文章编号:1001 - 9081(2009)01 - 0178 - 03

基于候选组合频繁模式的骨干网蠕虫检测研究

许晓东[1,3],杨 甦[2],朱士瑞[3]
(1. 南京理工大学 计算机科学与技术学院,南京 210094; 2. 江苏大学 计算机与通信工程学院,江苏 镇江 212013;
3. 江苏大学 网络中心,江苏 镇江 212013)
(xdxu@ujs.edu.cn; ikki2002@sohu.com; zsr@ujs.edu.cn)

摘 要:现有的网络蠕虫检测方法大多都是基于包的检测,针对骨干网 IP 流检测的研究较少,同时也不能很好地描述蠕虫的攻击模式。为此研究了一种在骨干网 IP 流数据环境下的蠕虫检测方法,通过流活跃度增长系数和目的地址增长系数定位可疑源主机,接着采用基于候选组合频繁模式的挖掘算法(CCFPM),将候选频繁端口模式在 FP 树路径中进行匹配来发现蠕虫及其攻击特性,实验证明该方法能快速地发现未知蠕虫及其端口扫描模式。

关键词:蠕虫攻击检测;IP 流;候选组合频繁模式挖掘;FP-树
中图分类号: TP393.08 **文献标志码**:A

Detecting worms based on candidate combination frequent pattern in Internet backbones

XU Xiao-dong[1,3], YANG Su[2], ZHU Shi-rui[3]
(1. *College of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing Jiangsu* 210094, *China*;
2. *College of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang Jiangsu* 212013, *China*;
3. *Network Center, Jiangsu University, Zhenjiang Jiangsu* 212013, *China*)

Abstract: The present worm detection methods have been mostly based on packets and less with IP flows in Internet backbones. They also cannot accurately describe the worm's scan-pattern. A method was presented to detect worms in Internet Backbones with flow data circumstance. First, find suspicious hosts by checking the increasing coefficients of Flow Activity Degree and Destination IP Address. Then, detect worms based on Candidate Combination Frequent Pattern Mining (CCFPM) algorithm. The results show that this method can effectively find out unknown worms and attackspattern of ports.

Key words: worm attack detection; IP flows; candidate combination frequent pattern mining; FP-tree

0 引言

20 世纪 80 年代末,一个名为“莫里斯”的蠕虫导致了当时互联网的大面积瘫痪。进入 21 世纪以后,随着网络技术的飞速发展,新一代的网络蠕虫也远远超过其前辈的破坏能力,逐渐成为互联网所面临的最严重的威胁之一[1],网络蠕虫极大地威胁到 Internet 的正常运行。因此,有效地检测和防御蠕虫攻击,对网络的安全和正常运行有深远的意义。本文将寻找一种在骨干网 IP 流数据环境下的蠕虫检测和端口扫描特征的发现方法。

1 相关工作

文献[2]根据在蠕虫攻击爆发早期,其繁殖速度呈指数增长的原理,通过检测动态流量的趋势来进行蠕虫攻击的早期检测,这种方法的虽有效,但是系统计算量过大,无法及时定位感染源。

文献[3]采用基于 IP 协议包中 TTL 值进行蠕虫攻击早期检测,但是存在取多少包进行分析的问题,且这种方法不适合骨干网。

文献[4]在流级别数据的基础上,通过监视主机的字节流量、连接数和响应数,采用基于主机行为分析的方法来检测蠕虫攻击,实验证明,该方法能检测未知蠕虫,但是该方法不能发现蠕虫的端口攻击模式。

文献[5]提出的模型能有效检测未知蠕虫,但是检测效率低下,并且由于要直接对包进行监视,因此算法的鲁棒性也不强。

文献[6]在分析了蠕虫攻击 IP 和端口特性的基础上,提出基于威胁兴趣关系(TIR)模型的树型数据结构,采用统计的方法检测蠕虫攻击,该方法在检测超过三个端口的组合模式攻击时就会影响效率。

目前许多关于蠕虫检测的理论大都局限于统计模型,针对骨干网 IP 流数据检测的研究较少,一旦发现蠕虫攻击,也不能很好地挖掘其攻击模式。因此,为了能够检测新型蠕虫及其攻击行为模式,并加快处理的速度,本文在 IP 流数据环境下,通过实时地监控用户的流活跃度、流活跃度增长系数和目的地址增长系数来定位可疑 IP,然后采用基于候选组合频繁模式挖掘(Candidate Combination Frequent Pattern Mining, CCFPM)算法来发现蠕虫及其端口扫描模式。

2 蠕虫攻击特征及 FP-Growth 算法

2.1 蠕虫攻击特征

蠕虫是一种利用远程主机的某种服务漏洞,自动寻找缺陷主机,并积极进行自我复制和传播的一段攻击代码。被蠕虫感染的主机对外呈现以下特点:在一个较短的时间窗内产

收稿日期:2008 - 07 - 18;**修回日期**:2008 - 09 - 22。 **基金项目**:江苏省教育厅高校科学研究基金资助项目(03KJD520073)。
作者简介:许晓东(1965 -),男,福建漳州人,副教授,主要研究方向:网络管理、系统集成; 杨甦(1982 -),男,江苏常州人,硕士研究生,主要研究方向:网络安全; 朱士瑞(1983 -),男,江苏泗洪人,硕士,主要研究方向:网络安全。

生大量重复模式的对外请求,即不断扫描某个端口或者端口组合[6-7,9](图 1)。

2.2　FP-Growth 算法

文献[12]提出了不产生候选挖掘频繁项集的方法——FP-Growth 频繁模式增长算法。该算法是在大型数据库中挖掘频繁项集的一个有效的算法,当数据库很大时,FP-增长算法首先进行数据库投影,得到频繁项;然后通过构造一个压缩的数据库结构——FP 树来对它们进行挖掘。实验表明,FP-Growth 算法对不同长度的规则都有很好的适应性。蠕虫扫描具有频繁的特征,"频繁"性正好成为了这两者之间的接合点,CCFPM 算法即建立在 FP-Growth 算法的基础上。

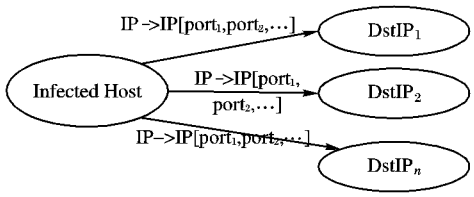

图 1　蠕虫攻击模式

3　检测方法与结果分析

通常情况下,蠕虫发作会导致网络流量增大。但在骨干网大流量环境下以及蠕虫攻击爆发早期,单从宏观流量上很难发现蠕虫扫描攻击,因为这些攻击流量往往会被淹没在大的宏观流量中。为了尽早发现蠕虫攻击,必须寻找其他方法。

本文的工作基于这样一个事实:某个用户主机一旦被蠕虫感染,那该主机就开始向随机的 IP 地址发送扫描数据包,这些数据包具有相同的源 IP 地址、不同的目的 IP 地址,按照流聚合的定义,这些包不可能汇聚成一条流,而是多个流。那么通过实时地监控每个主机发出的流个数,一旦发现某个时间段内流数大量增加,就极有可能感染了蠕虫病毒。

3.1　相关定义

定义 1　支持数。样本数据库 T 中与模式 P 匹配的记录数称为模式 P 的支持数。

定义 2　频繁模式。频繁模式是一个项集或规则,其支持数大于阈值。

定义 3　IP 流。IP 流是指符合特定的流规范和超时约束的一系列数据报文的集合[8]。本文采用五元组(源宿 IP 地址、协议、源宿端口)流规范进行流聚合。

定义 4　流活跃度。对于某个源主机 $SIPx$ 在特定的单位时间间隔 Time_Interval 内,发出的 IP 流的个数 $SumFlow(i)_{SIPx}$ 称为该 IP 地址在该时间段内的流活跃度(Flow Activity Degree),记为 $FAD(i)_{SIPx}$(其中 i 代表某个时间段的序号)。

定义 5　目的地址增长系数。在定义 4 所述的时间间隔 Time_Interval 内,设某个源主机 $SIPx$ 访问过的不同目的 IP 地址的个数为 $SumDIPx(i)$ (i 代表某个时间段的序号),则该 IP 在前三个时间段访问过的不同 IP 地址个数分别为 $SumDIPx(i-1)$、$SumDIPx(i-2)$ 和 $SumDIPx(i-3)$,定义在第 i 个时间段内 $SIPx$ 访问的目的地址增长系数 $DipGD(i) = [(3 \times SumDIPx(i)) / (SumDIPx(i-1) + SumDIPx(i-2) + SumDIPx(i-3))] - 1$,该值表明了该 $SIPx$ 在第 i 个时间段所访问的目的地址与前三个时间段均值的增幅(与均值比较有利于发现某些扫描速率较慢的蠕虫)。

定义 6　流活跃度增长系数。由定义 4 和定义 5 可推得流活跃度增长系数 $FADGD(i)_{SIPx}$ 的定义:某个源主机 SIP_x 在第 i 个时间段的流活跃度与前三个时间段流活跃度均值的增幅,即 $FADGD(i)_{SIPx} = [(3 \times FAD(i)_{SIPx}) / (FAD(i-1)_{SIPx} + FAD(i-2)_{SIPx} + FAD(i-3)_{SIPx})] - 1$。

3.2　基于 IP 流的蠕虫攻击检测算法

第 1 步　定位可疑 IP。实时地监控当前在线的源 IP 地址,每隔 Time_Interval 计算一次流活跃度 FAD。对超过流活跃度阈值的 IP,则计算它们的流活跃度增长系数 FADGD 以及访问目标 IP 地址的增长系数 DipGD。设定这两个参数的阈值分别为 $LimitF$ 和 $LimitD$(本文设定 $FAD = 1\,000$, $LimitF = 0.3$, $LimitD = 0.25$),若某 IP 在某个时间段系数分别超过了 $LimitF$ 和 $LimitD$,则认为该 IP 发生了异常,并置该 IP 地址为可疑 IP。

第 2 步　计算频繁目的端口列表。针对第 1 步中得到的可疑 IP 地址,在原始流数据中扫描对应异常时间段的流数据,统计出该 IP 访问的目的端口频度列表,并按降序排序。指定频繁访问的端口频度阈值 k,将上述端口列表中小于该阈值的端口去掉,得到频繁端口列表集合 L:{(端口号 $port$,出现频度 num) | $num \geqslant k$}。

第 3 步　转换数据表格式。对可疑 IP,再次扫描原始流数据中对应异常时间段的流数据,对目的 IP、目的端口两个字段进行投影,并将其转化为表 1 所示的数据表格式,该表表示可疑源 IP 地址访问的每个不同目的 IP 地址及其端口列表 PList,转化数据格式是为了挖掘出该可疑 IP 访问的端口模式。为了加速挖掘,将每个源宿 IP 地址对所对应的目的端口列表分成频繁端口列表 FPList 和非频繁端口列表 NFPList,即 FPList = {a | $a \in$ PList 且 $a \in L$}, NFPList = {a | $a \in$ PList 且 $a \notin L$},转化后的数据存储在一个临时数据库表 TempTable 中。

表 1　转化后的数据表格式定义

字段名称	类型	含义
SrcIP	Varchar(18)	可疑源地址
DstIP	Varchar(18)	不同的目的地址
FPList	nText	频繁端口列表
NFPList	nText	非频繁端口列表

第 4 步　采用 CCFPM 算法挖掘频繁端口模式。指定 CCFPM 算法的支持数 Sup,对 TempTable 表的 FPList 字段进行频繁模式的挖掘,挖掘该可疑 IP 所访问目的端口的模式,从而验证该 IP 是否感染蠕虫,并求得其攻击的端口模式。

与 FP-Growth 算法类似,CCFPM 算法也分两步:

第 1 步　构建 FP-树。

由于数据库中的事务已被分成频繁和非频繁端口列表,因此可大大加速构建 FP 树的速度,不同于建立 FP 树的常规方法,本文使用非递归的算法来产生 FP 树,算法代码如下,其中,Ports 是与 TempTable 表中某个(SrcIp, DstIp)对相对应的频繁目的端口列表,Tree 代表 FP 树,为了便于处理,算法将树根节点添加子节点和孩子节点添加子节点的过程分开,因此设置了 Flag 标志位。

算法 1　构建 FP 树的非递归算法

```
FPTree_Set( Ports,  Tree)
{ Flag  =  1;
  for each port a in L
  for each port b in Ports
```

```
{ if (b == a)
  { if (Flag! =1)
    { if (currentNode.get(b) == null)
                    //如果当前树节点没有对应的孩子节点
      { currentNode.add (new node(b));
        tempNode = currentNode;
        currentNode = currentNode.get(b);
                        //将新建的节点设置为当前节点
        curentNode.setPreviousNode(tempNode);
                    //将当前节点的父指针指向先前的节点
      }
      Else{//访问次数加1
        currentNode.get(b).Add_appearNum(); }
      Else {
        If (Tree.getRoot.get(b) == null)
        {              //若Root下无该端口对应的孩子节点
          Tree.getRoot.add(new node(b));
                  //新建一个树节点,作为Root的孩子节点
          currentNode = Tree.getRoot.get(b);
                        //将新申请的节点作为当前节点 }
        Else Tree.getRoot.get(b).Add_appearNum();
                                        //访问次数加1
        Flag = 2; }
      Break;            //若已发现匹配端口则跳出循环 }
    Else                                // 若b与a不相等
      Continue;
}}}
```

对TempTable表中每个元组的FPList字段都执行FPTree_Set建树算法,为了方便对树路径的遍历,一方面对树中每个节点都建立指向其父节点的指针;另一方面对L集中的每个元素都创建一个项头表[12],使得列表L中的每个项通过节点链指向它在树中的节点。

第2步 挖掘大于支持数*Sup*的频繁模式。

建立完FP树之后,开始执行CCFPM算法。这是一种基于候选模式组合匹配的挖掘方法,来进行频繁模式的挖掘,无需存储条件FP-Tree集。该算法的伪代码如下。

算法2 基于候选模式组合匹配的频繁模式挖掘

```
1)  int j = L.length;                    //j为L中元素的个数
    for(int i = 1; i <= L.length; i++)
2)  { port = L[j - i];          // 逆序访问L中每一个端口号
3)  根据项头表以及FP树节点的父指针,在建立的FP树中找
      到包含port且不包含L中从第j - i - 1到第L.length的
      端口号的所有路径,构成路径集合PathList{(Path, s)
      | s = Path中port节点的支持数};
4)  找出PathList集合各路径中不同于port的端口列表
      portList;
      // 本次挖掘的模式肯定包含端口port,为了减少组合
      // 的个数,加速挖掘,故除去port
5)  对portList列表,计算所有可能的候选组合频繁端口模式;
6)  For(int k = 0; k < PathList.Length; k++)
7)  { 将5)中求得的各个候选频繁模式在PathList[k]中进行
      匹配;
8)    If(匹配成功) 对该模式的频度m进行累加;
                    // 类加的是该路径对应的样本支持数s
9)    If(该模式最终的频度m ≥ sup) 则输出该模式;} }
```

对表2所示的TempTable表,针对FPList字段建立的FP树如图2所示,指定CCFPM算法挖掘的支持数$sup = 3$,则运行CCFPM算法后得到所有的频繁组合端口模式为:{80,139,600|3},{80,139|3}。

表2 转化后的TempTable表

SrcIP	DstIP	FPList	NFPList
10.2.100.28	201.110.90.46	80,139,600	15000
10.2.100.28	202.197.201.11	80,600	
10.2.100.28	62.69.51.32	80,139,600	8080
10.2.100.28	61.89.201.39	80	1718,169
10.2.100.28	211.196.172.18	80,139,600	25
10.2.100.28	211.196.172.12	139	

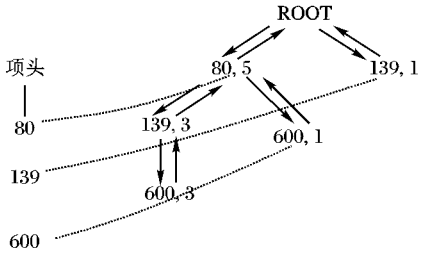

图2 表2数据最终生成的FP树

3.3 算法运行结果

3.3.1 实验环境

在校园网边界路由器上布置了NetFlow流量监控系统,实时地采集骨干网的数据包,且每隔5 min就将采集到的包汇聚成流,并将经过聚合的IP流数据按Netflow V5格式实时地存储到Oracle 9i数据库服务器中。

目前已实现了对校园网宏观流量实时的监控,并对在线用户进出口的包数、字节数和流数进行实时地统计,且可以得到对应的TopN排名。

3.3.2 CCFPM与FP-Growth性能比较

目前,本文所介绍的算法处理过程已经用Java实现。硬件环境为CPU:AMD Athlon 3600 +,内存:DDRII 1 024 MB,硬盘:希捷160 GB 7 200 rpm。为了测试CCFPM算法的性能,采集5 min正常的流数据,然后对其中某主机添加若干正常流的基础上,再添加具有红色代码、冲击波等蠕虫病毒特性的流数据,然后将其对应的流数据转成表1的格式后,对频繁端口FPList字段进行挖掘,算法的挖掘效率曲线如图3所示。

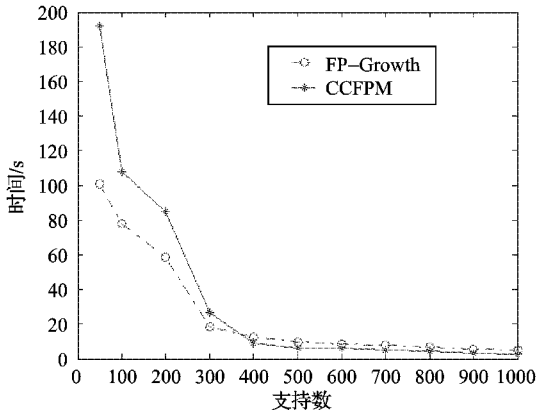

图3 算法性能比较

该图表明在支持数比较高的情况下,由于大于支持数的端口比较少,因此候选组合模式的个数也较少,CCFPM算法性能要优于FP-Growth算法。但是随着支持数的减少,候选组合模式大幅度增加,其中包括了许多非频繁的模式,算法性能与FP-Growth相比开始下降。但实际的蠕虫攻击扫描的端口一般在4个以内[6],因此候选组合频繁模式的个数较少,算法的挖掘速率较FP-Growth算法快。 (下转第192页)

新的兴趣度量方法,激励综合了支持度与效用的优点,较好地反映了项集的语义特性与统计特性,能更好地服务于决策。本文还分析了激励约束的性质,证明了激励上界特性的存在,并在HM-miner算法中利用此特性进行减枝。实验证明了算法的正确性与有效性。

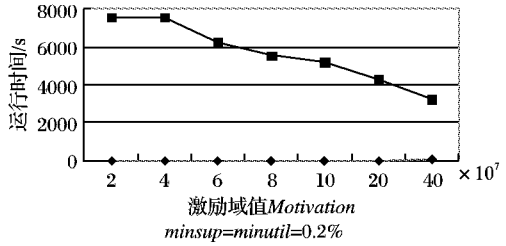

图2 激励的变化对算法性能的影响

参考文献:

[1] AGRAWAL R, SRIKANT R. Fast algorithms for mining association rules [C]// Proceedings of VLDB 1994. Santiago, Chile: VLDB Endowment, 1994: 487 - 499.

[2] LU S F, HU H P, LI F. Mining weighted association rules [J]. Intelligent Data Analysis, 2001, 5(3): 211 - 225.

[3] SHEN Y D, ZHANG Z, YANG Q. Objective - oriented utility - based association mining [C]// Proceedings of the 2002 IEEE International Conference on Data Mining. Maebashi, Japan: IEEE Computer Society, 2002: 426 - 433.

[4] YAO H, HAMILTON H J. Mining itemset utilities from transaction databases [J]. Data & Knowledge Engineering, 2006, 59(3): 603 - 626.

[5] LIU Y, LIAO W K, CHOUDHARY A. A fast high utility itemsets mining algorithm [C]// Proceedings of the 1st International Workshop on Utiliy-based Data Mining. New York: ACM Press, 2005: 90 - 99.

[6] 余光柱,李克清,易先军,等. 一种基于划分的高效用长项集的挖掘算法[J]. 计算机工程与应用, 2007, 43(29): 11 - 13.

[7] VROOM V H. Work and motivation [M]. Hoboken, NJ: John Wiley, 1964.

[8] GENG L, HOWARD J, HAMILTON H J. Interestingness measures for data mining: A survey [J]. ACM Computing Surveys (CSUR), 2006, 38 (3): 61 - 93.

[9] WANG J, LIU Y, ZHOU L, *et al.* Pushing frequency constraint to utility mining model [C] // ICCS: Proceedings of International Conference on Computational Science. Berlin: Springer-Verlag, 2007: 685 - 692.

(上接第180页)

3.3.3 实际检测结果

在真实的环境中经过多次实验,发现在把单端口出现频度阈值设置 k = 600, CCFPM 算法支持数阈值设置 sup = 500 的情况下,算法表现出了比较良好的检测效果,检测率达到80%以上。列举部分检测结果如表3所示(注:"检测时间"是指数据格式的转化以及挖掘的时间之和)。表3中各源IP地址在5 min的时间间隔内,发出的流数都在7 000个以上。

表3 实际检测结果

日 期	源 IP	端口模式	检测时间/s
2007—11—23	202.195.163.138	80	28
2007—12—24	10.2.89.52	80	22
2008—03—02	10.2.73.121	1433	41
2008—03—08	10.2.43.153	80,1755	28
2008—04—20	10.2.100.248	1433	39
2008—06—13	10.2.65.236	139,445	31

4 结语

本文针对骨干网IP流数据环境,通过实时地监控用户的出口流活跃度和访问目的IP地址的增幅来发现可疑源主机,然后将蠕虫攻击的频繁模式特征与频繁模式挖掘算法的特性相结合,提出了一种基于候选组合频繁模式挖掘的网络蠕虫检测算法。实验证明该算法能很好挖掘蠕虫的攻击模式,及时发现新型的蠕虫病毒。

参考文献:

[1] CNCERT/CC. 网络蠕虫灾害对应急处理的挑战[EB/OL]. [2008 - 05 - 01]. http://www.cert.org.cn.

[2] ZOU C C, GONG W, TOWSLEY D, *et al.* The monitoring and early detection of Internet worms [C]// IEEE/ACM Transactions on Networking. [S.l.]: IEEE Press, 2005, 13(5): 961 - 974.

[3] YAMADA Y, KATOH T, BISTA B B, *et al.* A new approach to early detection of an unknown worm [C]// AINAW '07: 21st International Conference on Advanced Information Networking and Applications Workshops. [S.l.]: IEEE Press, 2007, 1: 194 - 198.

[4] DUBENDORFER T, PLATTNER B. Host behaviour based early detection of worm outbreaks in Internet backbones[C]// WETICE '05: Proceeding of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise. [S.l.]: IEEE Press, 2005: 166 - 171.

[5] ZOU C C, GONG W, TOWSLEY D. Code red worm propagation modeling and analysis[C]// CCS '02: Proceeding of 9th ACM Conference on Computer and Communications Security. [S.l.]: ACM Press, 2002: 281 - 287.

[6] 顾荣杰, 晏蒲柳, 邹涛, 等. 基于频繁模式挖掘的Internet骨干网攻击发现方法研究[J]. 计算机科学, 2006, 33(9): 76 - 80.

[7] 王方伟, 张运凯, 王长广, 等. 网络蠕虫的扫描策略分析[J]. 计算机科学, 2007, 34(8): 105 - 108.

[8] CLAFFY K C. Internet traffic characterization. Dissertation for the degree Doctor of Philosophy. University of California[D]. San Diego: University of California, 1994.

[9] HUNG J C, LIN K C, CHANG A Y. A behavior based anti-worm system[C]// AINA '03: Proceeding of the 17th International Conference of Advanced Information Networking and Applications. Washington, DC: IEEE Press, 2003: 812 - 815.

[10] LIU BIN, LIN CHUANG, RUAN DONG - HUA, *et al*. Netflow based flow analysis and monitor [C]// International Conference on Communication Technology. Washington, DC: IEEE Press, 2006: 1 - 4.

[11] PAO TANG-LONG, WANG PO-WEI. NetFlow based intrusion detection system [C]// Proceeding of the 2004 IEEE International Conference on Networking, Sensing Control. Washington, DC: IEEE Press, 2004, 2: 731 - 736.

[12] KANTARDZIC M. 数据挖掘——概念、模型、方法和算法[M]. 闪四清, 陈茵, 程雁, 等译. 北京: 清华大学出版社, 2003: 144 - 153.

[13] Cisco. NetFlow overview[EB/OL]. [2008 - 05 - 01]. http://www.cisco.com.