基于抽样与实时监控的完整性快速检测

卢海阳 [1,2] 程 光 [1,2]

（[1]东南大学计算机科学与工程学院，南京 21189）

（[2]东南大学教育部计算机网络和信息集成重点实验室，南京 211189）

摘 要：保证数据的完整性是数据安全保护的关键问题。通过抽样检测与实时监控的方法可以有效提高文件完整性检测效率，实现快速检测。针对被保护的对象可以进行分类，应用一定的抽样规则来进行检测，在保证一定检测准确率的基础上提高检测效率。针对文件访问进行实时监控，建立文件监控列表，对于被访问文件进行完整性检测。

关键词：完整性；快速检测；文件监控；抽样

The rapid detection of file integrity

Lu Haiyang[1,2] Cheng Guang[1,2]

([1]School of Computer Science and Engineering, Southeast University, Nanjing 211189, China)

([2]Key Laboratory of Computer Network and Information Integration, Ministry of Education, Southeast University, Nanjing 211189, China)

Abstract: Ensure data integrity is a key issue of data security protection. The method of sampling and real-time monitoring file integrity can effectively improve the detection efficiency to achieve rapid detection.The object to be protected can be classified and apply certain rules to detect them to improve the detection efficiency on the basis of the guaranteed accurate detection rate. Real-time monitoring file access and establish a list of file monitoring to complete integrity testing of the file being accessed.

Key Words: Integrity;Rapid Detection;File Monitor;Sampling

1. 引言

随着互联网的发展，信息化进一步深入人们的日常生活。目前大量的数据都是以电子化的形式存储的。例如银行客户信息存储在数据库中，政府机构的涉密文件存储在专用的文件服务器上，企业级的数据库用来存储业务数据，如财务凭证、报表等，个人通讯录、照片直接存储在手机上。信息化给人们的生活带来了极大的便利，但是也存在诸多隐患。近年来，信息安全事故频频发生，如 CSDN 用户信息泄露，多个网站也发生类似情况，支付宝用户莫名其妙被捐款，团购价格遭到篡改。在这些事故中，数据大都被泄露或篡改。随着移动终端的迅猛发展，信息安全的形势更加严峻。因此，如何有效的保护数据成为了极为重要和关键的问题，而数据完整性保护是其中的重要方面[1]。

目前数据完整性检测的主要方法是通过哈希函数生成被保护文件的数字签名，然后在进行验证时重新生成数字签名来和初始化时的签名相对比，若吻合则完整性得到了保护，若签名不同则说明数据完整性被破坏[7]。目前使用较广泛的完整性检测软件有 tripwire 和 sentinel。采用传统方法进行完整性检测能够有效保护系统完整性，但是花费时间代价较大，在一些计算性能较差的设备上会带来较差的用户体验。目前研究人员采用了多种方法来提高完整性检测效率。宋宁楠[2] 、高伟[3]通过增量哈希的方法减少哈希值重新生成的时间，在对文件进行较小的改动时这种方法能够显著提高检测效率；温敏[4]采用文件实时监控的方法生成访问日志，通过访问日志来判断文件完整性是否得到保护。

本文主要研究基于抽样和实时监控的完整性快速检测。在 PC 端和移动端（移动端选取 android 平台）进行相关研究，采用抽样检测可以大大减少单次检测文件数量，大幅提高检测效率。 利用 android 平台的开源性，可以实现对文件访问的实时监控，建立相应的文件监控列表[6]，对于被访问的文件进行完整性检测，通过这样的方法使完整性检测更加具有针对性，效率更高。

本文的组织结构如下：第二节介绍文件完整性快速检测方法；第三节介绍实验方法和结果分析；

第四节总结和阐述下一步工作。

2. 文件完整性快速检测方法

传统的完整性检测方法需要对被保护的文件全部进行检测，所耗费的时间代价较大。移动平台近年来发展迅猛，但是其计算性能要落后于个人电脑，因而在移动端采用传统方法来检测完整性则效率更低，严重影响用户体验。在保持原有检测模型，即通过对比前后数字签名来确定完整性的基础上，本文研究通过减少单次检测的文件数量或者文件内容来提高检测效率。

进行完整性检测的先决条件是需要确定完整性检测策略。本文中我们采用的策略是保护系统内核和用户选择的重要文件，针对这些文件的任何非正常改动都应视为违规行为。

2.1 抽样检测方法

抽样检测可以有效减少单次检测的时间，提高检测效率，再用一定的规则将单次抽样的结果组合在一起，通过多次检测可以保证较高的检测准确率。针对文件的检测可以细分为大文件和小文件，大文件提取部分内容进行检测，而小文件则从小文件集合中选取部分来检测。

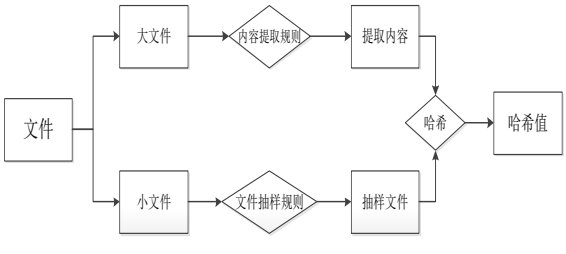

图 1 抽样规则分类图

针对大文件的检测采用内容提取规则对其选取部分内容来进行哈希，初始化得到原始签名保存，用户进行完整性检测时也采用同样的内容提取规则来进行哈希，对比两次签名即可得到相应大文件的完整性状态。

针对小文件的检测采用文件抽样规则来选取部分文件进行哈希。在小文件集合中，单次检测只检测部分文件，利用一定规则可以保证在多次检测后每个文件都得到了检测。

2.2 抽样规则确定

大文件可以采用多种方法来进行内容提取，这些方法各自具有一定的优缺点，列举如下：

方法一：直接哈希，对于全部内容都进行哈希，这种方法优点是检测准确率高，缺点是效率低下，计算时间较长。

方法二：概率抽样，将大文件内容分块，按照一定概率对于对分块抽样，组合到一起再进行哈希。这种方法优点是检测效率高，计算时间较短，缺点是准确率较低，检测结果是带单错的，若检测结果为完整性被破坏则一定正确，若检测结果为完整性未被破坏则不一定正确。

方法三：位置抽样，针对大文件特定位置的数据进行内容提取抽样，组合到一起再进行哈希。这种方法的优缺点与概率抽样类似，但是因为进一步减少了抽样内容，检测效率得到进一步提高，伴随的代价就是牺牲了检测准确率，检测准确率同样是带单错的。

方法四：分块异或，先将大文件分块进行异或，最后得到的分块再进行哈希。这种方法的优点是检测准确率高，对被保护的数据进行了完整的处理，缺点是检测效率较低，异或的时间复杂度低于哈希计算但是相对抽样还是显得效率偏低。

针对这四种大文件抽样方法分别进行了相关实验，实现过程和数据分析在后文。

在单次检测中采用小文件抽样规则只选取部分小文件来进行完整性检测，这样做可以显著提高小文件检测效率，提高的幅度取决于单次检测的文件数量，检测数量越少则检测效率越高但是检测覆盖率也就越低，牺牲了检测准确率。因而需要选取一个合适的小文件抽样规则，在提高单次检测效率的同时要满足一定的检测覆盖率和准确率。

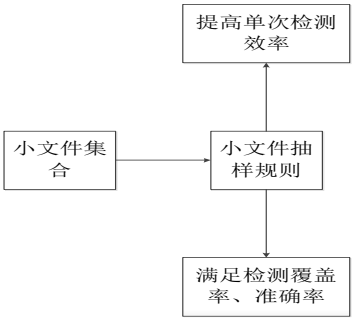

图 2 小文件集合抽样分析图

根据以上分析，对于小文件集合的抽样不能采用简单的概率抽样，这样的话在极端情况下会出现部分文件很长一段时间内都得不到检测，带来较差的检测覆盖率。

本文中采用了标志赋值的方法来提高多次检测的覆盖率和准确率。在文件哈希初始化时，给其设定一个标志值，每个标志值在文件集合中出现的次数基本相等，在之后的单次检测中都针对具有相同标志值的文件进行检测，在对所有标志值都检测过后即可保证所有的小文件都已经得到检测。

通过单次检测提高效率、多次检测提高覆盖率的方法，可以带来更好的用户体验。单次执行的效率提高后，在用户运行程序时对系统带来的影响更小，之后可以由程序判断系统空闲，进行自动检测，在多次检测后，即可完成对于小文件集合的完整检测。

2.3 文件访问监控

Android 系统是 google 公司开发的基于 linux2.6 内核的开源手机操作系统。inotify 是 Linux 内核提供的一个文件系统变化通知机制，从 2.6.13 版本的内核开始提供，比如你在创建一个文件时它可以通知你哪个文件被创建了，删除文件时通知你哪个文件被删除了，修改文件时通知你哪个文件被修改了，关闭文件时哪个文件被关闭了，是可写关闭还是不可写关闭等等。利用 inotify 机制，可以建立起有效的文件访问监控，进行完整性检测时只检测近期被访问过的文件，通过这样的方法使完整性检测更加具有针对性，能够大大提高检测效率。

利用 Android 系统提供的 api 可以方便的实现针对文件访问的实时监控。建立监视文件列表，当发生文件访问时，将被访问文件加入待检测文件列表，下一次进行完整性检测时即检测该文件完整性。

3．实验方法和结果分析

针对文件抽样规则在 win7 系统上进行实验。由于待检测的文件数量很多，而且可能存在一些较大的文件，因而在实现时，采用了内存映射文件。

内存映射文件，是由一个文件到一块内存的映射。Win32 提供了允许应用程序把文件映射到一个进程的函数。内存映射文件与虚拟内存有些类似，通过内存映射文件可以保留一个地址空间的区域，同时将物理存储器提交给此区域，内存文件映射的物理存储器来自一个已经存在于磁盘上的文件，而且在对该文件进行操作之前必须首先对文件进行映射。使用内存映射文件处理存储于磁盘上的文件时，将不必再对文件执行 I/O 操作，使得内存映射文件在处理大数据量的文件时能起到相当重要的作用。

使用内存映射文件来访问磁盘上的数据文件时必须执行下列的操作步骤：

1） 通过 CreatFile 函数创建或打开一个文件内核对象。
2） 通过 CreatFileMapping 函数创建一个文件映射内核对象。
3） 通过 MapViewOfFile 将文件的数据映射到进程的地址空间，完成这一步后就可以直接读取数据文件。
4） 通过 UnmapViewOfFile 从进程的地址空间中撤销文件数据的映射。
5） 用 CloseHandle 函数关闭文件映射对象和文件对象。

这里需要特别注明的是映射的文件大小必须是电脑内存分配精度的整数倍，一般设置为 64Kb 的整数倍。若映射大小设置不对则程序会报错。通过内存映射文件方法获取数据后就要对数据进行处理。

3.1 哈希时间对比

实验方法：首先针对大文件的内容提取规则进行实验。分别采用直接哈希、抽样哈希和异或哈希的方法来进行实验。这里抽样哈希采用的方法是概率抽样，按照 64Kb 大小对文件进行分块，每十个数据块中选择一个，组合所有抽样的数据块再进行哈希即得到文件的哈希值。异或哈希是将文件按照 64Kb 大小分块，先对所有数据块进行异或，通过这种预处理的方法只需要对最后异或生成的数据块进行哈希即可得到文件哈希值。

实验结果如下(实验中所有时间单位均为 ms)：

文件大小	直接哈希时间	抽样计算时间	异或计算时间
9MB	205	4	88
15MB	338	8	128
44MB	1070	20	404
105MB	2537	45	1165
568MB	17380	256	5954

表 1　哈希时间对比表

实验结果分析：直接哈希时间代价较大，不能满足快速检测的要求，采用概率抽样的方法可以大幅提高检测效率，减少哈希时间，但是被抽样文件的检测准确率会受到影响。采用异或哈希方法能提高检测效率，但是效果不如抽样方法显著。

3.2 抽样计算时间对比

实验方法：大文件内容提取规则中抽样方法主要分为概率抽样和位置抽样。概率抽样是将文件按照一定大小分块，然后按照一定概率来提取数据块，对于提取出的数据块再进行哈希。这里有两个重要的变量要确定，一个是概率，一个是分块大小。抽样概率对于时间复杂度显而易见是线性影响的，降低抽样概率能减少检测时间，但是同时也会降低检测准确率。概率对于检测时间的影响是可控而且明显的，因而这里不对概率进行过多研究，实验中选取 10%作为抽样概率来对数据分块大小进行研究。由于内存映射技术中要求映射数据块大小为 64kb 的整数倍，为了便于抽样这里选取的分块大小也为 64kb 的整数倍。位置抽样采取的方法参考迅雷中应用的方法，分别对文件的前中后三个位置各取 4Kb 大小数据进行哈希，得到的哈希值即为文件的哈希值。

实验结果如下：

	概率抽样					位置抽样	
文件大小	64K/分块	128K/分块	512K/分块	1M/分块	2M/分块	4K/块	8K/块
486K	1	1	1	1	1	0.35	0.7
848K	14	2	2	2	3	0.80	0.82
90M	37	25	24	26	27	0.88	0.88
133M	60	37	36	37	38	0.80	0.84
400M	121	110	108	106	108	0.78	0.78
1G	295	270	255	257	258	0.88	0.81

图 3　抽样时间对比表

实验结果分析：大文件采用概率抽样和位置抽样的实验结果对比明显。位置抽样由于检测数据大小固定，检测时间比较稳定，特别在文件较大时，位置抽样的检测时间比概率抽样小很多。概率抽样时针对单个文件的分块测试结果不明显，不同分块之间的计算时间比较接近。

由于电脑上运行效率不是非常稳定，在计算单个文件哈希时间时波动较大，因而针对一些文件集合进行了实验。

实验方法：实验中对文件按照不同分块大小进行概率抽样，不到分块大小的文件进行直接哈希。这种方法其实将分块大小默认为大文件小文件的分界点，并不是很严谨，在后面实验中会针对分界点进行研究。

实验结果如下：

文件目录	64K/分块	128K/分块	512K/分块	1M/分块	2M/分块
system32	84203	42294	40692	94237	98204
system32	81644	43175	43593	93265	98255
winsxs	219488	202856	193117	206129	233991
ubuntu	56266	28311	15085	9008	7647
F/常用软件	85191	51992	27361	16180	15710

图 4　文件集合时间对比图

实验中选取的文件集合说明如下：

1) system32 和 winsxs 都位于 c 盘 windows 目录下，其中包含了大量的小文件，大文件数目较少。
2) ubuntu 和 F/常用软件两个目录下包含了大量的大文件，小文件数据较少。

实验结果分析：针对不同类型的文件集合采用不同的分块大小能得到更好的检测性能。当文件集合中包含大量小文件，如 system32 和 winsxs 时，分块取得过大效果等同于对文件进行直接哈希，检测时间较长。而对于以大文件为主的文件集合时，采用较大的分块来进行抽样可以得到更短的检测时间。

3.3 综合抽样实验

实验方法：采用系统文件 system32 作为实验对象，针对大文件小文件分界点进行研究。实验中针对大文件的内容提取采用了位置抽样，对于抽样块大小也进行了不同的实验。分界点取的过大则加大了小文件的计算量，取的过小则把过多的文件归入了大文件，降低了检测的准确率。由于采用了内存映射文件，选取 64kb 的整数倍作为分界点来进行实验。实验中针对小文件集合分别采用了直接哈希和集合抽样的两种方法。给每个小文件初始化一个

标志值（1-5），每次抽样时依次选取不同的标志值来进行检测，这样五次检测后所有的小文件都被检测过了，而单次对于小文件的检测量降低到了 20%。

实验结果如下：

文件分界值	抽样大小	小文件直接哈希	小文件集合抽样 (20%)	
64K	4K	7092	5973	3896
64K	8K	7074	3781	3725
128K	4K	7465	3957	3492
128K	8K	7450	4406	3526
512K	4K	9276	4919	4266
512K	8K	8726	4306	3925
512K	16K	8682	4071	3733
1M	16K	78684	6617	4222
2M	16K	92202	9391	4877

图 5：检测时间对比图

实验结果分析：大文件、小文件的分界点取得过大会加大计算量，取得过小会降低检测准确率。针对 system32 文件夹进行的实验结果表明，选取 128k 为分界点，4K 为抽样大小进行抽样较为理想。分界点以下的小文件集合进行集合抽样，每次选取 20%的文件进行计算，通过这样的方法可以进一步较少计算时间，提升检测效率。

3.4 文件实时监控

Android 系统 API 提供了 FileObserver 抽象类（Linux 的 INotify 机制）来监听系统/sdcard 中的文件或文件夹，FileObserver 类能对 sdcard 中的文件及文件夹的打开、创建、移动和删除操作进行监控。

关键代码如下：

```
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
if(null == mFileObserver) {
mFileObserver = new
SDCardFileObserver(Environment.getExternalStorageDirectory().getPath());
mFileObserver.startWatching(); //开始监听
}
}
public void onDestory() {
if(null != mFileObserver) mFileObserver.stopWatching(); //停止监听
}
static class SDCardFileObserver extends FileObserver {
//mask:指定要监听的事件类型，默认为 FileObserver.ALL_EVENTS
public SDCardFileObserver(String path, int mask) {
super(path, mask);
```

实验中需要注意防止 FileObserver 类的对象被垃圾回收，否则将不能收到文件及文件夹的监听事件了，程序运行在 Android 模拟器中，使用 Eclipse 的 DDMS 中的 File 视图来对 Android 模拟器的 sdcard 中的文件及文件夹进行操作 。

4. 总结及下一步工作

本文对文件完整性快速检测进行研究，通过抽样检测和文件访问实时监控的方法可以较好的实现快速检测。对于大文件进行内容提取对于小文件进行集合抽样，通过抽样的方法能够大大减少单次检测的数据内容，较大的缩短了检测时间，提升了检测效率。由于 android 平台的一些特性，能够较好的实现文件访问实时监控，通过监控使完整性检测具有针对性，减少检测数据，提升检测性能。

目前采用的方法着重于缩短检测时间，通过一定策略可以提高小文件集合的准确率，但是对于大文件的内容抽样方法还不够精细，采用的抽样方法需要进一步改进。研究文件关键信息保存特征和文件被修改及访问特点，找出一定的规律，将抽样方法改进。

参考文献：

[1] 侯方勇.存储系统数据机密性与完整性保护的关键技术研究.[D]国防科技大学.2005

[2] 张学旺.MD5 算法及其在文件系统完整性保护中的应用.[C].计算机应用.2003

[3] 宋宁楠.磁盘数据完整性保护技术研究.[D]上海交通大学.2009

[4] 高伟.磁盘数据安全保护技术研究.[D]上海交通大学.2008

[5] 温敏.Android 智能手机系统中文件实时监控的研究与实现.[C] 科学技术与工程.2009

[6] Android project official. Android project[EB／OL】. http：／／www. android. corn／，2008.

[7] Enhancing File System Integrity Through Checksums [C] Technical Report FSL-04-04 of Stony Brook University[R].New York, 2004.7.