Attacking the BitLocker Boot Process

Sven Türpe, Andreas Poller, Jan Steffan, Jan-Peter Stotz
and Jan Trukenmüller
Fraunhofer-Institute for Secure Information Technology SIT
Rheinstrasse 75,
64295 Darmstadt, Germany
{tuerpe,poller,steffan,stotz,truken}@sit.fraunhofer.de

We discuss attack scenarios against the TPM-based boot process of BitLocker. BitLocker is a disk volume encryption feature included in some recent versions of Microsoft Windows. BitLocker is capable of using the TPM to manage all or a portion of its secret encryption keys. Specifically it uses the sealing feature to ensure keys are released only if the platform is in a predefined, trusted state. We present six ways in which an attacker may gain access to secret key material by manipulating the boot process in ways not prevented by the trusted computing technology. We also discuss their causes and contributing factors.

1 Introduction

The trusted computing platform as specified by the Trusted Computing Group does not support secure booting [1, 13]. Not in a strict sense, that is: there are indeed features in the platform that allow the TPM to keep track of the boot process and notice tampering with most (but not all [5]) components involved. Other functions of the TPM, such as remote attestation and sealing, are designed to use the result of it.

This leads to the question what exactly the capabilities and limitations of the existing functions are. In this paper we explore this question along one particular software design and implementation: BitLocker. Included with some editions of Microsoft Windows Vista and Windows Server, BitLocker encrypts volumes on disk and uses the sealing function of a v1.2 TPM for some aspects of its key management. We discuss several attack scenarios against the BitLocker boot process. So far, our work is limited to theoretical considerations and to analysis. We do not (yet) discuss practical implementation of the attacks we describe, and we try to fully understand the problem before devising solutions. Our analysis is based on the version of BitLocker included with pre-SP1 versions

of Windows Vista Ultimate. We expect our points to remain valid for the SP1 version but haven't verified this yet.

The remainder of this paper is organized as follows. Section 2 briefly describes the design of BitLocker, focusing on its key management and how it is using the TPM. Our adversary model is outlined in section 3. Section 4 describes attack scenarios that seem feasible and either yield secret key or data or achieve some important steps towards a successful attack. Causes and contributing factors are discussed in section 5, followed by the conclusions in section 6. Related literature is referenced where appropriate but not specifically discussed.

A Disclaimer Note that there are two distinct attack strategies against which BitLocker should ideally protect. *Opportunistic* attacks use only what is easily obtained under common real-world conditions. An example is recovering data on a disk or computer that has been bought in used condition from somebody else, or stolen somewhere. A *targeted* attack is different in that the attacker attempts to get access to data on a specific, predetermined disk or machine, usually within some time and resource constraints. According to Microsoft, BitLocker is designed to withstand at least opportunistic attacks. Considering targeted attacks as we are doing here may be beyond its specification. However, disk encryption along with TPM-based key management might be expected and perceived to be more powerful than what the manufacturer is willing to promise, and we deem it useful to explore the actual security properties and limitations regardless of claims and cautionary notes.

2 An Overview of BitLocker

2.1 Security and Attack Objectives

The primary security objective is confidentiality of any data stored on the encrypted volume. As a corollary, secret keys that could be used, along with the contents of the disk, to obtain the cleartext are to be kept confidential as well. An attack against BitLocker can be considered successful if through the attack:

- the attacker obtains the cleartext of all data on the encrypted volume, or a considerable portion thereof, or
- the attacker obtains such supposedly secret key material that obtaining the cleartext from ciphertext becomes trivial, and obtaining the ciphertext is not more difficult than it would be to obtain cleartext from an unencrypted volume, all side conditions being the same.

Manipulating either the BitLocker instance itself or its execution environment would be an obvious and straightforward way of obtaining both cleartext data and secret key material. Integrity of the BitLocker software and the platform executing it is therefore an important secondary security objective. A tertiary objective is availability; there

are few usage scenarios where it is acceptable to risk losing all data just to keep them confidential under all circumstances.

2.2 Integrity Model and Design Constraints

BitLocker works, at boot time, as a component of the boot loader and later as a driver of the operating system kernel. Its design assumes that the kernel boots from a BitLocker-protected volume, that BitLocker sufficiently protects the integrity of data on this volume, and that anything that happens after initiating the OS boot process is sufficiently controlled by other security mechanisms. We do not challenge these assumptions here; see [6, 8] for two known attacks against the running system.

According to these assumptions, BitLocker has to protect the integrity of the boot loader and its execution environment up to the point where the kernel can be read from the locked volume. This code is read from an unencrypted part of the disk and needs to be supplied with a secret key for the AES algorithm. This is where the TPM is being used in. BitLocker uses the *sealing* function to store all or part of its key material in such a way that it becomes accessible only if the platform configuration as represented by the PCR values is in line with the reference configuration. The reference configuration is determined by the administrator accepting the current system configuration at some point in time. This adoption of a reference configuration is initially done during BitLocker activation but can be repeated at any time from the running Windows system.

2.3 Key Management and Recovery Mechanisms

Apart from special cases—BitLocker can also be operated without a TPM or with all key material being managed by the TPM—key material is divided. One part is managed by the TPM and released only if the platform is in the trusted state, the other is supplied by the user as a password and/or key file on a USB memory stick.

If the TPM works as desired, there is no way according to the design to gain access to all required key material if the platform state measured is different from the reference state. This is intended if the platform state is modified by an attack, it is not, if state is modified for a legitimate reason and the change can not be reverted easily, e.g. after component failure and repair. BitLocker therefore offers two recovery mechanisms, the recovery password and the recovery key. Both are designed to circumvent the TPM and supply BitLocker with its secret key independent of the current platform state. The recovery mechanisms don't correct the problem, though. This is left to the administrator who, after the recovery boot, may set a new reference state from the running system.

The actual encryption key does not change during the recovery process.

2.4 User Experience

The user experience hides most of the details. When switching on their PC, users will experience a text-mode prompt for their PIN and/or USB stick if the platform is in reference state. Otherwise they will be prompted for their recovery key or password. Depending on how the computer is being used, users may experience a recovery prompt

from time to time, e.g. after accidentally leaving a bootable CD or DVD in the drive or a bootable USB stick plugged into their computer.

3 Adversary Model

As any disk encryption scheme, BitLocker is supposed to protect the confidentiality of data against an adversary who gets physical access to the computer or even brings the computer into his possession. We therefore assume the attacker is interested in the data stored on the BitLocker-protected volume. Furthermore, we limit our discussion to such attacks that do actually exploit physical access in some way.

- Copy the encrypted volume and any other data on disk but return the computer
- Modify the encrypted volume and any other data on disk and return the computer
- Modify the hardware and return the computer
- Take away the computer, returning it later or not
- Replace the computer with an identical-looking copy
- If modified software gets executed on the machine, use any peripheral component of the system

We will not discuss the effort required in each case but assume that each of these attack building blocks is considerably cheaper than brute-forcing AES. We further assume that each of these physical attack building blocks may be executed an arbitrary number of times in combination with any other, provided a single rule is observed: if the attack becomes obvious, the user and administrator will be cautious in all subsequent steps. In other words, the user and administrator will not knowingly support an attack.

4 Attack Scenarios

4.1 Replace and Relay

This is a hardware-level phishing attack. The attacker replaces the entire target machine with another computer prepared for the attack. The replacement, when turned on, produces all the messages and prompts that the original machine would have produced. Up to the point where BitLocker would start, it takes all user inputs (via keyboard or USB) and relays them to the attacker, e.g. using radio. The attacker, being in possession of the unmodified original system, uses this information to start up the stolen computer.

Requirements This attack requires that:

- the attacker is capable of replacing the BitLocker-protected machine altogether with an identically-looking copy, and
- the machine is plausibly turned off or in suspend-to-disk mode when the legitimate user returns, and
- the replacement device is capable of relaying user input to the attacker.

The attacker will have to remain—or leave some device—in proximity to the target until the next boot is initiated by the victim. The attacker will also need some prior knowledge of non-secret facts, specifically everything that might be needed to perfectly reproduce the user experience.

Result As a result of this attack, the attacker receives the user-controlled secrets. Depending on the mode in which BitLocker is deployed on the target system, the result is either key material or authentication credentials or both. Either one can be used in conjunction with the unmodified system to start up the operating system. Security mechanisms of the operating system remain intact; another attack will be required to actually access any encrypted data. Such attacks exist [6, 8]. The attack will likely be noticed right after the victim provided credentials or keys to the spoofed machine. This attack may be combined with any attack that yields the TPM-managed portion of the key material.

Extensions and Variants A more sophisticated version of this attack involves two-way communications, turning the replacement into a terminal of the stolen target machine. This would probably require quite some additional effort but might extend the time span between success and detection of the attack. All variants of this attack may also be attempted against recovery mechanisms, which yields sufficient key material to decrypt disk contents immediately.

4.2 Plausible Recovery

The attacker modifies the BitLocker code on disk, adding a backdoor. Such a backdoor could be as simple as saving a clear key in some location on disk or elsewhere in the system from where it can be retrieved later. This modification will of course be detected the next time the system is started by a legitimate user. However, the attacker hopes that the user applies one of the TPM-independent recovery mechanisms to overcome the problem. The attacker later visits the system again to collect the key. Encrypted volume data could be copied during each visit to the target system as the actual encryption key does not usually change.

Requirements This attack requires:

- that recovery mechanisms are used at all, and
- that the attacker can physically access the target machine at just the right time without taking it away permanently, and
- that the reported platform validation error seems plausible for the victim.

One obvious implementation of this attack would be to wait for a situation that plausibly changes the state of the platform, such as a repair. It may also be possible to provoke such a situation. The attacker will then have to sneak into the process somewhere before the user accepts the seemingly legitimate modification. This would mask the malicious change with the legitimate one.

Result If the attack succeeds, the attacker has successfully planted a backdoor into the system in such a way that *all* software-based security features could be circumvented. The attack is unlikely to be noticed by the victim. In order to get both the encrypted data and the secret key the attacker will have to visit the target system at least twice. However, the backdoor may also use other channels to leak cleartext data, possible increasing the risk of detection.

4.3 Spoofed Prompt

Similar to the plausible recovery attack, the attacker modifies BitLocker on the target system and lives with the fact that the TPM will detect this modification. The attacker adds code that spoofs the user interface of BitLocker up to the point where the user has given up his secrets. The malicious code may spoof either the normal-operations UI or the prompt for a recovery key.

Requirements This attack requires that the attacker can physically access the target system. It is not necessary that the attacker takes the system away permanently.

Result The attack is easily detected as soon as secrets have been provided to the spoofed prompt. After detection it is generally possible to prevent the attacker from interacting with the compromised system again. Also, the TPM will refuse to unseal its part of the key material while the platform is in this modified state. If a recovery prompt is successfully spoofed and operated by the user, the attack will yield sufficient key material for decryption of a volume.

Extensions Although it may work under some circumstances, this attack does not appear very critical. However, the next subsection describes a more critical extension.

4.4 Tamper and Revert

The tamper and revert attack extends the spoofed prompt attack. Instead of simply accepting that platform modifications can be detected, the attacker attempts to exploit tampering yet hiding it. This becomes surprisingly easy if one additional boot cycle is possible. The attacker could make a temporary modification to TPM-verified code. If we stick to the spoofed prompt example, this means to add a cleanup function to the malicious code, whose purpose it is to restore the former platform state. After a reboot—which might be initiated by the malicious code after showing a bogus error message—the platform state as measured will be compliant with the reference PCR values again.

Requirements Requirements are similar to those of the *spoofed prompt* attack. In addition the attacker needs to get away with a boot cycle after platform integrity failure without disturbing the victim so much as to spoil further steps of the attack. Depending on how the credentials or keys obtained are transmitted to the attacker, a further visit to the system may or may not be required.

Results This attack yields copies of keys controlled by the user. In a simple implementation these keys will end up in clear somewhere on the target system itself but more sophisticated approaches can be imagined, for instance sending key somewhere using a built-in WLAN interface. Additional effort is required on the attacker's part to gain access to TPM-managed key material.

4.5 Preemptive Modification

This attack is similar to the plausible recovery attack, but at a different point in time. The recovery attack targets systems on which BitLocker has already been activated. Preemptive modification attacks earlier, before BitLocker has been activated at all.

When defining the reference state for future booting, the operator has no choice other than using the current platform state. BitLocker does not provide the user with any means of verifying that this current state has or hasn't any particular property. If an attacker manages to modify critical parts of the platform before BitLocker is activated, this modification therefore goes unnoticed and will be incorporated into the trusted (but not trustworthy) platform state.

Requirements Preemptive modification requires that the attacker gets physical access to the target system *before* BitLocker is activated. Arbitrary modifications are possible at that time that would weaken the security of the BitLocker instance affected forever. Another physical visit may be required later to retrieve a disk image for decryption or leaked cleartext. However, the system may also be modified in such a way that it leaks data at runtime. Everyone who gets physical access to the machine or OS installation media before BitLocker setup is a potential attacker.

Results The attacker potentially gains read and write access to all data handled on the system throughout its lifetime. This attack is hard to detect unless there are additional means of verifying the integrity of executable code against external references.

4.6 TPM Reset

TPM reset attacks have been described in the literature before and are included here for the sake of completeness. The attacker, in temporary or permanent possession of the target system, attaches equipment to the hardware to record the measurements sent to the TPM during a clean boot process. Next the machine is booted with a system of the attacker's choice, e.g. from a CD. While this system is running, the attacker performs the reset attack and replays the clean sequence of measurements to the TPM. As a result, the attacker will be able to unseal the TPM-managed portion of the key material outside the trusted operating system.

Reset attacks have been described and demonstrated against implementations of version 1.1 of the TCG specification. The current version 1.2 contains mitigations that make such attacks more difficult to mount but by no means impossible.

Requirements The attacker needs to be in possession of or co-located with the target computer for some amount of time. This would be the case e.g. after the attacker has stolen the target computer. Mounting the reset attack may require irreversible modifications to the hardware, traces of which could be detected at least by close examination.

Results As a result of this attack, an attacker in possession of the target computer is able to extract key material sealed by the TPM. The only prerequisite is that the attacker must be able to boot the machine into the trusted platform state at least once. In other words, this attack cannot be applied after one that modified the platform in a detectable way unless this modification is reversible.

5 Causes and Contributing Factors

This section identifies factors that make the overall system—a PC with BitLocker and Trusted Computing technology—vulnerable to the attacks described above. Factors include fundamental properties of the security mechanisms involved as well as features in the design and implementation of BitLocker and the Trusted Computing platform.

Passive TPM. Theory suggests that secure booting requires an appropriate action if the measured state deviates from the reference. For instance the boot process might be halted, or it may be possible to fix the issue once it had been detected [13]. The Trusted Computing platform, however, has been designed to work with a *passive* TPM: functions like sealing and attestation depend on prior measurements of platform state, but the TPM does not actively enforce anything. Our attack scenarios confirm the requirement of active enforcement for secure boot. The spoofed prompt and tamper and revert attacks would be much harder to carry out

if the boot process would stop immediately after a modification had been detected. However, recovery features may also be harder to implement in this case.

No trusted path to the user. BitLocker uses secrets to authenticate the user: the PIN and key material. The channel between the legitimate user and the system in a trusted state is prone to spoofing and man-in-the-middle attacks (replace and relay; spoofed prompt; and tamper and revert). or specifically, the system lacks context-awareness and the user is unable of authenticating the system. Similar problems exist elsewhere, e.g. ATM skimming. Both directions of authentication can be discussed separately:

No context-awareness. The BitLocker has no means of determining whether the computer is under control of a legitimate user or somebody else. It simply assumes that whoever provides the correct key or credential is a legitimate user. Although requesting a PIN or key may be interpreted as authentication, it is not a very strong one, and adding stronger authentication may be difficult.

Lack of system authentication. While BitLocker is capable of authenticating its user at least in the weak sense described above, the user has no means of verifying authenticity and integrity of the device. Keys and passwords are to be entered into an unauthenticated computer.

History-bounded platform validation. The Trusted Computing platform detects and reports platform modifications only within the scope of the current boot cycle. BitLocker uses this feature through the sealing function of the TPM and does not add anything. The system is therefore unable to detect, and react to, any tampering in the past that has not left permanent traces in the system.

Incomplete diagnostic information. If current and reference state are out of sync, it is difficult or even impossible for the user or administrator to determine the exact cause(s). This leaves the user with a difficult choice: to use recovery mechanisms blindly, or not to use them at all. The lack of diagnostic information contributes to the plausible recovery attack. Note that detailed diagnostic information may not be required where a trusted state can be enforced, e.g. by re-installing software from trusted sources.

Lack of external reference. This is another issue that has already been discussed in the literature. BitLocker is capable only of using any current platform state as a reference for future boot cycles. There are no means of verifying that this reference state is trustworthy, opening the road to preemptive modification attacks.

TPM reset. TPM reset attacks imply that the TPM cannot reliably detect platform modifications if the attacker is in physical possession of the computer for sufficiently long time. This may be critical here since theft and other physical-access attacks are the key component of the adversary model.

Recovery mechanisms that circumvent the TPM altogether. Except for TPM reset and preemptive modification, all attacks described above do or may profit from the recovery mechanisms built into BitLocker. These mechanisms pose a particularly attractive target as they yield a key that is independent from the TPM and thus can be used more flexibly. The plausible recovery attack would not even be possible without recovery mechanisms.

Large amount of unprotected disk space. This is a secondary contributing factor to attacks involving purposeful, detectable modification of the platform (plausible recovery; spoofed prompt; tamper and revert). Large amounts of disk space are available for the attacker to install software or data in. This may be difficult to avoid, though.

Almost arbitrary sequence of partial attacks. Another, possibly application-specific, secondary factor is an effect of BitLocker's function and key management. In order to succeed, the attacker needs to accomplish several intermediate goals: copy ciphertext, and get access to various components of the encryption key. Due to the design of BitLocker, the respective attack operations can be executed in arbitrary order, unless one operation permanently changes the platform, the TPM or the knowledge of the victim in such a way that other operations become impossible.

The barn door property. This term has been coined by Whitten and Tygar [12], describing the fact that security often involves operations that are not easily reversed. There is often no *undo*. This is particularly true for confidentiality: once broken, it cannot be restored. There are many ways for the attacker to gain at least some partial success, but there are few situations where the attacker could lose anything achieved before. This may be different in applications with different security objectives and a different adversary model.

Table 1 shows how these causes and factors contribute to the attacks described before. Each column represents an attack, each line a cause or factor. If a factor contributes to an attack—makes it possible, makes it easier, or makes the result more useful for the attacker—the respective cell is marked with an X. The last two lines contain question marks in all cells: the authors do not fully understand the impact of these factors yet.

6 Conclusion

The caution exercised by Microsoft regarding claims about the security of BitLocker seems justified. While BitLocker may indeed protect against opportunistic theft of a computer that is turned off at the time, there are several plausible scenarios for targeted attacks. The trusted computing platform combined with the specific purpose, design and implementation of BitLocker fails to protect against these attacks. Although this does not necessarily imply grave deficiencies on either part, developers and users alike should be aware of these scenarios and the limitations of trusted computing.

Table 1: Attack scenarios and contributing factors.

	Replace and relay	Plausible Recovery	Spoofed prompt	Tamper and revert	Preemptive modification	TPM reset
Passive TPM			X	X		X
No trusted path to user	X		X	X		
No context awareness	X					
Lack of system authentication			X	X		
History-bounded platform validation				X		X
Incomplete diagnostic information		X				
Lack of external reference		X			X	
TPM reset	X		X	X		X
Recovery mechanisms circumventing TPM	X	X	X	X		
Unprotected disk space		X	X	X		
Arbitrary sequence of partial attacks	?	?	?	?	?	?
The barn door property	?	?	?	?	?	?

Our results yield various questions for further research. How easy or difficult is implementing these attacks in practice? Are there issues that we may have overlooked, or are some of the attacks even easier than they appear? Are there countermeasures that can be implemented easily? Where do countermeasures belong, into the application or the trusted computing technology? Is it possible to overcome the fundamental trade-off and implement secure recovery mechanisms? If so, how?

This paper represents a preliminary theoretical analysis. The authors intend to continue this work in three directions. First, the analysis needs to be refined. The final result shall contain an exact description of what an attacker can or cannot achieve in each scenario and what the side conditions are. It would be interesting to determine exact limits to attack optimization. Estimations of the effort required for and risk involved in each step of an attack will also be part of further analysis.

Second, we plan to implement the attacks described here. The purpose is not to create new hacker tools but to gain a deeper understanding of what works and what doesn't, and a better idea of parameters such as effort, time and side effects.

The final step is to devise specific countermeasures. They shall cover two distinct fields: improvements to the trusted computing technology, and recommendations for software implemented on top of it.

References

[1] Chris J. Mitchell (editor). Research workshop on future TPM functionality: Final report. http://www.softeng.ox.ac.uk/etiss/trusted/research/TPM.pdf.

[2] Niels Fergusson. AES-CBC + Elephant diffuser: A disk encryption algorithm for windows vista. Technical report, Microsoft, 2006.

[3] David Grawrock. *The Intel Safer Computing Initiative: Building Blocks for Trusted Computing*. Intel Press, 2006.

[4] Trusted Computing Group. TCG platform reset attack mitigation specification. https://www.trustedcomputinggroup.org/specs/PCClient/TCG_PlatformResetAttackMitigationSpecification_1.00_0340308-1.pdf.

[5] James Hendricks and Leendert van Doorn. Secure bootstrap is not enough: Shoring up the trusted computing base. In *Proceedings of the Eleventh SIGOPS European Workshop, ACM SIGOPS*. ACM Press, 2004.

[6] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold boot attacks on encryption keys. Technical report, Princeton University, 2008.

[7] J. D. Tygar and Bennet Yee. Dyad: A system for using physically secure coprocessors. Technical report, Proceedings of the Joint Harvard-MIT Workshop on Technological Strategies for the Protection of Intellectual Property in the Network Multimedia Environment, 1991.

[8] Michael Becher, Maximillian Dornseif, and Christian N. Klein. Firewire: all your memory are belong to us. Slides, http://md.hudora.de/presentations/#firewire-cansecwest.

[9] Saar Drimer and Steven J. Murdoch. Chip & PIN (EMV) relay attacks. http://www.cl.cam.ac.uk/research/security/banking/relay/.

[10] Saar Drimer and Steven J. Murdoch. Keep your enemies close: Distance bounding against smartcard relay attacks. In *USENIX Security 2007*, 2007.

[11] Evan R. Sparks. Security assessment of trusted platform modules. Technical report, Dartmouth College, 2007.

[12] Alma Whitten and J. D. Tygar. Why Johnny can't encrypt. In *Proceedings of the 8th USENIX Security Symposium*, 1999.

[13] William A. Arbaugh, David J. Farbert, and Jonathan M. Smith. A secure and reliable bootstrap architecture. In *In Proceedings 1997 IEEE Symposium on Security and Privacy*, pages 65–71. IEEE Computer Society, 1997.