Mastering TrueCrypt

Windows 8 and Server 2012 Memory Forensics

Michael Ligh (@iMHLv2)

Part I

Investigative Context

- Suspect uses TrueCrypt for disk encryption
  - Most recent version: 7.1a
- Suspect runs Windows 8 / Server 2012
- What can you do, given a disk and memory image?
  - Provide three of the most common scenarios

What is encrypted?

- Full disk encryption
- Non-system partitions (USB)
- Hidden operating systems
- Virtual hard disk
  - A.K.A file-based containers
  - Normal/Standard
  - Hidden (container within a container)
  - First step: isolate the encrypted file(s)

How was it encrypted?

- Algorithms
  - AES
  - Twofish
  - Serpent
  - AES-Twofish
  - AES-Twofish-Serpent
- Modes
  - XTS
  - LWR
  - CBC, Outer CBC, Inner CBC
- Password
  - Cached or non-cached

Disk Parameters

- Depending on the scenario, you may also need to know
  - File system
    - FAT, NTFS, etc.
  - Size of the disk
    - Host size versus real size

Existing Solutions

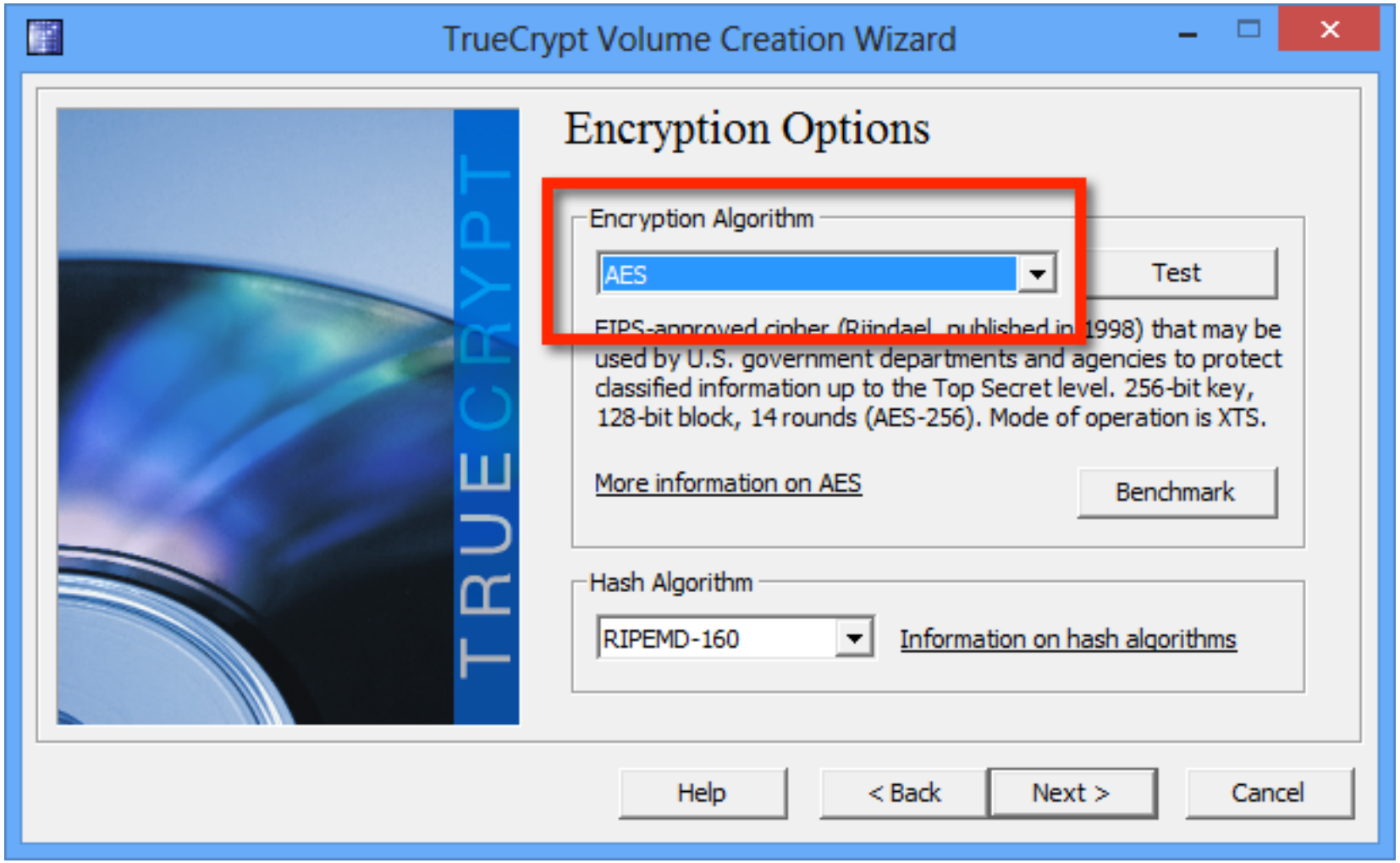
- Passware Kit Forensic
  - A disk image, memory image, and $995
- Elcomsoft Forensic Disk Decryptor
  - A disk image, memory image, and $299
- Cryptoscan, circa 2008
  - Passwords must be cached
  - Recent TrueCrypt not supported
- Key scanning
  - AESKeyfinder, Bulk Extractor, etc.
  - Only works if AES was used

Suspect #1 – Standard Container with Cached Password

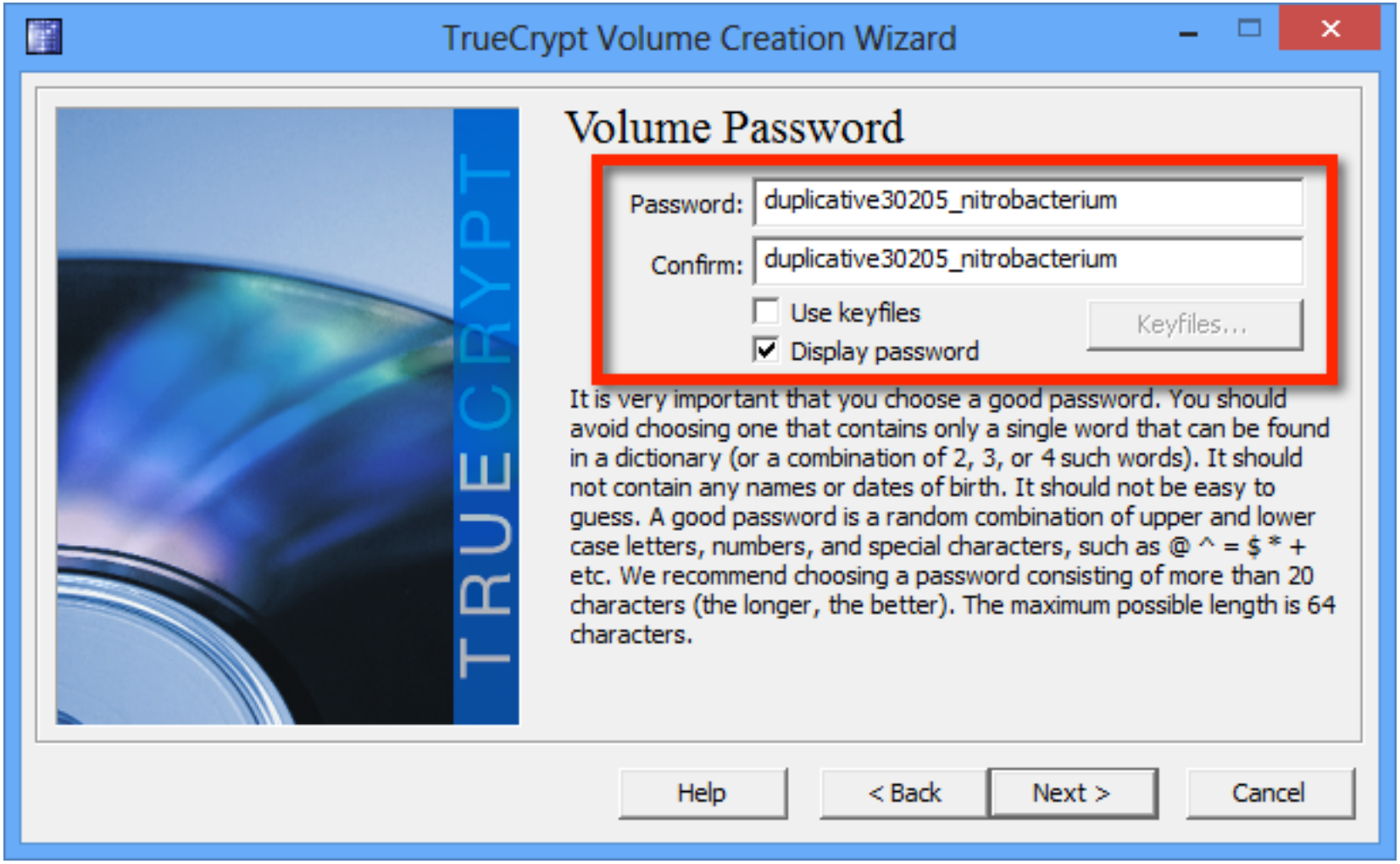
- Standard (not hidden) container
- Cached password
- Default encryption (AES)
- 20 MB with FAT file system
- 32-bit Windows 8
- Most common configuration, inexperienced suspect

Suspect #1 – Standard Container

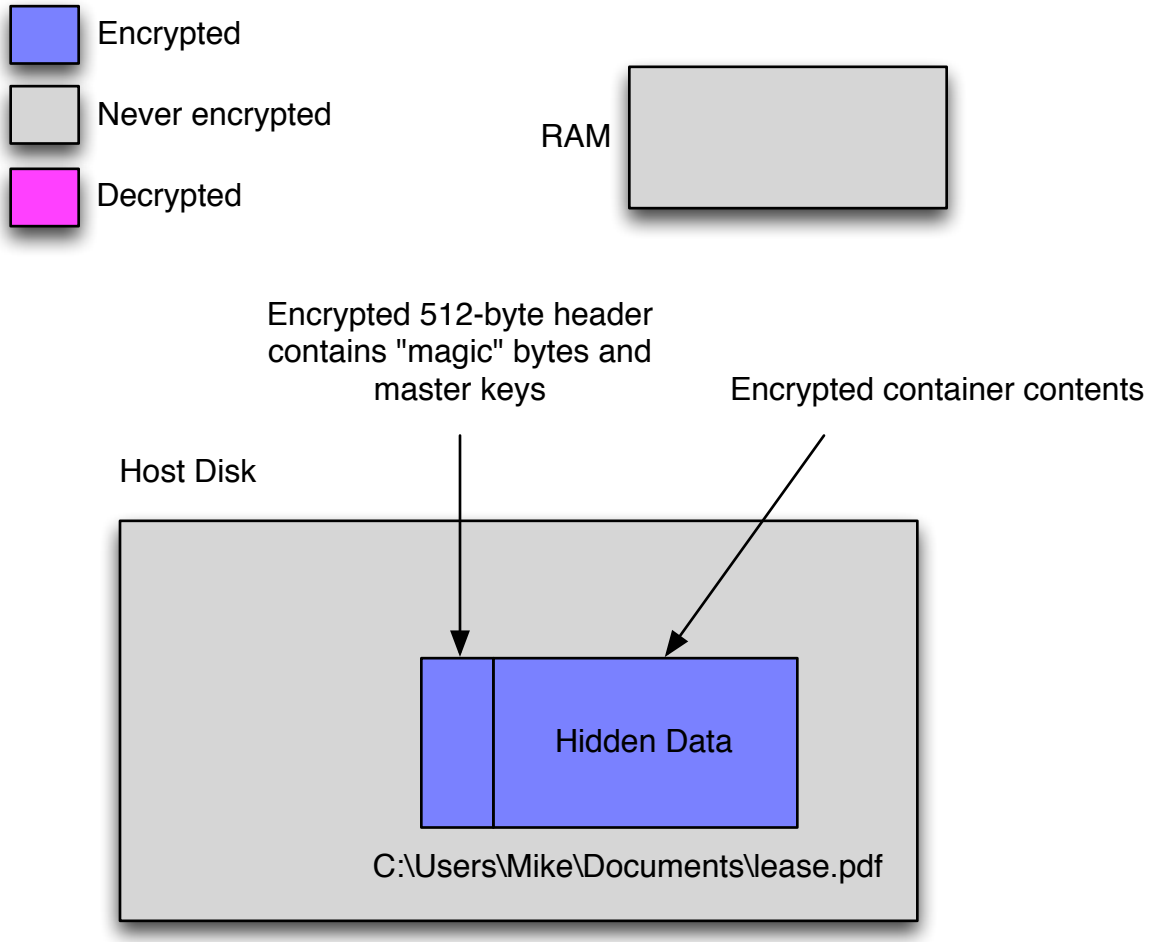
Suspect #1 – AES Encryption



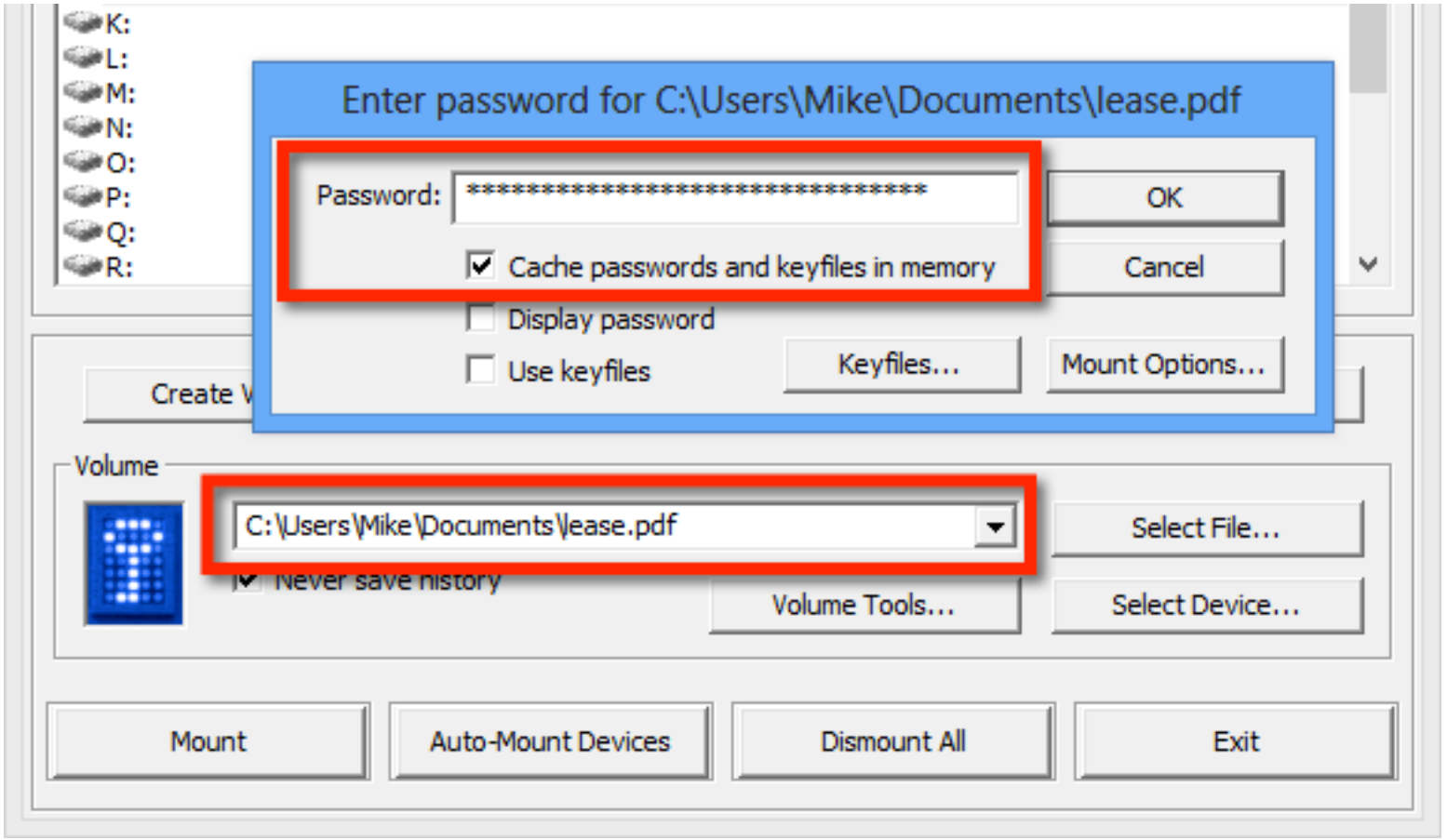
Suspect #1 – Volume Password



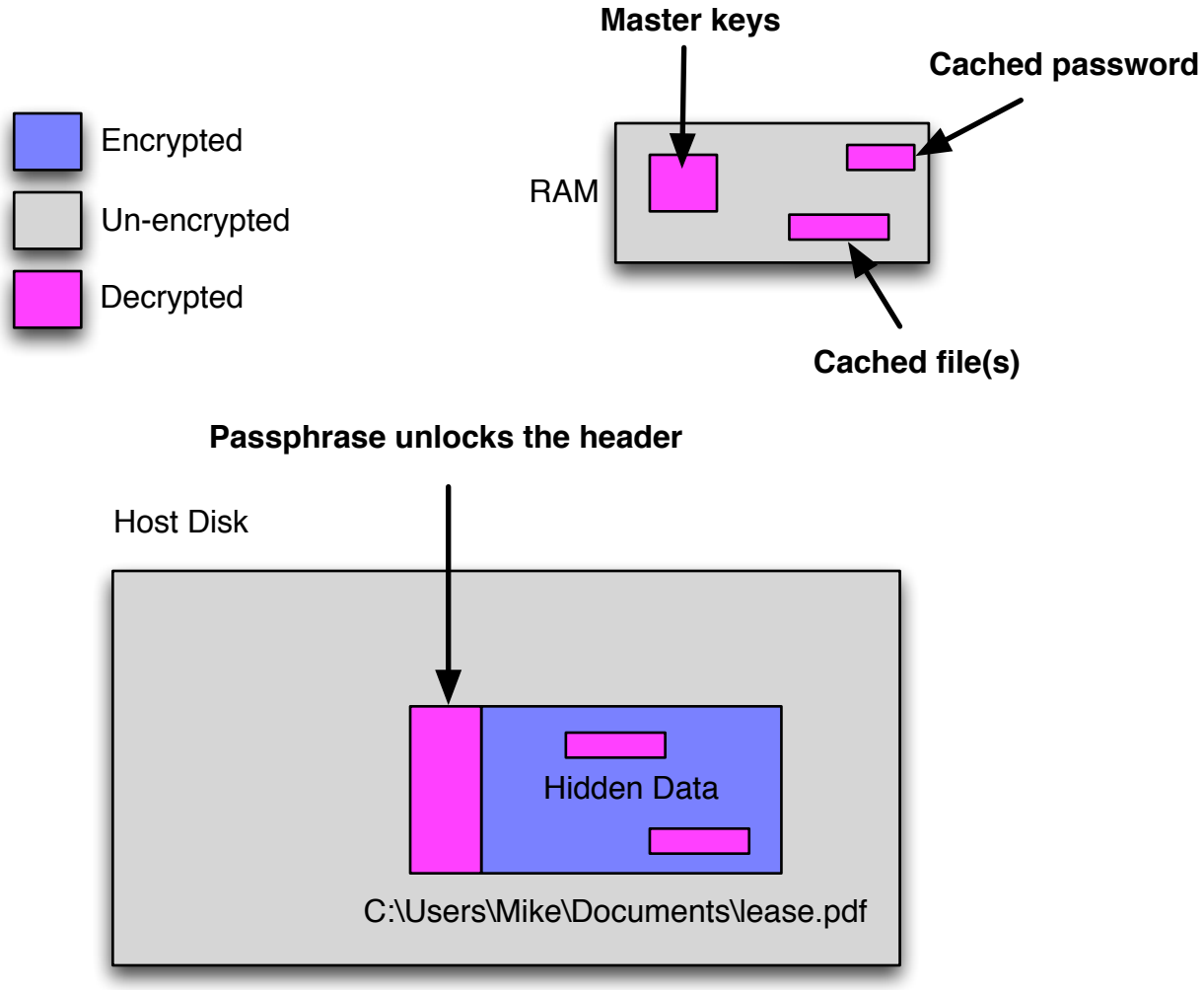
Suspect #1 – Container with Cached Passphrase



Suspect #1 - Mounting the Volume



Suspect #1 - Mounting and Accessing



Cached Passwords

- Allows auto-mounting containers at boot or several times during a session without re-entering the password
- Cached into driver memory in kernel space
  - Specifically the .data section of truecrypt.sys
  - Stays there even after un-mounting
  - Can “wipe cached passphrases”
- If you find the password, game over – no need to know encryption algorithm, master keys, etc.
  - All of this will be found when the password decrypts the 512-byte header

Password Structure

- Starting with TrueCrypt 4.x
- Previously the password was just a char *

```
'Password' : [ 0x48, {
    'Length' : [ 0x0, ['unsigned long']],
    'Text' : [ 0x4, ['array', 65, ['unsigned char']]],
    'Pad' : [ 0x45, ['array', 3, ['unsigned char']]],
    } ],
```

Investigator #1 – Password Recovery

```
$ time python vol.py -f Win8SP0x86-Pro.mem
--profile=Win8SP0x86 truecryptpassphrase
Volatility Foundation Volatility Framework
2.3

Found at 0x9cd8f064 length 31:
duplicative30205_nitrobacterium

real 0m2.746s
user 0m2.278s
sys  0m0.463s
```

Investigator #1 – Find the Container

```
$ python vol.py -f Win8SP0x86-Pro.mem --profile=Win8SP0x86 truecryptsummary
Volatility Foundation Volatility Framework 2.3

Registry Version      TrueCrypt Version 7.1a
Process               TrueCrypt.exe at 0x85d79880 pid 3796
Kernel Module         truecrypt.sys at 0x9cd5b000 - 0x9cd92000
Symbolic Link         Volume{ad5c0504-eb77-11e2-af9f-8c2daa411e3c} -> \Device
\TrueCryptVolumeJ mounted 2013-10-10 22:51:29 UTC+0000
File Object           \Device\TrueCryptVolumeJ\ at 0x6c1a038
File Object           \Device\TrueCryptVolumeJ\Chats\GOOGLE\Query
\modernimpact88@gmail.com.xml at 0x25e8e7e8
File Object           \Device\TrueCryptVolumeJ\Pictures\haile.jpg at 0x3d9d0810
File Object           \Device\TrueCryptVolumeJ\Pictures\nishikori.jpg at 0x3e44cc38
File Object           \Device\TrueCryptVolumeJ\$RECYCLE.BIN\desktop.ini at 0x3e45f790
File Object           \Device\TrueCryptVolumeJ\ at 0x3f14b8d0
File Object           \Device\TrueCryptVolumeJ\Chats\GOOGLE\Query
\modernimpact88@gmail.com.log at 0x3f3332f0
Driver                \Driver\truecrypt at 0x18c57ea0 range 0x9cd5b000 - 0x9cd91b80
Device                TrueCryptVolumeJ at 0x86bb1728 type FILE_DEVICE_DISK
Container             Path: \??\C:\Users\Mike\Documents\lease.pdf
Device                TrueCrypt at 0x85db6918 type FILE_DEVICE_UNKNOWN
```

Investigator #1 – Extract the Container

```
$ mmls rawdisk.dd
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

     Slot    Start        End          Length       Description
00:  Meta    0000000000   0000000000   0000000001   Primary Table (#0)
01:  -----   0000000000   0000002047   0000002048   Unallocated
02:  00:00   0000002048   0125827071   0125825024   NTFS (0x07)
03:  -----   0125827072   0125829119   0000002048   Unallocated

$ fls -o 2048 -r rawdisk.dd | grep lease.pdf
+++ r/r 62543-128-3:    lease.pdf

$ icat -o 2048 -r rawdisk.dd 62543 > lease.pdf
```

Investigator #1 – Decrypt the Container

```
$ truecrypt --text
      --mount-options=readonly
      --password='duplicative30205_nitrobacterium'
      lease.pdf
      /mnt/truecrypt

Enter keyfile [none]:
Enter your user password or administrator password:

$ ls /mnt/truecrypt
Chats  Emails  Pictures  $RECYCLE.BIN
```

What if the Suspect Does Not Cache Passwords?

Suspect #2 – Container with AES and No Cached Password

- Standard container
- No cached password
- Default encryption (AES)
- 20 MB with FAT file system
- 32-bit Windows 8

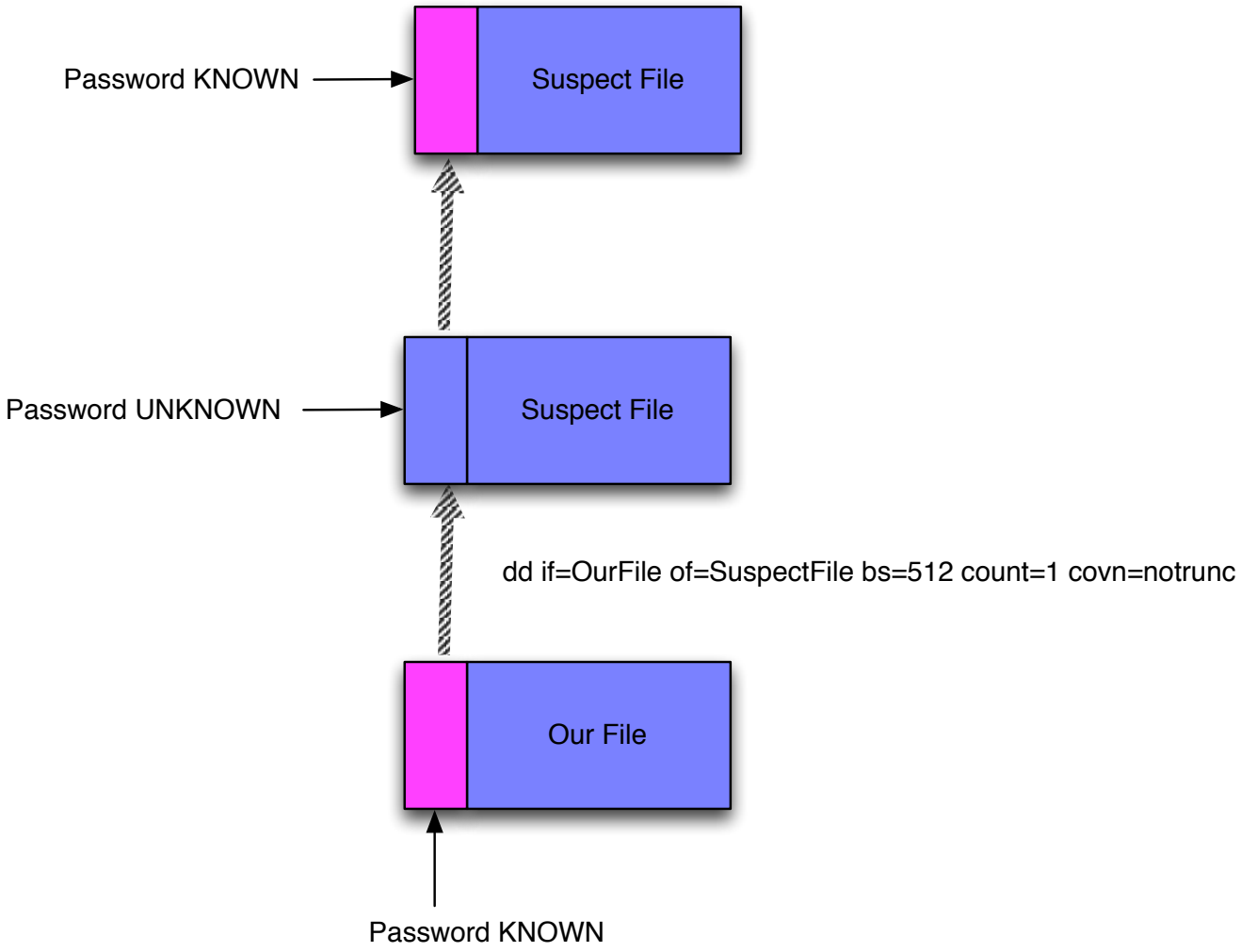
Master Keys

- Always in memory while the volume is mounted
  - Otherwise on-the-fly encryption/decryption would not work
- Default AES w/ XTS
  - Primary and secondary 256-bit keys concatenated together
  - 512-bit key (64 bytes)

Investigator #2 – Find AES Keys

```
$ ./aeskeyfind Win8SP0x86-Pro.mem
f12bffe602366806d453b3b290f89429
e6f5e6511496b3db550cc4a00a4bdb1b
4d81111573a789169fce790f4f13a7bd
a2cde593dd1023d89851049b8474b9a0
269493cfc103ee4ac7cb4dea937abb9b
4d81111573a789169fce790f4f13a7bd
4d81111573a789169fce790f4f13a7bd
269493cfc103ee4ac7cb4dea937abb9b
4d81111573a789169fce790f4f13a7bd
0f2eb916e673c76b359a932ef2b81a4b
7a9df9a5589f1d85fb2dfc62471764ef47d00f35890f1884d87c3a10d9eb5bf4
e786793c9da3574f63965803a909b8ef40b140b43be062850d5bb95d75273e41
Keyfind progress: 100%
```

Investigator #2 – Fake Header



Investigator #2 – Template Creation

```
$ truecrypt --text --create --encryption=AES
        --filesystem=FAT --hash=RIPEMD-160
        --password=ABC123
        --random-source=/dev/random
        --size=20971520
        --volume-type=normal
        our-lease.pdf

Enter keyfile path [none]:

Done: 100.000%  Speed:   76 MB/s  Left: 0 s

The TrueCrypt volume has been successfully created.
```

Investigator #2 – Header Merge

```
$ dd if=our-lease.pdf
      of=lease.pdf
      bs=512
      count=1
      conv=notrunc

1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.000658232 s, 778 kB/s
```

Investigator #2 – Patch TrueCrypt

- Force it to use ./master.key which came from the RAM dump
- Volume/VolumeHeader.cpp
- Patch based on code by Michael Weissbacher: http://mweissbacher.com/blog/tag/truecrypt/

Investigator #2 – Patch TrueCrypt

```
$ diff -u Volume/VolumeHeader.orig Volume/VolumeHeader.cpp
--- Volume/VolumeHeader.orig	2013-10-06 09:17:36.634314650 -0700
+++ Volume/VolumeHeader.cpp	2013-10-06 15:02:45.297023971 -0700
@@ -6,6 +6,10 @@
  packages.
  */
 		if (typeid (*mode) == typeid (EncryptionModeXTS))
 		{
-			ea->SetKey (header.GetRange (offset, ea->GetKeySize()));
-			mode->SetKey (header.GetRange (offset + ea->GetKeySize(), ea->GetKeySize()));
+			FILE *fh = fopen("./master.key", "rb");
+			if (fh == NULL) {
+				ea->SetKey (header.GetRange (offset, ea->GetKeySize()));
+				mode->SetKey (header.GetRange (offset + ea->GetKeySize(), ea->GetKeySize()));
+			}
+			else
+			{
+				char * buffer = (char *) malloc (65);
+				memset(buffer, 0, 65);
+				fread(buffer, 64, 1, fh);
+				ConstBufferPtr cbp = (ConstBufferPtr( (TrueCrypt::byte*) buffer, 32));
+				ea->SetKey (cbp);
+				ConstBufferPtr cbpm = (ConstBufferPtr( (TrueCrypt::byte*) buffer + 32, 32));
+				mode->SetKey (cbpm);
+				fclose(fh);
+			}
 		}
 		else
 		{
```

Investigator #2 – Decryption

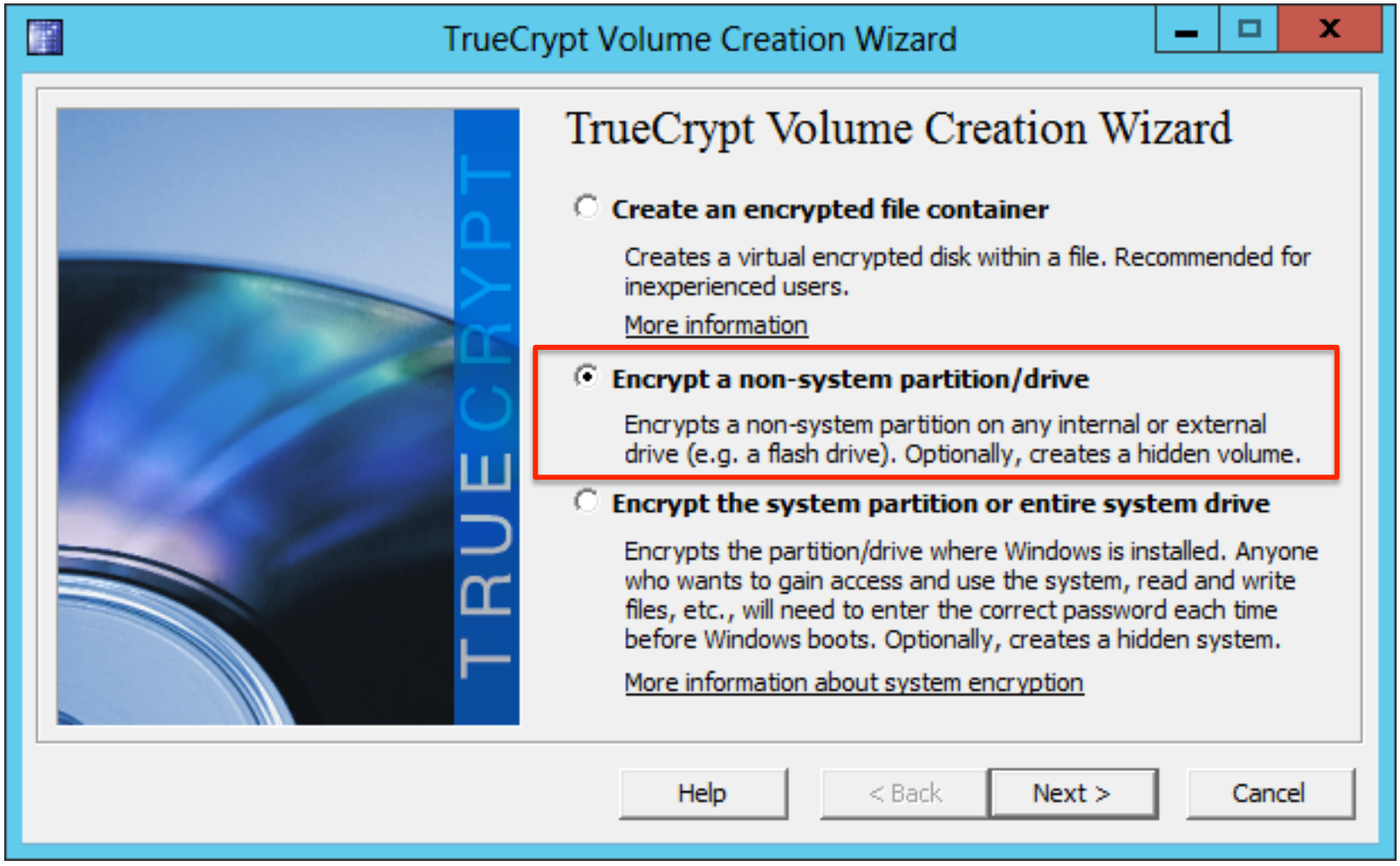
```
$ truecrypt --text
      --mount-options=readonly
      --password=ABC123
      lease.pdf
      /mnt/truecrypt

Enter keyfile [none]:
Enter your user password or administrator password:

$ ls /mnt/truecrypt
Chats  Emails  Pictures  $RECYCLE.BIN
```

What if the Suspect Does Not Use AES Encryption?

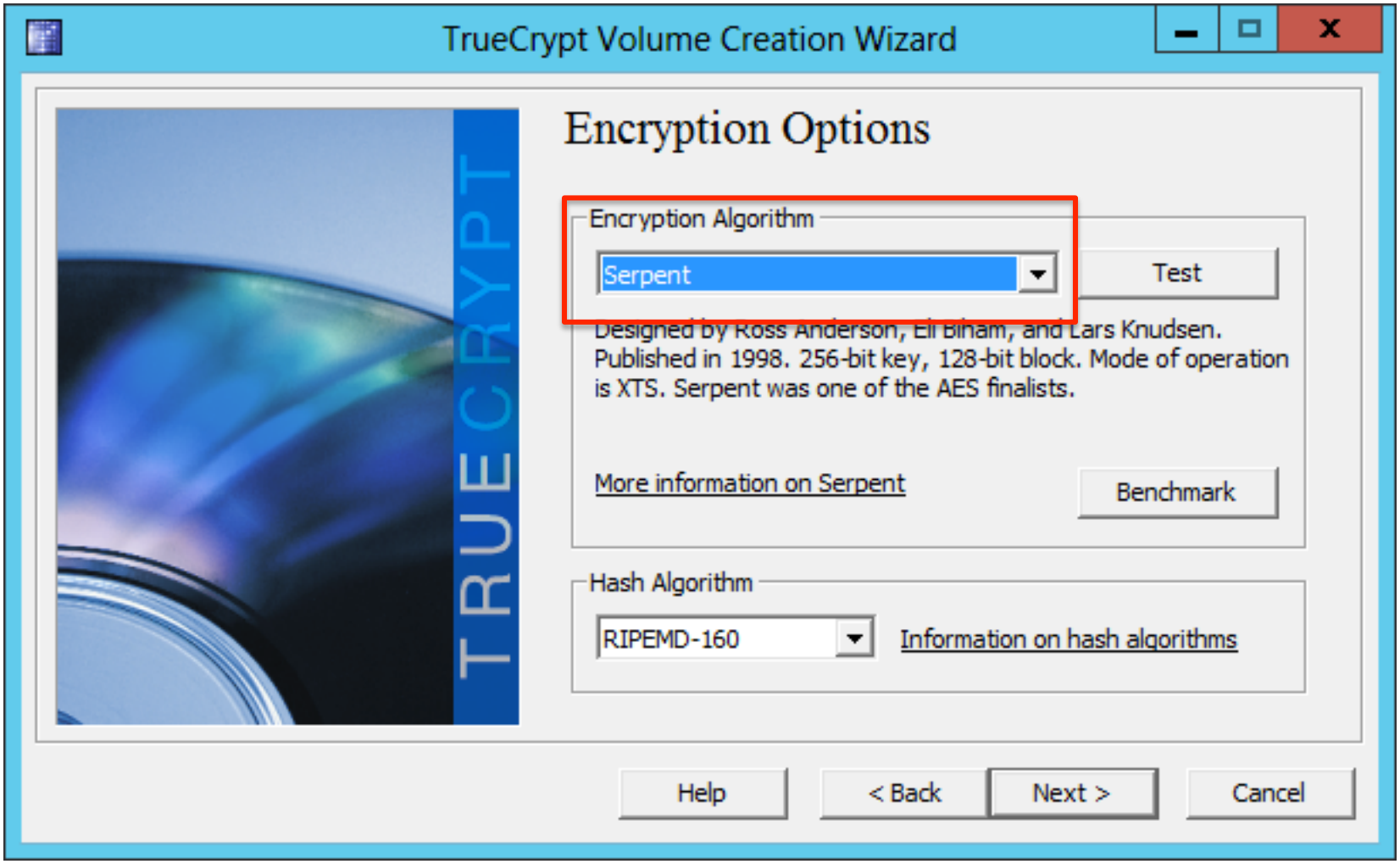
Suspect #3 – USB with Serpent and NTFS

- Non-system partition (USB)
- No cached password
- Non-default encryption (Serpent)
- 8 GB with NTFS file system
- 64-bit Windows Server 2012

Suspect #3 – Non System Partition



Suspect #3 – Serpent Encryption



Suspect #3 – NTFS File System

Investigator #3 – Initial Summary

```
$ python vol.py -f WIN-QBTA4959AO9.raw --profile=Win2012SP0x64 truecryptsummary
Volatility Foundation Volatility Framework 2.3

Process               TrueCrypt.exe at 0xfffffa801af43980 pid 2096
Kernel Module         truecrypt.sys at 0xfffff88009200000 - 0xfffff88009241000
Symbolic Link         Volume{52b24c47-eb79-11e2-93eb-000c29e29398} -> \Device
\TrueCryptVolumeZ mounted 2013-10-11 03:51:08 UTC+0000
Symbolic Link         Volume{52b24c50-eb79-11e2-93eb-000c29e29398} -> \Device
\TrueCryptVolumeR mounted 2013-10-11 03:55:13 UTC+0000
File Object           \Device\TrueCryptVolumeR\$Directory at 0x7c2f7070
File Object           \Device\TrueCryptVolumeR\$LogFile at 0x7c39d750
File Object           \Device\TrueCryptVolumeR\$MftMirr at 0x7c67cd40
File Object           \Device\TrueCryptVolumeR\$Mft at 0x7cf05230
File Object           \Device\TrueCryptVolumeR\$Directory at 0x7cf50330
File Object           \Device\TrueCryptVolumeR\$BitMap at 0x7cfa7a00
Driver                \Driver\truecrypt at 0x7c9c0530 range 0xfffff88009200000 -
0xfffff88009241000
Device                TrueCryptVolumeR at 0xfffffa801b4be080 type FILE_DEVICE_DISK
Container             Path: \Device\Harddisk1\Partition1
Device                TrueCrypt at 0xfffffa801ae3f500 type FILE_DEVICE_UNKNOWN
```

Investigator #3 – Mastering TrueCrypt

```
$ python vol.py -f WIN-QBTA4959AO9.raw --profile=Win2012SP0x64 truecryptmaster -D .
Volatility Foundation Volatility Framework 2.3

Container: \Device\Harddisk1\Partition1
Hidden Volume: No
Removable: No
Read Only: No
Disk Length: 7743733760 (bytes)
Host Length: 7743995904 (bytes)
Encryption Algorithm: SERPENT
Mode: XTS
Master Key
0xfffffa8018eb71a8  bb e1 dc 7a 8e 87 e9 f1 f7 ee f3 7e 6b b3 0a 25   ...z.......~k..%
0xfffffa8018eb71b8  90 b8 94 8f ef ee 42 5e 51 05 05 4e 32 58 b1 a7   ......B^Q..N2X..
0xfffffa8018eb71c8  a7 6c 5e 96 d6 78 92 33 50 08 a8 c6 0d 09 fb 69   .l^..x.3P......i
0xfffffa8018eb71d8  ef b0 b5 fc 75 9d 44 ec 8c 05 7f bc 94 ec 3c c9   ....u.D.......<.
Dumped 64 bytes to ./0xfffffa8018eb71a8_master.key
```

Investigator #3 – Imaging the USB

$ **sudo dd if=/dev/sdb of=usb.raw bs=1024**

7562496+0 records in

7562496+0 records out

7743995904 bytes (7.7 GB) copied, 523.606 s, 14.8 MB/s

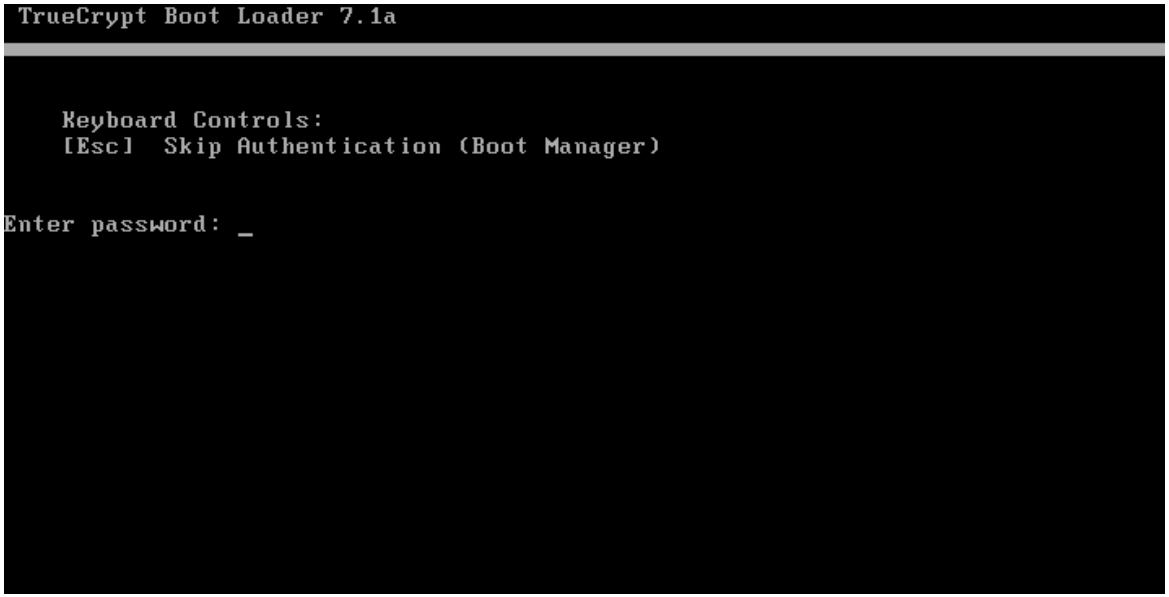
Investigator #3 – Faking the Header

- Same steps as before, but cannot use Linux (need NTFS format)
- Need a USB drive of the same model and capacity
- Between the truecryptsummary and truecryptmaster plugins, you know:
  - Encryption algorithm
  - Encryption mode
  - Size in bytes
  - File system
- Choose your own password
- Overwrite the 512-byte header

Investigator #3 – Mounting the USB as a File Container

```
$ truecrypt --text
        --volume-type=normal
        --encryption=SERPENT
        --hash=RIPEMD-160
        --mount-options=readonly
        --password=ABC123
        usb.raw
        /mnt/truecrypt/
Enter keyfile [none]:
Enter your user password or administrator password:

$ ls /mnt/truecrypt/
Credit Card Payment Form.doc  $RECYCLE.BIN          YARA User's Manual 1.6.pdf
Documents                     reviews.zip
NPA - Brooklyn.pdf            Veggie Cracker PR.doc
```

What if the Suspect Uses Full Disk Encryption?



UEFI on x64

- Support for full disk encryption on 64-bit Windows 8 and Server 2012 is listed as a future enhancement by TrueCrypt
- “All 64-bit versions of PCs running Windows certified by the Windows Certification Program will use UEFI instead of BIOS”
- http://windows.microsoft.com/en-us/windows-8/what-uefi

Dumping Cached Files

- You still have complete access to RAM
- All the cached files (exe, dll, sys, doc, pdf, txt, jpg, png, gif, etc.)
- Use the dumpfiles plugin to recover un-encrypted file contents
- Also works against the $Mft, $MftMirr

Support Matrix

Version	Release	Passphrase	Summary	Master Key
7.1a	2/2012	Yes	Yes	Yes
7.0a	6/2010	Yes	Yes	Yes
6.3a	11/2009	Yes	Yes	Yes
5.1a	3/2008	Yes	Yes	No**
4.3a	3/2007	Yes	Yes	No**
3.1a	1/2005	No*	Yes	No**

** cached passphrases are char[] buffers in the .data section – dump truecrypt.sys and run strings on it*

*** add support by compiling from source and extracting symbols from truecrypt.sys*

Part II

Windows 8 and Server 2012

- First new Windows OS supported by Volatility in the last two years
  - Volatility 2.0 supported Windows 7
  - Released August 2011

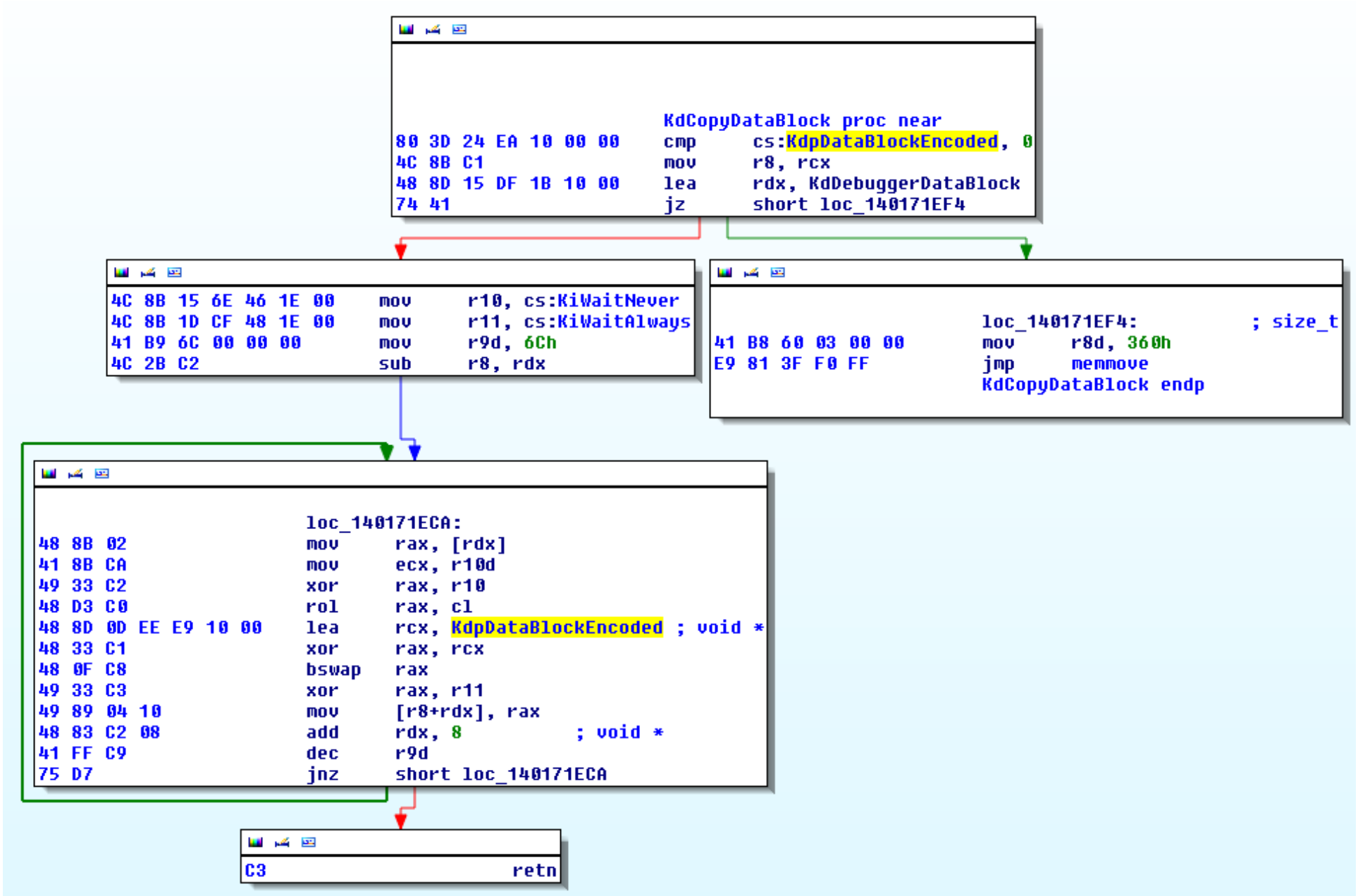
Profile Name	Operating System
Win8SP0x86	32-bit Windows 8
Win8SP1x86	32-bit Windows 8.1
Win8SP0x64	64-bit Windows 8
Win8SP1x64	64-bit Windows 8.1
Win2012SP0x64	64-bit Windows Server 2012
Win2012R2x64	64-bit Windows Server 2012 R2

Encoded KDBG

- Kernel Debugger Data Block (_KDDEBUGGER_DATA64)
- Used by memory forensics tools to find the active process and loaded module list heads
- Encoding possible since Vista
- Not enabled by default until 64-bit Windows 8 and Server 2012
- Decoded when you attach to a target kernel with Windows Debugger (windbg)
- What if you have a raw memory dump?

_KDDEBUGGER_DATA64

```
>>> dt("_KDDEBUGGER_DATA64")
'_KDDEBUGGER_DATA64' (832 bytes)
0x0    : Header                        ['_DBGKD_DEBUG_DATA_HEADER64']
0x18   : KernBase                      ['unsigned long long']
0x20   : BreakpointWithStatus          ['unsigned long long']
0x28   : SavedContext                  ['unsigned long long']
0x30   : ThCallbackStack               ['unsigned short']
0x32   : NextCallback                  ['unsigned short']
0x34   : FramePointer                  ['unsigned short']
0x38   : KiCallUserMode                ['unsigned long long']
0x40   : KeUserCallbackDispatcher      ['unsigned long long']
0x48   : PsLoadedModuleList            ['pointer', ['_LIST_ENTRY']]
0x50   : PsActiveProcessHead           ['pointer', ['_LIST_ENTRY']]
0x58   : PspCidTable                   ['pointer', ['pointer',
['_PSP_CID_TABLE']]]
[snip]
```


KdCopyDataBlock proc near
80 3D 24 EA 10 00 00 cmp cs:KdpDataBlockEncoded, 0
4C 8B C1 mov r8, rcx
48 8D 15 DF 1B 10 00 lea rdx, KdDebuggerDataBlock
74 41 jz short loc_140171EF4
4C 8B 15 6E 46 1E 00 mov r10, cs:KiWaitNever
4C 8B 1D CF 48 1E 00 mov r11, cs:KiWaitAlways
41 B9 6C 00 00 00 mov r9d, 6Ch
4C 2B C2 sub r8, rdx
loc_140171EF4: ; size_t
41 B8 60 03 00 00 mov r8d, 360h
E9 81 3F F0 FF jmp memmove
KdCopyDataBlock endp
loc_140171ECA:
48 8B 02 mov rax, [rdx]
41 8B CA mov ecx, r10d
49 33 C2 xor rax, r10
48 D3 C0 rol rax, cl
48 8D 0D EE E9 10 00 lea rcx, KdpDataBlockEncoded ; void *
48 33 C1 xor rax, rcx
48 0F C8 bswap rax
49 33 C3 xor rax, r11
49 89 04 10 mov [r8+rdx], rax
48 83 C2 08 add rdx, 8 ; void *
41 FF C9 dec r9d
75 D7 jnz short loc_140171ECA
C3 retn

KdCopyDataBlock

- Decoding depends on:
  - KiWaitNever
  - KiWaitAlways
  - Address of KdpDataBlockEncoded
- KiWait* values are also used by PatchGuard 2 and PatchGuard 3 for obfuscation of pointers and DPC objects
- Computed in KiInitializeKernel
- http://uninformed.org/index.cgi?v=8&a=5&p=10

Setting the Seeds

```
E8 9E 77 0B 00          call    KeInitializeProcess
41 C6 84 24 B5 01 00 00+mov     byte ptr [r12+1B5h], 7Fh
0F 31                   rdtsc
48 C1 E2 20             shl     rdx, 20h
48 0B C2                or      rax, rdx
8B C8                   mov     ecx, eax
83 E1 0F                and     ecx, 0Fh
48 8B D0                mov     rdx, rax
48 C1 C2 2B             rol     rdx, 2Bh
48 33 D0                xor     rdx, rax
48 D3 CA                ror     rdx, cl
48 89 15 87 17 FF FF    mov     cs:KiWaitNever, rdx
0F 31                   rdtsc
48 C1 E2 20             shl     rdx, 20h
48 0B C2                or      rax, rdx
8B C8                   mov     ecx, eax
83 E1 0F                and     ecx, 0Fh
48 8B D0                mov     rdx, rax
48 C1 CA 2F             ror     rdx, 2Fh
48 33 C2                xor     rax, rdx
48 D3 C0                rol     rax, cl
48 89 05 CD 19 FF FF    mov     cs:KiWaitAlways, rax
E9 28 FB FF FF          jmp     loc_1403648F0
```

Volatility += On-The-Fly KDBG Decoding

Algorithm

- Find the values
  - nt!KdDebuggerDataBlock
  - nt!KiWaitAlways
  - nt!KiWaitNever
  - nt!KdpDataBlockEncoded
- Use the last three to decode the first
- Buffer stored in Volatility's memory
  - Changes are not written back to disk

Other Major Changes

- Handle tables
  - Pointers to objects encoded on x64
    - SAR 0x13 (Win 8 and Server 2012)
    - SAR 0x10 (Win 8.1 and Server 2012 R2)
  - 32-bit uses a 7-bit mask but not _EX_FAST_REF
- Pool allocation strategy
  - No more protected bit (0x80000000) in pool tags
- New executive object types
  - IRTimer, WaitCompletionPacket, DxgkSharedResource, DxgkSharedSyncObject
- Structures service records, network connections are different

VAD Tree Family Tree

Operating System	Design Structure(s)
Windows XP	_MMVAD, _MMVAD_SHORT, _MMVAD_LONG
Windows 2003	Same as XP
Windows Vista	_MM_AVL_TABLE and _MMADDRESS_NODE
Windows 2008	Same as Vista
Windows 7	Same as Vista
Windows 8, 2012	_MM_AVL_TABLE and _MM_AVL_NODE
Windows 8.1, 2012 R2	_RTL_AVL_TREE and _RTL_BALANCED_NODE

Current Support

Yes

- Pool scanning
  - psscan, filescan, modscan, etc.
- List-walking plugins
  - pslist, modlist, etc.
- Process-based plugins
  - handles, vadinfo, privs, dlllist, getsids, etc.
- Registry and timeline plugins
- PE dumping plugins
  - procdump, dlldump, moddump
- Everything else
  - svcscan, consoles, ssdt, yarascan
- Conversion plugins
  - raw2dmp, imagecopy, etc.

No

- Hibernation file analysis
- GUI space plugins
- TCP connections (TCP and UDP listeners are OK)

Summary

- Pull cached passwords from TrueCrypt version 4.x (~2006) to the latest
- Identify standard and hidden container paths from (all time, or at least ~2005)
- Dump master keys for all algorithms starting with TrueCrypt 6.x (~2009) to the latest
- All Volatility plugins work on 32- and 64-bit Windows 8, 8.1, Server 2012, and Server 2012 R2 unless mentioned on the previous slide

The End

- Michael Ligh (@iMHLv)
- The Volatility Foundation (@volatility)
- http://volatility-labs.blogspot.com
- 5-day Hands-On Malware and Memory Forensics Training Course
  - Reston, VA, November 2013
  - San Diego, CA, January 2014
  - New York, NY, May 2014
  - London, UK, June 2014
- voltraining@memoryanalysis.net