基于协议状态有限机的系统扫描检测算法

邓一贵[1,2] 王 康[2] 邱全杰[1,2]

(重庆大学计算机学院 重庆 400044)[1] (重庆大学信息与网络管理中心 重庆 400044)[2]

摘 要 针对现有扫描检测算法对隐蔽扫描、慢扫描无法识别的不足,提出了基于协议状态有限机的检测算法,该算法能更准确地检测出普通扫描,对隐蔽扫描、慢扫描等现有技术难以检测的扫描也有较好的检测效果。实验测试表明该算法能提高系统扫描检测性能,降低误报率和报警次数。

关键词 入侵检测,系统扫描检测,协议状态有限机

A Scanning Detection Algorithm Based on Finite Machine of Protocol Status

DENG Yi-Gui[1,2] WANG Kang[2] QIU Quan-Jie[1,2]

(College of Computer Science, Chongqing University, Chongqing 400044)[1]

(Network Center, Chongqing University, Chongqing 400044)[2]

Abstract In order to resolve the problem that current scanning detection algorithms can not recognize hidden scanning and slow scanning, a scanning detection algorithm based on finite machine of protocol status is proposed. It can more exactly detect common scanning, and has effect on hidden scanning and slow scanning which current scanning detection algorithms can not recognize. Experiment indicates that the algorithm can augment the performance of scanning detection, decline the rate of misinformation and alarming times.

Keywords Intrusion detection, System scanning detection, Finite machine of protocol status

1 引言

系统扫描往往成为入侵的前奏,对系统扫描活动的准确检测是入侵检测系统不可忽略的部分。传统的扫描检测主要使用统计的方法。Snort 是一个目前获得广泛应用并且开放源代码的轻量级网络入侵检测系统[1]。其扫描检测规则为"对于任意一个目标主机 H1,如果在某个时间段 M 内 H2 与 H1 建立的连接数超过了阈值 N,则认为 H2 对 H1 发动了端口扫描"。Wenke Lee 使用数据挖掘方法提取入侵检测特征[2],其用于检测扫描的规则和 Snort 类似。Raj Basu 等人使用基于时间窗口的方法来检测扫描[3],用基于连接窗口的方法检测慢扫描,其原理和 Wenke Lee 的方法基本相同,其缺点也类似。Emerald 系统为被保护网络中每个主机在正常情形下的通信建立模型[4],该模型为 1025 维向量,其中前 1024 个属性对应于 1～1024 端口,最后一个属性对应于所有大于 1024 的端口,属性值为对应端口发送和接收的报文数。将短期和长期的模型进行比较,如果偏差较大,则可判断为异常。Emerald 系统能够检测高强度的扫描,但仍无法检测慢扫描。Stamford 等人提出一种启发式的基于网络包头异常分析的端口扫描检测方法[5],通过对每个 IP 包头部的 4 个字段(源 IP 地址,源端口,目的 IP 地址,目的端口)进行统计,计算联合概率分布,从而可以计算出每个报文的异常值。对异常值超过阈值的报文进行相关性分析,最终确定是否发生扫描行为。但通过测试发现,多个参数的设置对检测结果有很大影响,如何确定参数值没有可行的指导原则,无法检测一些隐蔽的扫描行为,误报率较高。

这些方法的主要缺点是参数 M、N 难以确定,检测准确性较低,难以检测慢扫描、分布式扫描等多种隐蔽扫描,而且要维护每个连接的状态,检测系统易受 DoS 攻击。

传统的扫描检测方法简单地根据连接的统计信息界定扫描没有考虑协议报文状态间的关系,无法检测攻击者越来越多地使用的隐蔽扫描、慢扫描、分布式扫描,以及 DoS 掩护下的扫描。

2 协议有限状态机及网络连接链表

2.1 协议有限状态机及其数据结构表示

协议的正确报文序列是可以通过协议有限状态机来刻画的,它表示了网络连接的不同状态之间的关系,它可以检测一个连接的报文序列的状态关系和连接的完整性。一个协议有限状态机的实际例子就是 TCP 协议,其服务器端和客户端的协议有限状态机的图形表示如图 1 和图 2。

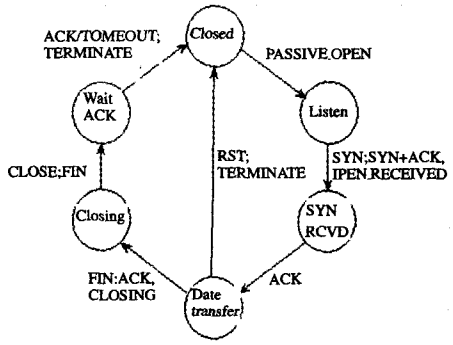

图 1 TCP 协议服务器端有限状态机

邓一贵 博士研究生,主要研究方向:计算机网络及信息安全;**王 康** 教授,硕士生导师,主要研究方向:Internet 网络技术,信息网络安全技术;**邱全杰** 硕士,主要究方向:Internet 网络技术,计算机网络信息安全。

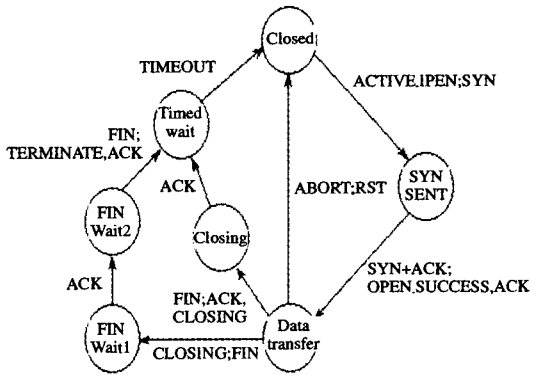

图 2　TCP 协议客户端有限状态机

协议用有限状态机表示有三个好处：首先，这种形式便于检查每种状态和事件的组合，找出可接收的状态和事件的组合，分辨出非法的状态和事件组合，识别对协议的扫描企图；其次，易于编程实现，可以用串表指示下一状态节点；最后，非常形象直观的协议描述，如 TCP 协议用图 1 和图 2 描述就非常直观，易于理解。

考虑到检测算法的实现的有效性以及灵活性，本文以多叉串表来表示协议转换模型，如图 3 所示。

图中 q_0 指向协议的起始节点，每个起始和中间链接节点包含四类信息：状态名称标识，转移到其他状态数目，转移到某个状态的条件（协议的标识位组合）以及每个标识位组合对应一个指向其他状态的指针。协议终止串接节点 finish 0 到 finish m 组成终止状态集合，只包含状态标识。

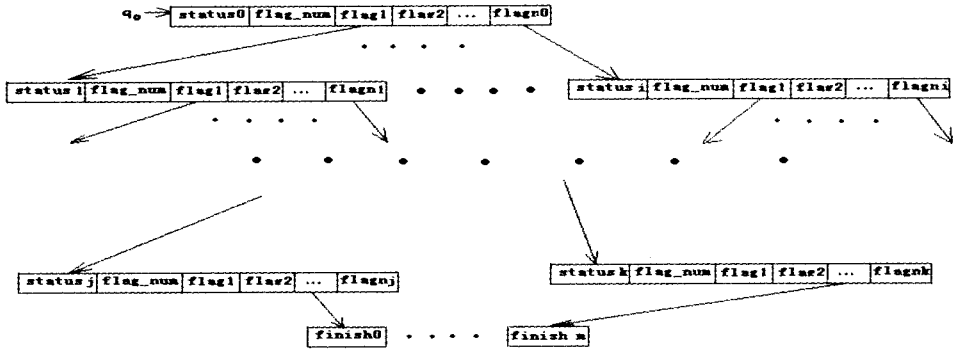

图 3　协议有限状态机的链表示意图

2.2　网络连接链表

为了对整个网络连接进行扫描检测，采用链表结构来组织网络连接信息（如图 4 所示）。

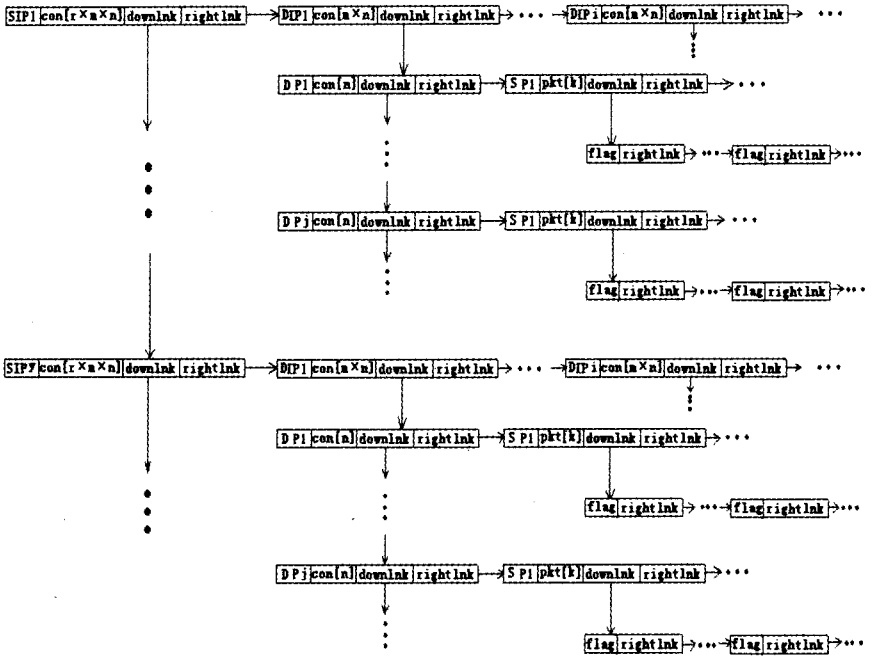

图 4　网络连接的链表存储结构

存储结构包含五类节点：源 IP 地址节点、目的 IP 地址节点、目的端口节点、源端口节点和报文节点。

不同的源 IP 地址节点构成主链表，每个源 IP 地址节点包含 SIP、con[$r\times m\times n$]、downlnk、rightlnk 字段，其中 SIP 指源 IP 地址，con[$r\times m\times n$]指与该源 IP 建立连接的情况数组、downlnk 指向下一个源 IP 地址链表、rightlnk 指向与该源 IP 地址建立连接的目的 IP 地址链表。

每个目的 IP 地址节点也包含 DIP、con[$m\times n$]、downlnk、rightlnk 字段一个指针，其中 DIP 指目的 IP 地址，con[$m\times n$]指与〈上级源 IP，该目的 IP〉相关连接的情况数组、downlnk

指向〈上级源 IP,该目的 IP〉相关的目的端口连接链表、rightlnk 指向〈上级源 IP,下一目的 IP〉连接链表。

每个目的端口节点是指与该节点的上级 IP 地址所对应的主机系统的目的端口的连接链表,它包含字段 DP、con[n]、downlnk 和 rightlnk,其中 DP 是目的端口号,数组 con[n]是与〈源 IP 地址,目的 IP 地址,目的端口号〉相关的各连接是否违反协议有限状态机的情况,其值为 1 表示正确连接,为 0 表示出错连接,downlnk 指向下一个与〈源 IP 地址,目的 IP 地址〉相关联的目的端口节点,rightlnk 指向与〈源 IP 地址,目的 IP 地址,目的端口号〉相关的源端口链表。

源端口节点有 SP、pkt[k]、downlnk 和 rightlnk 字段,其中 SP 是指源端口号,数组 pkt[k]的每个元素的值是指连接〈目的 IP 地址,源 IP 地址,目的端口,源端口〉的每个报文在协议状态有限机中的状态号,downlnk 指向报文链表,rightlnk 指向下一个源端口节点。

报文节点包括 pkt 和 rightlnk 字段,pkt 是指连接过程中每个报文根据时间先后次序所产生的报文标识位组合,rightlnk 指向下一个序号的报文节点。

3 基于协议状态有限自动机的系统扫描检测算法

为了对网络内所有连接进行系统扫描检测,需要对本节上两小节的定义的连接链表和协议状态有限自动机进行关联处理(对每个连接的报文状态按协议状态有限自动机进行检查):遍历每个与〈目的 IP 地址,源 IP 地址,目的端口,源端口〉的所有报文状态在相应协议状态有限自动机中进行检查,同时检查该连接是否完整。连接的第一个报文的状态必须属于协议状态有限自动机的起始状态,中间报文状态必须和其前一个报文的状态相同,最后一个报文的状态必须属于协议状态有限自动机的终止状态集合;符合上述条件则该连接为正常连接,否则为出错连接,将源地址的连接情况数组所对应的错误连接数加 1,同时记录出错报文的状态,在系统扫描检测规则中查找出错状态所对应的扫描类型名称。对每个连接的报文状态按协议状态有限自动机进行检查 Packet_Status_Check_in_Protocol_Status_Machine 具体描述如下:

```
函数 Packet_Status_Check_in_Protocol_Status_Machine(link ptrProtocolStatusMachinePointer, link ptrConnectionPointer)
输入:协议状态有限自动机链表起始指针,网络连接链表起始指针
输出:报文状态检查后的网络连接是否符合协议规则的网络连接链表
BEGIN
    ptrPositionSourceIPNode= ptrConnectionPointer;/* 初始化源 IP 地址节点工作位置指针 */
    ptrPositionDestinationIPNode = ptrPositionSourceIPNode->rightlnk;/* 初始化目的 IP 地址节点工作位置指针 */
    ptrPositionDestinationPortNode= ptrPositionDestinationIPNode->downlnk;/* 初始化目的端口节点工作位置指针 */
    ptrPositionSourcePortNode = ptrPositionDestinationPortNode->rightlnk;/* 初始化源端口节点工作位置指针 */
    ptrPositionPacketNode = ptrPositionSourcePortNode ->downlnk;/* 初始化某连接中报文节点工作位置指针 */
    ptrPositionProtocolNode = ptrProtocolStatusMachinePointer;/* 初始化协议状态有限自动机状态节点工作位置指针 */
    ptrPreviousPositionProtocolNode = ptrPositionProtocolNode; /* 初始化协议状态有限自动机前一状态节点工作位置指针 */
    packetStatusError=false; /* 初始化报文状态出错为假 */
    tempR=0; /* 初始化连接情况数组第一维下标位置变量 */
    tempM=0; /* 初始化连接情况数组第二维下标位置变量 */
    tempN=0; /* 初始化连接情况数组第三维下标位置变量 */
    tempK=0; /* 初始化报文状态情况数组下标位置变量 */
    for i=0 to k ptrPositionSourcePortNode->pkt[i]=0; /* 将报文状态情况数组中表示每个状态是否出错的值设为正常,用 0 表示,如果为 1 则表示出错 */
    for i=0 to ptrProtocolStatusMachinePointer->flag_num do
    /* 检查连接的第一个报文是否为协议起始标志位组合 */
    if ptrPositionPacketNode->flag = ptrProtocolStatusMachinePointer->flag
      then
        break;
        packetStatusError=true;
      else
        continue
      end if
      if packetStatusError=true then ptrPositionSourcePortNode->pkt[0]=1;
    /* 检查连接的后继报文状态是否符合协议规定 */
      do while (ptrPositionPacketNode! =null | packetStatusError==false){
        if ptrPositionPacketNode->statusCode== ptrPreviousPositionPacketNode->statusCode
        {
        ptrPositionPacketNode = ptrPositionPacketNode->rightlnk; /* 如果当前报文等于协议状态有限自动机的前一状态,则继续下一个报文状态检查 */
          }
          Else {
            packetStatusError=truse; /* 如果当前报文不等于协议状态有限自动机的前一状态,则报文出错,即为非规则报文,将报文状态出错标志设为真 */
            ptrPositionSourcePortNode->error=1; /* 将报文状态情况数组中该报文的出错标志设为 1,表示出错,违反了协议规则 */
            ptrPositionDestinationPortNode->con[tempN]=ptrPositionDestinationPortNode-> con[tempN]+1;
            ptrPositionDestinationIPNode-> con [tempM, tempN]= ptrPositionSourceIPNode-> con [tempM, tempN]+1;
            ptrPositionSourceIPNode -> con[tempR, tempM, tempN]= ptrPositionSourceIPNode -> con [tempR, tempM, tempN]+1;
            }
}
END
```

在对每个连接的报文状态检查完毕之后,就得到了每个连接的是否正常,进而可以得到某个源 IP 对应多个 IP 地址的连接是否正常,依据这些信息就可以对是否存在系统扫描进行判断了。基于这样的思路,系统扫描检测 Scanning_Attack_Detection 算法描述如下:

```
函数 String Array Scanning_Attack_Detection(link ptrConnectionPointer, int NumberOfConnectionIP-ErrorIsScanningAttack)
输入:报文状态检查后的网络连接链表起始指针
输出:包含存在扫描的 IP 地址数组
BEGIN
    String Scanning_Attack_IP_List[R];
    ScanningAttackExist=false;
    For i to R do Scanning_Attack_IP_List[i]=''; /* 初始化扫描攻击 IP 地址清单为空 */
    ptrPositionSourceIPNode=ptrConnectionPointer;/* 初始化源 IP 地址节点工作位置指针 */
    do while (ptrPositionSourceIPNode->downlink )! =null;
      for i=0 to R-1 do
      OneDestinationIPConnectionError=0;
      NumberOfDestinationIPConnectionError=0;
      for j=0 to M-1 do
        for k=0 to N-1 do
          if ptrPositionSourceIPNode-> con [i, j, k] = 1 then
            OneDestinationIPConnectionError=1;
            Break;
          end if
        end for;
        if OneDestinationIPConnectionError=1 then
          NumberOfDestinationIPConnectionError = NumberOfDestinationIPConnectionError+1;
          Break;
        end if
      end for
    end for
    if NumberOfDestinationIPConnectionError > NumberOfConnectionIPErrorIsScanningAttack then
      Scanning_Attack_IP_List [i] = ptrPositionSourceIPNode->SourceIPaddress;
      ScanningAttackExist=true;
    End if
    ptrPositionSourceIPNode = ptrPositionSourceIPNode->downlink;
  end while
  if ScanningAttackExist=true then
    return Scanning_Attack_IP_List;
  else
    return "Scanning Attack does not exist!";
END
```

4 算法测试及结果

算法的测试数据采用DARPA入侵检测系统评测数据集中第四、第五周的原始数据，测试的结果与Snort port-scan检测结果进行对比(port-scan检测规则设为“如果在3秒内连接数大于6，则报警”)，结果见表1。

表1 性能比较

系统扫描检测算法	检测率	误报率	报警连接数
Scanning Attack Detection	97%	0.07%	870
Snort port-scan	63%	0.15%	11180

由表1可知，基于协议状态有限自动机的系统扫描检测算法与Snort port-scan相比，检测率高，误报率低，报警次数大大减少。这是因为采用基于协议状态有限自动机的系统扫描检测算法对报文序列的状态根据协议状态机进行了检查，能够检测出慢扫描和隐蔽扫描，从而提高了检测率，同时降低了误报率。考虑到实际使用中可只对系统开启的服务端口检测(对连接使用所提出的算法进行合法性检查，如果非法则判断为扫描，对合法的连接的频繁程度采用阈值判别法界定是否发生扫描行为)，对关闭的端口建立连接的企图都界定为扫描，但只报警一次，在测试中对报警进行了合并。因为对外提供服务端口默认为0～1024，测试对0～1024端口进行了检测并报警，同时考虑到有的系统会使用大于1024的端口提供服务，对大于1024的端口以port mod 1024进行检测并报警合并，所以输出的报警数也大为减少。

总结 本文分析了现有检测算法的缺陷，针对现有扫描检测算法对隐蔽扫描、慢扫描无法识别的不足，提出了基于协议状态有限机的检测算法，算法通过对报文序列的状态在协议状态有限机中前后关系进行检查来识别连接的合法性，然后检查非法连接和端口的访问频率(采用现有的阈值判别法)来识别系统扫描。实验测试表明该算法能明显提高系统扫描检测性能，降低误报率和报警次数。

参 考 文 献

1 http://www.snort.org/;

2 Lee W, Nimbalkar R A, Yee K K. A data mining and CIDF based approach for detecting novel and distributed intrusions. In: Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection (RAID 200), Toulouse, France, Oct. 2000

3 Basu R, Cunninghuam R K, Webster S E, et al. Detecting Low-Profile Probes and Novel Denial-of-Service Attacks. In: Proceedings of the 2001 IEEE workshop on information assurance and security, June 2001

4 Porras P A, Neumann P G, Merald E. Event monitoring enabling responses to anomalous live disturbances. In: National Information Systems Security Conference, Baltimore MD, October 1997

5 Stamford S, Hoagland J, Mcalerney J. Practical Automated. Detection of Stealthy Port scans. ACM CCS IDS, Workshop, Athens, Greece, 2000

(上接第105页)

议进行了分析，认为在该协议的最后一则消息中，不能保证A发送过现时值N_b。他们给出了改进协议V4：

(1)$A\rightarrow B:A$

(2)$B\rightarrow A:B,N_b$

(3)$A\rightarrow B:\{B,N_b\}_{Kas}$

(4)$B\rightarrow S:A,\{B,N_b\}_{Kas}$

(5)$S\rightarrow B:\{A,B,N_b\}_{Kbs}$

协议V4避免了原Woo-Lam协议中的两个重放攻击，但是本文发现，该协议中存在另一个重放攻击：

攻击5：

(1)$P(A)\rightarrow B:A$

(2)$B\rightarrow P(A):B,N_b$

(1′)$B\rightarrow P(C):B$

(2′)$P(C)\rightarrow B:A,B,N_b$

(3′)$B\rightarrow P(C):\{A,B,N_b\}_{Kbs}$

(3)$P(A)\rightarrow B:X$

(4)$B\rightarrow P(S):A,X$

(5)$P(S)\rightarrow B:\{A,B,N_b\}_{Kbs}$

在协议V2中，“(3)A→B:$\{B,N_b\}_{Kas}$”消除了B和S可以在不同协议回合中生成相同消息项$\{A,B,N_b\}_{Kbs}$的漏洞，故避免了上述攻击：B在(3′)中向P(C)发送“$\{C,A,B,N_b\}_{Kbs}$”，使攻击者无法得到在(5)中执行重放攻击所需的“$\{A,B,N_b\}_{Kbs}$”。

小结 Abadi和Buttyan等人分别使用非形式化设计原则和BSW逻辑对Woo-Lam协议进行了改进，但是得到的两个新协议V3和V4中仍存在缺陷：V3的“(3)A→B:$\{N_b\}_{Kas}$”中缺少对主体B进行身份确认的标识；V4中，过多的主体标识不仅不能为协议带来更高的安全性保证，反而导致协议受到新的攻击。而使用多重认证测试方法设计出的协议V2，避免了上述缺陷，可以抵抗Abadi和Buttyan等人分别对原协议改进后存在的两个重放攻击。

参 考 文 献

1 Guttman J D, F'abrega F J T. Authentication tests[C]. In: Proceedings of the 2000 IEEE Symposium on Security and Privacy, Los Alamitos, 2000. 96～109

2 Guttman J D, F'abrega F J T. Authentication tests and the structure of bundles[J]. Theoretical Computer Science, 2002, 283(2): 333～380

3 Guttman J D. Security protocol design via authentication tests [C]. In: Proceedings of the 2002 IEEE Computer Security Foundations Workshop, Los Alamitos, 2002. 92～103

4 Perrig A, Song D X. Looking for diamonds in the desert-extending automatic protocol generation to three-party authentication and key agreement[C]. In: Proceedings of the 2000 IEEE Computer Security Foundations Workshop, Los Alamitos, 2000. 64～76

5 Choi Hyun-Jin. Security protocol design by composition [D]. Cambridge, United Kingdom: University of Cambridge, 2006

6 Debbabi M, Mejri M, Tawbi N, et al. A new algorithm for the automatic verification of authentication protocols: From specifications to flaws and attack scenarios[C]. In: DIMACS Workshop on Design and Formal Verifictaion of Security Protocols, 1997

7 Clark J, Jacob J. A survey of authentication protocol literature: Version 1.0[EB/OL]. http://www-users.cs.york.ac.uk/～jac/ under the link \Security Protocols Review, 1997

8 Abadi M, Needham R. Prudent Engineering Practice for Cryptographic Protocols[C]. In: Proceedings of the 1994 IEEE Computer Society Symposium on Security and Privacy, Los Alamitos, 1994. 122～136

9 Buttyan L, Staamann S, Wilhelm U. A simple logic for authentication protocol design[C]. In: Proceedings of the 1998 IEEE Computer Security Foundations Workshop, Los Alamitos, 1998. 153～162