特殊字符绕过 DNS 过滤

---vr_system

1. 介绍

DNS 域名系统是互联网关键的基础设施之一，它是一个将域名与 IP 地址互相映射的全球分布数据库。对于恶意 DNS 的过滤是安全防护设备中必不可少的一种手段。利用 DNS 流量检测恶意网站域名、僵尸网络和网络隐秘通道。

2. 攻击原理

RFC 1035 规定了域名每个标签不超过 63 字节，域名总长不超过 255 字节。可以含有任意 8bit 值，通常情况下域名标签由英文字母、数字和连字符构成。RFC 2181 进一步明确了，DNS 本身不对域名所含字符内容进行限制。一些文献中验证了 ISC BIND 等常用 DNS 服务器软件对二进制域名的支持。尽管在 RFC1 123 之中对于 DNS 软件支持无法转换为可打印格式的资源记录，内部存储不能使用文本格式。由于 Letter Digit Hyphen 规则的域名含有可打印字符，如此产生了两种问题，其一为大多数程序对于域名的处理采用字符串函数，可能会对于某些特定结束字符进行处理(例如 C 语言中对于\000 进行处理)，其二 DNS 服务器对于特殊字符进行处理后依然返回解析结果，某些程序过滤恶意 DNS 域名并未考虑。

3. 实验验证

结束符的方法已有详细的论述，请参见《NULL 字符欺骗 DNS 监控系统的研究》，本章中不多阐述。

3.1.测试方法和测试对象

在对 DNS 服务器测试时，我们想被测的服务器发送正常和带有特殊字符的 DNS 两种请求，如果 DNS 服务器两种响应存区别则证明其二失败，否则成功。

用 PYTHON socketserver 和 struct 开发简单的 DNS 服务器进行测试，再使用 DNSPython 模块作为 DNS 请求的测试。DNS 服务器脚本详情请见附录。

测试使用的版本:

Dnspython 1.16.0

Python 2.17.16

首先进行的是正常的测试,使用 www.aa.com.www.bb.com 能够正常的解析。如图 1 所示。

```
Python 2.7.16 (default, Apr  6 2019, 01:42:57)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import dns.resolver
>>> my_resolver=dns.resolver.Resolver()
>>> my_resolver.nameservers=['192.168.[illegible]']
>>> answer=my_resolver.query('www.aa.com.www.bb.com')
>>> print(answer.response)
id 46608
opcode QUERY
rcode NOERROR
flags QR RD RA
;QUESTION
www.aa.com.www.bb.com. IN A
;ANSWER
www.aa.com.www.bb.com. 190 IN A 192.168.0.2
;AUTHORITY
;ADDITIONAL
```

图 1

然后使用带有特殊字符的 www.aa.com\x09.www.bb.com 解析结果，如图 2 所示。

```
Python 2.7.16 (default, Apr  6 2019, 01:42:57)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import dns.resolver
>>> my_resolver=dns.resolver.Resolver()
>>> my_resolver.nameservers=['192.168.[illegible]']
>>> answer=my_resolver.query('www.aa.com\x09www.bb.com')
>>> print(answer.response)
id 44602
opcode QUERY
rcode NOERROR
flags QR RD RA
;QUESTION
www.aa.com\009www.bb.com. IN A
;ANSWER
www.aa.com\009www.bb.com. 190 IN A 192.168.0.2
;AUTHORITY
;ADDITIONAL
```

图 2

通过返回信息不难得出"\x09"等同于"."。为了进一步分析，通过数据报文查看传输的请求。正常的 DNS 请求如图 3 所示。

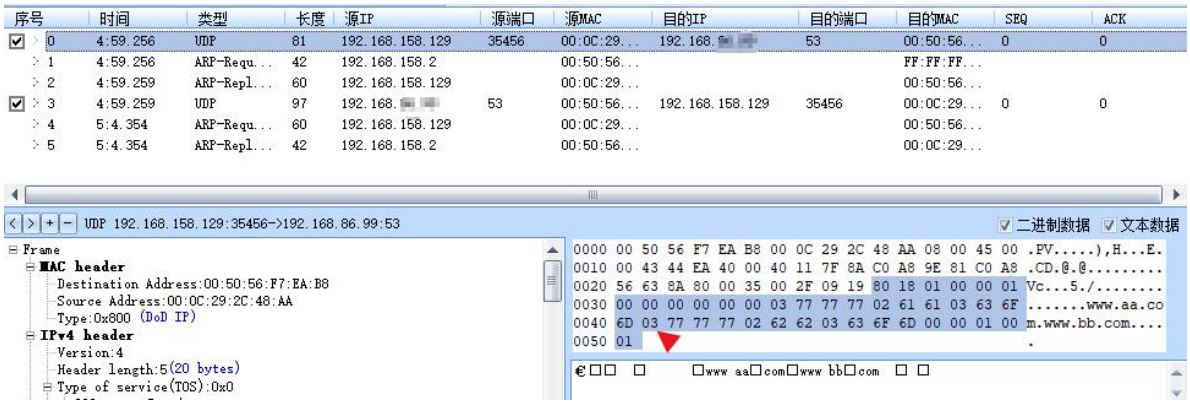

图 3

带有特殊字符的请求如图 4 所示。

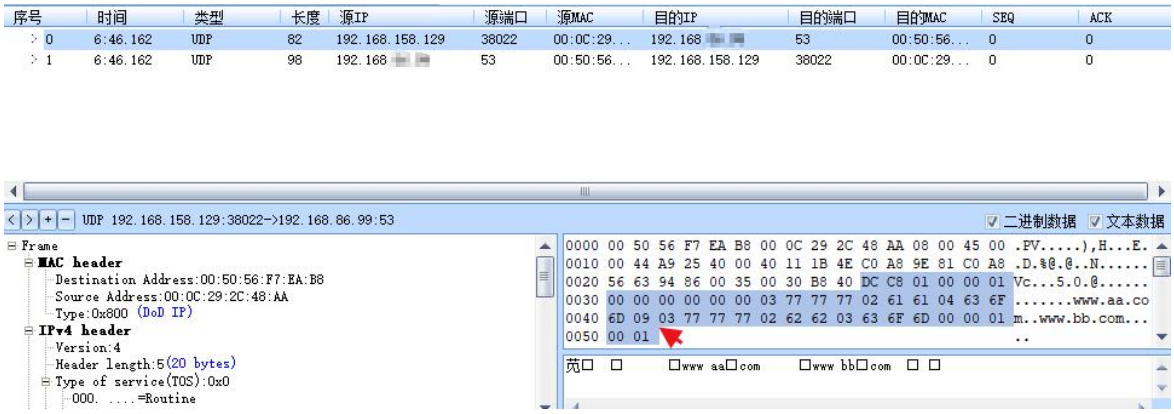

图 4

4. 绕过测试

测试能够绕过某些安全大厂的设备 DNS 域名过滤，一些原因不能在文章中进行展示。允许的特殊字符范围在\x01-\x19，这个范围内的特殊字符等同于“.”。绕过的测试方法如图 5 所示。

```
>>> answer=my_resolver.query('www.aa\x09com\x01www.bb\x06com')
>>> print(answer.response)
id 25122
opcode QUERY
rcode NOERROR
flags QR RD RA
;QUESTION
www.aa\009com\001www.bb\006com. IN A
;ANSWER
www.aa\009com\001www.bb\006com. 190 IN A 192.168.0.2
;AUTHORITY
;ADDITIONAL
```

图 5

5. 修复方法

1. 通过 DNSPython 修复，在 dns\resolver.py->Resolver()->query() 第 802 行，加入过滤异常的特殊字符。
2. 安全设备中扩大过滤 DNS 请求特殊字符的范围。

6. 附录

DNS 简单服务器：

```
import socketserver
import struct
# DNS Query
class SinDNSQuery:
    def __init__(self, data):
        i = 1
        self.name = ''
```

```
        while True:
            d = data[i]
            if d == 0:
                break;
            if d < 32:
                self.name = self.name + '.'
            else:
                self.name = self.name + chr(d)
            i = i + 1
        self.querybytes = data[0:i + 1]
        (self.type, self.classify) = struct.unpack('>HH', data[i + 1:i + 5])
        self.len = i + 5
    def getbytes(self):
        return self.querybytes + struct.pack('>HH', self.type, self.classify)
# DNS Answer RRS
# this class is also can be use as Authority RRS or Additional RRS
class SinDNSAnswer:
    def __init__(self, ip):
        self.name = 49164
        self.type = 1
        self.classify = 1
        self.timetolive = 190
        self.datalength = 4
        self.ip = ip
    def getbytes(self):
        res = struct.pack('>HHHLH', self.name, self.type, self.classify, self.timetolive,
self.datalength)
        s = self.ip.split('.')
        res = res + struct.pack('BBBB', int(s[0]), int(s[1]), int(s[2]), int(s[3]))
        return res
# DNS frame
# must initialized by a DNS query frame
class SinDNSFrame:
    def __init__(self, data):
        (self.id, self.flags, self.quests, self.answers, self.author, self.addition) =
struct.unpack('>HHHHHH', data[0:12])
        self.query = SinDNSQuery(data[12:])
    def getname(self):
        return self.query.name
    def setip(self, ip):
        self.answer = SinDNSAnswer(ip)
        self.answers = 1
        self.flags = 33152
    def getbytes(self):
```

```
        res = struct.pack('>HHHHHH', self.id, self.flags, self.quests, self.answers,
self.author, self.addition)
        res = res + self.query.getbytes()
        if self.answers != 0:
            res = res + self.answer.getbytes()
        return res
# A UDPHandler to handle DNS query
class SinDNSUDPHandler(socketserver.BaseRequestHandler):
    def handle(self):
        data = self.request[0].strip()
        dns = SinDNSFrame(data)
        socket = self.request[1]
        namemap = SinDNSServer.namemap
        if(dns.query.type==1):
            # If this is query a A record, then response it

            name = dns.getname();
            if namemap.__contains__(name):
                # If have record, response it
                dns.setip(namemap[name])
                socket.sendto(dns.getbytes(), self.client_address)
            elif namemap.__contains__('*'):
                # Response default address
                dns.setip(namemap['*'])
                socket.sendto(dns.getbytes(), self.client_address)
            else:
                # ignore it
                socket.sendto(data, self.client_address)
        else:
            # If this is not query a A record, ignore it
            socket.sendto(data, self.client_address)
# DNS Server
# It only support A record query
# user it, U can create a simple DNS server
class SinDNSServer:
    def __init__(self, port=53):
        SinDNSServer.namemap = {}
        self.port = port
    def addname(self, name, ip):
        SinDNSServer.namemap[name] = ip
    def start(self):
        HOST, PORT = "0.0.0.0", self.port
        server = socketserver.UDPServer((HOST, PORT), SinDNSUDPHandler)
        server.serve_forever()
```

```
# Now, test it
if __name__ == "__main__":
    sev = SinDNSServer()
    sev.addname('www.aa.com', '192.168.0.1')     # add a A record
    sev.addname('www.aa.com.www.bb.com', '192.168.0.2')     # add a A record
    sev.addname('*', '192.168.0.5') # default address
    sev.start() # start DNS server
# Now, U can use "nslookup" command to test it
# Such as "nslookup www.aa.com"
```