

# 基于特征串树的病毒特征码匹配算法

于 泠<sup>1</sup>, 李国建<sup>2</sup>

(1. 南京师范大学数学与计算机科学学院, 210097, 南京)

(2. 江苏油田计算中心, 210046, 南京)

[摘要] 在分析已有病毒特征码的精确和模糊模式匹配算法的基础上, 采用面向对象的程序设计方法, 设计实现了一个基于病毒特征串树的匹配算法, 实验证明该方法可以方便地进行病毒特征库的更新以及含通配符特征串的模糊匹配。

[关键词] 反病毒, 病毒扫描, 模式匹配, 算法, 计算机安全

[中图分类号] T309. 5, [文献标识码] B, [文章编号] 1672- 1292- (2003) 04- 0037- 04

## 0 引言

计算机病毒<sup>[1]</sup>对人类的影响可以称得上是灾难性的. 尽管人类已与计算机病毒斗争了数十年, 并已取得了可喜的成绩, 但是随着 INTERNET 的发展, 计算机病毒的种类急剧增多, 扩散速度大大加快, 对用户的破坏性越来越大. 如何有效地控制和杀灭病毒成为人们日益关心和研究的课题.

在与病毒的对抗中, 各种反病毒工具的设计思想和功能选择有所不同. 其中, 常用的检测病毒的方法有: 特征代码法、校验和法、行为监测法和感染实验法等<sup>[2]</sup>. 特征代码法中的关键技术主要是: 病毒特征串的提取、特征串的匹配. 病毒特征串匹配法因为准确度高, 被广泛用于病毒检测工具中. 目前很多反病毒技术都以病毒特征串的匹配为基础<sup>[3, 4]</sup>.

本文在分析已有病毒特征码的精确和模糊模式匹配算法<sup>[5, 6]</sup>的基础上, 采用面向对象的程序设计方法, 设计实现了一个基于病毒特征串树的匹配算法, 实验证明该方法可以方便地进行病毒特征库的更新以及含通配符特征串的模糊匹配.

## 1 基本的特征串扫描技术分析

### 1.1 病毒特征串的精确匹配过程

一个基本的病毒特征串匹配的宏汇编代码的执行过程是: 读取文件内容, 扫描并查找是否含有病毒特征码, 主要由 cmpsb 串比较指令完成匹配. 每读取一次文件内容, 将扫描所有的病毒特征码.

该种匹配法检测计算机病毒具有较高的准确度, 但是局限于病毒特征串的精确匹配, 对含有通配符的特征串还需要采用高效率的匹配算法.

### 1.2 通配符的设置

考虑到大多数简单加密的病毒运用基本的解密例程, 这些基本的解密例程在一次感染和另一次感染之间没有太大差别. 只有加密例程的键值在每一次感染中会改变. 例如, 下面显示的字节可能在同一种加密病毒感染的多个文件中发现:

sample1.com: B9 00 10 BE OC 01 80 34 **52** 46 E2 FA

sample2.com: B9 00 10 BE OC 01 80 34 **78** 46 E2 FA

sample3.com: B9 00 10 BE OC 01 80 34 **05** 46 E2 FA

收稿日期: 2003- 06- 25.

作者简介: 于 泠, 女, 1971- , 硕士, 南京师范大学数学与计算机科学学院讲师, 主要从事信息网络安全技术的研究.

因此可将特征标记为: B9 00 10 BE OC 01 80 34 %2 46 E2 FA

这个特征标记可以准确地匹配病毒解密例程中除第 9 个字节外其余的每一个字节. 如果不使用通配符功能, 检测同样的病毒就需要 256 种特征标记.

我们定义如下 4 种通配符: ? 表示同输入串中的任一个字符匹配; %d 表示跳过输入串中的 0~ d 个字符; \* d 表示跳过正好 d 个字符; \* \* 表示跳过输入串中任意个字符.

1.3 模糊模式匹配算法( FPM 算法)的基本思想

在每一趟匹配过程中:

- (1) 若模式串 p 出现通配符?, 则文本串 t 中的指针 i 向右移动一个字符;
- (2) 若模式串 p 出现通配符%d, 则 i 指针依次向右移动 0~ d 个字符;
- (3) 若模式串 p 出现通配符\* d, 则 i 指针向右移动 d 个字符;
- (4) 若模式串 p 出现通配符\* \*, 则 i 指针向右移动尽可能长的一段距离;
- (5) 若出现 p. ch[ j] 与 t. ch[ i] 不等时, 利用已得到的“部分匹配”结果, 将模式向右“滑过”文本串尽可能远的一段距离; 继续比较.

基于 FPM 算法的基本思想, 我们采用面向对象的程序设计方法, 定义了一个病毒信息抽象类, 这样可以在发现新病毒时方便地添加新病毒的名称、病毒类型、特征串等信息. 将病毒的特征串构造成一棵特征串稀疏树, 并设计了对该稀疏树的模式匹配算法.

2 病毒信息抽象类的设计

C++ 允许程序员声明一个没有实例对象的抽象类, 其唯一的用途就是被继承. 抽象类中只描述一组子类共同的操作接口, 而将完整的实现类给子类. 由此算法中定义了一个包括各类病毒共同信息的抽象类 VirInfo.

```
class VirInfo
{protected:
    string_ type; // 病毒类型
    string_ name; // 病毒名称
    string_ sig; // 病毒特征串
    int_ infects; // 感染类型
public:
    string &name() { return _ name; }
    string &sig() { return _ sig; }
```

```
int &infects() { return _ infects; }
void set_ name( char * n) { _ name= n; }
void set_ sig( char * n) { _ sig= n; }
virtual void print() = 0;
virtual int_ infects_ ftype( string &fname)= 0;
virtual ~ VirInfo() {}
virtual void read( istream&) = 0;
```

类 VirInfo 的所有数据成员受保护, 通过公有继承, 可以派生出所需的病毒实例对象.

3 特征串树的建立算法

将已知病毒特征串构成一棵病毒特征串树, 从树根到任何给定节点的路径标志了与该节点有关的病毒特征串.

该树的每个节点的结构如下:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	?	%d	* d	* *	virusp
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	----	-----	-----	--------

其中 0~ 9、A~ F 对应于 16 进制字符; ?、%d、\* d、\* \* 对应于 4 个通配符; 最后为病毒记录指针, 该记录中含有沿从树根到该节点的路径所对应的特征串.

特征串树建立算法的伪代码是:

```
void Create_ tree(node * t)
```

```

// 对病毒特征中的每个字符作处理
for( every character in the virus signature)
{if ( the character is a hexadecimal digit)
    //如果字符属于 0~9、A~F 中,为结点 t 中的该字符建立一个后继结点
    make a child of t corresponding to this fixed nibble;
    t= this child; }
else
    switch( character)//处理通配符,建立相应的后继结点
    {case?: create a child of t flag ?; t= this child; break;
    case %d: create a child of t flag %d; t= this child; break;
    case * d: create a child of t flag * d; t= this child; break;
    case * * : create a child of t flag * * ; t= this child; break;
    }
}
add this regular expression at node t; }

```

例如,对于如下病毒,采用上述算法可建立稀疏树如图 1 所示.

_ type: DOS	_ type: DOS
_ name: Virus_ A	_ name: Virus_ B
_ sig: 01BF	_ sig: 01 * 3C
_ infects: COM SYS	_ infects: COM ARC

## 4 匹配算法

假定输入流中每个字符的位置都可能是一个病毒序列的起始位置. 从起始位置进行匹配,直到从固定位置到当前位置的输入流模式与任何一个病毒特征完全吻合,匹配停止.

算法中主要用到了: `node * traverse( ifnibblestream &i, node &n)` 这一函数,其中 `i` 指输入流, `n` 表示开始进行匹配算法的节点位置. `traverse()` 返回一个指针,若返回的是非空指针,就有一个从节点 `n` 开始的部分特征与输入流匹配;若返回的是空指针,则从节点 `n` 开始的部分不与输入流匹配. 该算法并非寻找和列举输入文件中的所有病毒,而是仅对文件是否被某一种病毒感染作出回答.

```

node * traverse( ifNibbleStream &i, node &n)
{if(there are virus signatures associated with this node)//若结点 n 处有病毒信息,则返回 n
    return &n;
ch= next character from the input stream; // 从输入流读取下一个字符,并存入 ch 中
if(there is a link on ch from n) // 若结点 n 中的字符 ch 有后继结点,调用 traverse() 函数处理后继结点
    if( ret= traverse( i, the node linking on ch from n)) return ret;
for( all the?, %d, * d, * * links of node n)//处理通配符
{if( link is of type %d)//对通配符 %d 作处理
    {int k= d;
    for( int len= 0; len <= k; len++ )//输入流中跳过 len 个字符
        {skip exactly len character;
        if( ret= traverse( i, the node linking on %d from n)) // 处理结点 n 中 %d 处的后继结点
            return ret;
        }
    }
}
}

```

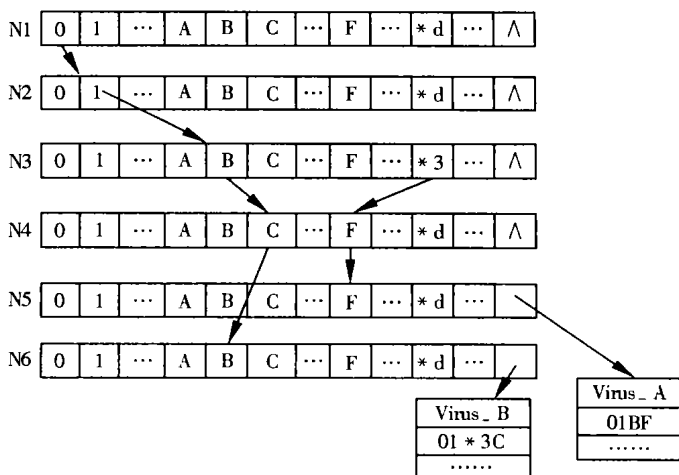


图1 特征串稀疏树

```

return ret;

}}
else if( link is of type * d) // 对通配符* d作处理, 通配符? 转换成* 1来处理
{
    skip exactly d characters; // 输入流中跳过 d 个字符
    if( ret= traverse( i, the node linking on * d from n)) // 处理结点 n 中* d 处的后继结点
        return ret;
}
else if( link is of type * * ) // 对通配符* * 作处理
{
    for( int len= 0; len< file_ len; len+ + ) // 输入流中跳过 len 个字符
    {
        skip exactly len character;
        if( ret= traverse( i, the node linking on * * from n)) // 处理结点 n 中* * 处的后继结点
            return ret;
    }
}
return 0; }

```

例如, 当前输入串为: 01BFCB, 利用以上产生的特征树进行匹配. 在节点 N1 处调用了 traverse(), 此时输入流 01BF 与病毒模式 Virus- A 的特征 01BF 匹配, 返回一个指向节点 N5 的指针; 当前输入串为: 01A4BC, 利用以上产生的特征树进行匹配. 在节点 N1 处调用了 traverse(), 此时输入流 01A4BC 与病毒模式 Virus- B 的特征 01\* 3C 匹配, 返回一个指向节点 N6 的指针.

算法性能分析: 病毒特征串对应于每个节点的顺序是很重要的. 因为当查找试图匹配某个病毒特征串时, 总是采用查找特征树中未被访问的最左路径. 这种排序法保证模式以这种顺序匹配: 0~ 9, A~ F, ?, %d, \* d 和 \* \*. 尽管在多种可能的匹配存在时, 这种顺序不能保证它是最短的匹配, 但它能保证, 如果输入串中能同一个固定模式匹配, 那么这一模式将第一个被找到.

### [参考文献]

- [1] Cohen Fred. Computer Viruses, Theory and Experiments[J]. Computer&Security, 1987, 6(1): 22~ 35.
- [2] 陈波, 于冷. 计算机病毒与反病毒技术的发展[J]. 微机发展, 2000, (6): 48~ 50.
- [3] Derek Atkins. Internet 网络安全专业参考手册[M]. 严伟, 刘晓丹, 王千祥译. 北京: 机械工业出版社, 1998.
- [4] 金晶, 何, 张世永. 基于智能扫描的病毒监视器的研究[J]. 计算机工程, 1999, 25(12): 86~ 88.
- [5] Sandeep Kumar, Eugene H Spafford. A generic virus scanner in C++ [A]. Proceedings of the 8th Computer Security Applications Conference. IEEE Press, 1992. 210~ 219.
- [6] Knuth D. The Art of Computer Programming[M]. Second Edition. Addison Wesley, 1981.

## The Algorithm of Virus Scan Based on Signature Tree

Yu Ling<sup>1</sup>, Li Guojian<sup>2</sup>

(1. College of Mathematics and Computer Sciences, Nanjing Normal University, 210097, Nanjing, PRC)

(2. Computer Center of Jiangsu Oil Field, 210046, Nanjing, PRC)

**Abstract:** Because of the high accuracy, virus pattern matching algorithms are widely used in virus scanner tools. After analyzing the precision and fuzzy patter matching algorithms, a patter matching algorithm based on signature tree, using OOP, is designed and implemented. It is convenient for updating the virus pattems and fuzzy matching.

**Key words:** anti-virus, virus scan, patter matching, algorithm, computer security

[责任编辑: 刘健]