第38卷第3期 中国科学技术大学学报 Vol. 38, No. 3
2008年3月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Mar. 2008

文章编号:0253-2778(2008)03-0230-05

分布式网络攻击测试系统中的代理选择算法

陈伟琳[1,2],周 颢[1,2],赵保华[1,2]

(1. 中国科学技术大学计算机科学与技术系,安徽合肥 230027;2. 安徽省计算与通讯软件重点实验室,安徽合肥 230027)

摘要:对网络设备的网络攻击测试有助于确定设备的稳定性和安全性,从而降低网络运行风险. 首先改进了分布式网络攻击测试系统,该系统由一个主控结点和多个代理结点组成,能适应多种网络拓扑结构,能对 Internet 网络中的目标设备进行各种常规的网络攻击和分布式攻击测试,并支持新攻击套件开发. 然后讨论了该系统在线模式下特有的代理选择问题,提出了一种合理的代理选择算法,有利于保证结点通信,分散测试流量.

关键词:分布式网络攻击测试系统;代理选择

中图分类号:TP393 **文献标识码**:A

An agent selection algorithm in distributed network attack testing system

CHEN Wei-lin[1,2], ZHOU Hao[1,2], ZHAO Bao-hua[1,2]

(1. *Department of Computer Science and Technology, University of Science and Technology of China, Hefei* 230027, *China*;
2. *Anhui Province Key Laboratory of Software in Computing and Communication, Hefei* 230027, *China*)

Abstract: Network attack testing on network equipment conduces to ensuring the equipment's stability and security during attacks, and reducing the risks that the network faces. Firstly a distributed network attack testing system we implemented is introduced. The system consists of a control node and several agent nodes, and is adapted to many different network topologies. It can test the target-equipment with diversified normal attacks and distributed attacks and also supports new attack suite development. Then the special problem of agent selection under the online testing mode of the system is discussed, and a proper agent selection algorithm is proposed. The algorithm can facilitate nodes communication fluency and testing stream dispersion.

Key words: distributed network attack testing system; agent selection

0 引言

近年来利用网络中存在的各种漏洞和安全缺陷对系统和资源进行的攻击越来越多,其中更有一部分针对路由器、交换机等网络通信关键设备,其后果往往非常严重. 因为路由器和交换机通常处于企业的基础设施内部,受监视器和安全政策的保护相对薄弱,一旦受到入侵,就可以被用作扫描行动、欺骗

收稿日期:2006-12-28;**修回日期**:2007-06-08

基金项目:国家自然科学基金重大研究计划(90104010),国家自然科学基金(60602016),国家重点基础研究发展(973)计划(2003CB314801),中国高技术研究发展(863)计划(2007AA01Z428),华为基金(YJCB2006044TS)资助.

作者简介:陈伟琳,女,1982年生,博士生. 研究方向:通信软件测试理论与方法. E-mail:wlchen@mail.ustc.edu.cn

通讯作者:赵保华,教授. E-mail:bhzhao@ustc.edu.cn

连接的平台，并作为发动攻击的一块跳板，进一步威胁整个网络的安全.

如果能在网络设备投入使用前对其抵御各种已知网络攻击的能力进行测试，将有助于了解可能存在的漏洞，采取相应的防御措施. 这种攻击测试本质上是一种渗透测试(penetration testing)[1]. 当前对网络设备进行攻击测试的一种主要方法是利用现有的网络攻击工具，如用于实施 DoS 攻击的 Trinoo 和 TFN2K，以及著名的扫描工具 Nmap 等. 另一种方法是使用专用的硬件测试设备. 攻击工具一般都只能涵盖部分攻击方式，且针对特定的网络攻击方式需要特定的实现，而对于新的网络攻击方式，则需要重新设计. 专用测试设备则代价很高，灵活性也不够. 同时，由于已有的网络攻击种类繁多，攻击目的、实施过程、结果和复杂度各不相同，且各个测试网络的结构也存在差别，攻击测试需要一个能支持多种攻击种类、适应多种网络拓扑结构的测试系统.

本文改进的分布式网络攻击测试系统正是出于上述目的而构建的. 该系统是一个由运行于网络上的多个独立结点组成的分布式环境，包括一个主控结点和多个代理结点，根据网络攻击原理用 TCL 语言[2]编写的测试套件在平台上分布式运行，从而模拟现实中的网络攻击，并对相关对象进行攻击测试. 系统基于普通微机就可运行，能对 Internet 网络中的目标设备进行各种网络攻击测试，尤其适于实现分布式和多角色参与的攻击.

在分布式网络攻击测试系统中，代理结点的分布情况对攻击测试的过程和结果都是有影响的. 如何根据网络状况部署代理结点，以更好地利用分布式能力获取更真实的测试结果，是该系统的一个特有问题. 该系统下，本文提出一个代理选择算法，在攻击之前先进行合理选择，以避免测试流量过于集中，减小对测试网络的不必要冲击，有利于得到更准确的测试结果.

1　分布式网络攻击测试系统

分布式网络攻击测试系统的逻辑结构如图 1. 此系统包含网络攻击测试套件开发系统和网络攻击测试执行系统两个子系统. 其中测试套件开发子系统只位于主控结点中，执行子系统由主控结点和代理结点共同组成. 系统内置有多种协议类攻击的测试套件，可以直接使用.

根据测试组网环境的差异，该系统定义了三种

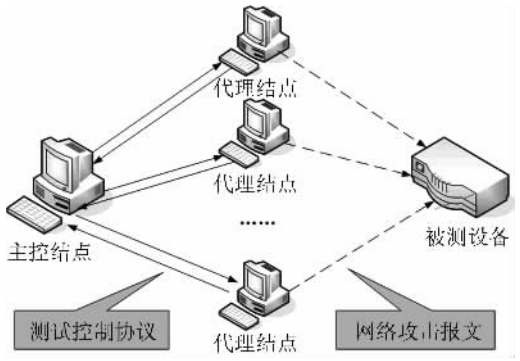

图 1　分布式网络攻击测试平台逻辑结构图

Fig. 1　Distributed network attack test platform logical structure

测试模式.

单机模式：主控进程和代理进程在同一台主机上运行. 主控进程直接控制代理进程进行攻击测试.

专用组网模式：一个主控结点和多个代理结点通过以太网互联成一个测试专用网络. 主控进程运行于主控结点，多个代理进程运行于多个代理结点. 被测设备也处在该专用网络内，此外网络内没有其他设备.

在线模式：测试环境就是测试目标所在的实际环境. 若测试目标是整个网络本身，则只能使用该模式. 主控结点和代理结点都分布在 Internet 中，通过 TCP/IP 协议实现互联. 该模式下得到的测试结果相对最为客观可信，但在这种模式下网络结构复杂多变，主控结点和代理结点之间的同步较困难，测试对其他无关设备和主机的影响不可忽视，不确定因素很多.

基于此测试系统，网络攻击测试过程可按如下步骤进行.

(Ⅰ) 如果当前攻击方式在系统内置的测试套件中有对应项，则直接选择内置测试套件，进行适当参数配置即可；如果不存在对应项，则先在测试开发子系统中使用 TCL 语言编写攻击测试套件，描述该特定攻击的前提、过程以及可能观察到的结果. 开发子系统提供了一组底层协议的 TCL 扩展命令，方便用户编写测试套件操纵协议栈.

(Ⅱ) 主控结点将测试套件分布到各代理结点，并控制各代理结点执行攻击测试套件，并从各代理结点实时回收测试数据和执行结果，此过程由测试控制协议支持. 测试控制协议定义了控制命令报文和数据报文的格式，同时也支持各代理间的同步和协作操作.

(Ⅲ) 主控结点检查被测目标的当前状态,结合在测试中获取的测试结果和相关数据,判定被测目标抵御当前攻击的能力,生成测试报告.

在已知的网络攻击测试系统中,文献[3]仅仅提出了一种平台通用模型,文献[4]中基于 APC 的测试模型则是针对 IDS 的测试,对网络设备的测试有不适应性. 而思博伦开发的 ThreatEx 安全测试方案[5]需要专用的测试设备,代价高昂. 相比较而言,本文的分布式网络攻击测试系统具有组网灵活、测试代价低等优点,并且使用 TCL 语言编写测试脚本是协议一致性测试中的成熟技术,易学易用. 系统提供的 TCL 扩展命令也使得测试人员能方便地操纵路由协议等底层协议和对被测设备进行自动化配置,利于对路由器和交换机等设备进行测试. 同时系统也允许测试人员根据自己的理解实现各种新的可能攻击,有良好的扩展性.

2 在线模式下的代理选择

在分布式攻击测试系统的在线模式下,可能存在许多可作代理结点的主机,且它们所处的位置、网络情况互不相同. 网络中同时也存在许多其他设备和主机在运行. 如果到被测设备的路径上的其他设备先于被测设备出现错误或崩溃,这首先可能影响到其他的不相关通信流量,其次也可能造成对测试结果的误判. 同时网络攻击测试不同于真正的网络攻击,攻击者不需要考虑后果,只要通信受到影响,攻击的目的就算达到了. 测试者必须考虑网络对测试后果的承受能力,尽量减少不必要的冲击. 因此,怎样根据实际网络的情况来选择最有利于测试的主机作为代理,是在线测试模式下不可回避的特有问题.

2.1 网络信息获取

在线模式下实际网络环境可能十分复杂. 可以期待网络管理者能提供一些已知的主机给我们做结点. 这些主机满足主控结点和代理结点的硬件要求,具有独立的 IP 地址,且不处于防火墙后,各已知主机与被测设备之间都是连通的. 如果网络管理者不能给出该网络的具体结构的详细信息,那么我们就只能自己主动获取一些网络的相关信息.

假定已知有主控结点 C,主机集合 $A'=\{a_1,a_2,\cdots,a_n\}$,被测设备 I. 我们需要知道 C 到各个已知主机 A_i 的时延和路径,以及各 A_i 到 I 的路径.

本系统并不需要特别精确的网络时延数值,大致估计即可. 获得网络时延的方法有多种[6]. 如利用 ICMP 协议、UDP 协议或 TCP 协议实现. 以 ICMP 实现为例,C 向各个 A_i 发送 ICMP echo 报文,然后接收返回的 ICMP echo reply 报文,计算经历的时间再除以 2 就能得到大致的时延 t_i. 如果某个 A_i 或到 A_i 途中的某个设备禁止响应 ICMP echo 报文,则 t_i 会被设为∞. 获得各 A_i 的时延后,剔除那些超过允许值 $t_{\max}$的 A_i,得到集合 A''.

测试网络的相关路径可以使用多种网络拓扑发现方法[7]得到. 例如,若测试网络支持 SNMP 协议[8],则可以向网管站发送 SNMP 查询请求,利用获取的相关路由表项和网络信息勾画网络拓扑结构. 其缺点在于存在一些设备不支持 SNMP,并且构造拓扑结构的算法相对复杂. 另一种常用的方法是从 C 向各个 A_i 进行 Tracert 扫描[9],能得到以 C 为根的树形结构. Tracert 扫描的缺点是,它要求涉及的所有路由器 R_i 在收到 TTL 为 1 的报文时都返回 ICMP 超时报文,但这并非任何时候都能保证. 其他方法还包括 Ping 方法、DNS 搜索等. 综合使用以上方法可以分析得到从 C 到各个 A_i 以及从各个 A_i 到 I 的路径.

如果 C 到某个 A_j 的通信路径经过 I,则此 A_j 不适合作为攻击代理,因为 I 受攻击影响出错的话就会直接影响 A_j 和 C 之间的通信. 从 A'中删去所有这样的 A_j,得到最终的待选主机集合 A,从 C 到各个 $A_i\in A$ 的通信路径集合 P,和从各个 $A_i\in A$ 到 I 的通信路径集合 Q.

需要说明的是,由于本文算法的目的是在攻击前对代理预先进行选择,因此考虑的各网络参数均是正常状态的. 在攻击中,网络相关参数的动态变化不在本文的考虑范围内.

2.2 通信路径图生成

在执行选择算法之前先构造通信路径有向图 G. G 的顶点集合 $V=C\cup A\cup R\cup I$,R 表示路由器集合;C,A,I 如 2.1 节所述. G 的边集 T,初始为 $\emptyset$.

每个 R_i 有四个属性值,初始均为 0.

used:表示在主控节点 C 和各待选主机 A_i 的通信路径中被使用次数;

attackused:表示从各个待选主机 A_i 到被测设备 I 的攻击路径上被使用的次数;

selected:表示在实际选取的代理路径上被使用的次数;

finished:表示从 R_i 出发的搜索是否已无必要.

则路径图生成算法伪码如下:

算法 2.1

```
输入:C,I,待选主机集合 A={A1,…,Ak},通信路径集合
      P={p1,…,p2},Q={q1,…,q2}
输出:通信路径图 G
将 I 加入 G,I. selected=0;//I 的 selected 值记录着算法中已选择的代理数
将 A 中所有 Ai 加入 G,Ai. selected=0;//当前没有代理被选中
while (Q 不为空)
{//根据从各 Ai 到 I 的路径向 G 中添加结点或修改结点属性值,向 T 中添加边
  对每个 qi∈Q,qi=AiRi1Ri2…RimI,考察 qi 中的每个 Rij
  {
    if Rij∈G,Rij. attackused++
    else {Rij加入 G,Rij. attackused=1,Rij. selected=0,Rij.
    finished=0,Rij. used=0}
  }
  添加有向边(I,Rim),…,(Ri2,Ri1)(Ri1,Ai)到 T
  从 Q 删除 qi
}
while P 不空
{//根据从 C 到各个 Ai 的路径修改某些 R 结点的属性值.
  对每个 pi∈P,pi=CRi1Ri2…RinAi,考察 pi 中的每个 Rij
  {
    if Rij∈G,Rij. used++;
    if Rij. used>Pmax,Rij. selected=1;/* 如果有超过 Pmax
                条通信路径都经过同一个 Rij,为推迟 Rij
                被选中,令 Rij. selected=1(也可以是一个
                适当的正整数,其大小取决于该路由器在
                通信中的重要性) */
  }
}
```

关于算法 2.1 我们有:

命题 2.1　图 G 中,从 I 出发的任何一条路径必终止于一个可选代理结点.

证明　算法 2.1 中只有第一个 while 循环才向 G 中添加结点. 从这个 while 循环可知,任何一个路由器结点 R_i 被加入 G,它必然处在从某个待选主机 A_i 到 I 的路径 p_i 上. 由于添加有向边时是按照 p_i 反向进行的,这使得最终在 G 中,从 I 出发的任何一条路径都必然终结于某个待选主机结点.

2.3　代理选择

代理选择算法的主要考虑有两个. 首先,对于结点通信中涉及的路由器,流经的攻击测试流量应尽量少,以免影响主控结点和各代理结点之间的通信;其次,考虑任意两个不同代理 A_i 和 A_j 到 I 的攻击路径 q_i 和 q_j,如果二者存在交集,则表明从 A_i 到 I 和从 A_j 到 I 的测试流量会经过一个或若干个相同的中间路由器. 当测试流量较大或测试数据有破坏性时,若攻击对其中的某个中间路由器先于 I 产生了效果,则通过该中间路由器的流量都会受到影响. 这造成了测试的一个不确定因素,可能影响系统对测试结果的正确判定,也可能影响其他通信. 因此,在选择代理时,应使测试流量尽量分散.

算法采用贪心策略[10],从 I 开始搜索图 G. 在选择下一步时,先优先选择没有被选择经过或经过次数少的结点(selected 值最小);其次优先选择对通信影响小的结点(used 值最小);再考虑选择利于分散测试流量的结点(attackused 值最小). 如果搜索到一个可选代理 A_i,则标记并递归返回,沿途修改各个经过的 R 结点的相关参数,并删除那些已经搜索完毕的结点;再基于已经改变了的图 G 重新进行新一轮代理搜索过程,直到选出的代理数量达到要求,或所有的可选代理都被选择.

算法 2.2

```
输入:图 G,需要选取的代理数 Anum
输出:选取的代理集合 SA
Proc searchforagent (p:pointer)
{/* 指针 p 指向当前需要做分支选择的结点,从 p 出发的所有边的终点的集合为 Q,
    Q 中的 R 结点集合为 QR,Q 中的主机结点集合为 QA */
  if (QA 中存在 q 未被选择)
  {
    标记 q 为已选择;q 加入 SA;
    从 p. QP 中删除 q;
    if (p. QR=Φ 且 p. QA=Φ) p. finished=1;
  } else if (QA 中不存在空闲的代理结点,但 QR 非空)
  {
    先选取 QR 中 selected 值最小的结点 qsmin;若这样的结点有多个,则选取其中 used 值最小的结点 qumin;若这样的结点仍有多个,则选取其中 attackused 值最小的结点 qamin;
  searchforagent(qxmin);//递归调用,这里的 qxmin 指上述选择中选出的结点
  p. selected++;
  if (qxmin. finished) 从 p. QR 中删除 qxmin;
  } else error;
}
agent_select (Anum:integer)
{
  ifAnum<A. size) then
```

```
{//需要的代理数小于代理集合大小
  for {I. selected<Anum} do
      searchforagent (I);
  返回集合 SA;
} else  返回整个集合 A;
}
```

关于算法 2.2,只需证明在需求代理数小于可选代理数时,算法能找到所要求个数的代理.

命题 2.2 searchforagent 过程递归执行结束后,命题 2.1 对新图 G' 仍然成立.

证明 假设在新图 G' 中出现某个路由器结点 R_j,存在某条 I 出发的路径 q 终止于 R_j. 则由算法 2.2 有 R_j. finished = = true. 而 finished 值为 true 的结点 R_j,在返回它的上层调用时,会被上层调用从 p. QR 删除,即它根本不可能出现在递归结束后的图中,导出矛盾,这样的 R_j 不存在. 因此,searchforagent 每次递归结束后,命题 2.1 对新图 G' 仍然成立.

定理 2.3 当需求代理数小于可选代理数时,算法 2.2 总能返回所需求数量的代理.

证明 由命题 2.2 知,searchforagent 过程的执行始终保证命题 2.1 对图 G 成立. 而 for 循环中的每一次搜索相当于重新建立从 I 出发的一条路径,则由命题 2.1 知,这样的路径必然终止于一个代理结点,也即每次搜索必然找到一个代理结点,即证.

2.4 算法示例

假设用 2.1 节中提到的方法得到了实际网络结构如图 2 所示. $A_1 \sim A_9$ 是网络管理员提供给我们的待选主机,C 是主控结点. $R_1 \sim R_7$ 是中间路由器,I 是被测设备. 该示例网络的通信路径和攻击路径信息如表 1 所示.

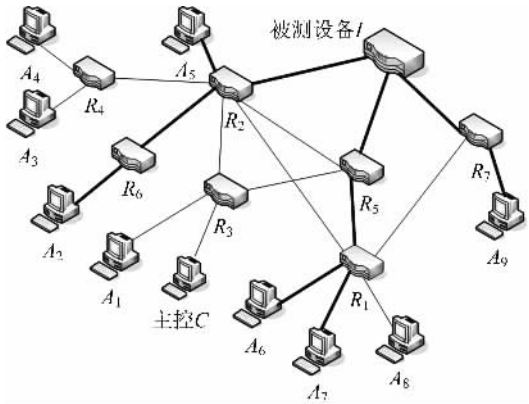

图 2 示例网络

Fig. 2 Example network

表 1 示例网络路径信息

Tab. 1 Example network paths

	通信路径		攻击路径
$C \to A_1$	无	$A_1 \to I$	R_3R_5
$C \to A_2$	$R_3R_2R_6$	$A_2 \to I$	R_6R_2
$C \to A_3$	$R_3R_2R_4$	$A_3 \to I$	R_4R_2
$C \to A_4$	$R_3R_2R_4$	$A_4 \to I$	R_4R_2
$C \to A_5$	R_3R_2	$A_5 \to I$	R_2
$C \to A_6$	$R_3R_5R_1$	$A_6 \to I$	R_1R_5
$C \to A_7$	$R_3R_5R_1$	$A_7 \to I$	R_1R_5
$C \to A_8$	$R_3R_5R_1$	$A_8 \to I$	R_1R_5
$C \to A_9$	$R_3R_5R_1R_7$	$A_9 \to I$	R_7

由此得到图 G 如图 3 所示. 注意到,由于 R_3 在通信路径中大量出现(used 值为 8),根据算法 2.1,其 selected 值开始时就被设为 1,以推迟 R_3 被选中的时机.

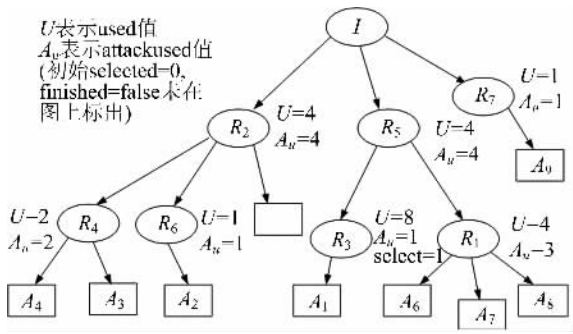

图 3 图 G

Fig. 3 Graph G

假定现在需要 5 个代理参与测试,则对 G 执行算法 agent_selection(5)得到代理列表$\{A_2, A_5, A_6, A_7, A_9\}$. 图 2 中的粗实线表示实际得到的攻击路径. 由图可以看到,经过选择后测试流量会被均匀分散到各条路径上,并且避开了主控结点和代理结点通信的关键路由器 R_3.

3 结论

本文通过介绍我们开发的分布式网络攻击测试系统,并重点给出了该系统下一个代理预选择算法,减轻了攻击测试对网络局部造成的负担,保障了结点通信,更可靠地反映了真实测试结果.

参考文献(References)

[1] Thompson H H. Application penetration testing [J]. Security & Privacy Magazine, 2005,3(1): 66-69.

(下转第 240 页)