具网络爬虫与端口扫描屏蔽功能的自动化Linux防火墙实现

李宗航 2014010912008

摘要: 网络爬虫是一种按照一定规则自动抓取万维网资源的程序或脚本，被广泛用于互联网领域，例如搜索引擎就使用的分布式网络爬虫来获取各站点的信息。但网络爬虫会盗取目标网站的高价值资源，并且由于其请求资源频率远高于正常用户以及请求操作的持续性，网络爬虫会占用网络带宽并增加Web服务器开销。恶意用户可以利用网络爬虫发起DoS攻击，使Web服务器资源耗尽而无法提供正常服务，更严重的可能造成站点内敏感资源的大量外流。而端口扫描能够暴露出潜在的入侵通道，使得病毒能够侵入服务器或者发掘出管理者希望隐藏的端口。因此，本文实现的是针对网络爬虫与端口扫描的、基于Linux系统下iptables的、具备自动检测与屏蔽功能的防火墙。

1. **设计思路与方案**

**1、设计思路**

防火墙是指设置在不同网络或网络安全域之间的一系列部件的组合。它可以监测、限制、更改跨越防火墙的数据流，尽可能地对外部屏蔽网络内部的信息、结构和运行状况，以此来实现网络或主机的安全保护。逻辑上，防火墙是一个分离器、限制器、分析器，有效地监控了内部网和Internet之间的数据交互，保证了内部网络的安全。

需要设计一个具有网络爬虫与端口扫描屏蔽功能的自动化Linux防火墙。具备基本的IP过滤和端口过滤功能，同时具备日志分析能力与请求辨别能力，以及自动添加与删除规则的功能。涉及到iptables规则设置、Python日志分析以及Python对Linux命令的调用等。优点在于能够自动屏蔽网络爬虫的攻击和端口扫描以保证服务器的安全性，缺点在于该方法可能对正常用户请求进行误判，导致用户不能正常访问服务器。

设计思路简单的说是将Linux虚拟机作为防火墙，IP与Port过滤功能主要由Linux系统下的iptables软件来实现，匹配iptables规则的条项有对应的动作与记录，通过使用Python分析记录文件，判断网络爬虫与端口扫描的IP源，并将该IP源加入iptables黑名单以屏蔽来自该IP的请求。

**2、设计方案**

在虚拟机Paralles Desktop下新建Linux虚拟机，选择网络为WiFi桥接网络，其IP地址为100.64.132.29，作为Web服务器，并且在该服务器上配置防火墙。

防火墙默认INPUT策略为丢弃所有包，OUTPUT与FORWARD接收所有包，接收状态为ESTABLISHED与RELATED的包，接收ICMP报文以检验网络情况，开放8000端口作为Web服务端口(默认为80端口)并记录请求信息，接收本机发出的所有数据包，接收白名单内的IP源发来的数据包，默认丢弃所有非本机发来的非8000端口的包。

iptables记录流过防火墙的数据包，将记录写入/var/log/messages文件中，通过使用Python读取该文件的内容，筛选出有效记录并将其对象化存储于数组中，便于后续的数据分析。

动态防火墙每20秒对日志记录采样分析。

第一步进行网络爬虫检测与屏蔽。通过分析来自非信任IP源的8000端口请求记录，统计其请求频率，判断该IP源是正常用户请求(低速且间断)还是网络爬虫的请求(高速且连续)，若是正常用户请求，清空日志文件并重新记录分析，若是网络爬虫请求，将该IP源写入iptables的配置文件mSetting.sh的黑名单中，并启用新的iptables配置方案。

第二步进行端口扫描的检测与屏蔽。通过分析被防火墙丢弃的数据包，这类数据包属于来自非信任IP源并且请求端口非8000端口的请求，通过收集统计这类被丢弃的数据包，筛选出各个IP请求的端口号，分析这些端口号的分布规律，简单地说如果端口号连续三次就可以判定为端口扫描。判定为端口扫描后，记录该IP源，将其写入iptables配置文件mSetting.sh的黑名单中，并启用新的iptables配置方案。

测试时使用受信任主机100.64.132.21与另一个不在信任列表中的Linux虚拟机100.64.132.22进行测试，分别在两个IP源发起爬虫请求与端口扫描脚本，观察iptables的记录文件messages输出信息，以及爬虫和端口扫描程序的调试信息，也可以在防火墙控制脚本运行的输出窗口查看并分析检测与屏蔽情况。

1. **软件设计**

本文的防火墙实现了任意非信任来源的网络爬虫与端口扫描脚本的自动检测与屏蔽，具备IP过滤和TCP端口过滤功能，具有白名单与黑名单策略，白名单内的IP源可以发起爬虫请求与端口扫描而不受防火墙影响，黑名单内的IP源将不能请求服务器的任何资源，仅能使用ICMP查看网络是否连通。防火墙搭建在parallel desktop虚拟机的ubuntu linux虚拟系统上。

1. **配置方案**
2. **iptables配置方案**

防火墙默认INPUT策略为丢弃所有包，OUTPUT与FORWARD接收所有包，接收状态为ESTABLISHED与RELATED的包，接收ICMP报文以检验网络情况，开放8000端口作为Web服务端口(默认为80端口)并记录请求信息，接收本机发出的所有数据包，接收白名单内的IP源发来的数据包，默认丢弃所有非本机发来的非8000端口的包。初始配置文件见附录文件mSetting.sh。

1. **网络爬虫与端口扫描检测与屏蔽程序**
   1. **models.py**

models.py定义了记录数据的模型，使用面向对象的方法将一个记录对象化，将相应记录的字段作为对象的属性存储，便于后期结构化分析记录数据。models.py中定义了LogItem模型，定义如下：

class LogItem:

def \_\_init\_\_(self, datetime, timestamp, prefix, IN, OUT, MAC, SRC, DST, PROTO, SPT, DPT):

self.datetime = datetime

self.timestamp = timestamp

self.prefix = prefix

self.IN = IN

self.OUT = OUT

self.MAC = MAC

self.SRC = SRC

self.DST = DST

self.PROTO = PROTO

self.SPT = SPT

self.DPT = DPT

以某条记录为例：

Nov 15 20:16:45 ubuntu kernel: [ 5998.355970] [DROP External Request]IN=eth1 OUT= MAC=01:00:5e:00:00:fb:a4:5e:60:cc:ee:bf:08:00 SRC=100.64.132.199 DST=224.0.0.251 LEN=32 TOS=0x00 PREC=0x00 TTL=1 ID=61214 PROTO=2

模型中定义的各个字段分别为：

* datetime – 记录产生日期，Nov 15 20:16:45
* timestamp – 时间戳，以秒为单位，[ 5998.355970]
* prefix – 记录前缀，标记数据包处理类型，[DROP External Request]
* IN – 数据包流入的网卡，eth1
* OUT – 数据包流出的网卡
* MAC – MAC地址，01:00:5e:00:00:fb:a4:5e:60:cc:ee:bf:08:00
* SRC – 源IP地址，100.64.132.199
* DST – 目的IP地址，224.0.0.251
* PROTO – 协议类型，2
* SPT – 源端口号
* DPT – 目的端口号

程序中实际使用的仅有timestamp、prefix、SRC、DPT四个字段。

* 1. **message\_handler.py**

message\_handler.MessageHandler类定义了对/var/log/messages文件的操作接口，提供了日志筛选与对象化、获取外网8000端口请求IP统计及弃包统计接口。需要分析的日志状态为ACCEPT External 8000与DROP External Request两个状态，分别用于分析网络爬虫与端口扫描。类MessageHandler实现见附录文件message\_handler.py。

* 1. **setting\_handler.py**

setting\_handler.SettingHandler类定义了对配置文件mSetting.sh的操作接口，SettingHandler将mSetting.sh中的配置内容以列表的方式存储，提供了iptables配置命令定位、在适当位置将IP插入黑名单、将IP从黑名单中移除的接口，依照列表settings中的内容修改对应配置文件mSetting.sh中的配置内容，最后使用os模块调用命令行命令，使iptables加载并使用新的配置文件。类SettingHandler实现见附录文件setting\_handler.py

* 1. **scrawl\_detector.py**

scrawl\_detector.ScrawlDetector类定义了爬虫检测的方法，提供了日志规范化输出、爬虫检测的接口。利用ScrawlDetector.parse\_log\_dict()方便管理员查看8000端口的访问情况，容易快速的判断出异常请求。利用ScrawlDetector.detect\_scrawl()可以分析日志信息并筛选出高频请求的IP源。一般来说，按正常步骤连续操作的正常用户平均每3秒访问一次服务器，而网络爬虫则以更高频率如每秒1-10次访问服务器。防火墙每20秒对日志文件采样分析，通过检查每个IP的请求频率，即可判断该IP是否正常的用户请求。为了避免浏览器多次请求静态资源引起的判断失误，以及考虑到后台的设计方案，选取15作为采样间隔，以15秒作为判断标准，即平均每秒内若该IP产生了一次或以上请求则判定为网络爬虫，否则为正常用户请求。ScrawlDetector类实现见附录文件scrawl\_detector.py。

判断网络请求并不只有判断请求频率一种方法，实际上这种方法容易造成误判，较为常用的作法为限制单个IP对某API或Token的访问次数，或在页面中设置隐藏的链接，该链接不会显示给用户，但爬虫会误入这个链接，随后后台服务器便可封禁该IP，或检查请求头部信息与校验用户提交的表单，以分离出用户请求、搜索引擎爬虫、私人爬虫。

由于iptables不能提供相应的日志内容，因此实现方案采用检查请求频率来辨别网络爬虫，可以修改despire\_sample\_interval与despire\_time\_interval以适应实际系统的需要，这两个参数为经验参数。

* 1. **port\_scan\_detector.py**

port\_scan\_detector.PortScanDetector类提供了端口扫描检测的实现，PortScanDetector.detect\_scanner()提供了端口扫描脚本的检测功能，其返回一个对本机进行端口扫描的IP列表。PortScanDetector类实现见附录文件port\_scan\_detector.py。

考虑到端口扫描脚本的常用技巧为逐端口扫描，这里也采用检测三连续端口的方式来检测端口扫描脚本。

* 1. **dynamic\_firewall.py**

dynamic\_firewall.DynamicFirewall类是执行自动化检测与屏蔽功能的主类，集合了上述提到的所有类，该类定义了白名单过滤、扫描启动器、爬虫处理函数、端口扫描处理函数、IP封禁函数。DynamicFirewall类实现见附录文件dynamic\_firewall.py。

扫描启动后，每20秒对日志文件一次样，分别进行爬虫识别和端口扫描识别后得到的IP列表经过白名单过滤最终交给scrawl\_banner函数封禁该IP列表中所有IP，scrawl\_banner调用了SettingHandler类的insert\_ip\_banner接口，操作iptables的配置文件mSetting.sh的写操作，并调用Linux命令加载与启用新配置。一次识别完成后，清空日志文件，等待20秒后再次采样。

1. **网络爬虫与端口扫描脚本实现**

形象的说，网络爬虫就是按照一定的规则访问一个页面，提取中页面中的链接存储到待爬取的url队列，在本页面筛选出需要的数据后又进入队列中下一个url提取数据的程序。它像一只蜘蛛，在蜘蛛网上不断地爬走，因此爬虫文件或模块常用Spider命名。

网络爬虫的本质是高频率的网络请求与数据处理程序，由于本项目不需要对获取的数据进行处理，因此测试用的网络爬虫仅实现高频率的网络请求即可。由于队列中维护的url都指向本地服务器，也不需要对各个url进行爬取，因此也省去了对url队列的维护，由于也不需要高效率的爬取服务器上的资源，也不必考虑多线程爬虫或分布式爬虫。测试用的爬虫本质是一个高频率的循环网络请求。SimpleScrawl类实现见附录文件ext\_spider.py。

如果服务器开启了防火墙，并设置INPUT策略为DROP，管理员只想对外网主机开放了8000端口提供Web服务，而由于疏忽或某些需求需要开放且隐藏8001端口，8001端口是不希望外网主机访问的。而端口扫描可以对某个指定端口区间如7000-9000逐个进行端口扫描，正常情况下7000-7999，8002-9000端口的数据包都会被防火墙过滤，使得端口扫描程序被挂起直至请求超时异常，而扫描到8000端口时会返回404页面不存在，表示8000是开放端口并提供Web服务，而8001端口会返回连接错误而不是请求超时，表明8001端口是一个开放的端口，这将暴露服务器管理员希望隐藏的端口或被忽略的系统漏洞，从而给攻击者提供了攻击的入口。一个简单的端口扫描类实现见附录文件ext\_spider.py。

端口扫描也可以用nmap来实现，它包含四项基本功能：1.主机发现 2.端口扫描 3.版本侦测 4.操作系统侦测。例如命令nmap –F –sT –v 100.64.132.21就能够采用TCP扫描服务器100.64.132.21上100个最有可能开放的端口。此处为了体现扫描的规律采用requests进行扫描。

1. **测试方案与方法**

假设服务器S(开启防火墙)IP为100.64.132.73，受信任的“外网”主机PC1的IP为100.64.132.72，不在受信任列表里的“外网”主机PC2的IP地址为100.64.132.74(Linux虚拟机，桥接网络)。

1. **初始配置下测试ICMP协议**

依照初始配置，PC1和PC2都应能ping通服务器S，保证网络连通。

1. **初始配置下测试内部8000端口**

开启Web服务器，将Web服务器开在8000端口，并在服务器S的浏览器上访问8000端口，应当能够正常访问。

1. **初始配置下测试外部8000端口**

在PC1与PC2上分别访问服务器S的8000端口的Web服务，应当能够正常访问。

1. **初始配置下测试内部任意端口**

在服务器S上访问任意端口，应当能够收到反馈信息(浏览器提示无法连接)。

1. **初始配置下测试白名单策略**

在PC1上访问服务器S的任意端口，应当能够收到反馈信息。

在PC2上访问服务器S的任意端口，不应收到任何反馈信息。

1. **测试网络爬虫检测与屏蔽功能**

在服务器S上启动程序dynamic\_firewall.py，使用tail –f /var/log/messages查看输出日志。

在PC1或PC2上打开浏览器正常访问网页，程序dynamic\_firewall不应检测到爬虫程序或扫描程序。

在PC1上运行爬虫脚本ext\_spider.py，程序dynamic\_firewall应检测到PC1为爬虫，但不应封禁PC1。

在PC2上运行爬虫脚本ext\_spider.py，程序dynamic\_firewall应检测到PC2为爬虫，且PC2的IP被封禁，查看配置文件mSetting.sh应有PC2被加入到黑名单中。

1. **测试端口扫描检测与屏蔽功能**

从配置文件mSetting.sh中清空黑名单，并让iptables重新启用配置。

在PC2上运行端口扫描脚本ext\_spider.py，dynamic\_firewall应检测到来自PC2的端口扫描，并将PC2的IP加入黑名单，此后PC2的扫描窗口扫描所有端口都应显示端口不可达的提示信息。

1. **后期屏蔽功能**

不从配置文件mSetting.sh中清空黑名单，PC2对服务器的任何请求都应被丢弃，只有ICMP协议报文能够到达服务器，其他请求都被挂起。

1. **测试结果与分析**

注: 视频中左侧上方命令行属于受信任主机PC1，左侧下方命令行属于PC2，右侧四个命令行都属于服务器(防火墙)S1。

1. **初始配置下测试ICMP协议(00:10-00:22)**

PC1和PC2都能ping通服务器S，说明网络连通，且ICMP报文没有被防火墙过滤。

1. **初始配置下测试内部8000端口(00:24-00:44)**

开启Web服务器，将Web服务器开在8000端口，并在服务器S的浏览器上访问8000端口，浏览器能够正常访问网站，且防火墙记录[ACCEPT White Loopback]中SRC为本机，说明防火墙接收了所有本机发出的请求。

1. **初始配置下测试外部8000端口(00:52-01:24)**

在PC1与PC2上分别访问服务器S的8000端口的Web服务，能够正常访问，并且/var/log/messages日志中记录了ACCEPT日志，分别来自于PC1和PC2，说明防火墙接收了其他主机对本机8000端口的请求。

1. **初始配置下测试内部任意端口(01:25-01:37)**

在服务器S上访问8001端口，显示无法连接，说明浏览器收到了来自8001端口的反馈信息而提示无法连接，并且日志文件中有前缀为[ACCEPT White Loopback]的记录，说明防火墙接收了来自本机的对任意端口的访问。

1. **初始配置下测试白名单策略(01:38-02:24)**

在PC1上访问服务器S的8001端口，能够收到反馈信息，日志文件中有前缀为[ACCEPT White Host]的记录，说明防火墙接收了来自受信任主机PC1对任意端口的访问。

在PC2上访问服务器S的任意端口，请求被挂起，不能收到任何反馈信息，日志文件中有前缀为[DROP External Request]的记录，说明防火墙拒绝了来自非信任主机PC2对任意端口的访问。

1. **测试网络爬虫检测与屏蔽功能(02:25-03:52)**

注: 部分耗时较长的操作在视频处理时采取加速措施以节省时间。

在PC1或PC2上打开浏览器正常访问网页，程序dynamic\_firewall输出：No scrawl detected，并且日志文件有少数前缀为[ACCEPT External 8000]的SRC为PC1或PC2的日志记录，说明正常的用户请求没有被判定为爬虫请求。

在PC1上运行爬虫脚本ext\_spider.py，爬虫窗口以每秒2次输出200成功状态反馈，同时日志文件出现大量前缀为[ACCEPT External 8000]的SRC为PC1的日志记录，程序dynamic\_firewall检测到PC1为爬虫，但由于PC1在受信任列表中，因此没有IP被封禁，PC1的爬虫可以持续访问，说明程序正确辨别出了爬虫请求，也正确执行了白名单策略。

在PC2上运行爬虫脚本ext\_spider.py，爬虫窗口前期以每秒2次输出200成功状态反馈，同时日志文件出现大量前缀为[ACCEPT External 8000]的SRC为PC2的日志记录，程序dynamic\_firewall检测到PC2为爬虫，并且PC2不在受信任名单中，PC2的IP被封禁，随后爬虫进程输出 [WARNING] Timeout, the spider has been banned并终止，日志输出前缀为[DROP Black Host]的SRC为PC2的日志记录，程序dynamic\_firewall恢复输出No scrawl detected，查看配置文件mSetting.sh可看到PC2被加入到黑名单中。表明程序dynamic\_firewall正确检测并屏蔽了PC2的爬虫程序。

1. **测试端口扫描检测与屏蔽功能(03:52-05:10)**

在PC2上运行端口扫描脚本ext\_spider.py，扫描窗口先输出port 799x is unreachable，随后dynamic\_firewall检测到来自PC2的端口扫描，将PC2的IP加入黑名单，此后PC2的扫描窗口扫描到开放端口8000时显示端口不可达的提示信息，日志中也记录了对于PC2的处理方案[DROP Black Host]。表明程序dynamic\_firewall正确检测并屏蔽了PC2的端口扫描程序。

由于PC1在受信任名单中，检测程序会忽略来自PC1的端口扫描操作，此处不展示。

1. **后期屏蔽功能(04:43-05:41)**

不清空黑名单，PC2对服务器的任何请求都被丢弃，只有ICMP协议报文能够到达服务器，其他请求都被挂起。如爬虫操作直接返回 Timeout, the spider has been banned.表明屏蔽功能持续有效。