The Exclude Signature Vulnerability

Analysis of the Rescue File of BestCrypt Volume Encryption

Author: Jan Vojtěšek
Supervisor: Ing. Josef Kokeš
Czech Technical University in Prague
Faculty of Information Technology

May 11, 2017

1 The Exclude signature

The XTS mode of operation transforms plaintext into ciphertext of the same size [2]. This poses a problem if disk encryption software wants to store store some extra data on the encrypted volume, since this extra data would not fit. BCVE solves this by reserving some sectors on the encrypted volume [1]. Those reserved sectors are used to store configuration data, encrypted `KEY_AND_HASH` structures, encrypted content of the replaced first sector, etc. The BCVE driver does not encrypt those reserved sectors, since some of them need to be available in plaintext during mount. To indicate which sectors should not be encrypted, the contents of those reserved sectors always start with a 16-byte signature (Exclude signature). This signature is the same in all versions of BCVE.

The issue is that the BCVE driver relies on this signature—any sector that starts with this signature (even if it is not a sector reserved by BCVE) will not be encrypted. This is a highly unusual behavior and I consider it to be a design flaw. The rest of this section explores how might an adversary use this behavior to his advantage.

The Exclude signature is 16 bytes long and consists of hexadecimal digits of *pi*. It is therefore highly unlikely that someone would accidentally store those 16 bytes on an encrypted volume. However, some applications might use the same hexadecimal digits of *pi* for whatever reason. For example, many cryptographic algorithms use so called "nothing up my sleeve numbers" to demonstrate that some constants have no hidden backdoor properties [6]. A popular choice is to use constants generated from *pi*. For instance, Blowfish uses hexadecimal digits of *pi* for its S-boxes, so the Exclude signature is included in most of its implementations. If this signature gets aligned to sector boundary, a whole sector will stay unencrypted. This sector is likely to contain the rest of the Blowfish S-box, but an adversary will at least get information that Blowfish might be used. The Exclude signature is also present in most BCVE executable and rescue files.

The rest of this section is going to assume that NTFS is used. This should not be a big loss of generality, since most encrypted volumes are likely to be formatted with NTFS and most concepts should apply to other filesystems as well.

1.1 Leaking contents of a single file

In NTFS, contents of two different files will never be stored in the same cluster [4]. Since cluster size in NTFS is always a multiple of BCVE sector

size[1], contents of two different files will also never be stored in the same BCVE sector. So it is not possible for an adversary to leave contents of a file unencrypted by writing the Exclude signature to another file. However, in some scenarios, it might be possible for an adversary to (indirectly) write to a file whose contents he should not be able to read. If an adversary is able to write the Exclude signature to such a file and align it to sector boundary, the whole sector (possibly containing some secret data) will stay unencrypted.

For instance, if a victim is running a database server and an adversary is able to indirectly insert data into this database, the adversary might insert many records containing the Exclude signature. Some of them might get aligned to sector boundary and thus leave other records containing potentially sensitive data in plaintext.

Some major browsers such as Mozilla Firefox and Google Chrome also store persistent HTTP cookies in a single file. An adversary running a website or able to perform a man-in-the-middle attack on unencrypted HTTP traffic can make a victim store arbitrary cookies in this file. The adversary might thus be able to leave cookies belonging to other domains stored in plaintext. Note that this does not affect session cookies, because they are not supposed to be written to disk.

Some email clients (such as Microsoft Outlook) also store email messages in a single file. By sending an email, an adversary could leave some other email messages unencrypted. Log files are also a risk—an adversary could trigger an event that will get logged (and contain the Exclude signature) and potentially leave other log events in plaintext, etc.

More powerful adversaries might also be able to standardize the Exclude signature for a signature of another file format. Then (unless the files are compressed, encrypted, or small enough to be stored directly in the NTFS MFT file record) the first 512 bytes of files in this file format are going to be left in plaintext on all volumes encrypted with BCVE.

In conclusion, there are many ways that an adversary might be able to exploit this flaw. Many of them are hard to carry out or very unreliable. However, they are still a possibility and a resourceful and/or lucky adversary can succeed in leaving sensitive data in plaintext. That is why I am urging Jetico to get this fixed as soon as possible.

[1] BCVE always uses XTS encryption with sector size of 512 bytes—even if the underlying HDDs use 4 KiB sectors.

1.2 Leaking metadata of other files in the same directory

To enable fast directory lookups, NTFS stores directory contents in a B-tree that is sorted by filenames [4]. The nodes of this B-tree contain various metadata, such as a filename, a file size, and four timestamps. By creating a file with a name consisting of padding and eight UTF-16 characters, I was able to leave a sector containing a part of a B-tree node unencrypted. The unencrypted sector contained metadata of three other files. An adversary able to create files with arbitrary filenames might be able to exploit this as well and leak metadata of other files.

1.3 Modifying contents of an encrypted volume

The contents of sectors containing the Exclude signature also do not get decrypted on read operations. An adversary able to modify raw encrypted sectors on a volume is therefore also able to modify plaintext sectors—the only obstacles are that the plaintext sector has to start with the Exclude signature and that he has to write a whole sector at once. If the adversary knows where some files are physically stored on the encrypted volume[2], he can overwrite their contents. By doing this, he might be able to overwrite some key data files or insert a backdoor into an executable file[3]. If the adversary does not know the position of a file precisely, he can "spray" the volume contents with sectors containing the Exclude signature, a NOP slide, and a shellcode. This naturally entails the risk of overwriting some important data structures, but it is possible that the shellcode will get loaded into executable memory and executed.

The attacks described in this subsection require an adversary who is able to modify raw data on encrypted volumes and an unsuspecting victim who will use the system after this modification. This most likely means that the adversary had unsupervised physical access to the encrypted volume for a certain amount of time. Under this attack model, other attacks are possible—such as modifying the boot loader [5] or installing a hardware keylogger. If the system and boot volumes are not encrypted (but some other volume is encrypted), an attacker with physical access can also inject code directly into some frequently used unsigned executable files.

[2] Note that the location of some Windows system files after a clean Windows installation might be to a certain degree predictable.

[3] If the Exclude signature in any BCVE executable file gets aligned to sector boundary (and thus leak the position of this executable file inside the encrypted volume) and this file is not fragmented (i.e. stored in a single NTFS data run), an adversary should be able to reliably insert a backdoor into BCVE.

Most of those attacks requiring physical access are well known and some protections exist that make it harder for adversaries to perform those attacks. Protections that BCVE supports include UEFI Secure Boot [3] and checksumming parts of the boot loader [1]. Those protections are completely ineffective against the newly presented attacks described in this subsection.

References

[1] Horňák, J. *Bezpečnostní analýza programu BestCrypt Volume Encryption*. Master's thesis, Czech Technical University in Prague, Faculty of Information Technology, 2016.

[2] IEEE. *Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices*. Technical report, 2007, `http://grouper.ieee.org/groups/1619/email/pdf00086.pdf` [Online; accessed 03-May-2017].

[3] Jetico Inc. Oy. *Disk Encryption Supporting UEFI Secure Boot Now Complete In BestCrypt By Jetico*. `http://www.jetico.com/about-jetico/newsroom/374` [Online; accessed 05-May-2017].

[4] Russinovich, M. E.; Solomon, D. A.; Ionescu, A. *Windows Internals, Part 2*. Microsoft Press, 6th edition, 2012, ISBN 978-0-7356-6587-3.

[5] Rutkowska, J. *Evil Maid goes after TrueCrypt*. 2009, `https://theinvisiblethings.blogspot.cz/2009/10/evil-maid-goes-after-truecrypt.html` [Online; accessed 05-May-2017].

[6] Schneier, B. *Applied Cryptography (2Nd Ed.): Protocols, Algorithms, and Source Code in C*. New York, NY, USA: John Wiley & Sons, Inc., 1995, ISBN 0-471-11709-9.