

基于信息摘要函数的文件完整性检测工具

孙凌宇¹, 冷 明¹, 周 宇², 夏洁武¹

(1 井冈山大学 计算机科学系, 江西 吉安 343009; 2 井冈山大学 网络中心, 江西 吉安 343009)

摘 要: 文中在 Microsoft Visual Studio 2005 集成开发环境下, 用 C# 实现了基于信息摘要函数的文件完整性检测工具. 该工具使用 CRC32、MD5 和 SHA1 信息摘要函数, 通过计算文件的数字文摘值, 向系统管理员提供文件完整性检测功能, 从而为发现入侵提供依据. 实验表明, 该工具在对多个文件的完整性测试过程中, 计算出的 CRC32、MD5、SHA1 数字文摘值正确无误.

关键词: 信息摘要函数; 文件; 完整性检测

中图分类号: TP391

文献标识码: A

文章编号: 1000-7180(2010)10-0182-04

File Integrity Checking System Based on Message Digest Function

SUN Ling-yu¹, LENG Ming¹, ZHOU Yu², XIA Jie-wu¹

(1 Computer Science Department, Jinggangshan University, Ji'an 343009, China;

2 Network Information Center, Jinggangshan University, Ji'an 343009, China)

Abstract: We design and implement the file integrity checking system based on message digest function, using C# language in Microsoft Visual Studio 2005 integrated development environment. The system provides file integrity detection function by calculating the message digest value of CRC32, MD5 and SHA1, so as to provide the evidence of invasion discovery for the system administrator. The experimental evaluations show that the system generates the correct message digest value of CRC32, MD5 and SHA1 in the testing process of multiple files integrity.

Key words: message digest function; file; integrity check

1 引言

作为计算机系统核心的文件系统是比较普遍的被攻击目标, 入侵者进入系统后让系统文件不被改动是非常困难的. 入侵者通常修改系统配置和程序以便于下次进入系统, 修改系统日志来掩盖他们的入侵路径和阻碍入侵的发现. 入侵检测系统是系统安全的一道重要防线, 主要通过多种手段监控系统来发现入侵, 其中检测文件系统的非授权改变和不可预料的改变给系统管理员提供了宝贵的数据, 他们可以紧密地检测文件内容的完整性, 从而为发现入侵提供依据^[1].

文件完整性检测主要关注某个文件或目录是否被更改, 包括文件和目录的内容及属性等. 它利用信

息摘要函数强有力的加密机制, 识别文件发生的微小变化, 只要是入侵者成功的攻击导致文件或目录的任何改变, 文件完整性检测都能够发现^[2]. 可以借助该检测工具对所需保护的文件计算出一个数字文摘并将它保存, 每次检查时, 重新计算文件的数字文摘并将它与历史数字文摘值相比较, 从而判断文件是否遭到修改. 如果不同, 则文件已被修改, 如果相同, 则未发生变化.

2 数字文摘和信息摘要函数

文件的数字文摘通过信息摘要函数计算得到. 不管文件长度如何, 其信息摘要函数的计算结果是一个固定长度的数字. 与加密算法不同, 信息摘要算法是一个不可逆的单向函数. 采用安全性高的信息

收稿日期: 2009-09-10; 修回日期: 2009-11-11

基金项目: 上海市教育委员会科研创新项目(08YZ13); 江西省教育厅科学技术研究项目(GJJ09590)

摘要算法,如 MD5、SHA、CRC32 等,两个不同的文件几乎不可能得到相同的数字文摘。

MD5 是 Message Digest Algorithm 5(信息摘要算法)的缩写,被广泛用于加密和解密技术上。MD5 可以说是文件的“数字指纹”,通过不可逆的字符串变换算法产生数字文摘,以此检查、验证信息的完整一致^[3]。MD5 具有全球唯一特性。任何一个文件,无论是可执行程序、图像文件、临时文件或者其他任何类型的文件,也不管它多大,都有且只有一个独一无二的 MD5 信息值。如果这个文件被修改过,它的 MD5 值也将随之改变。因此,我们可以通过对比同一文件的 MD5 值,来校验这个文件是否被“篡改”过。

SHA1 是 Secure Hash Algorithm(安全哈希算法)的缩写,由美国国家安全和美国国家标准与技术研究院联合研发,主要适用于数字签名标准(DSS: Digital Signature Standard)定义的数字签名算法(DSA: Digital Signature Algorithm)^[4]。对于长度小于 264 位的信息,SHA1 会产生一个 160 位的数字文摘。当接收到信息时,这个数字文摘可以用来验证数据的完整性。在传输过程中,数据很可能会发生变化,此时会产生不同的数字文摘。SHA1 有如下特性:①不可以从数字文摘中复原信息;②两个不同的信息不会产生同样的数字文摘。

CRC 是 Cyclical Redundancy Check(循环冗余校验)的缩写,是一种根据网络数据封包等数据产生简短固定位数的一种散列函数,主要用来检测、校验数据传输或者保存后可能出现的错误。生成的数字在传输或者储存之前计算出来并且附加到数据后面,然后接收方进行检验,确定数据是否发生变化。一般来说,循环冗余校验的值都是 32 位的整数^[5]。

3 基于信息摘要函数的文件完整性检测工具

3.1 系统设计

基于信息摘要函数的文件完整性检测工具的功能设计要求为:使用 CRC32、MD5 和 SHA1 信息摘要函数,通过计算文件的数字文摘值,向系统管理员提供文件完整性检测功能,从而为发现入侵提供依据。

根据文件完整性检测工具的功能设计要求,结合软件工程中软件设计步骤,划分程序功能模块为: CRC32 数字文摘计算模块、MD5 数字文摘计算模块、SHA1 数字文摘计算模块;然后在 Windows 平台下用 C# 语言编写各模块实现代码;最终在 Microsoft Visual Studio 2005 集成开发环境下实现该文

件完整性检测工具。工具运行的主界面如图 1 所示,其中给出了文件的大小、全文件名、以及文件相应的 CRC32、MD5 和 SHA1 数字文摘值。



图 1 基于信息摘要函数的文件完整性检测工具主界面图

3.2 CRC32 数字文摘计算模块

对于 CRC32 数字文摘计算模块,采用查表法来实现,具体的计算步骤为:(1)将上次计算出的 CRC 校验码右移一个字节;(2)将移出的这个字节与新的要校验的字节进行 XOR 运算;(3)用运算出的值在预先生成码表中进行索引,获取对应的余式值;(4)用获取的余式值与第 1 步右移后的值进行 XOR 运算;(5)如果要校验的数据已经处理完,则第 4 步的结果就是最终的 CRC 校验码。如果还有数据要进行处理,则再转到第 1 步运行^[6]。C# 语言查表法实现 CRC32 计算的关键代码如下:

```
static UInt32 [ 256 ] crc32Table = { 0x00000000,
0x77073096, 0xee0e612c, ..., 0x5a05df1b, 0x2d02ef8d };

public static int GetCRC32 ( byte [ ] bytes, int offset, int
length )
{
    int iCount = length;
    UInt32 crc32 = 0xFFFFFFFF;
    for ( int i = offset; i < iCount; i ++ )
    {
        crc32 = (( crc32 >> 8 ) & 0x00FFFFFF ) ^
        crc32Table [ ( crc32 ^ bytes [ i ] ) & 0xFF ];
    }
    UInt32 temp = crc32 ^ 0xFFFFFFFF;
    int t = ( int ) temp;
    return ( t );
}
```

3.3 MD5 数字文摘计算模块

MD5 数字文摘计算模块以 512 位分组来处理输入的信息,且每一分组又被划分为 16 个 32 位子分组。经过了一系列处理后,算法的输出由 4 个 32

位分组组成,将这 4 个 32 位分组级联后生成一个 128 位散列值.具体的计算步骤为:(1)加填充位,使其位长对 512 求余的结果等于 448.填充的方法是:在信息的后面填充一个 1 和若干个 0,直到满足上面的条件时才停止用 0 对信息的填充.(2)附加一个以 64 位二进制表示的填充前信息长度(低位字节优先),使其总位长等于 $N \times 512 + 448 + 64$,即长度恰好是 512 的 $N+1$ 整数倍.(3)初始化 MD5 的缓存,使用一个 128 位的缓存来存放该散列算法的中间及最终结果,该缓存表示为 4 个 32 位的寄存器 A、B、C、D,且初始化 A 为 0x01234567、B 为 0x89abcdef、C 为 0xfedcba98、D 为 0x76543210.(4)循环执行变换函数来处理 512 位信息分组序列,且循环的次数是信息中 512 位信息分组的数目 $N+1$,操作具体描述请见 <http://www.ietf.org/rfc/rfc1321.txt>.(5)所有 $N+1$ 个 512 位信息分组处理完成后,输出 A、B、C、D 连续存放的 128 位的散列值^[7-8].C# 语言分组处理实现 MD5 计算的关键代码如下:

```
public static byte[] MD5Array(byte[] input)
{
    MD5 _Init();
    UInt32[] block = MD5 _Append(input);
    UInt32[] bits = MD5 _Trasform(block);
    byte[] output = new byte[bits.Length * 4];
    for (int i = 0, j = 0; i < bits.Length; i++, j += 4)
    {
        output[j] = (byte)(bits[i] & 0xff);
        output[j + 1] = (byte)((bits[i] >> 8) & 0xff);
        output[j + 2] = (byte)((bits[i] >> 16) & 0xff);
        output[j + 3] = (byte)((bits[i] >> 24) & 0xff);
    }
    return output;
}
```

对于 MD5 数字文摘计算模块,还可以采用 C# 语言在命名空间 System.Security.Cryptography 提供的 MD5CryptoServiceProvider 类和 MD5 类来实现,具体实现 MD5 计算的关键代码如下:

```
private void ShowFileMD5CryptoInfor(byte[] bytes, int offset, int length)
{
    System.Security.Cryptography.MD5 md5 =
        new MD5CryptoServiceProvider();
```

```
byte[] md5Result = md5.ComputeHash(bytes, offset, length);
String md5Ret = "";
for (int i = 0; i < md5Result.Length; i++)
    md5Ret += md5Result[i].ToString("X").PadLeft(2, '0');
this.tbFileInfor.Text += "MD5 : " + md5Ret + "\r\n";
}
```

3.4 SHA1 数字文摘计算模块

SHA1 数字文摘计算模块以 512 位分组来处理输入的信息,最终产生一个 160 位散列值.具体的计算步骤为:(1)加填充位,使其位长对 512 求余的结果等于 448,填充的方法与 MD5 算法相同.(2)附加一个以 64 位二进制表示的填充前信息长度,使其总位长等于 $N \times 512 + 448 + 64$,即长度恰好是 512 的 $N+1$ 整数倍.(3)初始化缓存,使用一个 160 位的缓存来存放该散列算法的中间及最终结果,该缓存表示为 5 个 32 位的寄存器 A、B、C、D、E,且初始化 A 为 0x67452301、B 为 0xefcdab89、C 为 0x98badcfe、D 为 0x10325476、E 为 0xc3d2e1f0.(4)循环执行压缩函数处理 512 位信息分组序列,且循环的次数是信息中 512 位信息分组的数目 $N+1$,压缩函数包括 4 次循环,每次循环又包含 20 个处理步骤,每个处理步骤的具体描述请见 <http://www.itl.nist.gov/fip-spubs/fip180-1.htm>.(5)所有 $N+1$ 个 512 位信息分组处理完成后,输出 A、B、C、D、E 连续存放的 160 位的散列值^[9].C# 语言分组处理实现 SHA1 计算的关键代码如下:

```
public static string Compute(Stream stream, long index, long count)
{
    .....
    sha1 _init();
    while ((b = stream.ReadByte()) != -1 && count-- > 0)
    {
        block[size++ % 64] = (byte)b;
        if (size % 64 == 0)
            sha1 _transform(block);
    }
    .....
```

对于 SHA1 数字文摘计算模块,还可以采用 C# 语言在命名空间 System.Security.Cryptography 提供的 SHA1CryptoServiceProvider 类和 SHA1 类来

实现,具体实现 SHA1 计算的关键代码如下:

```
private void ShowFileSHA1CryptoInfor(byte[] bytes, int
offset, int length)
{
    System.Security.Cryptography.SHA1 sha1 =
        new SHA1CryptoServiceProvider();
    byte[] sha1Result = sha1.ComputeHash(bytes, off-
set, length);
    String sha1Ret = "";
    for (int i = 0; i < sha1Result.Length; i++)
        sha1Ret += sha1Result[i].ToString("X").
            PadLeft(2, '0');
    this.tbFileInfor.Text += "SHA1: " + sha1Ret +
        "\r\n";
}
```

4 结束语

文中在基于 Microsoft Visual Studio 2005 集成开发环境下,用 C# 实现了基于信息摘要函数的文件完整性检测工具.在对多个文件的完整性测试过程中,计算出的 CRC32、MD5、SHA1 数字文摘值分别和 CRC Verification Utility 软件 3.05 版本、WinMD5 软件 2.05 版本、File Checksum Integrity Verifier 软件 2.0.5 版本计算的相应数字文摘值相同.今后将增加对 1G 以上超大文件的完整性检测支持,减少数字文摘计算耗时,并增加计算过程的进度条,避免计算比较大的文件时处于假死状态等.

参考文献:

[1] 张学旺,唐贤伦. MD5 算法及其在文件系统完整性保

护中的应用[J]. 计算机应用, 2007,23(1):430-432.

[2] 郑光明,胡博. 基于 MD5 的文件完整性检测软件设计[J]. 湖南理工学院学报:自然科学版,2007,20(1):35-38.

[3] 彭文波. MD5 算法原理及应用[J]. 信息网络安全, 2005,5(3):43-46.

[4] 蔺守河,戴紫彬. 基于 Nios 处理器用户指令的 SHA-1 算法的实现[J]. 微电子学与计算机,2006,23(4):104-106.

[5] 王爱英. 计算机组成与结构[M]. 北京:清华大学出版社,2001.

[6] 常晓明,潘卫华,王建东. CRC 校验及其软件实现[J]. 电子技术应用,1995,6(1):14-15.

[7] 李杰,张学旺. MD5 报文摘要算法与文件系统完整性保护[J]. 湘潭大学自然科学学报,2003,25(3):96-99.

[8] 李春科,李伟华. MD5 加密算法及其在 RADIUS 协议中的应用[J]. 微电子学与计算机,2004,21(2):21-24.

[9] 李杰,张学旺. 数字签名算法 SHA1 的 C++ 实现[J]. 华南金融电脑,2007,4(1):102-104.

作者简介:

孙凌宇 女,(1976-),硕士,副教授.研究方向为算法分析与设计等.

冷明 男,(1975-),博士,副教授.研究方向为算法分析与设计等.

周宇 男,(1974-),硕士,实验室.研究方向为网络安全等.

夏洁武 女,(1969-),硕士,教授.研究方向为算法分析与设计等.

(上接第 181 页)

[6] Wu Q X, McGinnity T M, Maguire L P, et al. Visual stimuli using hierarchical spiking neural networks[J]. International Journal of Neurocomputing, 2008,71(10/12): 2055-2068.

[7] Wu Q X, McGinnity T M, Maguire L P, et al. Edge detection based on spiking neural network model [M]. Berlin, Heidelberg: Springer-Verlag; 2007:26-34.

[8] Quian Quiroga R, Reddy L, Kreiman G, et al. Invariant visual representation by single neurons in the human brain [J]. Nature, 2005(435):1102-1107.

[9] Maria Jose Escobar, Guillaume S Masson, Thierry Vieville, et al. Action recognition using a Bio-inspired feedforward spiking network[J]. Internatioanl Journal of

Computer Vision, 2009(82):284-301.

[10] Anil Bharath, Maria Petrou. Next generation artificial vision systems—reverse engineering the human visual system [M]. London, Artech House, 2008.

[11] John Canny. A computational approach to edge detection [J]. IEEE transactions on pattern analysis and machine intelligence, 1986,8(6):679-69.

作者简介:

蔡荣太 男,(1979-),博士,讲师.研究方向为图像处理与计算机视觉、智能信息处理.

吴庆祥 男,(1958-),博士,教授.研究方向为智能信息处理、图像处理与机器学习.