(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 113010882 B
(45)授权公告日 2022.08.30

(21)申请号 202110292042.7

(22)申请日 2021.03.18

(65)同一申请的已公布的文献号
申请公布号 CN 113010882 A

(43)申请公布日 2021.06.22

(73)专利权人 哈尔滨工业大学
地址 150001 黑龙江省哈尔滨市南岗区西大直街92号

(72)发明人 刘立坤 余翔湛 韦贤葵 史建焘 叶麟 葛蒙蒙 李精卫 石开宇 车佳臻 王久金 冯帅 赵跃 宋赟祖

(74)专利代理机构 哈尔滨市伟晨专利代理事务所(普通合伙) 23209
专利代理师 陈润明

(51)Int.Cl.
G06F 21/52(2013.01)
G06F 21/78(2013.01)
G06K 9/62(2022.01)

(56)对比文件
CN 103500178 A,2014.01.08
CN 109977276 A,2019.07.05

审查员 宋梦玲

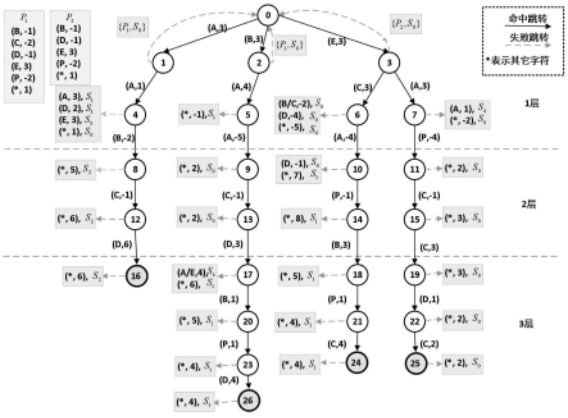
权利要求书2页 说明书7页 附图4页

(54)发明名称

一种适用于缓存丢失攻击的自定义位置顺序模式匹配方法

(57)摘要

本发明提出了一种适用于缓存丢失攻击的自定义位置顺序模式匹配方法,涉及一种匹配算法,尤其涉及一种适用于缓存丢失攻击的自定义位置顺序模式匹配方法,通过建立自定义位置顺序的自动机,将当前扫描字符与当前状态节点进行匹配;当前扫描字符与当前状态节点边值或失败指针中的chr匹配成功,自动机沿着边或失败指针跳转到下一个节点,如果新节点为模式自定义顺序的尾节点,即自动机的叶子节点,命中模式,输出OUTPUT表中记录的模式,自动机当前状态跳转到当前节点记录的下一个节点,继续扫描匹配;自定义位置顺序匹配算法解决了模式匹配算法总是向深度扫描,因大量自动机节点不在CPU缓存内,造成缓存命中率低,系统处理性能大幅下降的问题。



CN 113010882 B

1.一种适用于缓存丢失攻击的自定义位置顺序模式匹配方法,其特征在于,包括以下步骤:

S1.构建自动机,步骤如下:

S1.1.按照自定义位置顺序构建自动机;创建根节点,编号为0;

S1.2.根据1层的第一个字符创建新节点,编号递增,每个模式新增1个节点;根节点与新增节点之间绘制一条边,GOTO表为边值*edge*(*chr*,*mov*),*chr*为1层第一个字符,*mov*为窗口滑动法计算的移动距离,FAIL表为根节点除了创建边的字符外,其它所有字符的*edge*值;

所述窗口滑动法方法步骤如下:

S1.2.1.设置每个模式首字符与窗口首字符对齐;

S1.2.2.窗口中已扫描位置与每个模式对应位置比对字符,所有位置完全匹配成功,执行步骤S1.2.3,匹配失败,执行步骤S1.2.4;

S1.2.3.找到了最佳模式,按自定义位置顺序返回模式的下一个待扫描位置,*mov*值为最近扫描字符位置到下一个待扫描位置的距离,算法结束;

S1.2.4.窗口向左移动一个字符,若窗口尾部位置小于所有模式首字符位置,算法结束,否则,执行步骤S1.2.2;

S1.3.根据1层的第二个字符创建新节点,编号递增,每个模式新增1个节点,每个模式当前节点与新增节点之间绘制一条边,GOTO表为边值*edge*(*chr*,*mov*),*chr*为1层第二个字符,*mov*为窗口滑动法计算的移动距离,FAIL表为模式当前节点除了创建边的字符外,其它所有字符的*edge*值;

S1.4.根据1层的第三个字符创建新节点,编号递增,每个模式新增1个节点,每个模式当前节点与新增节点之间绘制一条边,GOTO表为边值*edge*(*chr*,*mov*),*chr*为1层第三个字符,*mov*为窗口滑动法计算的移动距离,FAIL表为模式当前节点除了创建边的字符外,其它所有字符的*edge*值;

S1.5.根据2层的第一个字符创建新节点,编号递增,每个模式新增1个节点,每个模式当前节点与新增节点之间绘制一条边,GOTO表为边值*edge*(*chr*,*mov*),*chr*为2层第一个字符,*mov*为窗口滑动法计算的移动距离,FAIL表为模式当前节点除了创建边的字符外,其它所有字符的*edge*值;

S1.6.每个模式当前节点移动到最新创建节点,重复步骤S1.1.5,直至2层的最后一个字符完成节点创建;

S1.7.根据3层的第一个字符创建新节点,编号递增,每个模式新增1个节点,每个模式当前节点与新增节点之间绘制一条边,GOTO表为边值*edge*(*chr*,*mov*),*chr*为3层第一个字符,*mov*为窗口滑动法计算的移动距离,FAIL表为模式当前节点除了创建边的字符外,其它所有字符的*edge*值;

S1.8.每个模式当前节点移动到最新创建节点,重复步骤S1.1.7,直至3层的最后一个字符完成节点创建;

S2.扫描指针指向待匹配数据首字符,自动机当前状态设置为根节点;

S3.当前扫描字符与当前状态节点进行匹配;当前扫描字符与当前状态节点所有边值和失败指针中的*chr*匹配成功,执行步骤S4,当前扫描字符与当前状态节点所有边值和失败指针中的*chr*匹配失败,若匹配失败指针中某一个值的*chr*成功,扫描指针根据其*mov*值移动

相应距离，自动机根据其失败指针中记录的 S_t 值跳转到下一个节点，重新扫描；

S4.自动机沿着边跳转到下一个节点，如果新节点为模式自定义顺序的尾节点，即自动机的叶子节点，执行步骤S5，否则，扫描指针根据边的*mov*值移动相应距离，如果移动后超出待匹配数据尾字符，扫描终止，否则重新扫描；

S5.命中模式，输出OUTPUT表中记录的模式，自动机当前状态跳转到当前节点记录的下一个节点，执行步骤S3。

2.根据权利要求1所述的一种适用于缓存丢失攻击的自定义位置顺序模式匹配方法，其特征在于，自动机由GOTO表、FAIL表和OUTPUT表组成。

3.根据权利要求2所述的一种适用于缓存丢失攻击的自定义位置顺序模式匹配方法，其特征在于，自动机构建时设置两类窗口，分别为固定窗口和可变窗口，固定大小窗口 w_1，多个可变大小窗口 w_2，其中 w_1 大小为[1,最短模式长度-1]，w_2 大小为[2，w_1 第二个字符为首字符的最长模式尾字符位置]，w_2 有多个，数量为模式首字符集的大小。

4.根据权利要求3所述的一种适用于缓存丢失攻击的自定义位置顺序模式匹配方法，其特征在于，自动机构建顺序为1层、2层、3层；各层自定义位置顺序如下：

(1)1层：包括3个字符，w_1 首字符、w_1 尾字符、w_2 最长窗口尾字符；

(2)2层：w_1 去掉首尾2个字符的最大子串，并以逆序重新排序；

(3)3层：w_1 尾字符与最长 w_2 尾字符之间的最大子串。

5.根据权利要求4所述的一种适用于缓存丢失攻击的自定义位置顺序模式匹配方法，其特征在于，根据步骤窗口滑动法计算*mov*值，通过查找符合已扫描位置的模式和计算查找到的第一个未被扫描的位置计算*mov*值；计算*mov*值公式如下：

$$mov=\{a_p-k \mid p=\min\{p>a_j, Scan(P_{u_{a_j}})=1, 1\le j<p\}\}$$

式中 $Scan(P_{u_{a_j}})$ 表示字符 $P_{u_{a_j}}$ 是否被扫描过，1为已扫描，0为未扫描；k表示当前扫描位置；a_p 表示模式 P_u 的第一个未被扫描字符的位置。

一种适用于缓存丢失攻击的自定义位置顺序模式匹配方法

技术领域

[0001] 本申请涉及一种匹配算法,尤其涉及一种适用于缓存丢失攻击的自定义位置顺序模式匹配方法。

背景技术

[0002] 模式匹配一直是计算机科学的研究热点和难点。在信息安全领域中,针对模式匹配算法的缓存丢失攻击严重威胁网络安全系统。攻击者利用模式匹配算法中扫描指针移动顺序,伪造攻击数据使得模式匹配算法扫描时总是扫描到路径的深处,增加了系统的缓存丢失率。多模式精确匹配算法按搜索方式可以分为三类:前缀搜索、后缀搜索和子串搜索。在前缀搜索方法中,Aho-Corasick(AC)是最典型的算法。该算法通过计算文本和模式之间的最长公共前缀来移动窗口。在后缀搜索方法中,经典的算法Wu-Manber(WM),其特点是在窗口内从右向左向后搜索。在子串搜索方法中,代表算法为Set Backward Oracle Matching (SBOM)。

[0003] AC算法构造了一个确定性有限自动机(DFA),将模式集记录为Trie树。AC算法包括三个表:GOTO表、FAIL表和OUTPUT表,GOTO表根据当前状态和下一个字符记录下一个状态,FAIL表决定当GOTO表获得的下一个状态无效时返回到哪个状态,而输出表将匹配的模式保存在一个状态中。在AC自动机中,如果有个节点,则该图包含条GOTO跳转边和个失败指针。

[0004] WM算法包括两步:预处理和扫描,预处理建立3个表:SHIFT表、HASH表和PREFIX表。扫描过程中,找出单个字符在模式串中的位置,基于固定长度的子串匹配块进行字符匹配。扫描步骤使用大小的滑动窗口,该窗口从文本的初始字符开始。窗口的第一个HASH值是为窗口后缀计算的。如果HASH值大于零,则在文本字符流中使用相同的值向前移动窗口。当SHIFT值为零,则检查HASH表,并使用后缀散列值找到可能匹配的候选列表。

[0005] SBOM算法使用Factor Oracle 结构为字符串集合构建自动机,自动机对字符串集中所有字符字串的超集。预处理时,取所有字符串前缀最小长度反向构建Trie结构,然后在Trie树上构建Factor Oracle自动机。匹配时,通过字符串前缀最小长度的滑动窗口来扫描文本,在窗口中从右往左扫描最长的从初始状态开始的后缀,该后缀为字符串的一个factor,若命中后缀,扫描字符串剩余部分,若为命中,窗口向后移动,状态转移到初始状态。

[0006] AC和SBOM算法中自动机状态每跳转一次,扫描位置对应移动一个字符,WM算法扫描位置在滑动窗口(w)内移动,w为最短模式长度。可见,现有常用算法的扫描位置每次移动范围在[1, w]内,并不能覆盖最长模式。

[0007] 现有这些算法都是基于固定位置顺序扫描,很容易被攻击者利用实施缓存攻击,攻击者只需要事先获取一些模式后,将模式扫描顺序的最后位置字符去掉或修改,通过大量重复发送,使得模式匹配算法总是向深度扫描,因大量自动机节点不在CPU缓存内,造成缓存命中率低,系统处理性能大幅下降。

发明内容

[0008] 为解决现有技术中存在的容易被攻击者利用实施缓存攻击造成的缓存命中率低、系统处理性能下降的技术问题，本发明提供了一种适用于缓存丢失攻击的自定义位置顺序模式匹配方法。

[0009] 一种适用于缓存丢失攻击的自定义位置顺序模式匹配方法包括以下步骤：

[0010] S1.构建自动机；

[0011] S2.扫描指针指向待匹配数据首字符，自动机当前状态设置为根节点；

[0012] S3.当前扫描字符与当前状态节点进行匹配；当前扫描字符与当前状态节点边值或失败指针中的*chr*匹配成功，执行步骤S4，当前扫描字符与当前状态节点所有边值和失败指针中的*chr*匹配失败，若匹配失败指针中某一个值的*chr*成功，扫描指针根据其*mov*值移动相应距离，自动机根据其失败指针中记录的 S_i 值跳转到下一个节点，重新扫描；

[0013] S4.自动机沿着边跳转到下一个节点，如果新节点为模式自定义顺序的尾节点，即自动机的叶子节点，执行步骤S5，否则，扫描指针根据边的*mov*值移动相应距离，如果移动后超出待匹配数据尾字符，扫描终止，否则重新扫描；

[0014] S5.命中模式，输出OUTPUT表中记录的模式，自动机当前状态跳转到当前节点记录的下一个节点，执行步骤S3。

[0015] 优选的，自动机由GOTO表、FAIL表和OUTPUT表组成。

[0016] 优选的，自动机构建时设置两类窗口，分别为固定窗口和可变窗口，固定大小窗口 w_1，可变大小窗口 w_2，其中 w_1 窗口大小为[1，最短模式长度-1]，w_2 窗口大小为[2，w_1 第二个字符为首字符的最长模式尾字符位置]，w_2 有多个，数量为模式首字符集的大小。

[0017] 优选的，自动机构建顺序为1层、2层、3层；各层自定义位置顺序如下：

[0018] (1)1层：包括3个字符，w_1 窗口首字符、w_1 窗口尾字符、w_2 最长窗口尾字符；

[0019] (2)2层：w_1 窗口去掉首尾2个字符的最大子串，并以逆序重新排序；

[0020] (3)3层：w_1 窗口尾字符与最长 w_2 窗口尾字符之间的最大子串。

[0021] 优选的，步骤S1所述构建自动机的步骤如下：

[0022] S1.1.按照自定义位置顺序构建自动机；创建根节点，编号为0；

[0023] S1.2.根据1层的第一个字符创建新节点，编号递增，每个模式新增1个节点；根节点与新增节点之间绘制一条边，GOTO表为边值*edge*(*chr*，*mov*)，*chr*为1层第一个字符，*mov*为窗口滑动法计算的移动距离，FAIL表为根节点除了创建边的字符外，其它所有字符的*edge*值；

[0024] S1.3.根据1层的第二个字符创建新节点，编号递增，每个模式新增1个节点，每个模式当前节点与新增节点之间绘制一条边，GOTO表为边值*edge*(*chr*，*mov*)，*chr*为1层第二个字符，*mov*为窗口滑动法计算的移动距离，FAIL表为模式当前节点除了创建边的字符外，其它所有字符的*edge*值；

[0025] S1.4.根据1层的第三个字符创建新节点，编号递增，每个模式新增1个节点，每个模式当前节点与新增节点之间绘制一条边，GOTO表为边值*edge*(*chr*，*mov*)，*chr*为1层第三个

字符，*mov*为窗口滑动法计算的移动距离，FAIL表为模式当前节点除了创建边的字符外，其它所有字符的*edge*值；

[0026] S1.5.根据2层的第一个字符创建新节点，编号递增，每个模式新增1个节点，每个模式当前节点与新增节点之间绘制一条边，GOTO表为边值*edge*(*chr*，*mov*)，*chr*为2层第一个字符，*mov*为窗口滑动法计算的移动距离，FAIL表为模式当前节点除了创建边的字符外，其它所有字符的*edge*值；

[0027] S1.6.每个模式当前节点移动到最新创建节点，重复步骤S1.1.5，直至2层的最后一个字符完成节点创建；

[0028] S1.7.根据3层的第一个字符创建新节点，编号递增，每个模式新增1个节点，每个模式当前节点与新增节点之间绘制一条边，GOTO表为边值*edge*(*chr*，*mov*)，*chr*为3层第一个字符，*mov*为窗口滑动法计算的移动距离，FAIL表为模式当前节点除了创建边的字符外，其它所有字符的*edge*值；

[0029] S1.8.每个模式当前节点移动到最新创建节点，重复步骤S1.1.7，直至3层的最后一个字符完成节点创建。

[0030] 优选的，步骤S1.2所述窗口滑动法方法步骤如下：

[0031] S1.2.1.设置每个模式首字符与窗口首字符对齐；

[0032] S1.2.2.窗口中已扫描位置与每个模式对应位置比对字符，所有位置完全匹配成功，执行步骤S1.2.3，匹配失败，执行步骤S1.2.4；

[0033] S1.2.3.找到了最佳模式，按自定义位置顺序返回模式的下一个待扫描位置，*mov*值为最近扫描字符位置到下一个待扫描位置的距离，算法结束；

[0034] S1.2.4.窗口向左移动一个字符，若窗口尾部位置小于所有模式首字符位置，算法结束，否则，执行步骤S1. 2.2。

[0035] 优选的，根据步骤窗口滑动法计算*mov*值，通过查找符合已扫描位置的模式和计算查找到的第一个未被扫描的位置计算*mov*值；计算*mov*值公式如下：

[0036] $$mov=\{a_p-k\mid p=\min\{p>a_j, Scan(P_{u_{a_j}})=1, 1\le j<p\}\}$$

[0037] 式中$Scan(P_{u_{a_j}})$表示字符$P_{u_{a_j}}$是否被扫描过，1为已扫描，0为未扫描；k表示当前扫描位置；a_p表示模式P_u的第一个未被扫描字符的位置。

[0038] 本发明的有益效果如下：自定义位置顺序模式匹配方法以自动机的形式存在，与常用AC算法有两点不同：一是自动机构建不是按模式字节顺序进行遍历，而是按自定义位置顺序；二是自动机的边值定义不同。在缓存丢失攻击下，自定义索引顺序的模式匹配算法所有节点都集中在1个层次内，不会出现缓存丢失率显著上升，因此，匹配性能趋于稳定。每个节点有一个失败列表，所有的失败转移状态都在1层内，使得攻击数据访问的节点主要位于1层，大大降低了缓存丢失率，提高了CPU性能。

附图说明

[0039] 此处所说明的附图用来提供对本申请的进一步理解，构成本申请的一部分，本申

请的示意性实施例及其说明用于解释本申请，并不构成对本申请的不当限定。在附图中：

[0040] 图1为本发明实施例所述的自动机构建示例图；

[0041] 图2为本发明实施例所述的自动机扫描流程图；

[0042] 图3为本发明实施例所述的自动机扫描示例图；

[0043] 图4为本发明实施例所述的窗口滑动法流程图；

[0044] 图5为本发明实施例所述的窗口和自定义位置顺序示例图。

具体实施方式

[0045] 为了使本申请实施例中的技术方案及优点更加清楚明白，以下结合附图对本申请的示例性实施例进行进一步详细的说明，显然，所描述的实施例仅是本申请的一部分实施例，而不是所有实施例的穷举。需要说明的是，在不冲突的情况下，本申请中的实施例及实施例中的特征可以相互组合。

[0046] 实施例1、参照图1至图5，说明本实施例，本实施例的一种适用于缓存丢失攻击的自定义位置顺序模式匹配方法，包括以下步骤：

[0047] 步骤一、构建自动机，按照自定义位置顺序构建自动机；自动机由GOTO表、FAIL表和OUTPUT表组成，GOTO表根据当前状态和当前字符记录下一个状态和扫描位置移动距离，FAIL表决定当GOTO表获得的下一个状态无效时返回到哪个状态和扫描位置移动距离，而OUTPUT表将命中的模式保存在一个状态中以及下一个状态。自动机构建时设置了两类窗口，分别为固定窗口和可变窗口。固定大小窗口 w_1，多个可变大小窗口 w_2，其中 w_1 大小为[1，最短模式长度-1]，w_2 大小为[2，w_1 第二个字符为首字符的最长模式尾字符位置]，w_2 有多个，数量为模式首字符集的大小。本算法对每个模式首字符、尾字符和 w_1 窗口尾部字符进行优先匹配，将不能命中的字符放到自动机的前几层节点。算法将所有模式按3个层次进行切分，每个层次包括不同自定义位置顺序，自动机构建顺序为1层、2层、3层，各层自定义位置顺序如下：

[0048] (1)1层：包括3个字符，w_1 首字符、w_1 尾字符、w_2 最长窗口尾字符；

[0049] (2)2层：w_1 去掉首尾2个字符的最大子串，并以逆序重新排序；

[0050] (3)3层：w_1 尾字符与最长 w_2 尾字符之间的最大子串。

[0051] 参照图5，自定义位置顺序，模式集{ADCAB，EBPCPCA，ECCADCP，BDCABPDA}最短长度为5，窗口为[1，4]，首字符A的最大长度为5，窗口为[2，5]，E的最大长度为7，窗口为[2，7]，B的最大长度为8，窗口为[2，8]。1层由所有模式的第1、4和尾字符组成，长度固定为3；2层由所有模式的第2、3字符倒序组成，长度固定为2；3层由所有模式的第5至倒数第二个字符顺序组成，每个模式的3层长度不固定。

[0052] 边的定义为*edge(chr，mov)*，*chr*表示当前字符，*mov*表示当前字符匹配后扫描指针移动距离。

[0053] 步骤一.一、创建根节点，编号为0

[0054] 步骤一.二、根据1层的第一个字符创建新节点，编号递增，每个模式新增1个节点；根节点与新增节点之间绘制一条边，GOTO表为边值*edge(chr，mov)*，*chr*为1层第一个字符，

*mov*为窗口滑动法计算的移动距离,FAIL表为根节点除了创建边的字符外,其它所有字符的edge值;

[0055] 所述窗口滑动法设置步骤如下:

[0056] 步骤一.二.一、设置每个模式首字符与窗口首字符对齐;

[0057] 步骤一.二.二、窗口中已扫描位置与每个模式对应位置比对字符,所有位置完全匹配成功,执行步骤一.二.三,匹配失败,执行步骤一.二.四;

[0058] 步骤一.二.三、找到了最佳模式,按自定义位置顺序返回模式的下一个待扫描位置,mov值为最近扫描字符位置到下一个待扫描位置的距离,算法结束;

[0059] 步骤一.二.四、窗口向左移动一个字符,若窗口尾部位置小于所有模式首字符位置,算法结束,否则,执行步骤一.二.二。

[0060] 所述mov计算方法如下:

[0061] 假设当前扫描位置为k,k表示与窗口起始位置的相对位置,L_{w_1}为窗口 w_1 尾字符位置,L_{w_2}为窗口 w_2 尾字符位置。本发明设计了窗口滑动法计算mov值,包括两步:第一步,查找符合已扫描位置的模式;第二步:计算查找到的模式第一个未被扫描位置,计算mov;

[0062] 第一步:假设模式集为 $P=\{P_1,P_2,...P_n\}$,窗口 w_1+w_2 中,已扫描字符和位置为 $P_{m_1}^{'},P_{m_2}^{'},....,P_{m_t}^{'}$。查找这样的模式 P_u,对于所有i,$1\le i<t$,满足 $P_{m_i}^{'}=P_{u_i}$ 且 $m_i-m_{i-1}=u_i-u_{i-1}$。

[0063] 第二步:模式 P_u,假设自定义位置顺序为 $\{a_1,a_2,...,a_q\}$,mov计算公式如下:

[0064] $$mov=\{a_p-k\mid p=\min\{p>a_j,Scan(P_{u_{a_j}})=1,1\le j<p\}\}$$

[0065] 式中,$Scan(P_{u_{a_j}})$ 表示字符 $P_{u_{a_j}}$ 是否被扫描过,1为已扫描,0为未扫描;k表示当前扫描位置;a_p 表示模式 P_u 的第一个未被扫描字符的位置。

[0066] 步骤一.三、根据1层的第二个字符创建新节点,编号递增,每个模式新增1个节点,每个模式当前节点与新增节点之间绘制一条边,GOTO表为边值*edge(chr,mov)*,*chr*为1层第二个字符,*mov*为窗口滑动法计算的移动距离,FAIL表为模式当前节点除了创建边的字符外,其它所有字符的*edge*值;

[0067] 步骤一.四、根据1层的第三个字符创建新节点,编号递增,每个模式新增1个节点,每个模式当前节点与新增节点之间绘制一条边,GOTO表为边值*edge(chr,mov)*,*chr*为1层第三个字符,*mov*为窗口滑动法计算的移动距离,FAIL表为模式当前节点除了创建边的字符外,其它所有字符的*edge*值;

[0068] 步骤一.五、根据2层的第一个字符创建新节点,编号递增,每个模式新增1个节点,每个模式当前节点与新增节点之间绘制一条边,GOTO表为边值*edge(chr,mov)*,*chr*为2层第一个字符,*mov*为窗口滑动法计算的移动距离,FAIL表为模式当前节点除了创建边的字符

外,其它所有字符的*edge*值;

[0069] 步骤一.六、每个模式当前节点移动到最新创建节点,重复步骤步骤一.五,直至2层的最后一个字符完成节点创建;

[0070] 步骤一.七、根据3层的第一个字符创建新节点,编号递增,每个模式新增1个节点,每个模式当前节点与新增节点之间绘制一条边,GOTO表为边值*edge(chr,mov)*,*chr*为3层第一个字符,*mov*为窗口滑动法计算的移动距离,FAIL表为模式当前节点除了创建边的字符外,其它所有字符的*edge*值;

[0071] 步骤一.八、每个模式当前节点移动到最新创建节点,重复步骤步骤一.七,直至3层的最后一个字符完成节点创建。

[0072] 参照图1,自动机构建结果:以{ADCAB, EBPCPCA, ECCADCP, BDCABPDA}为例展示自定义位置顺序算法自动机的构建结果,首先创建根节点,编号为S_0;根据4个模式1层的第一个字符{A, B, E}创建新节点,编号S_1, S_2, S_3。根节点S_0分别与3个新增节点之间绘制边,边值分别为*edge(A,3)*,*edge(B,3)*,*edge(E,3)*。根节点的失败列表为{(*,1),S_0}。根据4个模式1层的第二个字符{A, A, C,A}创建新节点,编号S_4, S_5, S_6, S_7。与S_4之间绘制边,边值为*edge(A,1)*,S_1的失败列表为$\{P_1, S_0\}$,其中P_1表示一个集,即不同字符对应的*mov*,{(B,-1) (C,-2) (D,-1) (E,3) (P,-2) (*,1)}。S_2和S_3的边和失败列表与S_1类似。剩下的节点构建与步骤(3)相似,不再赘述。树的每个分支对应一个模式,按自定义位置顺序以每个字符创建一个节点,边为一对值,第一个为字符,第二个为扫描后位置移动距离。每个节点有一个失败列表,所有的失败转移状态都在1层内,使得攻击数据访问的节点主要位于1层,大大降低了缓存丢失率,提高了CPU性能。

[0073] 步骤二、扫描指针指向待匹配数据首字符,自动机当前状态设置为根节点;

[0074] 步骤三、当前扫描字符与当前状态节点进行匹配;当前扫描字符与当前状态节点所有边值和失败指针中的*chr*匹配成功,执行步骤四,当前扫描字符与当前状态节点所有边值和失败指针中的*chr*匹配失败,若匹配失败指针中某一个值的*chr*成功,扫描指针根据其*mov*值移动相应距离,自动机根据其失败指针中记录的值跳转到下一个节点,重新扫描;参照图2自动机扫描流程图理解本步骤。

[0075] 步骤四、自动机沿着边跳转到下一个节点,如果新节点为模式自定义顺序的尾节点,即自动机的叶子节点,执行步骤五,否则,扫描指针根据边的*mov*值移动相应距离,如果移动后超出待匹配数据尾字符,扫描终止,否则重新扫描;

[0076] 步骤五、匹配一个模式,输出OUTPUT表中记录的模式,自动机当前状态跳转到当前节点记录的下一个节点,执行步骤三。

[0077] 参照图3,说明自动机扫描攻击样本过程,首先以{ADCAB, EBPCPCA, ECCADCP, BDCABPDA}为例构建攻击数据,攻击AC算法数据{ECCADC, BDCABPD},攻击WM算法数据{EBMCP},攻击SBOM算法数据{CDCAB},根据这些攻击数据生成一个攻击样本{CDCAB EBMCP ECCADC BDCABPD},图中边上的数字表示执行第几次扫描,S_t为自动机下一个状态节点。具

体地：

[0078] 首先状态机进入根节点S_0，此时，扫描第1个字符C，状态机匹配C失败，从失败列表中找到{(*,1)， S_0}，状态仍为S_0，扫描指针向右移动1个字符，当前字符为第2个字符D；

[0079] 其次，扫描第2个字符D，状态机匹配D失败，从失败列表中找到{(*,1)，S_0}，状态仍为S_0，扫描指针向右移动1个字符，当前字符为第3个字符C；

[0080] 其次，扫描第3个字符C，状态机匹配C失败，从失败列表中找到{(*,1)，S_0}，状态仍为S_0，扫描指针向右移动1个字符，当前字符为第4个字符A；

[0081] 其次，扫描第4个字符A，状态机匹配A成功，根据S_0边值edge(A,3)，扫描指针向右移动3个字符，当前字符为第7个字符B，自动机沿边跳转到状态S_1；

[0082] 其次，扫描第7个字符B，状态机匹配B失败，S_1的从失败列表中找到{(B,-1)，S_0}，扫描指向向左移动1个字符，当前字符为第6个字符E，自动机沿失败指针跳转到S_0；

[0083] 其次，扫描第6个字符E，状态机匹配E成功，根据S_0边值edge(E,3)，扫描指针向右移动3个字符，当前字符为第9个字符C，自动机沿边跳转到状态S_4；

[0084] 最后，后续步骤以此方法进行扫描，直至扫描指针超过待匹配字符串最后一个字符D。

[0085] 该示例中待扫描数据包括23个字符，自动机一共扫描了16次，占比69.6%，有8个字符被跳过，占34.8%。自动机节点跳转顺序

$\{S_0,S_0,S_0,S_1,S_2,S_3,S_6,S_0,S_0,S_3,S_7,S_0,S_0,S_0,S_2,S_5\}$，16个节点都在1层中，根节点$S_0$有8个，占比50%。可见，在缓存丢失攻击下，自定义顺序的模式匹配算法所有节点都集中在1个层次内，不会出现缓存丢失，因此，匹配性能稳定。

[0086] 显然，本领域的技术人员可以对本申请进行各种改动和变型而不脱离本申请的精神和范围。这样，倘若本申请的这些修改和变型属于本申请权利要求及其等同技术的范围之内，则本申请也意图包含这些改动和变型在内。

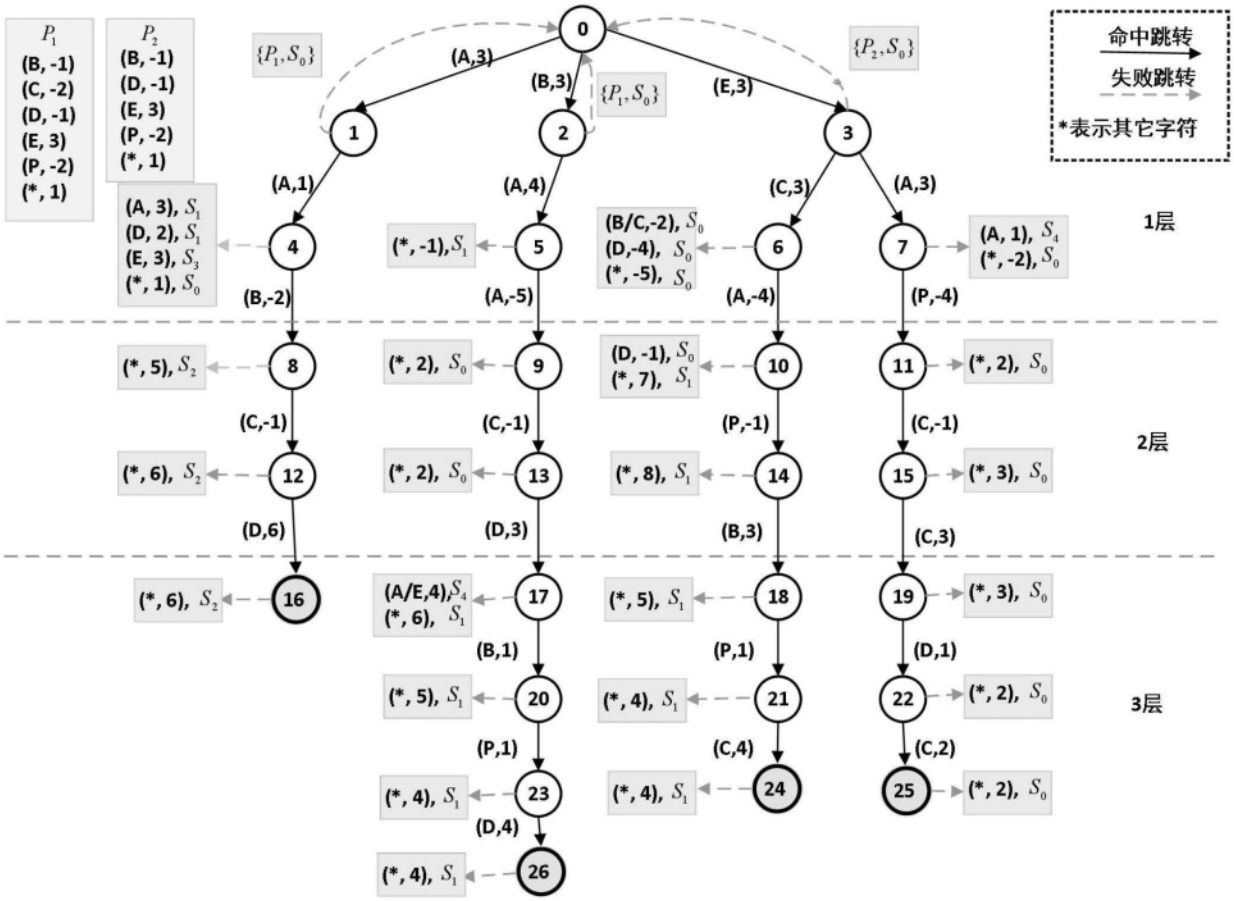

P_1
(B, -1)
(C, -2)
(D, -1)
(E, 3)
(P, -2)
(*, 1)
P_2
(B, -1)
(D, -1)
(E, 3)
(P, -2)
(*, 1)
$\{P_1, S_0\}$
$\{P_1, S_0\}$
$\{P_2, S_0\}$
命中跳转
失败跳转
*表示其它字符
0
1
2
3
(A,3)
(B,3)
(E,3)
(A,1)
(A,4)
(C,3)
(A,3)
4
5
6
7
(A, 3), S_1
(D, 2), S_1
(E, 3), S_3
(*, 1), S_0
(*, -1), S_1
(B/C,-2), S_0
(D,-4), S_0
(*, -5), S_0
(A, 1), S_4
(*, -2), S_0
1层
(B,-2)
(A,-5)
(A,-4)
(P,-4)
8
9
10
11
(*, 5), S_2
(*, 2), S_0
(D, -1), S_0
(*, 7), S_1
(*, 2), S_0
(C,-1)
(C,-1)
(P,-1)
(C,-1)
12
13
14
15
(*, 6), S_2
(*, 2), S_0
(*, 8), S_1
(*, 3), S_0
2层
(D,6)
(D,3)
(B,3)
(C,3)
16
17
18
19
(*, 6), S_2
(A/E,4), S_4
(*, 6), S_1
(*, 5), S_1
(*, 3), S_0
(B,1)
(P,1)
(D,1)
20
21
22
(*, 5), S_1
(*, 4), S_1
(*, 2), S_0
3层
(P,1)
(C,4)
(C,2)
23
24
25
(*, 4), S_1
(*, 4), S_1
(*, 2), S_0
(D,4)
26
(*, 4), S_1

图1

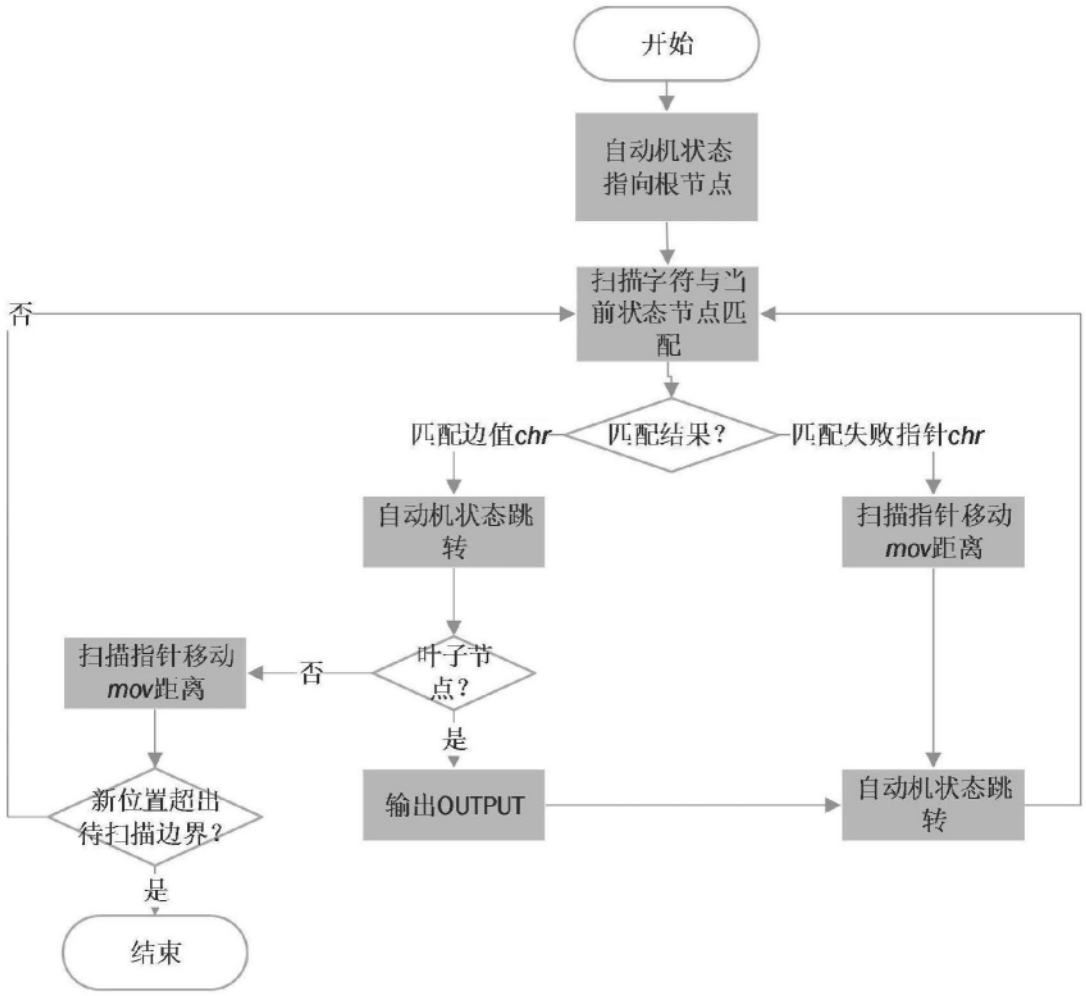

开始
自动机状态指向根节点
扫描字符与当前状态节点匹配
否
匹配边值chr
匹配结果？
匹配失败指针chr
自动机状态跳转
扫描指针移动mov距离
扫描指针移动mov距离
否
叶子节点？
是
新位置超出待扫描边界？
输出OUTPUT
自动机状态跳转
是
结束

图2

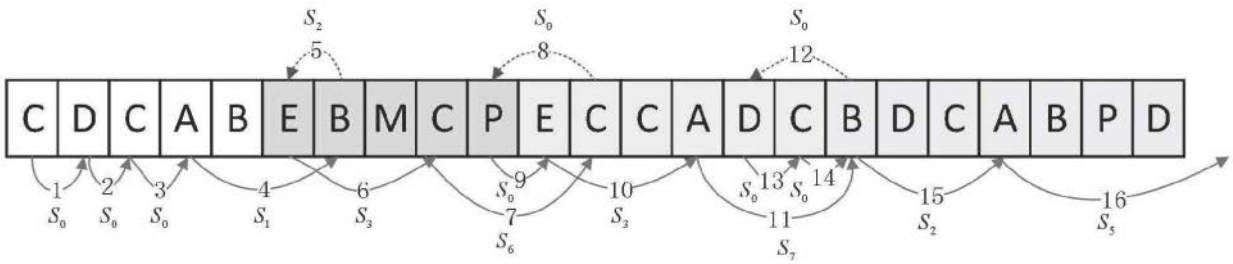

C D C A B E B M C P E C C A D C B D C A B P D

图3

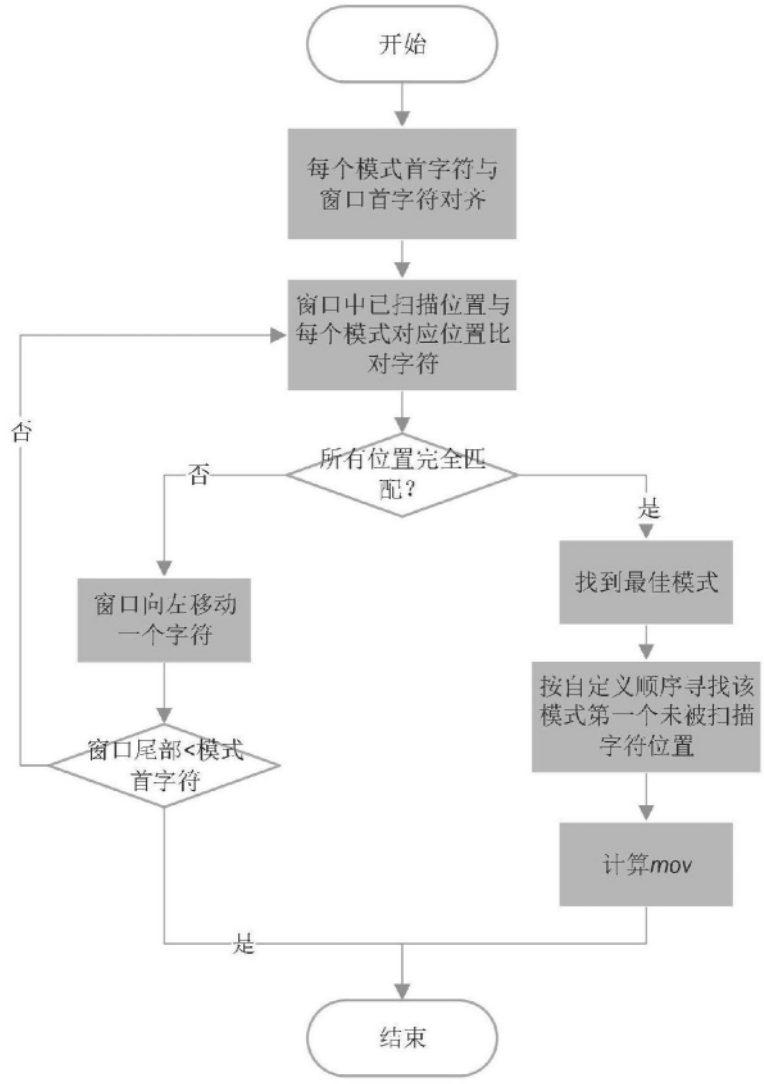

开始
每个模式首字符与窗口首字符对齐
窗口中已扫描位置与每个模式对应位置比对字符
所有位置完全匹配?
否
是
否
窗口向左移动一个字符
找到最佳模式
按自定义顺序寻找该模式第一个未被扫描字符位置
窗口尾部<模式首字符
计算mov
是
结束

图4

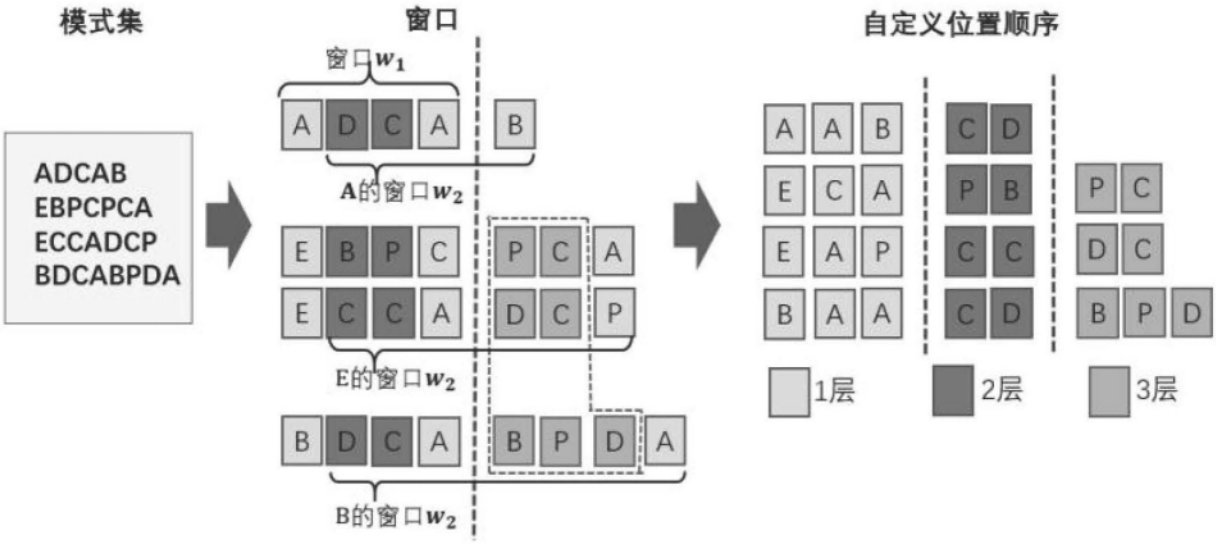

模式集
窗口
自定义位置顺序
ADCAB
EBPCPCA
ECCADCP
BDCABPDA
窗口w_1
A的窗口w_2
E的窗口w_2
B的窗口w_2
1层
2层
3层

图5