Analysis
of
Window Vista
Bitlocker Drive Encryption

This is an INCOMPLETE draft version.
Visit www.nvlabs.in for updates

Nitin Kumar
Independent Security Researcher

Vipin Kumar
Independent Security Researcher
What we do?

Analyzing malware
Custom Development of S/W
Code Reviewing
Network PenTests
and anything that seems interesting!
Presentation Outline

• Bitlocker Introduction
• Modes of Operation
• Available algorithms
• Structure of Bitlocker Volume
• Different Keys used in Bitlocker
• Key Generation
• Key Storage
• Key Usage
• Data Encryption
• In non-diffuser mode
• In diffuser mode
• References
• Questionnaire
Bitlocker introduction

BitLocker Drive Encryption is a full disk encryption feature included with Microsoft's Windows Vista and Windows Server 2008 operating systems designed to protect data by providing encryption for entire volumes.

However, BitLocker is only available in the Enterprise and Ultimate editions of Windows Vista.
Modes of Operation

Bitlocker operates in one or more modes for every volume. Available modes are:-

**Basic**
- TPM only :- all keys are stored within TPM

**Advanced**
- USB:- Key is stored on an external device
- TPM + PIN:- TPM stores key with a user specific PIN
- TPM + USB:- TPM stores ½ key and USB stores another ½ half.
- TPM + USB + PIN ( available in Vista SP1):- TPM stores ½ key, USB stores another ½ half, together with a user specific PIN.
Available Algorithms

User can select encryption algorithm at the time of enabling bitlocker. Algorithm can be selected per volume. And it cannot be changed during reseal. To change algorithm, turn off bitlocker & then turn it on.

Available algorithms are

- AES 128 bit
- AES 256 bit
- AES 128 bit + Diffuser (Elephant) Default
- AES 256 bit + Diffuser (Elephant)
Bitlocker Volume Structure
Structure of Bitlocker Volume

Bitlocker volume has almost all its sectors encrypted except a few which contain metadata.

[Diagram showing the structure of Bitlocker volume with labeled components: Boot Sector, Key Meta Data VMK,FVEK, Encrypted Data]
Different Keys used in Bitlocker

Bitlocker uses a total of 5 different types of keys which are as follows:-

- VMK unlockers (These keys decrypt VMK)
- VMK (Volume Master Key is used to decrypt FVEK)
- FVEK (Full Volume Encryption Key decrypts DATA)
- TWEAK Key (Generates Sector Key)
- SECTOR Key (decrypts DATA)

Each of these will be detailed in the subsequent slides
Key Generation

Whole encryption chain depends on keys, so keys should be derived in an as random as possible method.

The above method is employed to generate all keys except Sector Key.
Key Storage

The keys are stored in the meta data of the Bitlocker Volume. Total number of meta data blocks is 3.

Key storage meta data structure as stored in Bitlocker volume
Encrypted Key Storage

The header contains size of encrypted data

- 8 byte header
- 12 byte counter
- Encrypted Data

Contains time when Bitlocker was enabled

Sample Encrypted Key
Key Encryption

The keys are encrypted either using RSA 2048 bit key or AES 256 bit. AES mode used is AES-CCM (AES–Counter with CBC-MAC)

In AES, 12 byte Counter is expanded as given below to 16 bytes

| 2 | 12 byte counter value | 0 | 0 | 0 |

Expansion of Partial Counter to 16 byte Initialization Vector
Storage of VMK

N number of VMKs can be stored. Each one having a similar structure.

<table>
<thead>
<tr>
<th>8 byte header</th>
<th>Key type Label</th>
<th>Key encrypted using itself</th>
<th>VMK encrypted using key</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>00602990</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>F2</td>
<td>00</td>
<td>02</td>
<td>00</td>
<td>08</td>
<td>00</td>
<td>01</td>
<td>00</td>
<td>00</td>
<td>93 45</td>
</tr>
<tr>
<td>006029A0</td>
<td>25</td>
<td>82</td>
<td>B3</td>
<td>B6</td>
<td>20</td>
<td>43</td>
<td>99</td>
<td>FC</td>
<td>67</td>
<td>24</td>
<td>D6</td>
<td>7C</td>
<td>ED</td>
<td>AA</td>
<td>50</td>
<td>C8</td>
</tr>
<tr>
<td>006029B0</td>
<td>48</td>
<td>48</td>
<td>24</td>
<td>3B</td>
<td>C8</td>
<td>01</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>08</td>
<td>22</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>02</td>
<td>00</td>
</tr>
<tr>
<td>006029C0</td>
<td>01</td>
<td>00</td>
<td>44</td>
<td>00</td>
<td>69</td>
<td>00</td>
<td>73</td>
<td>00</td>
<td>6B</td>
<td>00</td>
<td>50</td>
<td>00</td>
<td>61</td>
<td>00</td>
<td>73</td>
<td>00</td>
</tr>
<tr>
<td>006029D0</td>
<td>73</td>
<td>00</td>
<td>77</td>
<td>00</td>
<td>6F</td>
<td>00</td>
<td>72</td>
<td>00</td>
<td>64</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>5C</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>006029E0</td>
<td>03</td>
<td>00</td>
<td>01</td>
<td>00</td>
<td>00</td>
<td>10</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>OC</td>
<td>13</td>
<td>38</td>
<td>E1</td>
<td>6C</td>
<td>65</td>
<td>F3</td>
</tr>
<tr>
<td>006029F0</td>
<td>70</td>
<td>00</td>
<td>C7</td>
<td>BE</td>
<td>71</td>
<td>DD</td>
<td>E7</td>
<td>92</td>
<td>40</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>05</td>
<td>00</td>
<td>01</td>
<td>00</td>
</tr>
</tbody>
</table>
Generating Recovery Key from Recovery Password

In case of system modification, user is asked to type a 48 digit key which will unlock the volume. Pseudocode given below

1. Divide each block by 11, if the remainder not 0 in all cases the key is not valid
2. Collect the quotients, and concatenate them to obtain a 128 bit key.
3. Take a 88 byte buffer and zero it. The structure of the buffer is as follows
   struct {
     unsigned char sha_current[32];
     unsigned char sha_password[32];
     unsigned char salt[16];
     int64 hash_count;
   };
4. Take SHA256 of the key and place it in the above structure in sha_password
5. The salt is place in the salt field of the above structure
6. Now run a loop 0x100000 (1048576) times
7. Find SHA256 of the entire structure and place it in sha_current field
8. Increment hash_count field counter in the structure
9. Repeat steps 6 through 9, till the loop is over
10. Take the first 32 bytes of the structure as the 256 bit key which can be used to decrypt the VMK corresponding to this key
Generating Recovery Key from Recovery Password

Block Diagram showing conversion from Recovery Password to Recovery Key
Startup Key and/or USB Key

Block Diagram showing usage of Startup Key and USB Key
TPM

Block Diagram showing usage of Startup Key and USB Key
Full volume Encryption Key (FVEK)
FVEK

FVEK is used to store data ion the volume.

Its size is different according to:

- AES 128 bit               size 128 bits
- AES 256 bit               size 256 bits
- AES 128 + diffuser    size 512 bits (half of the bits are unused)
- AES 256 + diffuser     size 512 bits
# FVEK Structure

FVEK is broken into two parts if larger than 256 bits

<table>
<thead>
<tr>
<th>FVEK 128 bit</th>
<th>FVEK 128</th>
<th>Unused 128</th>
<th>Tweak key 128</th>
<th>Unused 128</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES 128</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FVEK 256 bit</th>
<th>FVEK 256 bit</th>
<th>Tweak key 256 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES 256</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AES 128 + diffuser

AES 256 + diffuser
Sector key from TWEAK key

Pseudocode

- Take a buffer of 16 bytes, zero it.
- Now copy the Sector Number in little endian format and encrypt it with TWEAK key to obtain first 16 bytes of Sector key.
- Take a buffer of 16 bytes, zero it.
- Now copy the Sector Number in little endian format and make the 16th byte as 128 or 0x80, now encrypt it with TWEAK key to obtain remaining 16 bytes of Sector Key.
- Concatenate both part to obtain full 32 byte or 512 bit Sector Key
Sector key from TWEAK key

Sector Key
first 16 bytes

Sector Key
last 16 bytes

32 byte

1
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
1
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
80
Diffusers A & B

The Diffusers just diffuse the data ie they mingle up the bits
Bitlocker has 2 diffusers called Diffuser A and Diffuser B

Diffuser doesn't need any keys and thus doesn't need to be broken to defeat bitlocker.

It's just based on XOR and mod operation
Diffuser B

Diffuser B in decryption direction
It's represented by
for i = 0, 1, 2, ..., n
\[ d[i] = d[i] + (d[i+2] \ XOR (d[i+5] << Rb[n \mod 4])) \]

where Rb = [ 0 ,10 ,0,25 ]

To obtain encryption function, just change first + to -

NOTE:- data is processed in 32 bit blocks
<<< is left rotate operation
Diffuser A

Diffuser A in decryption direction
It's represented by
for i = 0, 1, 2, ..; n
\[ d[i] = d[i] + (d[i-2] \text{ XOR} (d[i-5] \ll \text{Ra}[n \mod 4])) \]

where Ra = \[ 9, 0, 13, 0 \]

To obtain encryption function, just change first + to -

NOTE:- data is processed in 32 bit blocks

\ll\ll\ll\ll is left rotate operation
Data Encryption

In AES 128 bit mode and AES 256 bit mode, AES-CBC mode is used with initialization vector (16 zero bytes)

However, if a diffuser capable mode is selected, then things turn out to be little bit more complex
Data decryption in diffuser capable mode

CipherText

AES CBC

Diffuser B 3 times

Diffuser A 5 times

XOR

Plain text

FVEK

TWEAK key

Sector Key 512 bits
Quick Rewind
Recovery Password
- 48 Digit Recovery Password
- 128 bit key
- SHA 256 with salt
- 256 bit Recovery Key

TPM + PIN
- PIN
- SHA 256
- 2048 bit RSA key
- PCR
- XOR

TPM + USB
- PIN
- SHA 256
- 256 bit Key
- PCR
- XOR

TPM + PIN + USB
- PIN
- SHA 256
- 2048 bit RSA Key
- PCR
- XOR

VOLUME MASTER KEY (VMK) 256 bit

Clear Key
- 256 bit Recovery key

USB key
- 256 bit Recovery key

TPM
- 2048 bit RSA key
- PCR

256 bit Recovery Key

2048 bit Key
Data decryption in diffuser capable mode

CipherText

AES CBC

Diffuser B 3 times

Diffuser A 5 times

XOR

Plain text

FVEK

TWEAK key

Sector Key 512 bits
Tool Release
Tool features

- Transparent access to Bitlocker volumes (if user supplies appropriate keys)
- 2 modes are supported (using Recovery Password/USB startup key)
- Currently provides only read-only access but write access can be added
- Ability to process partition image files
- Ability to convert Bitlocker Volume to NTFS volumes permanently.
References

- Nitin Kumar, Vipin Kumar. Vbootkit: Compromising Windows Vista Security
- Randall Hyde. Art of assembly Language
Questionaire ?

Questions
Comments
email us at

nitin@nvlabs.in
vipin@nvlabs.in

http://www.nvlabs.in
Thank you