the drive back to the manufacturer's vanilla state.

The Locking SP has 20 userids for some manufacturers, less for others. Admin1, which initially has the MSID as its password, is always enabled. The remaining 19 userids consist of three admin passwords (Admin2, Admin3, and Admin4) and 16 users (User1 to User16).

The differences are particularly important if the user wants to divide a drive into multiple ranges, and only allow each user to be able to enable reading of a range for which they have been associated. In some drives, administrator privileges do NOT allow unlocking a range associated with a user, only being allowed to reset it. These drives are known as eDrives.

3.6 Table Driven

All commands on the drive are executed by reading and writing table entries in the drive. This is somewhat confusing, but it is abstracted for the end user by the various tools that now exist. However, to play with the drive directly, this is important to know.

This page intentionally left blank.

4. USE CASES FOR OPAL DRIVES

4.1 Secure Recycling

Perhaps the most popular use case for an Opal drive is for secure recycling. As most professionals know, simply reformatting a drive is not sufficient to erase the data stored on it; neither is simply erasing the files on the drive.

In the case of solid state drives (SSDs), even overwriting the data *n* times does not work (because of the wear-leveling routines used inside the drive.) Instead, the only way to securely erase an SSD is to physically crush it, grinding it to dust.

Opal drives are always encrypting the data stored on them. Therefore, executing a command to the drive to "spin" the key (forgetting all trace of the key that was used for encrypting that data) is sufficient to crypto-erase the drive. This operation takes a fraction of a second. It can be done with either the PSID printed on the drive itself or the Locking SP's admin1, admin2, admin3, or admin4 userids (assuming the last three have been enabled).

Furthermore, a single range can have its encryption key reset if the drive is to be repurposed, and the main OS has been kept separate from user data.

4.2 Full Disk Encryption

The first thing people think about when they hear about Opal drives is the ability to use it to encrypt all their data. This is a very good use, especially because Opal drives typically have no performance delta between when they are encrypting or not. To the drive, the OS being run on it is irrelevant, and therefore can be used in any environment. However, to use the Opal drive in this way, a user must first provide an authorization password from which the drive will derive the symmetric key used to encrypt and decrypt data sent to or received from the drive.

There is a caution to encrypting the entire drive. If the entire drive is encrypted, the application that asks the user for runtime authorization may be encrypted and hence may not be able to run. There are several workarounds to this problem:

- **ATA compatibility mode.** All Opal drives can be run in compatibility mode. In this mode, the normal ATA hard disk password is used to unlock the drive. Depending on implementation, this may or may not simply hook into the Opal encryption capabilities of the global range. Several vendors, like Seagate, guarantee that it does, whereas others have no comment. If it does not, then many protections may be absent.

- **Using the Shadow MBR to run software that asks for authorization for the drive.** Specialized software that asks for a password for the global range can be bought, written, or downloaded. It allows the user to type in a password to unlock the drive from this read-only range. The software then sets the Done bit and performs a warm reboot into the normal (and now apparently decrypted) drive. This has the advantage

of being able to be installed in a drive that already has a full OS on it without disturbing the existing OS at all. The open-source solution *sedutil-cli* has such a pre-boot image available for free at https://github.com/Drive-Trust-Alliance/sedutil.

- **Creating a read-only pre-boot range.** If starting from scratch, users may wish to create a range at the beginning of the drive from which to control the booting of the system and install their own pre-boot OS. After completion, the user may then wish to make that region read-only to prevent malware attacks on the authorization value. Making a read-only OS image is not particularly difficult because there are many "live" CDs that can boot in a read-only state.

4.3 Encryption To Meet U.S. Government Standards

Commercial Solutions for Classified (CSfC) standards describe how commercial solutions can be used to create systems to protect classified data. Generally, two layers of encryption must be used, and both must use strong cryptography based on different encryption engines.[3] Both layers must also be certified. Using an Opal drive for one layer of the data at rest solution is ideal because it is done at a different level than file system cryptography and as such definitely will be using a different encryption engine than any software applications such as BitLocker. Currently, some SEDs are Federal Information Processing Standard (FIPS) certified and therefore sufficient for sensitive data (like Social Security Numbers); however, approval as a CSfC component[4] is required before a SED can be used for handling classified data.

4.4 Resilience

A favorite use of the Opal drive does not involve using the encryption at all; rather, it uses the fact that regions of the drive may be made read-only. Making the partition table, boot manager, and boot loader read-only can guarantee that no software will be able manipulate the hard drive in such a way that it will not boot to the boot loader. Placing read-only OS images in a read-only range can guarantee that no matter what happens, the drive itself will always allow booting to an OS.

Choosing these various OSs so that repair of a system can be done from them makes a system immune to a lot of the problems an OS is prone to (especially if a hostile nation is targeting a specific organization and/or its employees). This is somewhat tricky to accomplish, but this document provides in-depth details on how to do it.

4.5 Secure Backup

Ransomware is an unfortunate truth today, with malicious parties deciding it would be a good idea to encrypt people's precious personal data (e.g., wedding photos, pictures of deceased relatives) and then sending a ransom demand to retrieve the data. Opal drives provide one way to defend against this. Storing precious memories in read-only ranges can exempt them from this type of malicious meddling.

[3] https://www.nsa.gov/resources/everyone/csfc/capability-packages/#data-at-rest
[4] https://www.nsa.gov/Resources/Everyone/csfc/Components-List/

A read-only OS utility, which automatically boots from the Shadow MBR, retrieves a password from the Trusted Platform Module (TPM) and uses it to perform either a remote or local backup of a user's data. Local backup is achieved by unlocking a partition that is normally read-only using the retrieved password, making an incremental backup (to prevent encrypted files from overwriting unencrypted ones), and then re-locking the partition to be read-only. Once this is achieved, the system is warm booted back to the main OS.

4.6 Secure Scanning

Similar to the problem of secure backup, to be sure the software doing the scanning has not been corrupted, it is necessary to have a safe place from which it executes. Having the scanning software resident in a read-only region of the drive, which executes first at boot time, provides such a place. The idea is to have sensitive information only available if the system has booted to a secure utility OS, which it can use to certify that the main OS is not infected. The secure utility OS can be implemented using TrustedGrub2[5]. Then, the system periodically hibernates; clears the Done bit; warm boots into the trusted Grub2 boot into a read-only OS, which does the necessary scan; and then sets the Done bit and reboots back (through hibernation) into the normal OS. While running, because the scanner has access to sensitive information, it can verify itself to an online database to update itself by making itself read-write, downloading updates, and then making itself read-only again.

4.7 Periods Processing based Cross Domain Solutions

Cross-domain solutions (CDS) provide the "ability to manually and/or automatically access and/or transfer information between different security domains."[6] Many CDSs work simultaneously; that is, both domains can be accessed at the same time. Using an Opal drive, the user can do something that works for many situations but does not provide simultaneous access to more than one domain. In this case, periods processing, a method of sequential operation that processes information of different classification or security levels at distinctly different times[7], is used. In this case, the administrator sets up a system that first boots into a read-only image (locked down by an administrator) and provides an opportunity to boot into one of multiple systems. The read-only system has access to keys sealed in a TPM and is only accessible when the machine has booted into the read-only boot system. When booted, the first action performed is to make certain all partitions are locked and the memory is cleaned upon reboot. The latter is done by setting the Memory Overwrite upon Reset (MOR) bit.

Once the user provides a password and security selection, the software retrieves the appropriate passwords from the sealed data, using the password to authorize the decryption of those passwords. Next, the Done bit is set so that the software can see the normal partition table, which is overwritten with an appropriate partition table that sees the partition that corresponds to the correct security level. The partition table is then again made read-only. The TPM then uses the appropriate password to unlock the appropriate partition, pointed to by the partition table, so that

[5] https://github.com/Rohde-Schwarz-Cybersecurity/TrustedGRUB2
[6] NIST SP800-53 Rev 1
[7] DoD 5220.22-M

it is both readable and writable during that power cycle. A warm boot then allows the system to boot into the OS at the unlocked partition.

Once the system is powered off, all partitions are automatically fully encrypted, and upon power on, the system will again boot into the Shadow MBR. If the system is warm booted, it will continue to boot the same way as its last boot. Figure 4-1 presents a graphical depiction of the various boot cases.

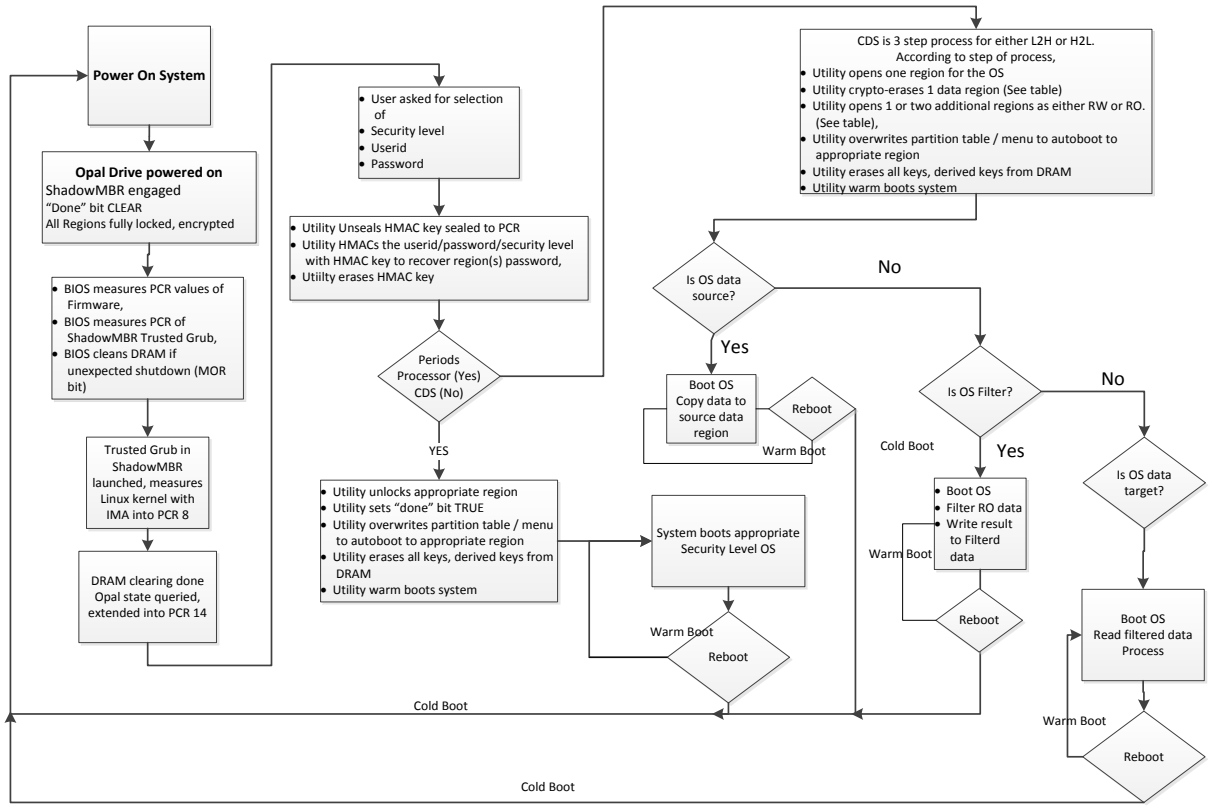

Figure 4-1 Flow Diagram Showing Opal Support for Various Boot Options

This design requires some extra work on the part of the designer—auto-locking of partitions and memory cleaning. It is possible to make a two-level-only CDS (classified and non-classified) that is easier, particularly if Windows is booting in both sides. In this case, booting to the secure side will require unlocking the secure partition, setting the Done bit and warm booting. The partition table on the Normal MBR is made read-only, and the partition table on the Shadow MBR shows the non-classified Windows partition as formatted, but the classified partition as not being a partition. The partition table on the Normal MBR side shows the non-classified Windows as not being a partition and the classified Windows partition as being a partition. The multiple partition tables are not for security, but to prevent Windows from crashing when it sees a non writeable NTFS partition.

The real challenge is preventing the following scenario:

- Booting into normal Windows (thus unlocking that partition).

- Warm rebooting into the secure Windows (thus unlocking that partition).

- While in secure Windows, using special software to copy classified data onto the partition that appears not to be a partition, but which is not encrypted.

This appears to be possible only by actually having code in the read-only section of the boot sequence that locks the normal Windows partition before booting into the secure partition. Although not particularly difficult, it does mean that booting into the normal Windows partition will require a warm boot after requesting a password for the normal Windows OS (but not setting the Done bit).

It is not possible to do the opposite because booting to normal Windows after booting to secure Windows requires a cold boot to get back into the Shadow MBR that can see the normal Windows partition. A cold boot locks the secure windows.

This page intentionally left blank.

5. GETTING STARTED

5.1 Choosing a Drive

Most drive manufacturers make Opal-compliant drives. The following is a list of Opal-complient drives that also are supported by Wave Systems https://wavesys.com/self-encrypting-drive-compatibility-list. Additional drives appear to be added to this list over time, so there may be some missing from this list that had not been verified by Wave Systems as working with their software. Note that The Drive Trust Alliance, which puts out *sedutil-cli*, is doing its own verification of drives, although it is unlikely they have as large a set of drives to test as Wave Systems did.

Note that Samsung announced in 2015 that all of their SSD drives would be Opal compliant. However, because some personal computer (PC) manufacturers have asked Samsung to disable this capability for the drives that they buy, some scrutiny is necessary to determine whether a used drive is Opal-compliant. The easiest way to do this is to look on the drive label for a line with the PSID that looks similar to this:

PSID: 747UV877 6879DYE3 DI93HLVE 97NCD38G

Of course, the alphanumeric numbers will be different on different drives.

5.2 Resetting the Drive

With a new drive, the drive probably does not need to be reset. However, for a drive that has been previously used, or to wipe a drive and start all over, resetting the drive is easy to do. The command can be executed using either *opaltool* or *sedutil-cli*.

5.2.1 Using *opaltoolc* To Revert a Drive to Manufacturing Defaults

In this example, the Seagate drive is attached as a universal serial bus (USB) drive. *opaltoolc* works equally well with the drive attached as a USB or on a Serial Advanced Technology Attachment (SATA) bus.

```
opaltoolc --list
Drive (0): \\.\PhysicalDrive0, Model = TOSHIBA MQ01ACF032,Serial = 24GLC2P6T
Drive (2): \\.\PhysicalDrive2, Model = ST320LT015-9YK142,Serial = W0Q17NX0
```

```
opaltoolc --revertDrive --DRIVE 2 --UID PSID --PW 01234567ABCDEFGH01234567ABCDEFGH
```

5.2.2 Using *sedutil-cli* To Revert a Drive to Manufacturing Defaults

In this example, the Seagate drive is inside the machine. *sedutil-cli* does not currently work with USB-attached drives.

```
sedutil-cli --scan
```

Scanning for Opal 2.0 compliant disks:

\\.\PhysicalDrive0 2 ST500LT025-1DH142 0012DM7

sedutil-cli --yesIreallywantoERASEALLmydatausingthePSID –password <insert-password-here> \\.\PhysicalDrive0

5.3 Enabling Security Providers

Usually, the Admin SP will already be enabled. However, if the drive has been reverted, the Admin SP may not be enabled. These commands will enable both the Admin and Locking SPs.

5.3.1 Using *opaltoolc* to Enable SPs

opaltoolc --activateSP --DRIVE 2 --SP Admin --UID SID --PW MSID
opaltoolc --activateSP --DRIVE 2 --SP Locking --UID SID --PW MSID

5.3.2 Using *sedutil-cli* to Enable SPs

sedutil-cli --activatelockingsp --password MSID

5.4 Taking Ownership of SPs

In this example, the user has to make up a new password called NewPW.

5.4.1 Using *opaltoolc* To Take Ownership

opaltoolc --changePassword --DRIVE 2 --SP Locking --UID Admin1 --PW MSID --TUID Admin1 --TPW <insert new password here> NewPW
opaltoolc --changePassword --DRIVE 2 --SP Admin --UID SID --PW MSID --TUID SID --TPW NewPW

5.4.2 Using *sedutil-cli* To Take Ownership

sedutil-cli --takeownership --password newPassword \\.\PhysicalDisk0
sedutil-cli --setPassword --password SEDUTIL-CLI --newPassword NewPW

5.5 Enabling Users of Locking SPs

This can only be done with *opaltool*. In this example, the admins and the first four users are enabled.

opaltoolc --enableUser --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID Admin2
opaltoolc --enableUser --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID Admin3
opaltoolc --enableUser --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID Admin4

opaltoolc --enableUser --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID User1
opaltoolc --enableUser --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID User2
opaltoolc --enableUser --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID User3
opaltoolc --enableUser --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID User4

5.6 Setting Passwords for Users of Locking SPs

This also can only be done with *opaltool*. In this example, passwords for the users just enabled are set.

```
opaltoolc --changePassword --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID Admin2 --TPW Admin2PW
opaltoolc --changePassword --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID Admin3 --TPW Admin3PW
opaltoolc --changePassword --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID Admin4 --TPW Admin4PW

opaltoolc --changePassword --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID User1 --TPW User1PW
opaltoolc --changePassword --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID User2 --TPW User2PW
opaltoolc --changePassword --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID User3 --TPW User3PW
opaltoolc --changePassword --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID User4 --TPW User4PW
opaltoolc --changePassword --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID User5 --TPW User5PW
```

This page intentionally left blank.

6. DESIGN CONSIDERATIONS

6.1 Tools: *sedutil-cli* or *opaltool*

There are some significant differences and drawbacks to using either of the two tools: *opaltoolc* vs. *sedutil-cli*. Currently, *opaltoolc* only works in Windows. *sedutil-cli* works with either Linux or Windows. *sedutil-cli* only works on drives attached to a SATA connection, whereas *opaltoolc* works with either SATA or a USB-connected drive (although not all USB-to-SATA convertors correctly transmit all ATA commands). Although seemingly ideal to use both tools, there is a significant problem. *sedutil-cli* does not pass the password directly into the drive; instead, it uses pbkdf2 on the password, which performs a number of hash operations on the password before it is passed into the drive. Fortunately, the *sedutil-cli* developer modified the design so a recent version can insert an "-n" after the command, before the parameters, that tells the executable NOT to execute pbkdf2 on the input password. If this is done, the two tools become compatible. A little modification to the code can also remove the use of pbkdf2, or the user can set up a drive with two admin passwords, with Admin1 having the *sedutil-cli* version of the password and Admin2 having the *opaltool* version of the password.

sedutil-cli uses pbkdf2 with Secure Hash Algorithm (SHA)-1 as its hashing algorithm, an iteration count of 75,000 and a salt of

CB AB CC B7 7B 27 47 13 87 73 EC 72 63 3F 34 A0 DB BB DC D7 8B 37 57 23 97 83 FC 82 7E 4F 44 B0

and produces a final size of 32 bytes.

6.2 OSs: Windows or Linux

Because of the automated checking performed by the OS of the underlying file systems on the drive, it makes a difference if the OS being used is Windows or Linux. Windows automatically checks file systems that are Network Terminal File System (NTFS) formatted, whereas some Linux implementations only check file systems on which it is directly running, and some, if they do check that file system, do it in a way that is not disturbed if the file system is read-only.

Part of the file system checking performed by Windows involves writing data to the file system in question, if a partition is not marked as read-only in the partition table itself. Therefore, if a file system is write-protected, Windows will present issues. It will take an inordinately long time to boot the system and thereafter crash periodically. Unfortunately, only Global Unique Identifier (GUID) Partition Table (GPT)-style partition tables are able to mark a partition as being read-only. When using a GPT-style partition table, there are other consequences as well, as discussed in Section 6.3.

6.3 Partition Table Types: MBR or GPT[8]

When disk sizes grew to be larger than 1 TB, the old-style MBR partition table could not cope with that size. A new-style partition table called GUID was created to handle larger hard disk sizes up to 8 ZiB, or 8 billion terabytes. This is about four times the data stored on the Web in 2013, which should be adequate for a while. However, this is not the first time this has happened in the history of hard disks, nor will it be the last.

Old-style MBR partition tables allow for only four primary partitions, one of which can be an "extended" partition. An extended partition then contains another partition table at the beginning of its partition, which allows for more "logical" partitions. Although four partitions have been enough for most drives, four might not be enough to accommodate the new capabilities that Opal drives support going forward.

GPTs allow for a virtually unlimited number of partitions; however, in most implementations, they are limited to 128 partitions, which seems more than adequate.

MBR partition tables only exist at the beginning of a disk, usually in the first 512 bytes.[9] With GPT, the situation is more complicated. Not only is the partition table size variable (according to the number of partitions that can be listed), but when Microsoft OSs are installed with a GPT partition layout, they will install a Microsoft Reserved Partition (MSR) that is usually 128 MB.

Additionally, Windows XP cannot boot from a GPT-formatted disk. (However, XP 64 can use such disks.) Windows Server 2003 and all later versions of Windows can use GPT-formatted disks for data, but may not be able to boot from them.

NOTE

To be able to boot, the version must be 64 bit, and the system must be Unified eXtensible Firmware Interface (UEFI)-based.

To make things even more complicated, when using a UEFI boot system, it **MUST** be GPT and it must have an eXtensible Firmware Interface (EFI) System Partition (ESP), which is about 100 MB. **Windows does not like a drive to have two separate ESPs on it.** Table 6-1 shows the contents of a GPT-formatted disk.

Table 6-1 Contents of GPT Formatted Disk

Title	Contents	Size
GPT	Data about up to 128 partitions	16,000 (typical)
EFI system partition	HAL, loader, other files needed to boot the OS	about 100 MB
MSR	space reserved for the OS for software that used to use hidden space	128 MB

[8] Much of Section 6.3 comes from https://en.wikipedia.org/wiki/GUID_Partition_Table

[9] If a Logical Disk Manager is also used, a 1-MB area at the end of the disk is also used to store metadata. This can cause problems if a Shadow MBR is used to create two partition tables for the drive that are not identical.

Redundant copy of GPT at end of disk	Copy of GPT pointed to within the partition table	16,000 (typical)

Table 6-2 shows details for Unix and Unix-like OSs.

Table 6-2 Details of GPT Support on UNIX and Unix-like Operating Systems

OS Family	Version or Edition	Platform	Read and Write Support	Boot Support	Notes
FreeBSD	Since 7.0	IA-32, x86-64, ARM	Yes	Yes	In a hybrid configuration, both GPT and MBR partition identifiers may be used.
Linux	Most x86 Linux distributions Fedora 8+ and Ubuntu 8.04+[12]	IA-32, x86-64	Yes	Yes	New tools such as gdisk,[13] GNU Parted,[14][15] util-linux v2.23+ fdisk,[16][17] Syslinux, GRUB 0.96 + patches and GRUB 2 have been GPT-enabled.
OS X	Since 10.4.0 (some features since 10.4.6)[18]	IA-32, x86-64	Yes	Yes	Only Intel Macintosh computers can boot from GPT.
MidnightBSD	Since 0.4 to present	IA-32, x86-64	Yes	Requires BIOS	In a hybrid configuration, both GPT and MBR partition identifiers may be used.
Solaris	Since Solaris 10	IA-32, x86-64, SPARC	Yes	Yes	[19]
HP-UX	Since HP-UX 11.20	IA-64	Yes	Yes	[20]

BIOS – Basic Input/Output System

Table 6-3 shows details for Windows 32-bit versions.

NOTE

Windows 7 and earlier do not support UEFI on 32-bit platforms, and therefore do not allow booting from GPT partitions.

Table 6-3 Details of GPT Support on 32-bit Editions of Microsoft Windows[11]

OS Version	Release Date	Platform	Read or Write Support	Boot Support	Notes
Windows XP	2001-10-25	IA-32	No	No	
Windows Server 2003	2003-04-24	IA-32	No	No	
Windows Server 2003 SP1	2005-03-30	IA-32	Yes	No	MBR takes precedence in hybrid configuration[10]
Windows Vista	2006-07-22	IA-32	Yes	No	MBR takes precedence in hybrid configuration[10]
Windows Server 2008	2008-02-27	IA-32	Yes	No	MBR takes precedence in hybrid configuration[10]
Windows 7	2009-10-22	IA-32	Yes	No	MBR takes precedence in hybrid configuration[10]
Windows 8	2012-08-01	IA-32	Yes	Requires UEFI[21]	MBR takes precedence in hybrid configuration[10]

Lastly, Table 6-4 shows details for Windows 64-bit versions.[10]

Table 6-4 Details of GPT Support on 64-bit Editions of Microsoft Windows[11]

OS Version	Release Date	Platform	Read and Write Support	Boot Support	Notes
Windows XP Professional x64 Edition Windows Server 2003	2005-04-25[22]	x64	Yes	No	MBR takes precedence in hybrid MBR configuration[10]
Windows Server 2003	2005-04-25	IA-64	Yes	Yes	MBR takes precedence in hybrid MBR configuration[10]
Windows Vista	2006-07-22	x64	Yes	Requires UEFI[b]	MBR takes precedence in hybrid configuration[10]
Windows Server 2008	2008-02-27	x64	Yes	Requires UEFI	MBR takes precedence in hybrid configuration[10]
Windows Server 2008	2008-02-27	IA-64	Yes	Yes	MBR takes precedence in hybrid configuration[10]
Windows 7 Windows Server 2008 R2	2009-10-22	x64	Yes	Requires UEFI[c]	MBR takes precedence in hybrid configuration.[10]

[10] Much of the information in this section comes from https://en.wikipedia.org/wiki/GUID_Partition_Table

Table 6-4 Details of GPT Support on 64-bit Editions of Microsoft Windows[11] (Continued)

OS Version	Release Date	Platform	Read and Write Support	Boot Support	Notes
Windows Server 2008 R2	2009-10-22	IA-64	Yes	Yes	MBR takes precedence in hybrid configuration[10]
Windows 8 Windows Server 2012	2012-08-01	x64	Yes	Requires UEFI	MBR takes precedence in hybrid configuration.[10]

Note, however, that if using GPT partitioning, the Shadow MBR itself cannot hold everything necessary to boot a system. This typically is not a problem because it is possible to make ranges other than just the Shadow MBR read-only. However, Microsoft Windows can cause problems if it expects to see something in a particular place and it is not there. This means that when using the Shadow MBR and GPT, the Shadow MBR should probably contain a duplicate of the Normal MBR's GPT.

6.4 Shadow MBR Usage

It is unfortunate that the Opal Drive Specification called the special 128-MB region of the drive—which can be pointed to in place of the "normal" first 128 MB of disk memory—the "Shadow" MBR. It is not really a shadow, it may or may not contain an MBR partition table (it may be GPT), and it is much bigger than 512 bytes.

As a result, the name confuses a lot of people, causing an unnecessary cognitive load on users who read the specification. A different name, for example, Alternative Initial Memory, would more reflect its true usage.

When making a design to use the Opal drive, one of the first questions to ask is: Will the Shadow MBR be used at all? Some of the reasons to use it are the following:

- One does not want to disturb the currently loaded OS in any way.
- One wants to have an alternative partition table for the hard disk, different from the one that main OS will see when it boots.
- One wishes to have a read-only (but small) OS that is available under certain circumstances, but not viewable by a casual viewer.

6.4.1 Do Not Disturb

The first reason is a good one. A machine has been already set up with the correct device drives and applications, and then takes advantage of the self-encrypting features of the drive. This is

why Shadow MBRs are often used today. In the case where an Opal drive is being used for its SED characteristics, it is clear that before the drive is booted, it needs to be unlocked. This entails setting the Read Protect and Write Protect bits of the drive to OFF, where they will remain until the next power cycle, and then booting into the main OS, which at that point appears to be unencrypted. However, the software used to unlock the drive has to reside somewhere, and the most convenient place is the drive itself.

Therefore, the Shadow MBR is often where the unlocking code is stored. The Shadow MBR is enabled and the system will boot into it. A small OS residing in the Shadow MBR then asks the user for authentication, either passwords alone or passwords and other forms of authentication. When it is satisfied, it presents the correct password to the drive to unlock the global partition by setting the Read Protect and Write Protect bits to OFF.

At this point, the Shadow MBR Done bit is set, which tells the drive that, until the drive powers off, when an LBA from the first 128 MB of the drive is requested, it should return the value stored in the Normal MBR instead of the value stored in the Shadow MBR. The system then goes through a warm boot, which does not drop power to the drive. At this point, the system will boot into the Normal MBR and proceed as though the added code was not there.

Note that most drives have much more space than is used. As a result, it is often possible to shrink the main partition and insert a new partition at the end of the drive without disturbing the existing loaded system. This partition can then use its Opal characteristics without disturbing the main load—with the proviso that if the main OS is Windows, and the partition is NTFS-formatted, it cannot be made read-only unless the partition table is GPT and the partition is set to be read-only before the partition is made write-protected.

6.4.2 Alternative Partition Tables

An alternative partition table for the system can be made using the Shadow MBR, so that when the system is booted into the Shadow MBR, a different partition table is seen than that which is seen when booting from the Normal MBR. The partition table used by the Shadow MBR should always be MBR, never GPT, because of the size constraints of the Shadow MBR compared to the size of the number of partitions required by Windows when using UEFI with GPT.

If the design requires having two separate Windows versions on the drive at the same time, which boot independently, having an alternative partition table may be the only realistic way of achieving this requirement. Windows does not like seeing NTFS partitions on a drive that it cannot write to, unless their GPT says they are read-only. To totally separate the two Windows version, two boot loaders and partition tables are required. Two separate version of this will not fit into the Shadow MBR if using a GPT (which requires using UEFI). If the PC is set up to boot into UEFI, and sees an MBR partition table, it may not boot at all. In any event, it will cause unnecessary complications.

Therefore, it is assumed that if the Shadow MBR is being used for a secondary partition table, it is running an MBR partition table. This means that the BIOS is set to not use UEFI, which means the Normal MBR must also be running an MBR partition table. (Note that this is not “fixed in

stone." If running a Linux system in the Shadow MBR, the boot sequence can be fit entirely inside the Shadow MBR.)

RECOMMENDATION

If not trying to simultaneously have two completely separate Windows systems on the drive, do not use an alternative partition table. If trying to simultaneously have two completely separate Windows systems on the drive, use MBR partitioning for both of them.

6.4.3 Separate Read-only OS, Not Generally Visible

Although 128 MB is not a lot of space, it is sufficient for running a small OS such as TinyCore. This OS would be running read-only because the Shadow MBR can only be written with privilege. It can be read using special Opal ATA commands without privilege; otherwise, it is not visible when the Shadow MBR enable bit is not set. It is possible to customize TinyCore by following the instructions at http://gr8idea.info/os/tutorials/tiny-core/modified-cd.html.

Usually the only advantage to putting the utility OS in the ShadowMBR is to save one of the 16 locking ranges. Otherwise, putting the utility OS in a read only locking range works fine.

This page intentionally left blank.

7. SETTING UP PARTITIONS

When setting up a partition table, some things need to be considered. If using an MBR-style partition table, only four primary partitions are available:

- If partitioning a drive larger than 1 TB, the user likely will have to use a GPT.
- Although Microsoft OSs can be extended to allow viewing of partition tables that are neither a File Allocation Table (FAT) nor an NTFS, they cannot boot from them.
  - It is possible to create a Random Access Memory (RAM) disk, format it with NTFS, copy an image into it, and then boot from it. However, this consumes a lot of precious DRAM.
- If the design requires making an NTFS partition read-only, which is visible to a running Microsoft OS by being recorded in the partition table, the partition table must be GPT-style, and the partition must be marked as read-only before the drive is used to enforce it being read-only.
- Ranges must be set before they are written because creation of a range appears to erase all of its contents.

7.1 Create Range(s) for Partition Table

Before creating a partition table, which is necessary before partitioning a drive, a range should be set up for where the partition table will reside. For an MBR-style partition table, only a single range (encompassing the first megabyte of the drive) needs to be set up. This usually corresponds to LBAs 0 to 2048.

When creating a GPT, the first 1 MB and last 1 MB of the drive should be set aside. This will cover both the partition table and its backup. After this is done, the partition table itself can be created. gparted (https://gparted.org/) is a Linux-based partition table utility with a live CD, or an application that can be installed on most Linux distributions. It is capable of creating a GPT or MBR partition table.[11]

7.2 Create Partitions

Once a partition table is created, it is now possible to create ranges on the drive. For any of the applications demonstrated, no more than four partitions will be created. However, if all of the applications were to be installed at the same time on a drive, it is possible more partitions would be needed. If using an MBR-style partition table, this will mean the creation of a primary "extended" partition, which will in turn contain "logical" partitions. If this is done, one must be careful to create ranges that cover the extended partition table, which resides in the beginning of the extended partition.

[11] A good tutorial on gparted is at http://www.dedoimedo.com/computers/gparted.html.

When creating ranges, it is a good idea to simultaneously format them. This can easily be done with gparted. Alternatively, they can be formatted using the Microsoft control panel tool. One difference is that gparted makes it easy to determine exactly which LBA a partition will begin on and its extent in terms of number of LBAs. The Windows tool is at a higher level, and usually only gives the number of hundredths of a gigabyte that is being set. Although this works for most uses, when setting the range, it is important to get the exactly correct information. This can be done by reading the partition table either using **sudo fdisk -l** in Linux or using a tool in Windows to read the partition table and print it out. The latter can be easily done using **dd** for Windows[12] to copy the partition table into a binary file that can then be parsed. From a command line opened with Administrator authority, the command looks like this for an MBR-style partition:

dd count=1 if=\\?\Device\Harddisk0\Partition0 of=.\partable.bin

This binary file can easily be read and then used to output a batch file that will in turn set the correct ranges that map to these partitions. The following example only reads four partitions and creates a batch file that maps to those ranges. (Note that it starts with range 2 because range 1 is expected to cover the first megabyte of the drive where the partition table and MBR reside.)

```
// fdiskLikeOutput.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <string.h>


int _tmain(int argc, _TCHAR* argv[])
{
    FILE *fin, *fout, *fout2;

    fin=fopen("parttable.bin","rb");
    fout=fopen("createRanges.scr","wb");
    fout2=fopen("createRanges.bat","wb");

    char myBuffer[2048];
    fread(myBuffer,1,512,fin);
    printf("partition   StartLBA           NumLBA          EndLBA\n");
    fprintf(fout,"--SP Locking --DRIVE %s --UID Admin1 --PW %%2 \n",argv[1]);

    int firstLBA;
    memcpy(&firstLBA,&myBuffer[454],4);

    int size;
    memcpy(&size,&myBuffer[458],4);

    printf("1       %12d     %12d     %12d\n",firstLBA,size,firstLBA+size-1);

    fprintf(fout,"--createRange --RANGE2 --RANGESTART %d --RANGELEN %d \n",firstLBA,
   size);
    fprintf(fout2,"opaltoolc --createRange --SP Locking --DRIVE %s --UID Admin1 --PW
   %%1"),argv[1]);
    fprintf(fout2," --RANGE2 --RANGESTART %d --RANGELEN %d \n",firstLBA, size);

    memcpy(&firstLBA,&myBuffer[470],4);
```

[12] http://www.chrysocome.net/dd.

```
        memcpy(&size,&myBuffer[474],4);
        printf("2       %12d     %12d     %12d\n",firstLBA,size,firstLBA+size-1);

        fprintf(fout,"--createRange --RANGE3 --RANGESTART %d --RANGELEN %d \n",firstLBA,
       size);
        fprintf(fout2,"opaltoolc --createRange --SP Locking --DRIVE %s --UID Admin1 --PW
       %%1"),argv[1]);
        fprintf(fout2," --RANGE3 --RANGESTART %d --RANGELEN %d \n",firstLBA, size);

        memcpy(&firstLBA,&myBuffer[486],4);
        memcpy(&size,&myBuffer[490],4);
        printf("3       %12d     %12d     %12d \n",firstLBA,size,firstLBA+size-1);
        fprintf(fout,"--createRange --RANGE4 --RANGESTART %d --RANGELEN %d \n",firstLBA,
       size);
        fprintf(fout2,"opaltoolc --createRange --SP Locking --DRIVE %s --UID Admin1 --PW
       %%1"),argv[1]);
        fprintf(fout2,"  --RANGE4 --RANGESTART %d --RANGELEN %d \n", firstLBA, size);

        memcpy(&firstLBA,&myBuffer[502],4);
        memcpy(&size,&myBuffer[506],4);
        printf("4       %12d     %12d     %12d \n",firstLBA,size,firstLBA+size-1);
        fprintf(fout,"--createRange --RANGE5 --RANGESTART %d --RANGELEN %d \n",firstLBA,
       size);
        fprintf(fout2,"opaltoolc --createRange --SP Locking --DRIVE %s --UID Admin1 --PW
       %%1"),argv[1]);
        fprintf(fout2,"  --RANGE5 --RANGESTART %d --RANGELEN %d \n", firstLBA, size);

            printf("Created createRanges.bat.  To use, enter: createRanges.bat
    Admin1Password \n");

            fclose(fin);
            fclose(fout);
            fclose(fout2);
            return 0;
    }
```

If a GPT was used instead, the code would have to be modified to read the partition information from that format. That format can be found in a Wikipedia article at https://en.wikipedia.org/wiki/GUID_Partition_Table.

7.3 Create Ranges for Partitions

To create a range that corresponds to a partition, the user needs to know the LBA of the first address of the partition and the number of LBAs in the partition. Because this is exactly the information stored in the partition table, this is not particularly difficult to obtain. Indeed, the previous code example shows one way to read the partition information from an MBR partition table and automatically create a batch file that will set up ranges that correspond to those partitions. See Appendix C for a brief explanation example of programs and scripts that can be used to create partitions and ranges. Note that Appendix A also has some related script information. This can be done manually, but experience shows a pretty high error rate when typing the starting LBA and LBA sizes.

7.4 Formatting Partitions

Partitions are formatted in a standard way. Some formatting algorithms actually touch the data inside a partition. For those that do that, formatting needs to be done after the range is set up. NTFS does not seem to mind being formatted before setting up the range.

8. OS AND BOOT LOADER CHOICES

There are a number of OS and boot loader choices that can be made when using an Opal drive. Not all of them have been tested. For the boot loader, the choices seem to be:

- Linux
  - Grub
  - Grub4Dos
  - Grub2
  - syslinux
  - lilo
- Windows
  - Binary Coded Decimal (BCD)

The two that seem best for booting International Organization for Standardization (ISO) images are Grub4Dos and Grub2. The JHU/APL team used Grub2 for its experiments.

8.1 Toggling the Shadow MBR Bits

Normally, the Shadow MBR is completely invisible to a file system. The Opal drive has two bits that control it. The Shadow MBR bit can be enabled or disabled. If enabled, then upon power off, the other bit, Done, is disabled.

Table 8-1 shows the state of the bits and the visibility of the Shadow MBR to the file system.

Table 8-1 Visibility Settings for Shadow MBR

Shadow MBR Enable Bit	Done Bit	
SET	CLEAR	VISIBLE to File System
SET	SET	NOT VISIBLE to File System
CLEAR	CLEAR	NOT VISIBLE to File System
CLEAR	SET	NOT VISIBLE to File System

From the table, it is clear that the Shadow MBR is only visible to normal drive operations when both the Enable bit is set and the Done bit is clear. Whenever power is removed from the drive, the Done bit is always reset to CLEAR. However, the fact that the Shadow MBR is not visible to normal drive operations does not protect data inside the Shadow MBR from being read. A special Opal command, which does not require authorization, is available to copy the contents of the Shadow MBR into a file.

9. SETTING RANGE CONFIGURATIONS

It is tempting to believe that the first 128 MB of data stored in the Shadow MBR are covered by a range that starts at LBA 0 and contains 128 LBAs. This is not true. The Shadow MBR is never covered by a range. However, the Shadow MBR is always read-only, but only writeable with Administrator privilege. It is probably a mistake that the Shadow MBR is readable when it is not ON or is ON when the Done bit is set. However, that is the way the specification reads. No privilege is necessary to read the Shadow MBR; although, if not turned on or the Done bit is set, it must be done via a special command.

9.1 Partition Table

The first range set is used to cover the partition table. It may be larger than that if a read-only OS is going to be set up in the drive to control the system. If only the partition table is being protected, 0–2048 is set as the first range, the drive is partitioned, the boot loader is installed, and the range is set read-only.

NOTE

If using GPT, this range should not be set read-only until after NTFS partitions that are being made read-only are set as such. It is also possible that if using GPT, the user will want to cover more than the first megabyte of data.

9.2 Boot Loader

It is not clear whether the Windows boot loader can work if the Windows boot loader is read-only. The fact that WinPE works just fine when booted read-only suggests that it will, but this has not been tested. However, Linux boot loaders most likely work just fine when in a read-only state. For this experiment, insert TrustedGrub2, along with the ancillary files necessary to boot Grub2, into the first 50+ MB. That is a lot larger than is necessary, so feel free to experiment with different amounts.

Installing TrustedGrub2 is easy. Boot from a Linux disk, download, compile and install TrustedGrub2 and then execute the command:

sudo ./INSTALLDIR/sbin/grub-install --directory=INSTALLDIR/lib/grub/i386-pc /dev/sda

This should create a system with the bootloader installed in the partition table of the root drive and a set of directories set up in the first partition of the root drive. Into the boot/grub directory, place the TrustedGrub2 menu file grub.cfg. Section 10 gives an example of how it can be set up. The boot loader partition will also be made read-only after it is set up correctly; make certain the grub.cfg menu file is correct before performing this step. Making the partition table and boot loader read-only will prevent anything from interfering with the system boot. However, make certain there is something to boot that cannot be compromised.

9.3 Read-only OS Partition

One other read-only partition also should be set up. (This partition can be combined with the last partition if desired.) This partition will contain images of OSs that can boot in a read-only mode. There are several methods of doing this. Section 10 describes one favorite—copying ISO images of the OSs into a read-only partition and booting directly to them. Because this does not work for every OS, a read-only OS can be directly installed to the partition. ISO images can be copied to this partition, giving the user both kinds of OSs booting. However, note that Windows will only boot from an NTFS partition unless special drivers are added to it. Also, a Windows system booting from MBR-style partition tables does not like seeing read-only NTFS partitions on the drive.

10. READ-ONLY OPERATING SYSTEMS

To make certain the machine can always be booted, it is necessary to set up the system so that there is a read-only OS that is in a read-only partition, and which can be booted from the grub menu.

Not all CD/DVD ISO images can be booted as a read-only OS. This is odd because they obviously can be booted and run from read-only media. Many of the ISO images can be booted from Grub2 directly and run. However, once the Opal drive is instructed to make the partition in which the images reside read-only, they can no longer boot. Some of this is explainable because the OS that is booting does not have the correct drivers to read directly from the ISO file.

However there are several OSs that can be booted from a drive that is made read-only. Sometimes all that needs to be done is to unpack the OS onto the partition that will be made read-only or to install a special driver into the OS. This is the case of the Windows install CD. Both solutions work. However, note that if an MBR partition manager is being used, a co-resident Windows OS will not like to have an NTFS partition that is read-only, and therefore will take a long time to boot and then periodically crash the Explorer executable. (If a GPT is being used instead, the drive can be marked as read-only in the partition table, which prevents Windows from having a problem with it.)

10.1 OSs That Can Boot from Read-only ISO Images

The following is a partial list of OSs that can boot from read-only ISO images:

- SystemRescueCD
- Department of Defense (DoD) Lightweight Portable Security (LPS) public version 1.5.5
- WinPE ISOs
- Hardware Detection Toolkit
- Seagate hard disk tools
- Free DOS
- Slitaz 4.0 with Firefox
- memtest
- Ultimate Boot CD

- TinyCore
- SuperUser

DSL can be modified somewhat to almost work; a file has to be added separate to the ISO image to make it work.

10.2 Grub Menu File To Work with These Images

```
##begin file
#!/bin/sh
exec tail -n +3 $0
menuentry "Windows Install" {
     set root=(hd0,msdos5)
     chainloader +1
}
menuentry "Windows 7" {
     set root=(hd0,msdos2)
     chainloader +1
}
menuentry "SystemRescueCD"{
     echo Loading SystemRescueCD
     loopback loop (hd0,msdos5)/systemrescuecd-x86-3.7.1.iso
     linux (loop)/isolinux/rescue32 isoloop=systemrescuecd-x86-
3.7.1.iso
     initrd (loop)/isolinux/initram.igz
}
menuentry "DoD LPS public" {
     set isoFile="/LPS-1.5.5_public.iso"
     echo Loading DoD LPS from Read-only Partition
     linux16 /memdisk iso
     initrd16 (hd0,msdos5)$isoFile
}
menuentry "Emergency Utilities"{
     echo Loading Emergency1.iso
     linux16 /memdisk iso
     initrd16 (hd0,msdos5)/Emergency1.iso
}
menuentry "Win8 PE"{
     echo Loading Gandalf, a Win 8 PE distribution
     set isoFile="Gandalf's_Win8PE_x86_no_press_any_key.ISO"
     linux16 /memdisk iso
     initrd16 (hd0,msdos5)/$isoFile
}
menuentry "Load Hardware Detection Toolkit" {
     echo Loading Hardware Detection Toolkit
     set isoFile="hdt-0.3.6-pre2.iso"
```

```
     linux16 /memdisk iso
     initrd16 (hd0,msdos5)/$isoFile
}
menuentry "Seagate hard disk tools" {
     echo Loading SeaTools -all
     set isoFile="SeaToolsDOS223ALL.ISO"
     linux16 /memdisk iso
     initrd16 (hd0,msdos5)/$isoFile
}
menuentry "Free Dos" {
     echo Loading FreeDos
     set isofile="fdfullcd.iso"
     linux16 /memdisk iso raw
     initrd16 (hd0,msdos5)/$isofile
}
menuentry "Slitaz 4.0 with Firefox"{
     echo Loading Slitaz
     set root=(hd0,msdos5)
     loopback loop (hd0,msdos5)/slitaz-4.0-firefox.iso
     linux (loop)/boot/vmlinuz-2.6.37-slitaz
isofrom=/dev/sda4/slitaz-4.0-firefox.iso boot=live noeject quiet
lang=en kmap=us
     initrd (loop)/boot/rootfs.gz
}
menuentry "Test the Memory" {
     echo Loading the memory tester
     linux16 (hd0,msdos5)/memtest86+-4.10.bin
}
menuentry "Ultimate Boot CD"{
     echo UBCD 5.3.1
     linux16 /memdisk iso raw
     initrd16 (hd0,msdos5)/ubcd531.iso
}
menuentry "Tinycore" {
     echo Loading TinyCore
     linux16 (hd0,msdos1)/vmlinuz quiet libata_allowtpm=1
     initrd16 (hd0,msdos1)/core.gz
}
menuentry "Tinycore with sedutil-ci" {
     echo Tinycore with sedutil-ci
     linux16 (hd0,msdos1)/vmlinuz1 quiet loglevel=0
libata_allowtpm=1 norestore noswap superuser
     initrd16 (hd0,msdos1)/core1.gz
}
menuentry "Slitaz 4.0 with Firefox"{
     echo Loading Slitaz
     set root=(hd0,msdos5)
```

```
     loopback loop (hd0,msdos5)/slitaz-4.0-firefox.iso
     linux (loop)/boot/vmlinuz-2.6.37-slitaz
isofrom=/dev/sda4/slitaz-4.0-firefox.iso boot=live noeject quiet
lang=en kmap=us
     initrd (loop)/boot/rootfs.gz
}
```

11. SETTING UP THE SHADOW MBR

The easiest way to set up the Shadow MBR was to set up a regular MBR on the drive, then copy the first 127 MB of data from the drive to a file. Next, this file was copied onto the Shadow MBR, using Admin1 as the userid of the Locking SP, and its password (in this example, Admin1Password) for its password. This is a slow process, much slower than copying a normal file. Supposedly, this is because no error checking is available for the process, so it is done using a slower, more reliable technique than is normally used for writing files.

11.1 Create the File

To create the file, the team first downloaded and installed **dd** for Windows from www.chrysocome.net/dd. This is basically just a Windows version of the standard Unix command. Next, the following a batch file was executed:

```
MakeShadow.bat 0

    erase ShadowMbr.bin

    dd bs=1M count=127 if=\\?\Device\Harddisk%1\Partition0 of=.\ShadowMbr.bin
```

This created a file called ShadowMbr.bin into which the Shadow MBR was copied.

11.2 Write the File to the Shadow MBR

Note that *opaltoolc* appears to be orders of magnitude faster than *sedutil-cli* in writing the Shadow MBR. However, it only works when attached through a USB interface, which is not always practical. The very latest version of *sedutil-cli* is about 1.9 times faster than previous versions, but that still can mean 1 hour versus 1 minute. To write this file onto the Shadow MBR, use another batch file:

WriteShadowMBR.bat *1* Admin1Password

```
opaltoolc --loadShadowMBR --DRIVE %1 --SP Locking --UID Admin1 --PW %2 --FILE ShadowMbr.bin
```

NOTE

This only works for drives that are not the current boot drive, which is why the example shows drive 1!

or

(In Linux, the **dd** command would be **dd bs=1M count=127 if=/dev/sda of=./ShadowMBR.bin**.)

sedutil-cliWriteShadowMBR.bat Admin1Password

sedutil-cli --loadPBAimage %1 ShadowMBR.bin /dev/sda

sedutil-cli will write to the Shadow MBR of the current root drive.

12. USES FOR OPAL DRIVES

12.1 Designs that Provide Secure Backup Safe from Ransomware Predators

The easiest way to prevent a ransomware attack from encrypting files and holding them for ransom is to have the files written to a read-only partition. This can be done manually, but that requires users to periodically open up the partition (typing in their password), copy files, and then relock the drive. This is a problem for two reasons. The first is that because it is irritating, a user will either not do it, make a bad copy, or forget to relock after copying. The second reason is that the password and data are exposed while the backup is happening. A good solution will automate the process so that the end user does not have to either remember to do it or expose his/her key to a malware author.

The JHU/APL team's solution is to have the system boot first into a read-only OS and software that retrieves the password, checks for any new files, unlocks any read-only partitions found, updates the backup file there, relocks the partition, and then boots into the normal OS. There are two questions on how to accomplish this.

- **Where is the key hidden?** It cannot be put in the Shadow MBR because it is generally readable using a special Opal command. Therefore, store it in the non-volatile TPM, sealed to the read-only OS and software.

- **How does one boot to Windows after having booted into something else?** Although there are many ways to do this, the recommendation is to put the first read-only OS and software into the Shadow MBR, then (after erasing the recovered password) set the MBR Done bit, and lastly warm boot into Windows.

It is expected that a machine will be booted once a week (for OS updates), so at the same time, a secure backup will be automatically be taken.

12.2 Designs to Provide an Inexpensive CDS

The requirements for trying for an inexpensive CDS (one level at a time) are as follows:

1. One partition, which is read-only, is used to manage the rest of the drive.

2. After being cold booted, only the read-only partition is accessible. (All other partitions are both encrypted and read/write protected.)

3. The read-only partition contains management software that will only allow one partition to be unlocked at a time, and only for a single boot cycle.

4. The read-only partition does not contain the password that is necessary to unlock a partition.

    a. The read-only partition obtains that sealed password from a TPM, only if the read-only partition booted.

    b. The read-only partition deletes that password from memory after each use.

5. The read-only partition will clean DRAM memory during each boot.

6. The read-only partition exists on the Shadow MBR, so it may only be reentered by a cold boot of the system.

    a. After a cold boot of the system, all but the read-only partitions will be automatically locked.

    b. After unlocking a partition, the system will perform a warm boot into the Normal MBR.

7. The Normal MBR provides a boot menu that has a selection for booting into any level, but a selection will only succeed if that level has been unlocked.

8. A warm boot will bring the user back to step 7.

12.2.1 Challenges with this Design

The main challenge with this design is with the partition table. One is using either the MBR or the GPT, and both will cause a problem with Windows.

In the case of MBR, Windows does not like to see a read-only NTFS partition (at least on the same drive as it is booting from). This means that each Windows partition being booted needs to see a different partition table. Two partition tables are available to the system when using MBR—the Shadow MBR can contain one and the regular partition table can contain one, and they need not be the same. If more than two partitions are wanted, this will not work. In this case, the Shadow MBR boot will have to rewrite the partition table based on a selection provided to it by the user who is logging in so that Windows will not see partitions that are hardware locked. At the same time, a menu will be rewritten so that the user will be directly booted into the correct partition when the system is warm booted.

When using a GPT, some things are easier and some are harder. When using a GPT, it is possible to mark partitions as read-only in the partition table itself, and to enforce this with the Opal drive settings. In this case, Windows has no problem with having an NTFS partition that is on the same drive as it booted from but that is marked read-only. However, when doing an inexpensive CDS, the GPT partition settings will need to be changed at the boot so that only the booting partition is marked read/write. This entails making sure all other partitions are marked read-only because a previous boot may have marked them as read/write as well as changing the chosen partition to be read/write. This can be done using a Windows tool; however, it is uncertain whether Linux has a tool for this. Changing the read-only status of a drive requires changing the

partition table in this event, so the major advantage is in working with drives that are greater than 2 TB in a single partition.

To change only the read-only bit in the paritition table for the correct partition, the process is relatively simple:

1. Boot into the Shadow MBR, which directly boots into a read-only Linux partition. This partition unseals a key from the local TPM, thus guaranteeing that only that Linux partition can obtain the key.

2. When presented with a menu, choose an OS to boot from. (In this example, partition X.)

3. When asked, provide a password for partition X.

4. The software then sets the Done bit of the MBR partition using its unsealed key.

5. The software then sets the GPT such that only partition X is read/write, and all other partition tables are read-only.

6. The software then uses the user's entered password, together with the unsealed key, to derive the actual password for partition X, and sets it to read/write for that boot cycle.

7. The software then does a warm boot.

8. Upon booting, a new menu appears, asking again for a selection of partitions. The user selects the correct partition, and the system boots into that partition, where a password is again requested for the second level of encryption (software based) for that partition. (If the wrong partition is selected, the user will have to start over from the beginning because it is not available.)

The following is an example of a minor enhancement that can be made to this process:

The menu of the normal boot can be modified at the same time the partition table is changed so that it boots directly into the correct partition, instead of asking the user to make a choice. This is not a difficult change.

12.2.2 Varieties of Cross Domain Solutions

Opal drives may be suitable to create the following different types of CDSs and CDS-like capabilities:

- A machine that can connect (not simultaneously) to various levels of secure networks but not store any data at rest.

- A machine that can execute (not simultaneously) at various levels of security, each with their own data at rest not available to any system at a different level.

- A machine that can execute (not simultaneously) at two levels of security and filter data appropriately before it is allowed to transit between the security levels.

Each of these examples can be constructed using periods processing. The general security approach for periods processing using an Opal drive is described in Section 12.2.2.1. Each example is then described in the following sections. For each example, the general architecture is described. Step-by-step guides for creating each architecture will be created as separate documents. Figure 12-1 Figure 12-1shows the general architecture supporting CDSs.

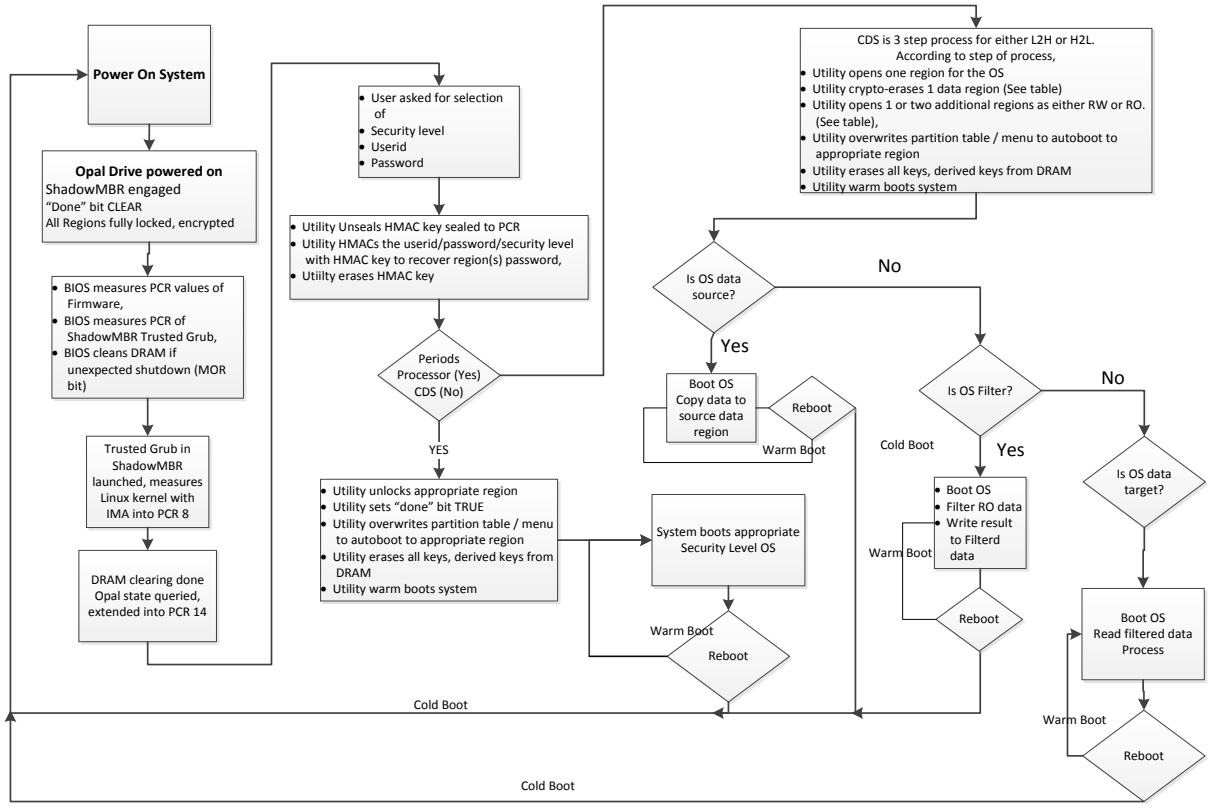

Figure 12-1 General Architecture Supporting Cross-Domain Solutions

12.2.2.1 Security Approach

Historically, the requirements for periods processing were satisfied by *clearing memory and media before and after periods of processing as a method of ensuring need-to-know protection.*[13] In order to guarantee the design will achieve this is, we follow the NEAT approach, with the main security component that has the responsibility for enforcing memory clearing having the following characteristics:

- **Non-bypassable.** A component cannot use another communication path, including lower level mechanisms, to bypass the security monitor.

[13] See Section 5 of the National Industrial Security Program Operating Manual (https://www.dss.mil/documents/odaa/nispom2006-5220.pdf)

- **Evaluatable.** Any trusted component can be evaluated to the level of assurance required of that component. This means the components are modular, well-designed, well-specified, well-implemented, small, low complexity, etc.

- **Always-invoked.** Each and every access or message is checked by the appropriate security monitors. (That is, a security monitor will not just check on a first access and then pass all subsequent accesses and messages through.)

- **Tamperproof.** The system controls "modify" rights to the security monitor code, configuration, and data, thereby preventing unauthorized changes.

The main security component in all of the designs will be the Shadow MBR software. It will work together with the Opal drive, TPM, National Institute of Standards and Technology (NIST) Special Publication (SP) 800-140–compliant BIOS, and a Filter OS (only used in some cases) to guarantee the following:

- To boot to a security level, the boot selector must be run (non-bypassability).

- The boot selector cannot be modified except by an administrator (tamperproof).

- To open a security level, all other security levels must be fully encrypted (guaranteed clearing of hard disk memory before and after use).

- DRAM will be cleared of any secrets before launching a user OS.

- Compromised firmware will not have the ability to gain access to secrets.

Additionally, when used as a kind of data diode in transferring data between security levels, the following must be guaranteed:

- Crypto-erasure of partitions is performed before they are written to during the process of transferring data between security boundaries (making it easier to evaluate).

- Filter OS code and settings used for review of data passed between security levels cannot be modified without administrator permissions (tamperproof).

12.2.2.1.1 Non-bypassable

To change security levels, it is necessary to obtain the password necessary to unlock the new security level. The architecture is designed to make it impossible to obtain this password without hard booting (after a power off) into the Shadow MBR. The password is obtained by unsealing the value from the TPM. Table 12-1 shows the required Platform Configuration Register (PCR) measurements and values that are used to seal the password to change security levels.

Table 12-1 Required PCR Measurements to Obtain Security Level Password

PCR Measurements	Required Values
PC firmware and configuration	MUST not have changed
Trusted Grub	MUST have booted
OS Kernel	MUST be expected kernel with IMA/EVM enabled
Opal Measurement Tool	MUST have run
Other kernel privileged code	MUST not include unexpected values
Opal Drive Configuration	MUST have booted from Shadow MBR

EVM – Extended Verification Module; IMA – Integrity Measurement Architecture

NIST SP 800-140 compliance guarantees the PCR measurements are made from a trusted base.

The TPM will not unseal the password unless all measurements match the expected values. Therefore, the PC must have booted from the Shadow MBR. With IMA and EVM enabled, EVM will white list all operations run on the OS so that no unexpected code can run. Only the utility that allows selecting which security level to boot to is allowed to run with user input. This utility is the only one that interacts with *opaltool*.

This utility erases all secrets after it uses them, before the MBR Done bit is set and before it warm boots the system. The OS traps all user attempts to warm boot the system so that if the system is warm booted from the Shadow MBR, it is done by the utility, and no secrets are in DRAM. Therefore, those secrets cannot escape processing done in the Shadow MBR.

To guarantee that the boot selector must be booted before any security level is booted, two things are done. First, the boot selector is placed in the Shadow MBR, and the Shadow MBR bit is set to ON by an administrator. This guarantees that after any full power down, the Shadow MBR is booted.

- The boot selector includes an application that queries the state of the Opal drive and extends it into PCR 14.
- Trusted Grub is used to boot the boot selector, thus guaranteeing the boot selector is measured into PCR 8.
- IMA is turned on, guaranteeing the boot selector executables are measured into PCR 10.
- Passwords used to unlock each region on the Opal drive are composed of a user-selected password that is used to release a key sealed to the TPM state when the Shadow MBR is fully booted with the correct Opal drive state, as shown by PCRs 0 to 15.
- The only software that has access to the key is the kernel, and it will only use it to unlock a particular security state. Kernel privilege is necessary to obtain access to this derived key; however, all this takes place before the user has a chance to log into the

system. Therefore, if a derived key is used, it must have been used by the kernel of the Shadow MBR

- All OS Opal regions are set to be fully encrypted when the Opal drive is powered down.

- Booting a selected security state will require unlocking the appropriate region for that power cycle as well as setting the Shadow MBR Done bit and warm booting into the Normal MBR. (A power cycle resets the Done bit to CLEAR.)

- BIOS settings do not allow booting except to the hard disk.

  – BIOS settings are locked by the administrator to prevent them from being changed.

  – If they are somehow changed, the PCR values would change, and the key would not unseal.

- PCs are required to conform to NIST SP 800-140 so that the firmware cannot be changed without administrative authorization. Similarly, the Opal drive, being FIPS certified, also has firmware that cannot be changed except to that authorized by the manufacturer.

NOTE

The result of these assertions means that:

- *If a system cold boots, it will boot into the Shadow MBR.*

- *If a system warm boots, it will boot into the Normal MBR or be authorized by an administrator.*

This is because of the following:

- The Shadow MBR traps all attempts to initiate a warm boot from the Shadow MBR, until after the Done bit is set. Therefore, warm booting into the Shadow MBR must be initiated from the Normal MBR.

- If the system booted into the Normal MBR, either the Shadow MBR is turned off (requiring administrator privilege) or the Done bit must be CLEAR (also requiring administrator privilege).

- Only the administrator and the Shadow MBR have administrator privileges.

Lastly, if a system boots into a Shadow MBR, all regions will be locked or the administrator will have initiated the boot. This is because the previous state the system was in was one of three states:

- Off, in which case, the system will cold boot, and the Opal drive will lock all regions.
- Booted from the Normal MBR.
  - In this case, the Done bit is set.
  - When the Done bit is set, a warm boot will send it to the Normal MBR.
  - Changing the Done bit requires one of the following (italic indicates cold booting, and bold indicates warm booting):
    - *Powering off (which leads to a cold boot, and the Opal drive locks all regions)*
    - **Being the Shadow MBR kernel (which it is <u>not</u>, so this cannot happen)**
    - *Being the administrator*
- Booted from the Shadow MBR.
  - In this case, there are only two ways to reboot (italic indicates cold booting, and bold indicates warm booting):
    - *Powering off (which leads to a cold boot, and the Opal drive locks all regions)*
    - **Warm boot (which only happens after the Done bit is set, so it boots to the Normal MBR)**

Figure 12-2 graphically shows the transition between these states, using red to depict warm booting, and green to depict cold booting.

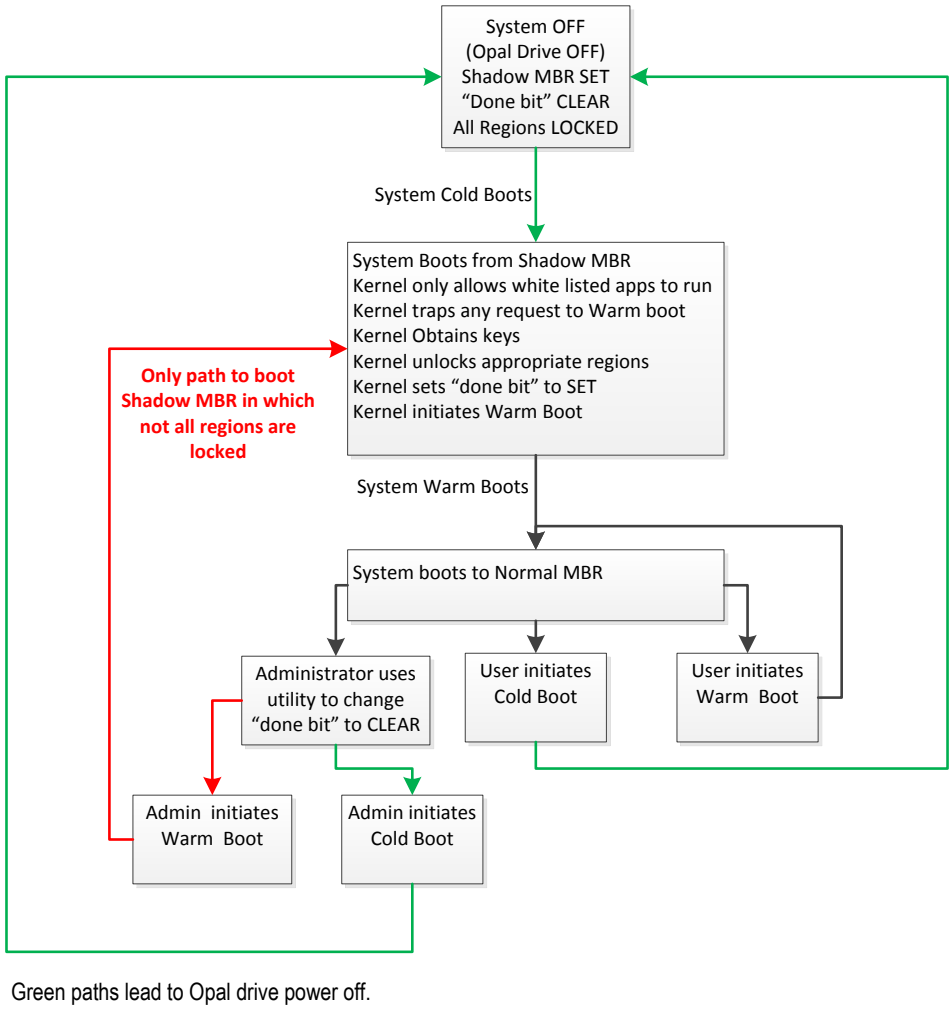

Figure 12-2 Paths To Boot Shadow MBR

12.2.2.1.2 Evaluatable

The Shadow MBR is small at only 128 MB, and consists of the following:

- Trusted Grub
- Small Linux kernel
- IMA and EVM (provides white listing)
- Opal drive querying software that extends results into PCR 14 (run with kernel privilege)
- Utility for managing keys and derived keys (run with kernel privilege)
  - Allows the user to select a security level and send in a userid and password

- Unseals a Hashed Message Authentication Code (HMAC) key from the TPM based on PCRs 0 to 15
- HMACs the key, userid, and password to recover region key(s)
- Erases the HMAC key from memory
- Uses region keys(s) to unlock appropriate portions of the drive based on configuration
- Sets Shadow MBR Done bit
- Performs a warm boot

- By design, the hard disk is fully locked when the Shadow MBR runs (except with Administrator privilege). See Figure 4-1.

12.2.2.1.3 Always Invoked

To change the security level, the system must go through the Shadow MBR (or be invoked with administrator privilege). Therefore, it is always invoked.

12.2.2.1.4 Tamperproof

The Shadow MBR is only writeable with administrator permissions, per the Opal specification. If the Opal drive is certified, that should provide a guarantee that the specification is followed in this security-sensitive operation. Therefore, the Shadow MBR is tamperproof because the hard disk itself controls "modify" rights to the boot selector code, configuration, and data. The same is true for any Filter OS together with its configuration.

12.2.2.2 Non-simultaneous, Multi-level Security Device Without Data at Rest

This section describes first example, the use of an Opal drive to create a machine that can connect (not simultaneously) to multiple networks of different classification levels but not store any data at rest.

12.2.2.2.1 Proposed Architecture

Because this machine has no data at rest, it is assumed to be attached to a network. An Opal drive first secure boots into a read-only boot selector running in the Shadow MBR, which will provide the user with a choice of networks from which to boot. Figure 12-3 Figure 12-3illustrates booting into this read-only system.

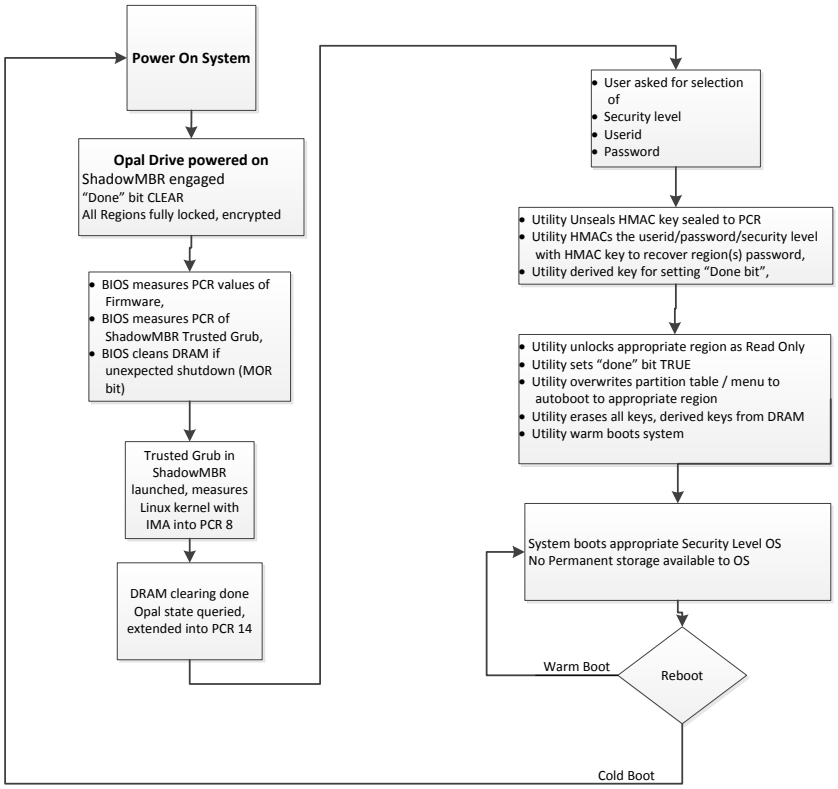

Figure 12-3 Booting to a Read-only Security Level OS

This boot selector is the main security solution in the architecture. It is responsible for making certain that only one security level boots at a time and that only the partitions that are associates with that security level are booted and have the correct read/write privileges. Additionally, it is responsible for cleanliness of the system, erasing any keys or derived keys it uses before the system is rebooted into a security level. (It does this by trapping the Ctrl-Alt-Boot sequence as necessary.) The drive itself is responsible for erasing all non-encrypted secrets it contains (including all key material that is in the clear) at each power off.

If the boot does not match what is supposed to boot, the system refuses to boot. This boot loader will unseal an HMAC key from the local TPM (based on the BIOS firmware being unchanged). The user then selects a security level to boot and provides a password for this level. The password will be HMACed together with the HMAC key and is unsealed by the boot loader, providing a region password. The HMAC key is then erased from DRAM.

The boot selector will overwrite all data stored in DRAM.

The boot selector will then use a well-known password to set the Done bit of the Shadow MBR, which will unlock the partition table of the main MBR. It will overwrite the main MBR partition so that it will see only one partition on the drive when it is rebooted into the main MBR, which corresponds to the correct security level. Using the region password, the boot selector will unlock (for that boot cycle) the readability of the region corresponding to that partition.

The boot selector will then perform a warm boot of the system, which will then automatically boot into the correct security level. If the system is booted to a classified level, performing a normal shutdown will also overwrite all the data in DRAM.

Once booted, the OS will use a standard CSfC virtual private network (VPN) to connect to the appropriate network. (LPS has a built-in Cisco VPN that can be used.) To provide an additional level of protection, TLS can be used to exchange keys necessary to set up a strong connection with a server.

12.2.2.2.2 Discussion of Vulnerability Mitigations

Evaluation of the design is based on the NEAT approach, which was discussed earlier in Subsection 12.2.2.1.

The initial boot selector is non-bypassable so as to change the security level booted. This is because to change the security level booted, the user must obtain the HMAC key. This is only available if the boot selector is running because that is the only way to get the PCRs into the correct state.

The boot selector should be small enough to fit in the Shadow MBR. This requires it to be modular, well-designed, well specified, well implemented, small and of low complexity.

The boot selector is guaranteed to be always invoked for every change of security level.

The boot selector is in non-writeable memory (without administrator privileges) because the hard disk itself controls "modify" rights to the Boot selector code, configuration, and data.

Additionally:

- Inadvertent error is not possible because the hard disk does not have to be swapped.
- Residual memory in DRAM should not be a problem because the system's DRAM is erased both at shutdown of a classified system and at boot up of any system.
- Firmware compromise is mitigated by not allowing the system to boot if the PCRs corresponding to firmware have changed and not allowing the HMAC key to be available if the PCRs have changed. Note that this is not a complete mitigation. A stronger mitigation would be to have an unclassified system run in a container (e.g., a hypervisor) and use VT-x technologies to prevent access to firmware states of the system that might be used for exfiltration.

12.2.2.3 Non-simultaneous, Multi-level Security Device with Data at Rest

This section describes the approach for the second example, a machine that can execute (not simultaneously) at various levels of security, each with their own data at rest not available to any system at a different level. This CDS replacement is similar to that discussed in Subsection 12.2.2.2. However, it allows data at rest and thus the partition is assumed to be unlocked for both read and write. With this change, it becomes possible to use the Windows OS.

The assumed certification of the Opal drive for one level of CSfC encryption provides one of the two levels of the needed encryption for classified operation. Advanced Encryption Standard (AES) 256 is assumed here so as to provide the necessary level of security for Top Secret data. A second level of encryption can be provided by using Windows' file-and-folder encryption (the EFS) for data stored on the system. This uses a different encryption engine, and can also be set up to be AES 256 using appropriate cryptographic modes. Figure 12-4Figure 12-4 shows the flow required to boot into the read/write partition, which supports two levels of encryption.

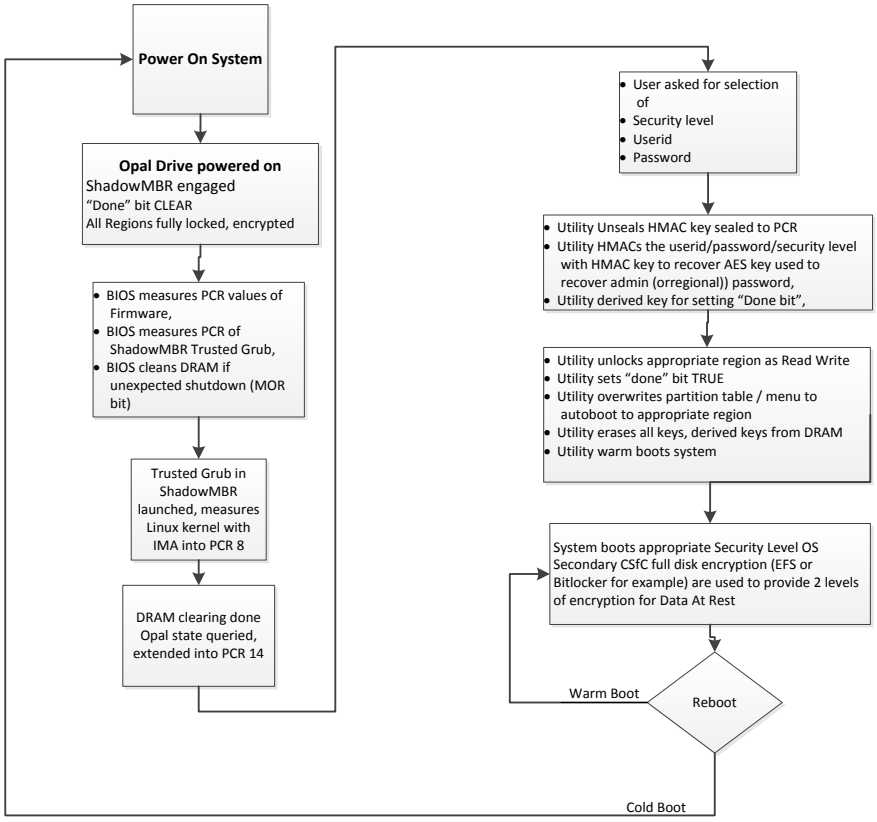

Figure 12-4 Booting to a Read/Write Security-Level OS

12.2.2.3.1 Proposed Architecture

The Opal drives will first secure boot into a read-only boot selector running in the Shadow MBR, which will provide the user with a choice of security levels from which to boot. (If the boot does not match what is supposed to boot, the system refuses to boot.) This boot loader will unseal an HMAC key from the local TPM (based on the BIOS firmware being unchanged). The user will then select a security level to boot into and provide a password for this level. The password will be HMACed together with the HMAC key unsealed by the boot loader, providing a region password. Additionally, it will derive a password for setting the Done bit. The region password will unlock the appropriate regions corresponding to the partition of the correct security level. The Done bit will then be set. At this point, the HMAC key and derived passwords will be erased from DRAM to prevent any type of warm or cold boot attack.

The utility will then overwrite the main MBR partition so that it will see only one partition on the drive when it is rebooted into the main MBR, which corresponds to the correct security level. The boot loader menu may also have to be overwritten. The boot selector will then warm boot the system, which will then automatically boot into the correct security level. If the system is booted to a classified level, performing a normal shutdown also will overwrite all the data in DRAM. Figure 12-5 Figure 12-5illustrates the normal shutdown process.

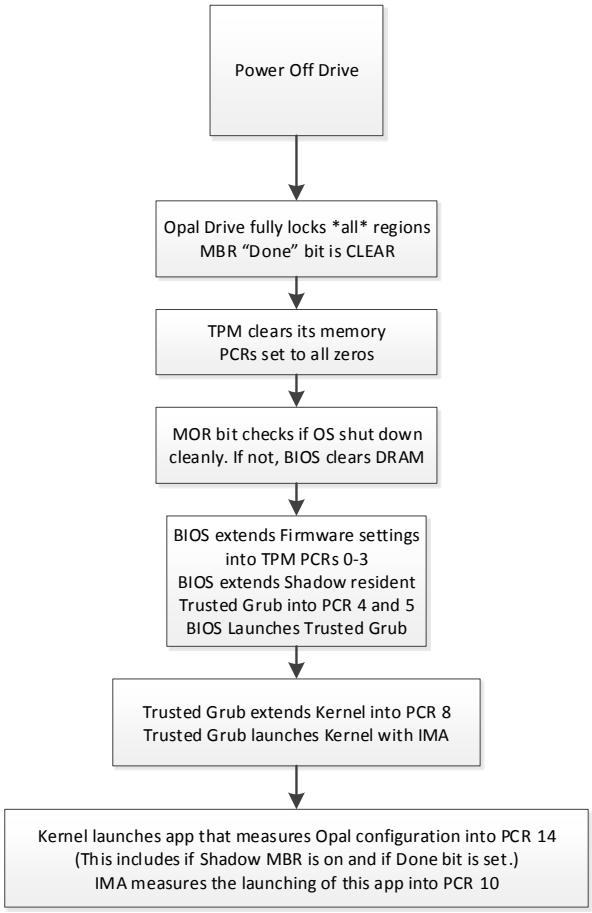

Figure 12-5 Normal Shutdown Process

If the shutdown is not normal, the MOR bit will be set and the BIOS will overwrite the DRAM upon the next boot.

- Inadvertent error is not possible because the hard disk does not have to be swapped.
- Residual memory in DRAM should not be a problem because the system's DRAM is erased by the BIOS in the advent of an unexpected shutdown.
- Warm booting will not allow changing the level of security (unless initiated by an administrator); it must be cold booted to go back to the boot selector to change security levels (see Figure 4-1). This automatically engages encryption on the OS that was previously booted. It is not possible to unlock a different security level once

booted into the Normal MBR because the HMAC key necessary to create the appropriate region unlock password is not available.

- Compromised firmware is mitigated by not allowing the system to boot if the PCRs corresponding to firmware have changed and not allowing the HMAC key to be available if the PCRs have changed.

12.2.2.4 Using Opal Drives in an Low-to-High (L2H) Data Diode-like System

This section describes the third example, a machine that can execute (not simultaneously) at two levels of security and filter data appropriately before it is allowed to transit between the security levels. In this case, the data transits from a low security level to a high security level. This type of machine allows an OS running on the high side to read data written on the low side, but it does not allow an OS running on the low side to read data written on the high side. The design looks very much like the general design described in Section 12.2.2.1 except that more than one partition will be visible at a time in the partition table.

When the system boots into the low side, two partitions become visible—one containing the OS and software and the other a repository to hold data for vetted before becoming visible on the high side. The repository partition is named Low Side Source Region or Temporary Partition. Before booting into the low side, the Shadow MBR will crypto-erase this partition. Next, the system will boot into an L2H Filter OS that is run as read-only and has access to two partitions. The first will be Low Side Source Region, which is read-only. The second partition, which is crypto-erased before the Filter OS is booted, is named L2H Filtered Data. The L2H Filter OS will filter out any data in Low Side Source Region that might contain malware, and will copy the rest into L2H Filtered Data. When the system boots into the high side, two partitions will be visible as well—one for the OS, and the other a read-only partition containing the L2H filtered data. The high-side OS can then see the filtered data. Figure 12-6Figure 12-6 illustrates the sequence of initialization steps and other conditions and requirements necessary to support L2H filtered data transfer. Figure 12-7Figure 12-7, Figure 12-8Figure 12-8, and Figure 12-9 Figure 12-9depict the three sequential L2H data transfer steps.

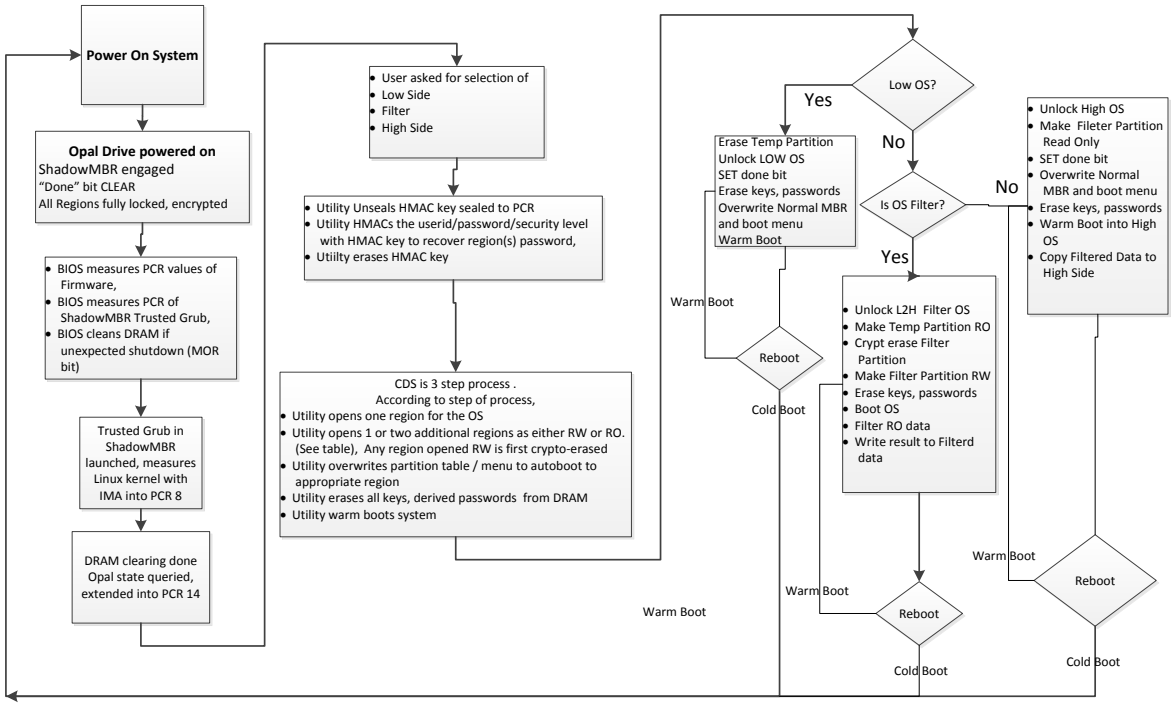

Figure 12-6 Process Flow for Opal Drive Enabled L2H Data Transfer

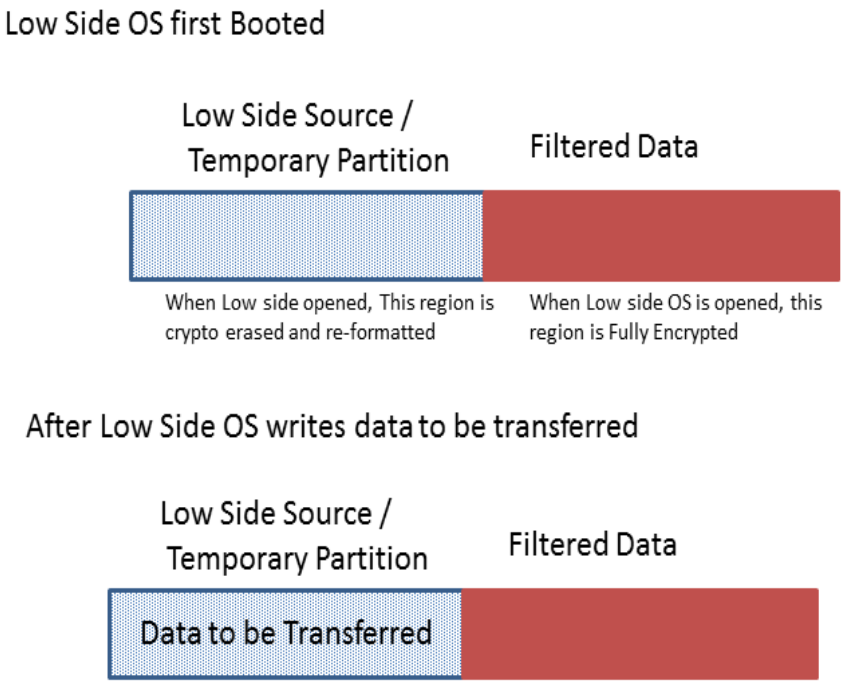

Figure 12-7 Step 1: Boot into Low Side

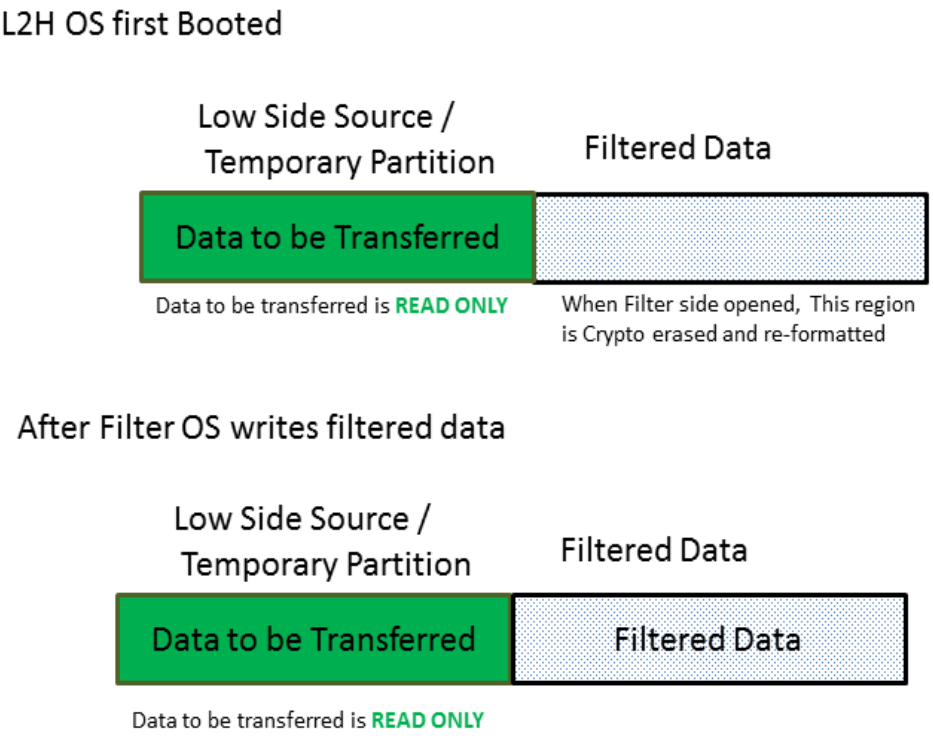

Figure 12-8 Step 2: Boot into L2H Filter

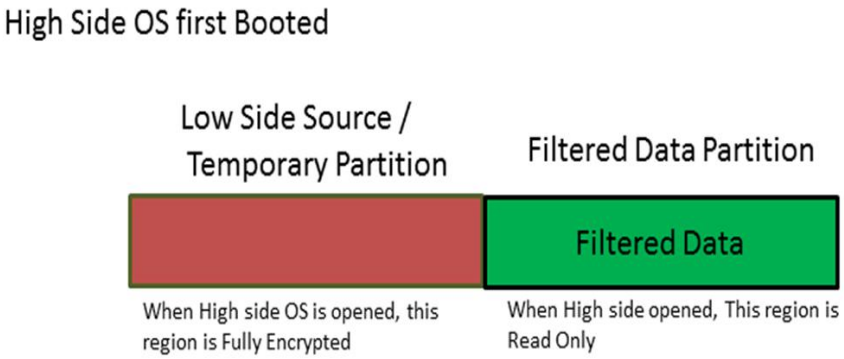

Figure 12-9 Step 3: Boot into High Side

12.2.2.5 Using an Opal Drive To Allow Data To Be Written Down

This section also describes the third example, a machine that can execute (not simultaneously) at two levels of security and filter data appropriately before it is allowed to transit between the security levels. In this case, the data transits from a high security level to a low security level. This type of machine allows an OS running on the high side to write data that can be readable on the low side. This design is similar to the design described in Section 12.2.2.4 except that the high-to-low (H2L) filter is more concerned about data leakage than malware.

When the system boots into the high side, two partitions will be visible—one containing the OS and software, and the other a repository to hold data for vetting before becoming visible on the low side. This latter partition will be named High Side Source Region or Temporary Partition. Before booting into the H2L Filter OS, the Shadow MBR will crypto-erase and reformat the Temporary Partition. Next, the system will boot into an H2L Filter OS that will filter any data out of the Temporary Partition that might be sensitive and copy the rest into the H2L Filtered Data partition. When the system boots into the low side, two partitions will be visible as well—one will be for the OS and the other a read-only partition containing the H2L filtered data. The low-side OS can then see the filtered data. Figure 12-10 Figure 12-10illustrates the sequence of initialization steps and other conditions and requirements necessary for achieving H2L filtered data. Figure 12-11Figure 12-11, Figure 12-12Figure 12-12, and Figure 12-13 depict the three sequential H2L data transfer steps.

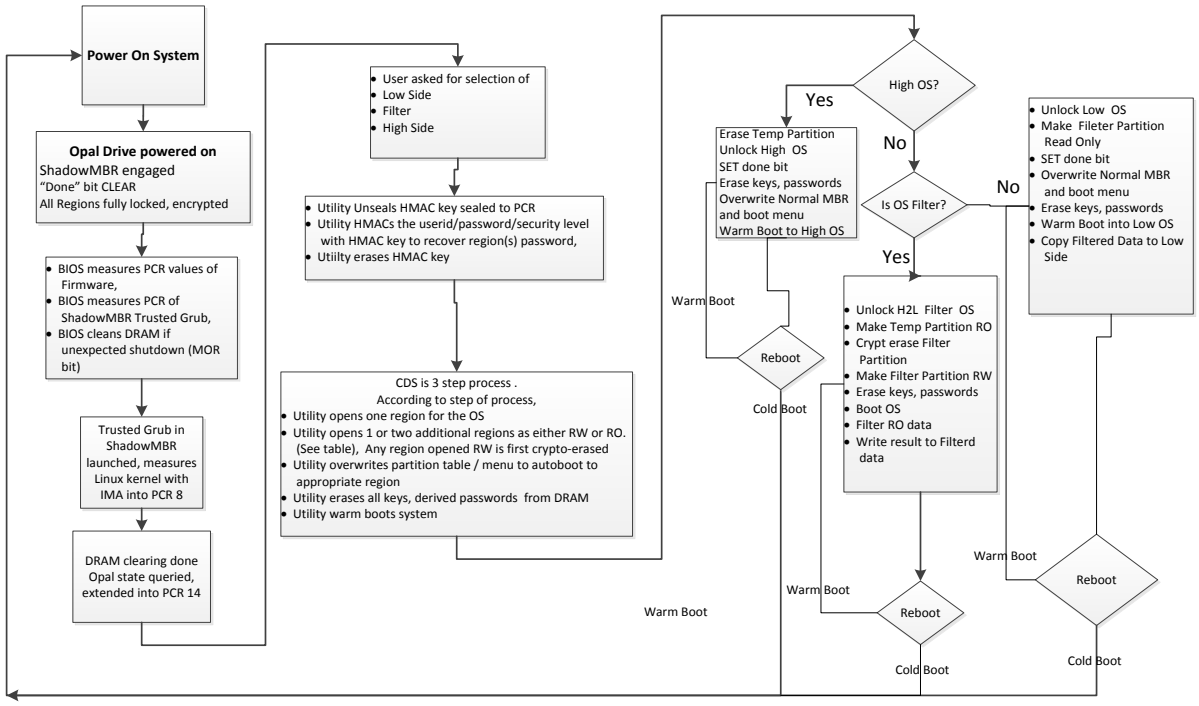

Figure 12-10 Process Flow for Opal Drive Enabled H2L Data Transfer

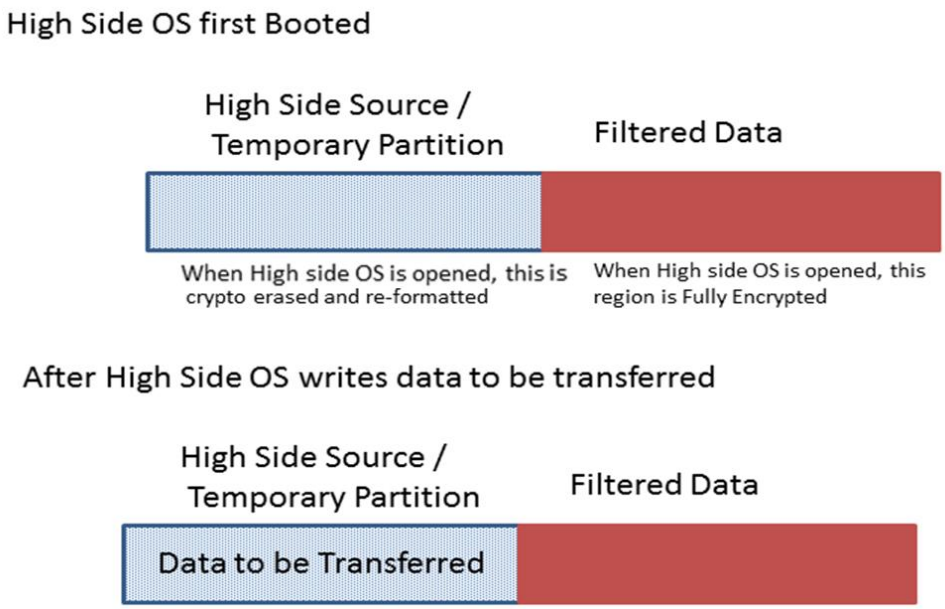

Figure 12-11 Step 1: H2L Data Transfer

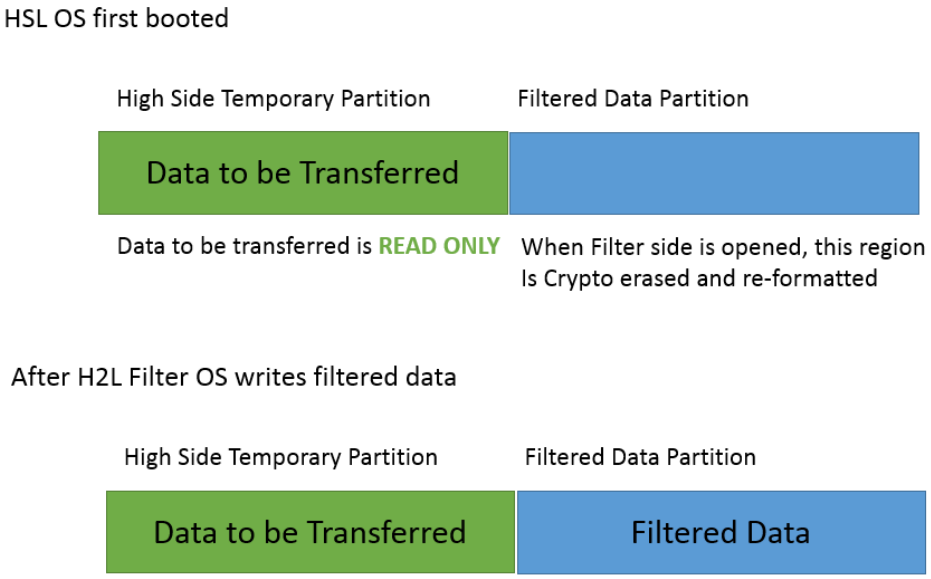

Figure 12-12 Step 2: H2L Data Transfer

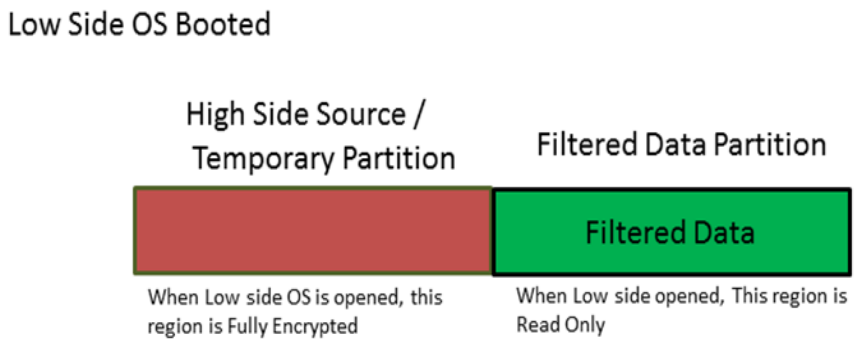

Figure 12-13 Step 3: H2L Data Transfer

12.2.2.6 Security

The main concern with regard to security is someone who knows the passwords for two different levels in the OS being able to open them both up simultaneously and then transfer data across the boundary. This cannot be done accidentally, but an insider threat could do it deliberately. The design needs to be robust against such an attack.

In the design of the CDS solutions described above, a full power off of the system locks down all partitions. Therefore, the design goal is to require going through a full power down to open up a partition. The architecture goal for this is to provide a requirement that the only way to unlock a partition is to boot into the Shadow MBR, and to then boot into the Normal MBR immediately upon unlocking a partition. (Once in the Normal MBR, the only way to return the system to boot into the Shadow MBR is through a full power-off reboot.)

Thus, the Shadow MBR boot is read-only; it cannot be changed and is set up so that a user may only perform one action—choose which partition to unlock and provide a password for that partition. No other options are provided. Attempts to warm boot the system are trapped by the OS and disallowed. The password necessary to clear the Done bit is kept secret by the Shadow MBR.

Because the Shadow MBR is readable without any authorization from the Normal MBR, secrets cannot be hidden in there. However, the TPM may be used to circumvent this issue. With trustedGrub, measurements can be made into the TPM that provide PCR-based evidence that the system booted into the read-only OS. At this point, the Shadow MBR may read a secret from the TPM that, together with the user credential, is able to derive the key used to unlock a partition. (Once derived, the Shadow MBR OS discards the key so that it is not in memory upon reboot.) This split-key design prevents an insider threat from knowing the password to unlock a partition of the Opal drive once in the normal boot.

There are three important aspects of this architecture:

- **The TPM must not allow any OS other than the correct Shadow MBR OS to unseal the HMAC key used for derivation of Opal credentials.**

- **The bootable read-only Shadow MBR OS must not leak information about the unsealed HMAC key to the user.**

- **The Shadow MBR must always be booted from a cold boot.**

Setup of such credentials is, of course, a weakness—an administrator who has access to the credentials can launch an insider attack. Also, it may be necessary at some point for an administrator to access those credentials for maintenance. The TPM's new method of authentication can help mitigate any threat by requiring two administrators to collaborate to retrieve information stored in the TPM.

This page intentionally left blank.

13. CONCLUSION

In conclusion, Opal-compliant drives have many underutilized capabilities, and there is sufficient open-source, free software to take advantage of those capabilities. Designs are provided in this document that should allow implementers to quickly start taking advantage of those capabilities, to solve the scourge of Ransomware, and make inexpensive CDS systems.

This page intentionally left blank.

APPENDIX A. BATCH FILES AND SCRIPTS

A.1 *opaltoolc*-based Batch Files and scripts

A.1.1 Administering an Opal Drive

A.1.1.1 Revert a Drive to Manufacturing Defaults

opaltoolc will work with the drive attached as a USB (which has been implemented, for example, using a Seagate drive attached as a USB drive) or attached on a SATA bus equally well for all commands except writing to the Shadow MBR. For the latter, it will not write to the root drive.

Script commands:

```
opaltoolc --list
```

Drive (0): \\.\PhysicalDrive0, Model = TOSHIBA MQ01ACF032,Serial = 24GLC2P6T
Drive (2): \\.\PhysicalDrive2, Model = ST320LT015-9YK142,Serial = W0Q17NX0

```
opaltoolc --revertDrive --DRIVE 2 --UID PSID --PW 01234567ABCDEFGH01234567ABCDEFGH
```

A.1.1.2 Enabling SPs

Usually, the Admin SP will already be enabled; however, if the drive has been reverted, the Admin SP may not be enabled. These commands will enable the Admin SP and the Locking SP.

Script commands:

```
opaltoolc --activateSP --DRIVE 2 --SP Admin --UID SID --PW MSID
opaltoolc --activateSP --DRIVE 2 --SP Locking --UID SID --PW MSID
```

A.1.1.3 Taking Ownership of SPs

This step requires a new password, which for this example is called NewPW. Here %1 is the drive number and %2 is the New password for the administrator.

Batch file takeOwnership.bat

```
opaltoolc --changePassword --DRIVE %1 --SP Locking --UID Admin1 --PW MSID --TUID Admin1 --TPW %2
opaltoolc --changePassword --DRIVE %1 --SP Admin --UID SID --PW MSID --TUID SID --TPW %2
```

Example usage:

```
takeOwnership 1 NewPW
```

A.1.1.4 Enabling Users of Locking SP

This can only be done using *opaltool*. This procedure enables the administrators and the first four users.

Batch file enableUsers.bat:

```
opaltoolc --enableUser --DRIVE %1 --SP Locking --UID Admin1 --PW %2 --TUID Admin2
opaltoolc --enableUser --DRIVE %1--SP Locking --UID Admin1 --PW %2 --TUID Admin3
opaltoolc --enableUser --DRIVE %1 --SP Locking --UID Admin1 --PW %2 --TUID Admin4

opaltoolc --enableUser --DRIVE %1 --SP Locking --UID Admin1 --PW %2 --TUID User1
opaltoolc --enableUser --DRIVE %1 --SP Locking --UID Admin1 --PW %2 --TUID User2
opaltoolc --enableUser --DRIVE %1 --SP Locking --UID Admin1 --PW %2 --TUID User3
opaltoolc --enableUser --DRIVE %1 --SP Locking --UID Admin1 --PW %2 --TUID User4
```

Example usage:

```
enableUsers 2 NewPW
```

A.1.1.5 Setting Passwords for Users of Locking SPs

This also can only be done with the *opaltool*. Here, this procedures sets the passwords for the users that were just enabled. Note that since the end user will set up different passwords for each user and administrator, this is not easily done with a batch file.

Script commands:

```
opaltoolc --changePassword --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID Admin2 --TPW Admin2PW
opaltoolc --changePassword --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID Admin3 --TPW Admin3PW
opaltoolc --changePassword --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID Admin4 --TPW Admin4PW

opaltoolc --changePassword --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID User1 --TPW User1PW
opaltoolc --changePassword --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID User2 --TPW User2PW
opaltoolc --changePassword --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID User3 --TPW User3PW
opaltoolc --changePassword --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID User4 --TPW User4PW
opaltoolc --changePassword --DRIVE 2 --SP Locking --UID Admin1 --PW NewPW --TUID User5 --TPW User5PW
```

A.1.1.6 Write the File to the Shadow MBR

To write a file onto the Shadow MBR, the WriteShadowMBR.bat is used:

Batch file WriteShadowMBR.bat:

```
opaltoolc --loadShadow MBR --DRIVE %1 --SP Locking --UID Admin1 --PW %2 --FILE ShadowMBR.bin
```

Example Usage:

WriteShadowMBR.bat *1* Admin1Password

NOTE*: This only works for drives that are not the one which was booted from. That's why the example shows drive 1!*

A.1.1.7 Enable the Shadow MBR

This command will enable the Shadow MBR, so that from a power off state, the system sees the Shadow MBR instead of the normal MBR.

Batch file EnableShadow.bat:

opaltoolc --MBRcontrol --DRIVE %1 --SP Locking --UID Admin1 --PW %2 SETENABLE

Example usage:

EnableShadow 1 Admin1Password

A.1.1.8 Disable the Shadow MBR

This will disable the ShadowMBR, so that upon power from a power off state, the system sees the Normal MBR

Batch file DisableShadow.bat:

opaltoolc --MBRcontrol --DRIVE %1 --SP Locking --UID Admin1 --PW %2 CLEARENABLE

Example usage:

DisableShadow 1 Admin1Password

A.1.1.9 Set the Done bit

This temporarily (for a power cycle) turns off the Shadow MBR, so that a warm boot will present the Normal MBR to the system.

Batch file EnableDone.bat:

opaltoolc --MBRcontrol --DRIVE %1 --SP Locking --UID Admin1 --PW %2 SETDONE

Example usage:

EnableDone 1 Admin1Password

A.1.1.10 Disable the Done bit

This will set the drive so that the ShadowMBR will be seen (if it is enabled) after either a warm or cold boot.

Batch file DisableDone.bat:

```
opaltoolc --MBRcontrol --DRIVE %1 --SP Locking --UID Admin1 --PW %2 CLEARDONE
```

Example usage:

```
DisableDone 1 Admin1Password
```

A.1.2 Setting Up Ranges

A.1.2.1 Create a Range

This is to create range number %3 on drive %1, from LBA %4 to LBA %5 using password %2.

Batch file CreateRange.bat

```
opaltoolc --createRange --SP Locking --DRIVE %1 --UID Admin1 --PW %2 –RANGE%3 --RANGESTART %4 --
RANGELEN %4
```

Example Usage:

```
CreateRange 1 Admin1Password 2 2048 260096
```

A.1.2.2 Administrating Ranges

A.1.2.2.1 Make a Range Read-only

This specifies that Range number %3 of drive %1 should be made read only, using the password given by %2.

Batch file MakeRangeReadOnly.bat :

```
opaltoolc --createRange --DRIVE %1 --SP Locking --UID Admin1 --PW %2 --RANGE%3 WRITELOCKENABLE
WRITELOCKED
```

Example usage:

```
MakeRangeReadOnly 1 Admin1Password 2
```

A.1.2.2.2 Make a Range Read/Write for a Power Cycle

This specifies that Range number %3 of drive %1 should be made read/write during that power cycle, using the password given by %2.

Batch file MakeRangeReadWrite.bat :

```
opaltoolc --createRange --DRIVE %1 --SP Locking --UID Admin1 --PW %2 --RANGE%3 WRITEUNLOCKED
READUNLOCKED
```

Example usage:

MakeRangeReadWrite 1 Admin1Password 2

A.1.2.2.3 Assign Range Control to a User

This assigns both read and write control to a userid, so that a password other than Admin can control the drive.

AssignRangeToUser.bat

```
opaltoolc --setRdLockedUID --DRIVE %1 --SP Locking --UID Admin1 --PW %2 --RANGE%3 --TUID %4
opaltoolc --setWrLockedUID --DRIVE %1 --SP Locking --UID Admin1 --PW %2 --RANGE%3 --TUID %4
```

Example usage:

AssignRangeToUser 0 Admin1Password 2 User1

A.1.2.2.4 Make a Region Encrypted

This sets up a range to be normally encrypted after a power cycle and also currently encrypted, with the drive not knowing the password.

Batch file MakeRangeEncrypted.bat:

opaltoolc --createRange --DRIVE %1 --SP Locking --UID Admin1 --PW %2 --RANGE%3 WRITELOCKENABLED WRITELOCKED READLOCKENABLED READLOCKED

Example usage:

MakeRangeEncrypted 1 Admin1Password 2

A.1.2.2.5 Make a Region Permanently Decrypted

This sets up a range to be decrypted (the drive uses its default password).
Batch file MakeRangeDecrypted.bat:

opaltoolc --createRange --DRIVE %1 --SP Locking --UID Admin1 --PW %2 --RANGE%3 WRITELOCKDISABLED READLOCKDISABLED

Example usage:

MakeRangeDecrypted 1 Admin1Password 2

A.1.2.2.6 Crypto Erase a Region

The only way to do this currently is to reset the whole drive.

A.2 *sedutil-cli*–based Batch Files

Normally, *sedutil-cli* does a password-based key derivation function to determine the actual Opal password from the typed in password. This makes its passwords incompatible with *opaltool*. To

use it interchangeably with *opaltool*, use the –n parameter, which tells it to use the password raw. This should be the **first** thing after *sedutil-cli*, as in

```
./sedutil-cli –n --loadPBAimage Admin1Password /dev/sda
```

NOTE*: There are multiple versions of the sedutil-cli executable, one each for 32- and 64 bit versions of Linux and Windows. There is also a version for Apple's OS. Users must use the correct version for their OS and bit length. In most Windows environments, the 32-bit version will work on a 64-bit OS, but not in the install command line. For that, the user must use the correct matching version of the executable. However, if using the command line of the Windows install DVD, the user must use the correct version. The 32-bit version will not work with the 64 bit command line.*

A.2.1 Administering the Opal Drive

sedutil-cli is an executable that needs no installation. It runs self-contained. It can only be run on a system that has the drive attached internally. It will NOT run on a USB attached drive.

A.2.1.1 Scandrives.bat

A.2.1.1.1 Description

This scans the drive to see what is connected. Do not believe the results except for drives connected via a SATA connection:

A.2.1.1.2 Contents of .bat file

```
sedutil-cli --scan
```

Example usage:

```
scandrives
```

A.2.1.2 QueryDrive.bat

A.2.1.2.1 Description

This gives more information about a particular drive, including the state of the Shadow MBR bits.

A.2.1.2.2 Contents of Batch and Script File

Windows

```
sedutil-cli –query \\.\PhysicalDisk%2
```

Linux

```
#!/bin/bash
sedutil-cli –query /dev/sda
```

A.2.1.3 IntialSetup.bat (Includes Initializing SPs)

A.2.1.3.1 Description

This enables the Locking SP, and changes the Locking SP Admin1 and Admin SP SID of drive %1 to have a password of %2. Because *sedutil-ci* only works on internal drives, unlike *opaltoolc,* the --DRIVE option is unnecessary.

A.2.1.3.2 Contents of .bat file

Windows

```
sedutil-cli --initialsetup \\.\PhysicalDisk%2
```

Linux

```
#!/bin/bash
sedutil-cli --initialsetup /dev/%2
```

Example usage:

```
InitialSetup Admin1Password
```

A.2.1.4 EnableUserAndAssignPassword.bat

A.2.1.4.1 Description

This creates and enables a user and sets a password for the user.

- %1 = Admin1's password
- %2 = User number
- %3 = User's new password
- %4= drive

A.2.1.4.2 Contents of .bat file

Windows

```
sedutil-cli --enableuser User%2 %1 %3 \\.\PhysicalDisk%4
```

Linux

```
#!/bin/bash
sedutil-cli --enableuser User%2 %1 %3 /dev/%4
```

Example usage:

This enables User2 and sets the user's password to User2Password.

Windows

```
EnableUserAndAssignPassword Admin1Password 2 User2Password 2
```

Linux

```
#!/bin/bash
EnableUserAndAssignPassword Admin1Password 2 User2Password sdb0
```

A.2.1.5 LockGlobal

A.2.1.5.1 Description

This locks the global range.

A.2.1.5.2 Contents of Batch File

Windows

```
sedutil-cli --enableLockingRange 0 %1 \\.\PhysicalDisk%2
```

Linux

```
#!/bin/bash
sedutil-cli --enableLockingRange 0 %1 /dev/sda
```

Example usage:

Windows

```
LockGlobal Admin1Password 2
```

Linux

```
LockGlobal Admin1Password
```

A.2.1.6 ShadowMaker

A.2.1.6.1 Description

This creates the Opal Shadow MBR using the original hard drive MBR. It takes the first 70 MB of a drive (including the partition table) and copies it to a file called ShadowMBR.bin. This assumes the **dd** for Windows program has been installed.

A.2.1.6.2 Windows

Contents of .bat file

```
erase ShadowMbr.bin
dd bs=1M count=70 if=\\?\Device\Harddisk0\Partition0 of=.\ShadowMbr.bin
```

A.2.1.6.2.1 Linux

```
#!/bin/bash
rm ShadowMBR.bin
dd bs=1M count=100 if=/dev/sda of=./ShadowMbr.bin
```

A.2.1.6.3 Example Usage

```
ShadowMaker
```

A.2.1.7 WriteShadowMBR

A.2.1.7.1 Description

This writes a file (less than 128 MB in size) to the Shadow MBR. This will only work for internal drives.

A.2.1.7.2 Contents of Batch File

Windows

```
sedutil-cli --loadPBAimage %1 %2 \\.\PhysicalDrive0
```

Linux

```
#!/bin/bash
sedutil-cli --loadPBAimage $1 $2 /dev/sda
```

Example usage:

Windows

```
WriteShadowMBR Admin1Password shadowMBR.bin
```

Linux

```
#!/bin/bash
WriteShadowMBR Admin1Password shadowMBR.bin
```

A.2.1.8 SetMBREnable.bat

A.2.1.8.1 Description

This sets the enable bit for the Shadow MBR for the main drive on the PC. The user needs to give it the admin1 password. It assumes this is for drive %1.

A.2.1.8.2 Contents of .bat File

Windows

```
sedutil-cli --setMBREnable on %1 \\.\PhysicalDrive0
```

Linux

```
#!/bin/bash
sedutil-cli --setMBREnable on $1 /dev/sda
```

Example usage:

```
SetMBREnable Admin1Password
```

A.2.1.9 ClearMBREnable.bat

A.2.1.9.1 Description

This clears the MBREnable bit for the main drive on the PC. The user needs to give it the admin1 password as %1. It assumes this is for drive 0.

A.2.1.9.2 Contents of .bat file

Windows

```
sedutil-cli --setMBREnable off %1 \\.\PhysicalDrive0
```

Linux

```
#!/bin/bash
sedutil-cli --setMBREnable off $1 /dev/sda
```

Example usage:

```
ClearMBREnable Admin1Password
```

A.2.1.10 SetShadowMBRDone.bat

A.2.1.10.1 Description

This sets the Done bit for the Shadow MBR for the main drive on the PC, which makes the regular MBR visible until the next power cycle. The user needs to give it the admin1 password. It assumes this is for drive 0.

A.2.1.10.2 Contents of .bat File

Windows

```
sedutil-cli --setMBRDone %1 \\.\PhysicalDrive0
```

Linux

```
#!/bin/bash
sedutil-cli --setMBRDone %1 /dev/sda0
```

A.2.1.11 ClearMBRDone.bat

A.2.1.11.1 Description

This clears the MBRDone bit for the main drive on the PC. The user needs to give it the admin1 password as %1. It assumes this is being done for drive 0.

A.2.1.11.2 Contents of .bat File

Windows

```
sedutil-cli --setMBRDone off \\.\PhysicalDrive0
```

Linux

```
#!/bin/bash
sedutil-cli --setMBRDone off /dev/sda
```

Example usage:

```
ClearMBRDone Admin1Password
```

A.2.1.12 Enabling SPs

Normally, enabling the SPs is incorporated into the Taking Ownership command. However, to do things manually, use this command:

```
sedutil-cli --activatelockingsp --password MSID
```

A.2.1.13 Revert a Drive to Manufacturing Defaults

Here, the Seagate drive is inside the machine. *sedutil-ci* does not currently work with USB-attached drives. Because this command is so dangerous, the full command should be typed instead of using a batch file.

Commands:

sedutil-cli –scan

```
Scanning for Opal 2.0 compliant disks
\\.\PhysicalDrive0 2 ST500LT025-1DH142                0012DM7
```

sedutil-cli --yesIreallywantoERASE*ALL*mydatausingthePSID -password
01234567ABCDEFGH01234567ABCDEFGH \\.\PhysicalDrive0

A.2.1.14 Taking Ownership

A.2.1.14.1 Windows

Command:

sedutil-cli --takeownership --password newPassword \\.\PhysicalDisk0
sedutil-cli --setPassword --password SEDUTIL-CLI --newPassword NewPW

A.2.1.14.2 Linux

Command:

#!/bin/bash
sedutil-cli --takeownership --password newPassword /dev/sda
sedutil-cli --setPassword --password SEDUTIL-CLI --newPassword NewPW /dev/sda

A.2.2 Setting Up Ranges

A.2.2.1 ListRanges

A.2.2.1.1 Description

This lists all the ranges in a drive %2, using password %1.

A.2.2.1.2 Contents of .bat File

Windows

sedutil-cli --listLockingRanges %1 \\.\PhysicalDrive0

Linux

#!/bin/bash
sedutil-cli --listLockingRanges %1 /dev/sda

Example usage:

```
listranges Admin1Password 0 >lr.txt
```

A.2.2.2 CreateR3

This is Windows only, MBR only.

A.2.2.2.1 Description

This reads the partition table of the drive to find the beginning and extent of the first four partitions and makes a batch file called createRanges. This batch file will create ranges that match those partitions, setting them to range numbers 2, 3, 4, and 5. (Range 1 is being reserved for the partition table.) It is assumed **dd** is installed on the system and the fdisklikeoutput executable is in the same directory as the batch file.

A.2.2.2.2 Contents of .bat File

```
erase parttable.bin
dd count=1 if=\\?\Device\Harddisk%1\Partition0 of=.\parttable.bin
fdisk3.exe
```

A.2.2.2.3 Example Usage

```
CreateR3 1
```

```
sedutil-cli createranges Admin1Password 1
```

A.2.2.3 CreateRange

A.2.2.3.1 Description

This creates a range on the first drive.

- %1=Admin1 password
- %2=range number
- %3=starting range logical block address
- %4=range size
- %5=drive identifier

A.2.2.3.2 Contents of .bat File

Windows

```
sedutil-cli -setupLockingRange %2 %3 %4 %1 \\.\PhysicalDrive0
```

Linux

```
#!/bin/bash
sedutil-cli -setupLockingRange %2 %3 %4 %1 /dev/sda
```

Example usage (this creates range 7 starting at LBA 410216448 of size 409600000):

```
CreateRange 7 410216448 409600000 Admin1Password
```

A.2.2.4 Create2Ranges

A.2.2.4.1 Description

This creates two ranges on the first drive, authorized by the %1 = Admin1 password, with ranges 1 and 2:

- Range 1 starting at %2 of size %3
- Range 2 starting at %4 of size %5

A.2.2.4.2 Contents of .bat File

Windows

```
CreateRange 1 %2 %3 %1 \\.\PhysicalDrive0
CreateRange 2 %4 %5 %1 \\.\PhysicalDrive0
```

Linux

```
CreateRange 1 %2 %3 %1 /dev/sda
CreateRange 2 %4 %5 %1 /dev/sda
```

Example usage:

```
Create2Ranges Admin1Password 0 409600000 410216448 409600000
```

A.2.3 Administering Ranges

A.2.3.1 MakeRangeWriteLocked

A.2.3.1.1 Description

This command will make a range read-only. The user passes it two parameters: the Admin1 password and the range number. It assumes the range in question is on the drive currently booted from.

- %1=Admin1Password
- %2 is the range number 0

A.2.3.1.2 Contents of .bat file

Windows

```
sedutil-cli --readonlyLockingRange %2 %1 \\.\PhysicalDrive0
```

Linux

```
sedutil-cli --readonlyLockingRange %2 %1 /dev/sda
```

Example usage (make the third region read-only):

```
MakeRangeWriteLocked Admin1Password 3
```

A.2.3.2 MakeRegionEncrypted

A.2.3.2.1 Description

%1=Admin1Password, %2 is the range number. This makes the range auto-encrypted after the next power cycle.

A.2.3.2.2 Contents of .bat file

Windows

```
sedutil-cli -- enableLockingRange %2 LK %1 \\.\PhysicalDrive0
```

Linux

```
#!/bin/bash
sedutil-cli -- enableLockingRange $2 LK $1 /dev/sda
```

Example usage (encrypt region 3 of drive 0):

```
MakeRegionEncrypted 3 Admin1Password
```

A.2.3.3 DecryptRegionForPowerCycleByUser

A.2.3.3.1 Description

This makes a region both read and writeable until the next power cycle of the drive. It uses %1 as the Admin1 password (currently, *sedutil-cli* can only use the Admin1 user) and %2 as the region number. It assumes the current booted drive.

A.2.3.3.2 Contents of .bat File

Windows

```
sedutil-cli --setLockingRange %2 %1 \\.\PhysicalDrive0
```

Linux

```
#!/bin/bash
sedutil-cli --setLockingRange %2 %1 /dev/sda
```

Example usage (decrypt region 3 by Admin1 for one power cycle):

```
DecryptRegionForPowerCycleByUser Admin1Password 3
```

A.2.3.4 Make Region Permanently Decrypted

A.2.3.4.1 Description

%1=Admin1Password, %2 is the range number, the drive is the one currently booted.

A.2.3.4.2 Contents of .bat file

Windows

```
sedutil-cli --disableLockingRange %2 %1 \\.\PhysicalDrive0
```

Linux

```
#!/bin/bash
sedutil-cli --disableLockingRange %2 %1 /dev/sda
```

Example usage (decrypt region 3 permanently):

```
MakeRegionPermanentlyDecrypted Admin1Password 3
```

A.2.3.5 Crypto-erase a range

A.2.3.5.1 Description

This will change the key used to encrypt and decrypt a region, thus crypto-erasing it.

A.2.3.5.2 Contents of .bat file

Windows

```
sedutil-cli --rekeyLockingRange %1 %2 \\.\PhysicalDrive0
```

Linux

```
#!/bin/bash
sedutil-cli --rekeyLockingRange %1 %2 /dev/sda0
```

Example usage:

```
cryptoErase 4 Admin1Password
```

APPENDIX B. OPAL COMPATIBLE DRIVES

There are a number of companies online that publish lists of opal drives that are compatible with their software. Using these lists, it is easy to find drives that are compatible. Samsung announced in 2015 that all of their SSD drives going forward would be Opal-compatible.[14]

Table B-1 lists several sites on the Internet where lists of compliant drives are recorded.

Table B-1 Links to Lists of Opal Compatible Drives

Manufacturer	List of Opal Compatible Drives
Wave systems	https://www.wavesys.com/self-encrypting-drive-compatibility-list
Symantec	https://support.symantec.com/en_US/article.TECH226779.html
Mcafee	https://kc.mcafee.com/corporate/index?page=content&id=KB81136
WinMagic	htps://www.winmagic.com/drive-compatibility?manufacturer=All

[14] At the request of some OEMs, there are some SSD drives that report that they are not compatible.

This page intentionally left blank.

APPENDIX C. MATCHING PARTITIONS AND RANGES

During the creation of demonstrations of various use cases, it became obvious that it is often useful for a partition to exactly match the LBAs of a range. This turned out to be extraordinarily difficult to do manually. For example, creating 10 ranges and partitions would entail manually entering 40 nine-digit numbers correctly—not an easy task. It is highly recommended that users create a batch file rather than try to do it by hand.

Additionally, with newer drives, it is often necessary that partitions and ranges begin on 4000 boundaries. This is not difficult for software to do, but can easily lead to errors if a user is trying to do it manually. The following paragraphs provide a brief explanation example of programs that can be used to create partitions and ranges.

Suppose you want to divide a disk into 10 partitions, each containing a different version of an OS (like Windows) at a different security level. You also want to have a utility partition that contains files that will manage which of those partitions is active at any given time. This will require using 12 ranges of the drive. The first range will encompass the first 2048 sectors (1 MB) of the drive, which will contain the partition table. It also is allowed to contain any leftover LBAs that were missed. This will become range 0, and it will not have to be customized.

Next, determine the size of the other 11 ranges. First, create a program that reads in the size of the drive in sectors, minus the 2048 sectors used by the partition table, and divide it by 11. Then round down to the nearest 8000 boundary by dividing by 8 and multiplying by 8. This will be the size of the first 10 ranges in sectors. The last range will be everything remaining. This program then creates script file that will be called to create each of these 11 ranges.

The script file will need to create both a partition table corresponding to the range and the range itself. Therefore, it will need to run as superuser and have the Opal drive admin1 password as an input. An easy way to do this is to name a file EachPartitionAndRange.

The script will first create the range, using *sedutil-cli* and then create a partition table. It will then format the partition table.

The following simple script assumes an MBR partition table is already present, with at most one partition extant. It operates as follows:

- Create the matching range.
- Delete the existing partition.
- Create a new partition using fdisk.
- Format that partition as NTFS.
- Copy the partition table into a binary file.
- Store a copy of that binary file on an attached USB key.

```
#!/bin/bash
# This script assumes the drive has already been taken ownership of
# It assumes there is (at most) one partition on the drive
# It assumes an MBR partition table type is present on the drive
# It assumes the drive being worked on is /dev/sda
# It assumes a USB key mounted at /mnt/sdb1
#
# %1 is number of range
# %2 is beginning sector of range
# %3 is the number of sectors in the range
# %4 is the admin password of the drive
# This should do as follows:
#    * delete the current 1st partition
#    * create a new primary 1st partition
#    * make it NTFS type
#    * make it active
#    * format it NTFS
#    * make a range that corresponds to it

echo "o
d
n
p
1
%2
+%3
t
9
a
1
w
q
"| fdisk /dev/sda;
sedutil-cli -setupLockingRange $1 $2 $3 $4 /dev/sda;
mkfs.ntfs -f /dev/sda1;

dd bs=1M count=1 if=/dev/sda of=$1.bin;
cp ./ $1.bin /mnt/sdb1
```

NOTE

Formatting has to take place <u>after</u> the range creation because the creation of the range effectively crypto-erases the range.

The script file that calls this script file can be created using a C program as follows:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int main(int argc, char **argv)
{
```

```
        long sizeOfDrive;
        int numWindowsRegions;
        int numRegions;
        long S;
        int B;
        int i;
        FILE *oFile;
        oFile=fopen("MakeRangesAndParititonFiles.scrd","wb");

        sizeOfDrive=atol(argv[1]);
        numWindowsRegions=atoi(argv[2]);
        numRegions=numWindowsRegions+1;
        S=sizeOfDrive-2048;
        B=8*((S/numRegions)/8);

        for(i=0;i<numWindowsRegions;++i)
        {
         fprintf(oFile,"EachPartitionAndRange %d %d %d %%1\n",i+1,2048+(i*B), B);
        }
        fprintf(oFile,
                "EachPartitionAndRange4 %d %d %ld %%1\n", numRegions,
                2048+(numWindowsRegions*B),
                sizeOfDrive-(2048+(numWindowsRegions*B)));
        return 1;
}
```

The EachPartitionAndRange4 script file looks similar to the EachPartitionAndRange script file, but produces a partition numbered 4.

```
#!/bin/bash
# This script assumes the drive has already been taken ownership of
# It assumes there is (at most) one partition on the drive
# It assumes an MBR partition table type is present on the drive
# It assumes the drive being worked on is /dev/sda
# It assumes a USB key mounted at /mnt/sdb1
#
# %1 is number of range
# %2 is beginning sector of range
# %3 is the number of sectors in the range
# %4 is the admin password of the drive
# This should do as follows:
#    * delete the current 1st partition
#    * create a new primary 1st partition
#    * make it NTFS type
#    * make it active
#    * format it NTFS
#    * make a range that corresponds to it

echo "o
d
n
p
1
%2
```

```
+%3
t
83
a
4
w
q
"| fdisk /dev/sda;
sedutil-cli -setupLockingRange $1 $2 $3 $4 /dev/sda;
mkfs –t ext3 /dev/sda4;

dd bs=1M count=1 if=/dev/sda of=$1.bin;
cp ./ $1.bin /mnt/sdb1
```

APPENDIX D. ACRONYMS

AES	Advanced Encryption Standard
ATA	Advanced Technology Attachment
BCD	Binary Coded Decimal
BIOS	Basic Input/Output System
CD	Compact Disk
CDS	Cross-Domain Solution
CHS	Common Hardware/Software
CSfC	Commercial Solutions for Classified
DES	Data Encryption Standard
DoD	Department of Defense
DRAM	Dynamic Random Access Memory
DSL	Damn Small Linux
DVD	Digital Video Disk
EFI	eXtensible Firmware Interface
EFS	Encrypting File System
ESC	EMBASSY® Security Center
ESP	EFI System Partition
EVM	Extended Verification Module
FAT	File Allocation Table
FIPS	Federal Information Processing Standard
GPT	GUID Partition Table
GUID	Global Unique Identifier
HMAC	Hashed Message Authentication Code
IBM	International Business Machines

IMA	Integrity Measurement Architecture
ISO	International Organization for Standardization
IT	Information Technology
JHU/APL	The Johns Hopkins University Applied Physics Laboratory
LBA	Logical Block Address
LPS	Lightweight Portable Security
MBR	Master Boot Record
MOR	Memory Overwrite upon Request
MSID	Manufacturer's Secure Identifier
MSR	Microsoft Reserved Partition
NEAT	Non-bypassable, Evaluatable, Always-invoked, and Tamperproof
NIAP	National Information Assurance Partnership
NIST	National Institute of Standards and Technology
NTFS	Network Terminal File System
OS	Operating System
PC	Personal Computer
PCR	Platform Configuration Register
PSID	Physical Security Identifier
RAM	Random Access Memory
SATA	Serial Advanced Technology Attachment
SED	Self-Encrypting Drive
SID	Security Identifier
SHA	Secure Hash Algorithm
SP	Security Provider Special Publication
SSD	Solid State Drive

TCG	Trusted Computing Group
TPM	Trusted Platform Module
UEFI	Unified eXtensible Firmware Interface
USB	Universal Serial Bus
VPN	Virtual Private Network

This page intentionally left blank.