**Linux完整性子系统概述**

IMA（完整性度量体系结构）是在Linux 2.6.30中引入的，它是整个Linux完整性子系统的一部分，由许多相关组件组成，包括：

在内核中：

IMA

IMA评估

IMA评估目录扩展

IMA评估签名扩展

EVM

可信和加密密钥

应用/支持：

Trousers/实用程序

OpenPTS

本文档概述了这些组件、它们的目标、功能、设计、状态和优点。由于这些组件处于不同的完成状态，本文档旨在帮助查看所有组件集成后的最终总体完整性体系结构。

***目标：***

构成Linux系统的程序和数据有三个主要的安全目标：完整性、真实性和机密性。这些目标相似且紧密耦合，因此清晰地定义它们是很有帮助的。

**完整性**-完整、完整、不受影响

完整性是最基本的安全属性；没有完整性，系统就不可能具有真实性或机密性，因为真实性和机密性的机制本身可能会受到损害。

对于一个文件，完整性通常被认为是“不变”的属性，也就是说，一旦安装，文件不会被恶意或不希望的修改所更改。在加密技术上，可以使用哈希来检测文件内容是否已更改，以及tripwire等应用程序(http://sourceforge.net/projects/tripwire/)还有助手(http://sourceforge.net/projects/aide/)通常由管理员用来将文件哈希值与预期值或“良好”值的数据库进行比较，寻找意想不到的变化。

在极端情况下，完整性不仅意味着不变，而且意味着“不可更改”或“不可变”，因此任何东西都不可能修改文件。BSD操作系统具有不可变文件的概念，它只能在单用户管理模式下更改。Android将其所有系统文件保存在一个分区中，该分区在正常使用期间是只读的，并且只能在引导期间通过恢复程序进行修改。这些不可变系统的问题是，虽然它们提供了更大的完整性，但它们使系统组件的正常安装或修补变得困难。

IMA遵循可信计算组（trustedcomputinggroup，TCG）的开放完整性标准，这些标准结合了这些完整性技术（散列和不变性）来创建一个新的完整性解决方案。与BSD和Android方法不同，BSD和Android方法使文件本身不可变，TCG方法是通过将它们存储在诸如可信平台模块（TPM）、移动可信模块（MTM）或类似设备中的硬件来实现文件的不可变。IMA在所有文件被访问时维护一个哈希列表，如果支持的硬件安全组件（如TPM）可用，它对软件攻击的不变性将被用来“证明”或锚定具有硬件签名的度量列表，以便远程验证度量列表。通过这种方式，管理员可以集中监控网络系统的完整性，并可以判断这些系统是否被更改或受损。

**真实性**-具有公认的真理和正确性的权威（韦伯斯特）

真实性是诚信概念的延伸。这里的想法不仅是要告诉文件具有完整性，因此是不变的，而且该文件的出处是已知的。例如，我们不仅知道一个文件没有被恶意修改，而且还知道该文件是由供应商（如RedHat）提供的原始文件。

简单的哈希不足以保证真实性。取而代之的是基于某种秘密的签名，这样攻击者就不能伪造出处的证据。通常，公钥签名（如RSA）用于验证真实性，但也可以使用对称密钥哈希，如HMAC。

公钥签名的一个优点是中央机构是唯一必须保护私钥机密性的实体，而所有其他系统只需要保护用于验证的公钥的完整性。公钥签名的问题是，它们只适用于从不更改的文件，因为如果文件发生更改，本地系统将无法重新签名。由于系统上的许多安全关键文件都会发生更改，因此这些更改的文件实际上只能通过基于本地对称密钥的签名来保护。必须小心保护用于身份验证的本地对称密钥，因为攻击者可以使用它在恶意更改的文件上伪造本地签名。

在Linux中，一个新的IMA评估扩展和一个新的扩展验证模块（EVM）是使用文件数据和元数据的对称密钥为允许更改的文件提供真实性的组件，因此这些文件必须在本地签名。对于不希望更改的文件，IMA评估签名扩展将存储授权的RSA公钥签名。带有RSA签名的文件将不允许更改，除非被删除，然后只能由特权进程（一个已被分配了拟议的新的“完整性”功能的进程）更改

由于EVM使用本地对称密钥来保证真实性，因此必须小心地保护该密钥。新的“可信”和“加密”内核密钥类型可用于保护EVM密钥，方法是在TPM中进行密封，并仅在指定的完整性度量值下释放。

**保密**-秘密的状态（韦伯斯特）

最后想要的特征是保密性。一种常见的误解是，加密本身可以保护文件的完整性，因此，人们只需要对文件系统进行加密，以实现完整性和机密性。

不幸的是，这并不是那么简单-加密不是完整性。首先，作为一个简单的例子，考虑一个流密码，如RC4，它在很长一段时间内被用作所有无线和因特网（https）通信的标准加密。RC4本身不提供任何完整性保护，攻击者在加密流中“摆弄”任何所需的位是微不足道的。即使使用块密码，使用当前的标准链接方法，加密的块也可以剪切、粘贴并在文件和会话之间重放，成功率很高。第二，加密系统文件的性能会受到很大的影响，在这种情况下，只需要完整性，而不是机密性。

在IPSEC中，经过深思熟虑的决定允许不加密的完整性（AH-身份验证标头），但不允许没有完整性的加密。

归根结底，加密本身并不能保证完整性，而您确实希望两者兼而有之。

保密中最困难的问题之一是密钥管理。使用密钥系统进行强加密是没有好处的，这使得攻击者很容易窃取密钥。

可信密钥组件做了两件事来帮助Linux上的安全密钥管理。首先，它提供了一个内核密钥环服务，在该服务中，对称加密密钥永远不会以明文形式显示给用户空间。密钥在内核中创建，并由诸如TPM之类的硬件设备密封，用户空间只能看到密封的blob。恶意或被破坏的应用程序不能窃取可信密钥，因为只有内核才能看到未密封的blob。其次，可信密钥可以将密钥解封与完整性度量联系起来，这样密钥就不会在离线攻击中被窃取，例如从CD或USB引导未锁定的Linux映像。由于测量结果不同，TPM芯片将拒绝解封密钥，即使对于内核也是如此。

通过这种方式，Linux将机密性和完整性联系在一起。系统的完整性是解密机密文件的先决条件。

***威胁模型：***

针对文件完整性、真实性和机密性有许多不同类型的攻击，根据需要防御的威胁，在安全性与成本和性能之间有着重要的权衡。

# 远程攻击

最常见的攻击是远程软件攻击，攻击者试图诱骗用户运行攻击者的恶意代码（特洛伊木马），或发送恶意数据，试图利用系统软件中的漏洞进行攻击（注入、溢出）。

# 局部攻击

本地攻击假定攻击者具有对系统的物理访问权限，例如与恶意内部人员（操作员）或系统被盗。本地攻击可以是基于软件的，例如离线攻击，也可以是基于硬件的，例如使用简单的JTAG内存探测，

甚至是极为复杂（昂贵）的攻击，其中芯片被拆开读取敏感数据。

最简单和最常见的本地攻击是脱机攻击，其中从CD或USB驱动器启动备用操作系统，攻击者使用此操作系统修改目标系统。离线攻击通常只会尝试破解现有密码，或插入已知密码，这样攻击者就可以简单地登录。在更复杂的离线攻击中，可以插入恶意代码并保留在原位，以便稍后捕获敏感数据，如银行密码。

***具体目标：***

虽然防止所有这些攻击是理想的，但至少要检测到这种攻击何时成功是至关重要的。保护Linux系统完整性的最好方法是使用强制访问控制（MAC），比如SELinux或Smack来限制不可信的代码和数据。即使使用MAC，一些远程和大多数本地攻击（尤其是离线攻击）仍然是可能的。

完整性子系统的目标是针对所有远程和本地攻击检测对文件的任何恶意更改，除非本地攻击者具有攻击硬件（特别是TPM芯片）的访问权限、时间和资源。恶意更改可能包括修改数据或元数据（例如重命名）、重放旧文件，以及删除文件。

***实施***

# IMA-基本测量和认证

IMA从2.6.30开始就包含在Linux内核中。

IMA是一个开源的可信计算组件。IMA维护一个运行时度量列表，如果锚定在硬件（例如TPM）中，则在该列表上维护聚合完整性值。在TPM中锚定聚合完整性值的好处是，度量列表不会因任何软件攻击而受损，且不会被检测到。即使恶意文件被访问，在访问该文件之前，它的度量将提交给TPM，恶意代码无法删除此度量。如果恶意软件破坏了认证软件，它就不能隐藏它的存在，因为它不能在伪造的测量列表上伪造签名。相反，如果没有硬件锚定，恶意代码很容易创建一个无法检测的假列表。因此，在受信任的引导系统上，可以使用IMA来证明系统的运行时完整性。需要注意的是，IMA度量和证明并不试图保护系统的完整性，它的目标至少是检测是否发生了这种损害，以便能够及时修复。

使用内核命令行参数IMA\_tcb=1启用IMA测量

这将启动一个默认策略，该策略测量所有已执行或mmap&apos;ed执行的普通文件，以及由具有根uid的进程读取的所有普通文件。

可以加载修改后的IMA策略，该策略可以基于LSM标签。例如，如果运行SELinux，一个理想的规则是dont\_measure obj\_type=var\_log\_t，这样就不会测量日志文件。

IMA测量列表可以通过IMA securityfs文件读取，通常安装在：

/sys/kernel/security/ima/ascii\_runtime\_度量，度量列表项如下所示：

PCR模板哈希\文件数据哈希文件名提示

10 7971593a7ad22a7cce5b234e4bc5d71b04696af4 ima\b5a166c10d153b7cc3e5b4f1eab1f71672b7c524启动集总成

散列被扩展成一个PCR（默认为PCR-10）。可信计算小组定义了一个标准的完整性证明XML格式PTS，其中包括测量列表和PCR-10中值上的TPM签名。远程系统可以验证列表是否生成PCR-10中的值，以及TPM是否已对该值进行签名。无论系统如何受到危害，恶意软件都无法伪造有效的度量列表和相应的TPM签名，因为恶意软件无法访问TPM的私有签名密钥，TPM中任何PCR值的签名都必须包含对恶意文件之前所取的任何恶意文件的测量已访问。恶意代码不能“收回”自己的度量，也不能伪造“干净”的度量签名。

# IMA评估-本地文件完整性检查

IMA评估已经发布了几次到LSM/LKML邮件列表中，但还没有向上传输，等待一些额外的扩展，包括稍后描述的签名和目录扩展。

IMA evaluation是IMA的一个新扩展，IMA将已评估文件的“good”哈希存储在安全.ima扩展属性，并验证文件的当前度量值是否与此“良好”值匹配。如果值不匹配，则拒绝访问该文件。默认情况下，安全.ima包含哈希，而不是键控签名。这很方便，尤其是在更改文件时，但由于哈希很容易伪造，因此无法提供强大的完整性和真实性保护，以抵御脱机和在线攻击。

# IMA评估签名扩展-数字签名文件

数字签名文件的IMA评估签名扩展旨在通过基于权限的数字签名（RSA）来增强标准IMA评估，以提高对不变文件的真实性和不变性的保证。它还没有发布。（第一个职位预计在2011年初发布。）

在IMA评估签名扩展中安全.ima扩展属性可以是RSA签名，通常由供应商提供。当设置了这个属性（需要完整性功能）时，文件被认为是不可变的，并且在访问之前验证签名。无法修改已签名的文件，但可以删除和替换这些文件以允许更新。这些签名很难被攻击者伪造，因为它们需要拥有权威机构的私钥。（引用公钥需要进行完整性保护，以防脱机攻击。）

# IMA评估目录扩展-本地目录完整性

IMA评估目录扩展尚未发布。它旨在增强文件评估功能，包括防止文件名更改攻击和签名文件重放攻击的保护，其中在脱机攻击中重放已签名的旧文件。

基本IMA评估使用RSA签名（用于不变的文件）或哈希（用于更改文件）来验证文件内容的真实性和完整性。仍然存在通过更改文件元数据（不受签名或哈希保护）而误用有效文件内容的攻击。敏感元数据的示例包括文件名、所有者和模式（访问控制）位。例如，只要将“rm”重命名为“ls”，攻击者就可以破坏系统。文件元数据既存在于目录（文件名）中，也存在于inode（所有者、组、模式、LSM标签）中。为了保护文件的元数据，目录扩展添加了目录内容的哈希（文件名、inode编号和inode评估值）。

# 扩展验证模块-可信元数据

EVM已经发布到LSM/LKML邮件列表中。它的目标是保护敏感的inode元数据免受脱机攻击。

IMA评估目录扩展名保护文件名元数据。第二个包含敏感元数据的区域在inode中，它包含owner、group、mode以及敏感的扩展属性。EVM保护这些安全扩展属性免受脱机攻击。EVM跨一组安全扩展属性维护HMAC-sha1，将HMAC存储为扩展属性&apos;安全.evm&apos;. 为了验证扩展属性的完整性，EVM导出EVM\_verifyxattr（）。EVM为所有安全扩展属性提供保护，包括

\* 安全.ima（文件的IMA哈希或签名）\*安全.selinux（文件上的selinux标签/上下文）

\* 安全.SMACK64（文件上有斯帕克的标签）

\* 安全性能（可执行文件上的功能标签）

由于用于HMAC元数据的EVM密钥非常敏感，它应该是可信密钥，或者是受信任密钥下的加密密钥，并且可信密钥应该完整性锁定到当前度量值。

**可信和加密密钥：**

受信任和加密的密钥修补程序目前处于securityŠnext中，排队等待2.6.38中的潜在包含，以便为敏感用途（如EVM主密钥）提供可信密钥。可信密钥和加密密钥是添加到现有内核密钥环服务中的两种新密钥类型。

这两种新类型都是可变长度的对称密钥，在这两种情况下，所有密钥都是在内核中创建的，用户空间只查看、存储和加载加密的blob。可信密钥需要可信平台模块（TPM）芯片的可用性以提高安全性，而加密密钥可用于任何系统。为了方便起见，所有用户级blob都以十六进制ascii格式显示和加载，并且经过完整性验证。

可信密钥使用TPM来生成和密封密钥。密钥在TPM中的2048位RSA密钥下密封，并且可以选择密封到指定的PCR（完整性测量）值，并且只有在PCR和blob完整性验证匹配的情况下才由TPM解封。加载的可信密钥可以用新的（未来的）PCR值更新，因此密钥很容易迁移到新的PCR值，例如当内核和initramfs更新时。同一个密钥可以在不同的PCR值下保存多个blob，因此很容易支持多个引导。

默认情况下，可信密钥被密封在SRK下，SRK具有默认的授权值（20个零）。这可以在takeownership时间使用裤管工具进行设置：

tpm公司所有权-u-z

可信密钥用法：

keyctl add trusted name“new keylen[选项]”ring keyctl add trusted name“加载十六进制blob[pcrlock=pcrnum]”环

keyctl update key“更新[选项]”keyctl打印keyid

选项：keyhandle=ascii hex-value of sealing key default 0x40000000（SRK）keyauth=ascii hex auth for sealing key default 0x00。。。

                                （40个ascii零）

blobauth=ascii hex auth for sealed data default 0x00。。。

                                （40个ascii零）

默认值为0xBAUTH十六进制数据。。。

                                （40个ascii零）

pcrinfo=PCR\_INFO或PCR\_INFO\_LONG的ascii hex（无默认值）pcrlock=要扩展到“lock”blob的PCR编号

migratable=0 | 1表示允许重新封装到新的PCR值，默认值为1（允许重新封装）

“keyctl print”返回密封密钥的ascii十六进制副本，这是标准的

TPM存储的数据格式。新密钥的密钥长度总是以字节为单位。可信密钥可以是32-128字节（256-1024位），上限是在2048位SRK（RSA）密钥长度范围内，以及所有必要的结构/填充。

加密密钥不依赖于TPM，而且速度更快，因为它们使用AES进行加密或解密。新密钥由内核生成的随机数创建，并使用指定的“主”密钥加密或解密。“主”密钥可以是可信密钥，也可以是用户密钥类型。加密密钥的主要缺点是，如果它们不是以可信密钥为根的，那么它们的安全性仅与加密密钥的用户密钥相同。因此，应该以尽可能安全的方式加载主用户密钥，最好是在启动时就加载。

加密密钥用法：

keyctl add encrypted name“新密钥-类型：主密钥名keylen“ring keyctl add encrypted name”加载hex\_blob“ring keyctl update keyid”更新密钥-类型：主密钥名称“其中&apos;key type&apos;是&apos;trusted&apos;或&apos;user&apos;。

可信和加密密钥用法示例：

创建并保存一个名为“kmk”的可信密钥，长度为32字节：

$keyctl add trusted kmk“新32”@u

    440502848

$keyctl显示

会话密钥环

-3——alswrv 500 500钥匙圈：

97833714--alswrv 500-1个密钥环：uid.500

440502848--alswrv 500 500\\可信：kmk

$keyctl打印440502848

0101000000000000000000001005D01B7E3F4A6BE5709930F3B70A743CBB42E0CC95E18E915

F60 DA455BF1144ADEF92BF52F6105F6105FF6105FF610FF610FFF610F610FF610FF610F610FF610FF610FF610FF610F610F610F610F29F12 F6 F610F6 F610F6 F6 F610FF12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 D4 D4 D4 D4 A40BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB是一次在该段中，该段以该段为该段的一次以该段为97F450B9EF9C22C6D5D型DD379F0FACD120281DFA3C70BA21A3FA6FC2471DC6D13ECF8298B946F65345FAA5EF0 F1F8FFF03AD0ACB083725535636ADDB08D73DEDB9832DA198081E5DEA84BFAF0409C22B e4a8aea2b607ec96931e6f4d4fe563ba$keyctl管道440502848>kmk.blob公司从保存的blob加载受信任的密钥：

$keyctl add trusted kmk“加载`cat”kmk.blob公司`“@u

    268728824

$keyctl打印268728824

0101000000000000000000001005D01B7E3F4A6BE5709930F3B70A743CBB42E0CC95E18E915

F60 DA455BF1144ADEF92BF52F6105F6105FF6105FF610FF610FFF610F610FF610FF610F610FF610FF610FF610FF610F610F610F610F29F12 F6 F610F6 F610F6 F6 F610FF12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 F12 D4 D4 D4 D4 A40BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB是一次在该段中，该段以该段为该段的一次以该段为97F450B9EF9C22C6D5D型DD379F0FACDCD020281DFA3C70BA21A3FA6FC2471DC6D13ECF8298B946F65345FAA5EF0 F1F8FFF03AD0ACB083725535636ADDB08D73DEDB9832DA198081E5DEA84BFAF0409C22B e4a8aea2b607ec96931e6f4d4fe563ba在新的PCR值下重新密封可信密钥：

$keyctl update 268728824“更新pcrinfo=`catpcr.blob`"

$keyctl打印268728824

一年二次发生额为2 0 0 0 0 0 0万元，2 0 0 0 0 0 8年，该公司以该公司为单位，以该公司为单位，以该公司为单位，以该公司为单位，以该公司为单位，以该公司为单位，以该公司为单位，以该公司为单位，以该公司为单位，以该公司为单位，以该公司为单位，以该公司为单位，以该公司为单位，以该公司为单位，以该公司为单位，以该公司为单位，以该公司为单位，以该公司为单位，以该公司为单位，以该公司为单位，以该公司为单位，该公司以该公司为单位，以该公司为单位，以该公司为单位，该公司以该公司为单位，该公司以该公司为单位，该公司以该公司为单位，该公司的68C354C5EAB4型第二种方法是用一种新的方法，用该方法对一种以该种方法为主的两种方法进行了比较，结果表明，该方法对该种方法的处理效果较好。该方法对该种方法的处理效果较好。该方法的结果表明，该方法能有效地减少对该种方法的处理。该方法对该种方法的处理效果好。该方法的结果表明，该方法是一种可靠的、可靠的、可靠的。该方法的结果表明，该方法能有效地减少对该种方法的处理。该方法对该种锅炉的使用该种方法的处理效果。该方法的结果表明，该方法对该种方法的处理效果良好。该方法的结果表明，该方法对该种方法的处理的效果良好。该方法在该方法中，该方法对该方法的处理的处理的处理的效果。该方法在该方法4434644EFdf8ae9a178e9f83ba9f08d10fa47e4226b98b0702f06b3b8

使用上述可信密钥“kmk”创建并保存加密密钥“evm”：

$keyctl add encrypted evm“新建可信：kmk 32“@u

    159771175

$keyctl打印159771175

可信：kmk 322375725ad57798846a9bbd240de8906f006e66c03af53b1b382dbbc55 BE244616E4959430436DC4F2A9659AA60BB4652AEB2120F149ED197C564E024717C64 5972dcb82ab2dde83376d82b2e3c09ffc

$keyctl管道159771175>evm.blob公司从保存的blob加载加密密钥“evm”：

$keyctl add encrypted evm“加载`cat”evm.blob公司`“@u

    831684262

$keyctl打印831684262

可信：kmk 322375725ad57798846a9bbd240de8906f006e66c03af53b1b382dbbc55 BE244616E4959430436DC4F2A9659AA60BB4652AEB2120F149ED197C564E024717C64 5972dcb82ab2dde83376d82b2e3c09ffc

受信任密钥的初始使用者是EVM，它在启动时需要一个高质量的对称密钥来保护文件元数据。使用可信密钥可以有力地保证EVM密钥不会因用户级别的问题而受损，并且当密封到特定的引导PCR值时，可以防止引导和脱机攻击。其他用于可信密钥和加密密钥的用途，如磁盘和文件加密。（正在处理ecryptfs的修补程序。）

***应用程序级实用程序***

# 裤子和公用设施

虽然内核包含一个基本的TPM设备驱动程序，并且内核包含直接使用TPM进行密钥管理的可信密钥，但是对于TPM的访问和配置，还需要一些库和实用程序。Trousers提供符合标准的TPM访问库和相关的TPM实用程序，用于TPM的初始化、管理和使用。

# 开放式

OpenPTS使用Trousers库访问TPM和IMA度量列表，并创建引用和当前完整性清单，并使用签名的TPM引号来锚定（验证）度量列表。这些报告以临时技术秘书处标准格式编制，用于应用程序和供应商之间的互操作性。

***使用案例：***

那么完整性子系统的好处是什么？在什么情况下可以获得这些好处呢？

**主机完整性**-确保托管主机不会受到远程或内部软件攻击的危害。

在过去，文件完整性的定期测量（例如使用Tripwire和Aide）遇到了两个问题。首先，它在所有文件的周期性散列方面存在性能问题；其次，作为应用程序级软件，它很容易受到损害。使用IMA，可以在内核中高效地维护度量，并与inode数据一起缓存。完整性管理应用程序（如Aide）可以获得所有缓存的度量，这样它们就不需要对已经访问过的文件执行冗余读取和散列操作。Aide还可以使用OpenPTS生成参考和当前完整性报告，这些报告的格式符合TCG标准，并且可以基于TPM的硬件签名进行加密验证，这样即使系统受到破坏，测量列表也可以验证。

**主机真实性**-保证所有文件的真实性。

对于一些用例，比如嵌入式移动设备，真实性，除了完整性之外，是一个重要的要求。以前的真实性机制，如签名可执行文件（DigSig）只提供了部分解决方案，只对ELF可执行文件起作用。通过IMA-evaluation和EVM，包括脚本和配置文件在内的所有文件都可以进行身份验证，并且可以防止脱机攻击。对于不变的文件，权威的数字签名提供了真实性和不变性。对于本地生成/更改的文件，EVM对称签名可保护本地真实性以防脱机攻击。

**可信加密**-通过将加密密钥绑定到系统完整性来提供更安全的加密，从而使敏感数据不会暴露在受损的系统中。

可信密钥为Linux带来了类似CCA的密钥管理和保护，因此密钥对用户空间和应用程序永远不可见，因此密钥的使用与系统的完整性息息相关。与以前基于可信计算的密钥不同，这些密钥可以通过标准的内核密钥环实用程序轻松管理，并且可以轻松地跨系统更新和多引导场景进行迁移。

**可信网络连接（TNC）**–在受信任节点之间建立网络连接期间，基于PTS的端到端可信状态保证。

对于企业级客户，基于TNC的网络准入控制对于阻止受损或配置错误的计算机访问敏感网络至关重要。通过基于IMA/PTS的完整性认证的网络准入，企业可以在开放的TCG标准下拥有强大的基于硬件的完整性验证。

***小结：***

本文档概述了与完整性相关的内核和应用程序级组件、它们的目标、设计和使用，以帮助理解它们提供的总体保护和好处。