A Novel Protection Mechanism for Encryption System

Lin Nan
College of Science
PLA Information Engineering University
Zhengzhou, China, 450002
zhengzhouln@163.com

***Abstract*—Software drive encryption system is difficult to prevent memory attacks, in which, an attacker acquire the physical accesses to the unattended computer, obtains the decryption keys from memory and consequently decrypts the drive. We propose a new method for protecting encryption systems against memory attacks, by converting them to use two tiers of keys, a single Master Key and a set of File or Sector keys. When the computer is unattended, the Master Key and part of the second-tier keys are erased from memory. The method is secure against any type of memory attack, including attackers who gain complete control of the unattended system. Compared to previous methods of protection, which erase keys and shut down the computer, our method allows to keep the computer operational by a combination of cryptographic and operating systems techniques.**

Keywords-Encryption systems, Memory attacks, two tiers of keys, Master Key

I. INTRODUCTION

Drive encryption systems attempt to provide data confidentiality against an attacker with physical access.Memory attacks break drive encryption systems by extracting the decryption keys from RAM on a running or recently turned off computer.This paper proposes a new method of protection against memory attacks on drive encryption systems.The method erases encryption keys to protect data from any attacker.Contrary to simpler methods of protection,where the computer is shut down after erasing keys,we describe cryptographic and operating system methods to keep the computer operational.The administrator may choose a trade-off between functionality and security,selectively making encrypted data available to running programs at the cost of making it vulnerable to memory attacks.We have converted an existing drive encryption system to implement the method.In this paper we describe the theoretical background and practical considerations that determine its design.

Based on the range of existing memory attacks we reviewed in this paper,we have concluded that an attacker can gain full control of the unattended computer.Therefore,we designed our method of protection to withstand an attacker with unlimited access,without depending on the properties of any particular memory attack.

II. MEMORY ATTACKS ON DRIVE ENCRYPTION SYSTEMS

A memory attack is a type of side-channel attack on drive encryption systems in which an attacker with physical access to a computer obtains the decryption key from RAM,and proceeds to decrypt the complete contents of the hard drive.Memory attacks destroy the ability of drive encryption systems to protect the data on a computer even after an attacker gains physical access to it.There are several types of memory attacks, each with a different set of requirements and effects,as we discuss below.

The Cold Boot Attack received wide attention in 2008,with the publication by[1].It is the first memory attack which requires no specialized hardware,whose minimum requirement to obtain memory contents is a reboot.The authors provide downloadable software,which boots the computer into a small program that dumps the contents of the physical memory to external storage.Additional tools then search the dump for encryption keys and key schedules,and help recover keys even in the presence of some bit flip errors. The attack relies on the memory remanence properties of RAM:although power to the RAM chips is temporarily lost,their most recent contents can be recovered.The longer the power loss, the more degradation occurs in the data.The remanence property has been the subject of earlier research in several types of memory, including Static RAM and Dynamic RAM[2].

At lower temperatures,memory contents last longer.Optional improvements to the attack involve cooling the RAM chips by spraying them with compressed air,then physically removing them from the target PC and further cooling them with liquid nitrogen before analyzing them in the attacker's computer.

Other attacks use memory remanence to create a snapshot of the user session active at the time of reboot[3], then resume that session live with modifications giving the attacker full visibility and control.

Memory attacks devised before Cold Boot used peripherals with Direct Memory Access(DMA) capabilities,such as Firewire[4]or USB[5]to dump the target computer's memory.The DMA attacks cause no power interruptions and therefore no data degradation,and in fact do not rely on memory remanence.On the downside,they require connecting a specially programmed external device.Once connected,the device can manipulate physical memory directly.As a result,the attacker gains full visibility and control,including the ability to run his code with system privileges.

In another attack scenario,the attacker induces either hibernation or a crash on the target computer. When entering hibernation or creating a crash dump,the computer makes an orderly snapshot of all virtual memory to a file.The attacker can recover the file and obtain the original memory contents,including drive decryption keys,without

Published by Atlantis Press, Paris, France.

corruption.Making the situation worse,on Microsoft Windows the hibernation and crash dump files are written using a special I/O path that bypasses most drivers,including disk encryption drivers,and is therefore difficult to encrypt.All this makes computers that have hibernation and crash dumps enabled more vulnerable to the attacker.

Together,the scenarios give rise to a more powerful model of the attacker with physical access to the computer.The attacker can obtain an exact snapshot of all memory contents,without degradation.He or she may observe the system and choose the time to make the snapshot,and even scan memory repeatedly.

III. PROTECTING DISK-BASED ENCRYPTION SYSTEMS

Our method of protection can be summarized as follows.It activates in the unattended state when it detects that all authorized users have stopped interacting with the computer.It erases the master encryption key, securing it and any unopened files against the attacker.It then uses a second tier of decryption keys,one key per file,to keep the open files available.To keep the system stable,it blocks file open requests to unopened encrypted files.To secure some of the open files,it selectively erases their keys from the second tier,suspending non-vital processes and taking other preventive measures to ensure that those files not be accessed,and the keys not be needed.Once an authorized user returns and logs in to the computer,the encryption system uses his credentials to restore the master decryption key;it then resumes suspended processes and returns the computer to regular functionality.

So far we have addressed the case of file-based encryption systems,where we require two tiers of keys, with a File Key for every encrypted file.Disk-based encryption systems cannot be protected the same way: as long as a single Master Key encrypts all sectors,it can never be erased.To extend our protection to disk-based encryption system,we isolate the needed properties of file-based systems,and apply them to disk-based systems.Although we have not implemented the protection in a disk-based system,we can define the requirements and propose a realistic design.

We note that all disk encryption modes currently accepted as standard-XTS,XEX,LRW,CMC,and EME[6]directly use the Master Key in all cipher operations.The modes above differ mostly in the tweaking schemes they use to transform the plaintext and ciphertext.The tweak operations involve XOR,multiplication,and sometimes exponentiation modulo a finite field,and their specifics are unimportant for this discussion.To enable erasing the Master Key,we must change the encryption to use two tiers of keys instead of one.

We propose to use a set of Sector Keys as an extra tier below the Master Key.The following key management requirements must be satisfied to protect against memory attacks:

- Each Sector Key SKi must be a function of the sector number and the Master Key.
- Each Sector Key SKi must not reveal the Master Key.
- Each Sector Key SKi must not reveal SKj,for i=j.
- Each Sector Key SKi must be retained in memory for the duration of the unattended mode,for every needed sector.
- These requirements are similar to those of the file-based systems.

The first three requirements are easy to satisfy,if we derive SKi by using a one-way pseudo-random function H of the Master Key and the sector number,i:

$$SK_i = H(MK, i)$$

Suitable options for H are encrypting i with the Master Key,or using the HMAC construction instantiated with a strong hash function to hash i with the Master Key.

The change above only covers key derivation.Similar requirements apply to tweaking schemes:they must not use the Master Key as a direct input,and they must not expose it.The first requirement makes the tweaking schemes usable in the unattended state.The second requirement means that tweaking schemes which use the Master Key must switch to a one-way function of it,or to another,unrelated key.Some of the existing tweaking schemes do not use the Master Key,and the rest can be adapted easily.

By mandating the use of different encryption keys per sector,we incur a computational cost.For ciphers using a key expansion stage,the standard disk encryption systems set up a single instance of the cipher, and have to go through key expansion only once.We now require one H operation and one key expansion operation per sector processed.

There are several complications which disk-based systems must address.First of all,instead of the issue of essential files as discussed in Section 3.6,disk-based systems must address the issue of essential sectors, the sectors containing data from essential files.The disk-based system can map essential files to a list of their sectors by using file system APIs.At first glance,the disk encryption system must collect the Sector Keys for all essential sectors before entering the unattended state,and retain them for the duration of that state.However,disk-based encryption systems typically encrypt the whole disk,including operating system files.The set of essential files for disk-based systems would by default encompass the whole file system.

Making matters worse,write operations may increase the size of a file,spreading it to new sectors.To support the encryption of new sectors without a Master Key,the encryption system must precompute and store a set of Sector Keys for some or all the free sectors.For example,with AES-256 and a typical hard drive geometry,the space cost is 32 bytes for each 512-byte sector of the hard drive.That would take up roughly 6%of the drive,or 64 gigabytes for a terabyte hard-drive(assuming the whole drive consists of essential files and free space).It would be impossible to hold all Sector Keys in memory and they would have to be kept on disk,occupying some of the disk space.

To avoid the complications,we propose to eliminate the need for storing sector keys for free space and non-essential files.

Note that free sectors are equivalent to newly created encrypted files in the file-based setting.By defining a

separate Master Key just for them,it is possible to eliminate the need to keep all of their sector keys in memory.There would be two different Master Keys:one for regular sectors(MK1),and one for free sectors(MK2).MK2 would not be erased when entering the unattended state.For this method to work,the encryption system must distinguish sectors encrypted with MK1 from those encrypted with MK2.It could use a bitmap to do so,consuming some disk space.A better solution is possible if the encryption system can cooperate with the disk sector allocation algorithms.For example,by allocating regular sectors from the beginning of the disk and newly occupied sectors from the end,it can use just two counters to keep track of the areas encrypted with each key.In the unattended state,newly written sectors should initially be encrypted with MK2,and can be moved to the regular area and re-encrypted with MK1 once the user logs back on.

Similarly,a separate Master Key can be used for essential files.The key would not be erased in the unattended state.The reasoning then proceeds as in the last paragraph,and in fact,the same key used for free sectors(MK2)can be used for sectors belonging to essential files.As in the file-based setting,essential files are not protected against memory attacks.

The proposed measures reduce the number of sector keys we need to retain.The encryption system would only need to retain sector keys corresponding to open non-essential files,and only if it does not intend to suspend the applications holding those files open.The reduced space requirements should allow for a practical implementation.

4. Open Issues and Future Work

This section discusses two types of data left unsecured against memory attacks:some system files remaining exposed to the attacker,and applications retaining confidential data in their address space.

There are special files to which the encryption system cannot deny access,most obviously the page file, constantly used by the virtual memory manager to swap memory pages out to disk and back.Those memory pages may contain confidential information from an application's memory space.The virtual memory manager is directly in charge of this file,and may access it at any time.Memory-mapped files are treated the same way.The encryption system must keep these files available,and therefore their contents will be vulnerable to memory attackers.

Finally, confidential data may be kept in application memory space, where memory attacks would expose it. An encryption system could suspend an application and evict its address space to disk, where it can encrypt the data. When returning to normal, the encryption system would restore the original memory contents and then resume the application. We have not yet attempted to implement such a system.

5. Summaries

We have described a technique which protects the keys and data in a drive encryption system against memory attacks.We have defined a combined model of the attacker,and detailed the steps needed to withstand memory attacks in a file-based encryption system,while keeping the computer in a limited but operational state.Finally,we have described the steps needed to implement our protection in disk-encryption systems.

If hardware-based drive encryption systems receive wider acceptance,we expect the impact of memory attacks to decline.Until then,we propose a solution which provides protection against memory attacks at no hardware cost.

REFERENCES

[1] 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.Commun.ACM,52(5):91–98,May 2009.

[2] Sergei P. Skorobogatov. Data remanence in flash memory devices. In CHES, volume 3659 of Lecture Notes in Computer Science, pages 339–353.Springer,2005.

[3] Ellick M.Chan, Jeffrey C. Carlyle, Francis M. David, Reza Farivar, and Roy H. Campbell. Bootjacker: compromising computers using forced restarts. In CCS'08: Proceedings of the 15th ACM conference on Computer and communications security, pages 555–564,New York, NY,USA,2008.ACM.

[4] Adam Boileau. Hit by a bus: Physical access attacks with firewire. Presentation at RUXCON 2006, Australia,2006.

[5] Darrin Barral and David Dewey. Plug and root: The USB key to the kingdom.http:// frozencache.blogspot.com/,2005.

[6] Morris Dworkin. Recommendation for block cipher modes of operation: The XTS-AES mode for confidentiality on block-oriented storage devices. NIST Special Publication 800-38E,2009.