第29卷 第20期 计 算 机 工 程 2003年11月
Vol.29 № 20 Computer Engineering November 2003
·安全技术· 文章编号：1000—3428(2003)20—0111—03 文献标识码：A 中图分类号：TP309.7

Linux加密文件系统：Crypt-FS

丁 成[1]，孙玉芳[2]
(1.中国科学技术大学研究生院计算机学部，北京 100080；2.中科院软件所)

摘 要：随着便携式计算设备的普及，保存数据的存储设备失窃或遗失的可能性增大。在这种情况下，能够保证敏感数据不被泄露的唯一方案是使用数据加密技术。相对于其它加密方式，使用加密文件系统对用户透明、可靠，有着不容忽视的优势。文章在Linux 2.4操作系统上设计和实现了一个加密文件系统——Crypt-FS，并对其进行了介绍。

关键词：Linux；加密文件系统；安全操作系统；Crypt-FS

Crypt-FS: A Cryptographic File System Designed for Linux

DING Cheng[1]，SUN Yufang[2]
(1.University of Science and Technology of China, Beijing 100080；2.Institute of Software,Chinese Academy of Sciences)

【**Abstract**】Portable computers become more and more popular. The storage device which stores sensitive data has the more chances to be lost or stolen. In this situation, the only way which can protect the sensitive data is using data encryption. Cryptographic file system is the best candidate in this scopes. Its lucider and reliable for the user. And it has more advantage which isn't to be ignored than other encrypt ways. This paper describes the design and implementation of the cryptographic file system Crypt-FS on Linux 2.4.

【**Key words**】Linux；Cryptographic file system；Security operation system；Crypt-FS

加密文件系统主要解决的是物理安全问题。通常依靠操作系统提供的安全机制来保护数据的安全。以普通的个人计算机来说，它们通常都允许通过启动软盘引导系统。这样对于在物理上能够接触到涉密计算机的入侵者来说，他可以简单启动另一个不同的操作系统，该系统完全由入侵者来控制，此时他就可以完全避开正常的权限检查。入侵者甚至有可能盗走保存有机密数据的硬盘，或者整台计算机。由于笔记本电脑在现今社会应用广泛，而且经常被随身携带，因此出现遗失和被窃的可能性自然大大增加。相对于通过计算机网络系统入侵，物理入侵带来的安全威胁相对要小一些，但是只要数据足够重要，这种安全威胁就绝对不能忽略。威胁的根源是数据在存储介质中以明文方式保存，而应对这种威胁的唯一方法就是使用数据加密技术。

1 加密文件系统介绍

1.1 基本原理

(1)文件系统

操作系统中负责管理和存取文件信息的软件机构称为文件管理系统,简称为文件系统。文件系统为用户提供了存取简便、格式统一、安全可靠的管理各种信息(文件)的方法。

(2)加密技术

采用密码技术将信息隐蔽起来，使信息即使被窃取或截获，窃取者也不能了解信息的内容，从而保证了信息的安全。

(3)加密文件系统

加密文件系统扩展了普通文件系统的功能，它以一种对用户透明的方式为用户提供了一个将数据以加密方式存放的功能。

1.2 存储系统的不同加密方式

存储系统的加密可以分为3大类。

(1)文件加密方式

最基本的文件加密方式是使用某种文件加密工具，它不需要操作系统支持。例如在E-mail通信时常用的PGP工具。用户通过自己选择的密码对文件进行加密。用户在访问加密过的文件的时候，需要使用工具对其先进行解密，使用完毕后再进行加密。这种繁琐的方式使得它在实际应用中缺乏足够的吸引力。而且这种方式由于需要用户参与较多，因此比较容易出错。如果用户遗忘了密码，文件内容将无法恢复，如果用户使用容易猜测的密码(用户通常倾向于使用比较容易记忆的口令，这往往是容易猜测的口令)，就会增加非法访问者解密的可能性。同时处于使用状态的明文数据文件完全没有保护，入侵系统的非法用户也可以使用替换加密工具，记录用户键盘输入等方法得到数据的访问权，所以只适合对安全性要求不高的场合使用。

(2)存储介质加密方式

存储介质加密方式通常在设备驱动程序层实现。由驱动程序透明地进行加解密工作。它们通常使用创建一个容器文件，通过某种机制，例如loop设备，在此容器文件中创建一个文件系统。整个容器文件是加密的，作为一个虚拟分区被安装和使用。使用这种方式实现的加密系统有BestCrypt，PGPDisk和Linux cryptoloop.

(3)加密文件系统

加密文件系统是在文件系统基础上实现加密功能。它和使用存储介质加密方式一样，能够提供对用户透明的文件加密功能。相对于使用存储介质加密方式，它的主要特点是可以支持文件粒度的加密。也就是说，用户可以选择对哪些文件加密。由于是直接在物理设备而不是在容器文件内创建文件系统，再加上不用对整个存储卷加密，加密文件系统就能够提供更好的性能。

1.3 加密文件系统的分类

加密文件系统可以分为两类，两者的实现方式和目标都有比较大的区别。一类是本地加密文件系统，它的目标是应对存储介质失窃的威胁，安全模型把加密文件系统所在的操作系统视为可信的。另一类是面向网络存储服务的，通常是基于NFS客户和服务器模型，可以称为加密网络文件系统。在加密网络文件系统中，数据以密文的方式保存在加密网络文件系统中，用户通过客户机服务进程与网络文件服务器交互，网络文件服务器负责将用户请求的密文传递到客户机服务进程，由客户机服务进程进行解密再交给应用程序。在这

作者简介：丁 成(1974－)，男，硕士生，主研方向为Linux操作系统；孙玉芳，博导
收稿日期：2003-08-21 E-mail：xld@ht.rol.cn.net

个模型中，只要求客户机操作系统是可信的,而网络服务器由于不接触明文数据，不要求其可信。Crypt-FS属于前一类。

2 相关的工作

这里列举一些重要的加密文件系统，它们有的属于实验性质的文件系统，有的则属于商业软件，在这里不作详细介绍，仅供参考。

· CFS (Cryptograhpic File System)：AT&T的Matt Blaze 开发，是比较早的加密文件系统；

· TCFS：意大利Salerno大学在CFS基础上开发的加密文件系统；

· Cryptfs：由Erez Zadok开发的加密文件系统，基于他本人提出的Stackable File System实现的加密文件系统(注:本文所论述的Crypt-FS名称和Cryptfs很相象，但是是完全不同的两个系统)；

· ReiserFS version 4：Linux上的著名日志文件系统，从v4开始支持加密功能；

· EFS：集成在微软公司的Window2000操作系统中的加密文件系统。

3 Crypt-FS加密文件系统的密码体系

Crypt-FS加密文件系统中存在3种密钥，分别是FEK，UEK和SEK.

(1) FEK

正如大多数现实中的安全体系一样，使用公开密钥算法和对称密钥算法的组合设计出加密文件系统的安全体系。文件数据通过一个快速而安全的对称加密算法进行加密，文件加密密钥简称为FEK (File Encryption Key), 是一个通过随机数产生器按某种的方式产生的随机数，它应该满足对称加密算法对密钥的长度和其它限制。

FEK和被加密文件是1对1对应关系，也就是说每次对一个文件加密时，就会产生一个随机的FEK，加密文件系统使用产生的FEK对文件数据进行加密，一个文件的FEK一旦产生通常就不再改变，FEK本身以加密的形式被保存在加密文件系统中。

使用1对1的FEK好处是加密文件系统中，不存在多个以相同对称密钥加密的文件。这样就增加了安全性，同时这也是在加密文件系统中支持加密文件共享的基础。考虑某个文件需要被多个用户共享时，每个用户都需要拥有此文件的密钥，由于密钥只用于这一个文件，因此不会有安全性的问题。

(2) UEK

每个用户拥有的自己的公开密钥算法的密码对， 称为UEK (User Encryption Key)，其中的公钥部分被用来对用户的FEK加密，在加密文件系统中FEK只以加密的形式保存在存储设备中，UEK的公钥部分则以明文方式存放。UEK的私钥被用来解密加密过的FEK。它是整个加密文件系统的关键信息，需要被安全地存放。一种合理的方式是存放在SmartCard中或其它物理上安全的地方。目前在Crypt-FS加密文件系统中，使用用户设置的口令通过对称加密算法对UEK的私钥进行加密并保存在加密文件系统中。此方案对安全性有一定的损害，特别是存在入侵者用暴力尝试的方式来攻击用户的私钥的可能性。以后我们会考虑使用其他更安全的方式解决这个问题。

(3) SEK

FEK和UEK都是重要信息，丢失后就会造成数据丢失，所以需要在系统中保存它们的备份。这同样需要通过公钥算法来保护。对加密文件系统来说，存在着一个系统加密密钥SEK(System Encryption Key)。 所有的FEK和UEK都使用SEK的公钥进行加密后保存在加密文件系统中，在需要恢复丢失的FEK或UEK的时候，系统管理员只要使用SEK的私钥就可以成功地恢复出FEK或UEK。由于丢失FEK或UEK的情况不会经常发生，因此SEK的私钥不在加密文件系统中保存，只在恢复时才导入到系统，恢复完就立即卸出。

图1描述了数据加密的过程，由 UEK私钥解出FEK，由FEK对数据块加密。解密的过程和加密的过程相似。在任何时候，用户或应用程序注意到的都是解密后的明文数据，他们不会观察到数据已经被加密了，而在存储介质中永远都不会有明文数据存在。其中加解密都是针对数据块进行的。无论是加密还是解密的过程，只对用户进行读写操作的数据所在的数据块(页对齐的)进行加解密的操作。

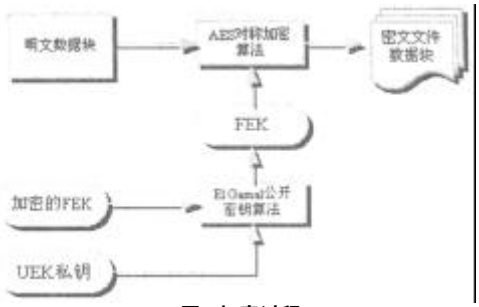

图1 加密过程

图2描述了加密文件系统的完整视图，其中标注着1的步骤是文件数据块加密/解密的步骤，发生在用户数据从磁盘读出和写回到磁盘的时候。步骤2是文件加密密钥产生的过程，发生在对一个未加密的文件设置加密标志时，这个过程对用户来说是透明的。步骤3发生在用户打开一个加密文件的时候，系统使用UEK的私钥将加密的FEK解密。步骤4发生在用户登录加密文件系统的时候。用户的加密文件系统口令和用户账户的口令是不同的，主要是考虑到这样可以使用强度更高的加密算法，同时用户也应该使用长度更长的口令作为他的加密文件系统的口令。步骤5发生在用户的FEK或UEK丢失的时候，这时用户应该请求系统管理员协助使用备份的FEK或UEK恢复用户丢失的FEK或UEK。

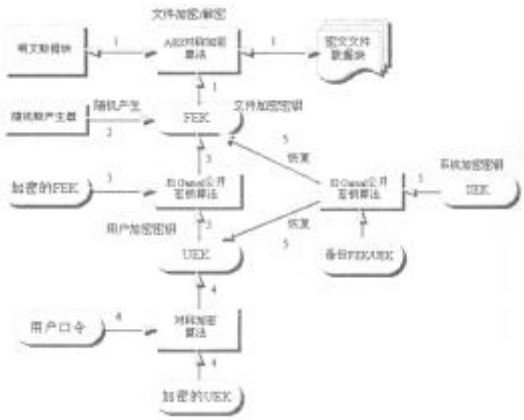

图2 加密系统的完整视图

4 加密文件系统的实现

在设计加密文件系统的时候，面临两种选择：一是在现有的文件系统的基础上对文件系统的存储结构加以调整以容纳加密信息，并修改文件系统代码来实现加密文件系统。StegFS和Reiserfs V4都以这种方式实现；另一种是在Linux操作系统的虚拟文件系统中实现加密文件功能。我们选择了后者。这样我们的实现独立于具体的文件系统，用户可以自由地选择满足需求的文件系统，同时又拥有了加密文件系统的功能。

4.1 加密属性的存放

我们采用在目标文件系统中创建加密文件系统元文件来实现文件加密属性的保存。加密文件系统元文件是目标文件系统根目录中的普通文件，但是通过安全操作系统的强制访问控制机制加以保护(Crypt-FS是安全操作系统的一个子模块)，所有用户都没有此元文件的访问权限，所以它是足够安全的。元文件的内容在系统启动前将由操作系统核心读取并构造相应的内存中的数据结构，在系统空闲的时候由一个核心进程将被修改的信息保存回加密文件系统元文件中。

4.2 加密的位置

Crypt-FS加密文件系统实现的难点在于找到Linux核心代码中合适的地点进行加解密的工作。这个地点有2个条件必须得到满足：(1)代码应该处理对齐的磁盘块数据；(2)应该在Page Cache层之下，以便充分利用Linux的缓冲机制提高加密文件系统的性能。 磁盘读写速度相对于计算机系统的处理能力实在是太慢了，Page Cache极大地增加了Linux操作系统的文件系统的效率。如果在Linux Page Cache 层之上进行加解密的话，意味着对于加密的文件，Page Cache中存在的将是密文，每次核心需要Page Cache中内容的话就要进行加解密的工作。而根据我们屏蔽了缓冲机制后的测试显示(使用同步方式安装文件系统，进行写操作) 对加密文件的操作由于涉及到费时的加密算法，对CPU为PII 266的服务器，CPU负荷相对于非加密文件的操作将提高5倍左右。可见开销还是相当大的。所以合理的方式是在数据块刚刚从磁盘中读取时即解密，对数据实际需要写入磁盘前才进行加密。这样Linux缓冲机制中缓存中存在的将是明文。只要数据在缓冲区内，实际性能将不会有明显的损失。

Crypt-FS的主要的加解密操作位置如下：

(1)在generic_make_request( )函数中进行加密。Linux操作系统中generic_make_request( )函数完成将实际的读写请求传递到下层的设备驱动的任务。

(2)在do_generic_file_read( ) 中将实际的数据解密。

4.3 文件名加密

虽然文件名不如文件数据敏感，但考虑到用户一般会使用有意义的命名文件，这样就会给入侵者提供某些信息，所以加密文件系统通常都考虑使用密文的方式保存加密文件的文件名。由于加密后的文件名通常包含不适合文件名要求的字符例如'/0'和其它不可读的字符，因此加密后要作一个转换，将加密后的文件名利用类似于BASE64的编码方式转化为符合文件系统对文件名规范的要求。这样有两个问题比较麻烦，一是减少了文件名可用长度，如果用户对足够长的文件名的文件加密，结果将超出文件系统对文件名长度的限制。第2个问题是加密后的文件名可能会和其它文件名重名。由于文件名加密算法不改变文件名的长度，而加密后对文件名的转换过程对文件名长度的变化是确定的。因此通过在加密工具中对原文件名长度作检测，避免了第一个问题，如果发现文件名过长，就拒绝加密请求。关于重名的问题还没有好的方法解决。

5 性能测试

测试系统硬件环境为PII 266, 192 MB内存，WD IDE硬盘，基础文件系统为ext3，在其上使用Crypt-FS。

表1列出了3次测试的结果，第1和第3列是完成任务总的时间，第2和第4列是用在核心态的时间。从表中可以看到，由于是同步方式，对文件的写操作立即被写回到磁盘，所以每次测试的数值相差不大。可以看到处理加密文件核心态的使用时间是处理非加密文件的5.18倍，开销还是相当大的。由于测试机CPU性能比较弱，导致对加密文件任务处理时间是非加密文件任务处理时间的1.34倍。

表1 同步方式写文件 (以mount o sync方式安装文件系统)

	1024*4kB 未加密文件(s)		1024*4kB 加密文件(s)	
第1次	3.967	0.300	6.133	1.980
第2次	3.909	0.270	5.051	1.380
第3次	3.913	0.310	4.869	1.200

表2的格式和表1一样。从表中可以看到由于第1次操作时，数据完全不在缓存里，因此对加密文件的读写，在核心态的运行时间分别为0.910 s, 0.930 s , 为非加密文件的6.5倍和15倍，可见CPU负载仍然很大。而后续的第2次、第3次操作由于数据已经在缓存中，所使用时间，对加密文件和非加密文件几乎完全相同，可见我们的实现很好地利用了Linux高效的缓存机制。

表2 非同步方式读写文件

写操作	1024*4kB 未加密文件(s)		1024*4kB 加密文件(s)	
第1次	0.153	0.140	1.023	0.910
第2次	0.145	0.140	0.154	0.140
第3次	0.142	0.120	0.145	0.150
读操作	1024*4kB 未加密文件(s)		1024*4kB 加密文件(s)	
第1次	0.389	0.060	1.014	0.930
第2次	0.038	0.030	0.039	0.020
第3次	0.037	0.030	0.037	0.030

参考文献

1 Blaze M. A Cryptographic File System for Unix. Proceedings of the First ACM Conference on Computer and Communications Security, 1993

2 Cattaneo G. The Design and Implementation of a Transparent Cryptographic File System.USENLX Annual Technical Conference, 2001

3 冯登国.计算机通信网络安全北京: 清华大学出版社,2001

4 毛德操.Linux核心源代码情景分析.浙江:浙江大学出版社出版,2002

☆☆

(上接第56页)

参考文献

1 蔡自兴,徐光佑人工智能及其应用[M].北京:清华大学出版社,1996

2 Lee S C,Lee E T.Fuzzy Sets and Neural Networks[J].Journal of Cybernetics,2000,4:83-103

3 Morita A. Fuzzy Knowledge Model of Neural Network Type [J]. Fuzzy Set sand Systems,2000,8:253-283

4 Narendra K S,Parthasarathy K.Identification and Control of Dynamic Systems Using Neural Networks[J].IEEE Trans. on Neural Network, 1999,1:4-28