BadTunnel：跨网段劫持广播协议

作者 **-Y.c** - 2016年6月19日

Author：tombkeeper@腾讯玄武实验室

0x00 简介

本文提出了一种新的攻击模型，可以跨网段劫持TCP/IP广播协议，我们把它命名为“BadTunnel”。

利用这种方法，可以实现跨网段的NetBIOS Name Service Spoofing攻击。无论攻击者和用户是否在同一网段，甚至中间存在防火墙或NAT，只要用户打开IE或Edge浏览器访问一个恶意页面，或打开一个特殊构造的Office文档，攻击者就可以劫持用户系统对任意NetBIOS名称的解析，从而实现仿冒本地网络的打印服务器、文件服务器等。

通过劫持“WPAD”名称，还可以进一步实现劫持用户的所有网络通信，包括一般网络访问，和Windows Update service以及Microsoft Crypto API更新Certificate revocation list的通信等。而一旦能劫持网络通信，配合类似Evilgrade的工具（参考链接【1】），也很容易在系统上运行任意程序。

这种方法对没有安装2016年6月补丁的所有版本Windows都有效。可以通过所有版本的IE和Edge、所有版本的MS Office、以及大量第三方软件触发。事实上只要存在能嵌入file URI scheme或UNC path的地方，就可以触发BadTunnel 攻击。如果在一个快捷方式中将图标路径设置为恶意file URI scheme或UNC path，只要用户在资源管理器看见这个快捷方式，就会触发BadTunnel攻击。所以BadTunnel可以通过网页、邮件、U盘等多种手段进行利用。甚至还可能威胁WEB服务器和SQL服务器等（参考链接【2】）。

（本文并未包含BadTunnel相关研究的所有内容，其余部分将在BlackHat US 2016的演讲“BadTunnel: How do I get Big Brother power?”中发布。）

0x01 背景知识

NetBIOS是一套古老的协议。1987年IETF发布RFC 1001与RFC 1002，定义了NetBIOS over TCP/IP，简称NBT。NetBIOS包含三种服务，其中之一是名称服务（Name service），即NetBIOS-NS，简称NBNS。NBNS可以通过发送局域网内广播来实现本地名称解析。

当你试图访问 `//Tencent/XuanwuLab/tk.txt` 时，NBNS会向广播地址发出NBNS NB query：

谁是“TENCENT”？

而本地局域网内的任何主机都可以回应：

192.168.2.9是“TENCENT”。

然后你的电脑就会接受这个回应，然后去访问 `//192.168.2.9/XuanwuLab/tk.txt` 。

这套机制谈不上安全，但由于发生在局域网内，而局域网通常被认为是相对可信的环境。所以虽然很早就有人意识到可以在局域网内假冒任意主机，但这并不被认为是漏洞——就像ARP Spoofing并不被认为是漏洞一样。

WPAD（Web Proxy Auto-Discovery Protocol）是另一套有超过二十年历史的古老协议，用于自动发现和配置系统的代理服务器。几乎所有操作系统都支持WPAD，但只有Windows系统默认启用这个协议。按照WPAD协议，系统会试图访问 `http://WPAD/wpad.dat` ，以获取代理配置脚本。

在Windows上，对“WPAD”这个名称的请求很自然会由NBNS来处理。而如前所述，在局域网内，任何主机都可以声称自己是“WPAD”。所以，这套机制也谈不上安全，但由于同样发生在局域网内，而局域网通常被认为是相对可信的环境，所以虽然十几年前就有人意识到可以在局域网内利用WPAD劫持假冒任意主机，2012年被发现的Flame蠕虫也使用了这种攻击方式，但这并不被认为是漏洞——就像ARP Spoofing并不被认为是漏洞一样。

接下来还得再提一下TCP/IP协议。NBNS是用UDP实现的。UDP协议最主要的特点是无会话。无论是防火墙、NAT还是任何其它网络设备，都无法分辨一个UDP包属于哪个会话。只要网络设备允许IP1:Port1->IP2:Port2，就必然同时允许IP2:Port2->IP1:Port1。

刚才我们说过NBNS使用广播协议，通过向本地广播地址发送查询来实现名称解析。但NBNS和绝大多数使用广播协议的应用一样，并不会拒绝来自本网段之外的回应。也就是说，如果192.168.2.2向192.168.2.255发送了一个请求，而10.10.10.10及时返回了一个回应，也会被192.168.2.2接受。在某些企业网络里，这个特性是网络结构所需要的。

0x02 实现方法

所以，假如我们能在NBNS发出名称解析请求的时候，从本网段之外返回一个回应，也同样会被NBNS接受，就可以实现跨网段NBNS Spoofing。但存在几个问题：

1、大多数主机都开启了防火墙，从本地网络之外主动向系统发送数据似乎是不可能的。即使不考虑防火墙，从互联网上主动向一个局域网IP发送数据似乎更是不可能的。也就是说只能对有公网IP又没有防火墙的系统进行NBNS Spoofing？

2、NBNS协议内部封装的几乎就是DNS报文，所以也有Transaction ID。只有Transaction ID匹配的回应包才会被接受。这个问题如何解决？

3、本地网络之外的主机接收不到NBNS NB query广播，又怎么知道该在什么时候发出NBNS Spoofing数据包？

幸运的是，这些问题都可以解决。

首先，Windows系统的NBNS使用且只使用137/UDP端口。“使用且只使用”的意思是：系统发起的NBNS通信，源端口和目标端口都永远是137/UDP。也就是说，如果一台内网的主机192.168.2.2向10.10.10.10发起NBNS查询请求，大概会是这样：

192.168.2.2:137 -> NAT:54231 -> 10.10.10.10:137

而10.10.10.10返回查询结果时会是这样：

192.168.2.2:137 <- NAT:54231 <- 10.10.10.10:137

也就是说，无论192.168.2.2的本机防火墙，还是NAT，还是中间的任何其它网络设备，只要允许查询请求发出，并允许查询结果返回，就至少需要在一段时间内，允许10.10.10.10:137发出任何UDP包到192.168.2.2:137。这其实就开启了一条双向UDP隧道。BadTunnel，指的就是这个Tunnel：

192.168.2.2:137 <-> NAT:54231<-> 10.10.10.10:137

有个简单的实验可以帮助你理解这个隧道。准备两台开启了防火墙的系统，IP地址分别是192.168.2.2和192.168.3.3：

首先在192.168.2.2上执行“`nbtstat -A 192.168.3.3`”，会失败。
然后在192.168.3.3上执行“`nbtstat -A 192.168.2.2`”，会成功。
再次在192.168.2.2上执行“`nbtstat -A 192.168.3.3`”，会成功。

那么怎么让192.168.2.2向10.10.10.10发出NBNS请求呢？当Windows系统试图访问一个带有IP地址的file URI scheme或UNC path时，如果目标IP地址的139、445端口不可访问（超时或收到TCP重置报文），系统会再向该IP地址发送NBNS NBSTAT query查询。而让系统访问file URI scheme或UNC path的途径太多了。

无论是Edge浏览器还是IE，都会试图解析页面中的fileURI scheme或UNC path：

```
<img src="//10.10.10.10/BadTunnel">
```

所有类型的MS Office文档都可以嵌入file URI scheme或UNC path。还有很多第三方软件的文件格式也都可以。

特别是如果我们将任何快捷方式的图标设置为一个UNC path，只要这个快捷方式显示在屏幕上，系统就会试图访问UNC path。

而如果目标是一台Web服务器，可能只需一个HTTP请求：

```
http://web.server/reader.aspx?ID=//10.10.10.10/BadTunnel
```

至于TransactionID，NBNS的Transaction ID并不是随机的，而是递增的。前面提到，NBNS解析名称时，会发出NBNS NB query；而系统访问file URI scheme或UNC path失败时，会发出NBNS NBSTAT query。NBNS NB query和NBNS NBSTAT query除了都使用且只使用137/UDP外，它们还共享同一个Transaction ID计数器。也就是说，当192.168.2.2访问 `//10.10.10.10/BadTunnel` 失败，向10.10.10.10发出的NBNS NBSTAT query不但打开了一条双向UDP隧道，还将系统的Transaction ID计数器当前值告诉了10.10.10.10。

也就是说，一个NBNS NBSTAT query同时解决了第一个问题和第二个问题。而第三个问题就更容易解决了。我们既然能在网页中嵌入 `<img src="//10.10.10.10/BadTunnel">` ，当然也可以同时嵌入：

```
<img src="http://WPAD/wpad.dat">
```

这样，我们可以控制对“WPAD”的NBNS NBquery的发出时间。也就可以及时返回伪造的回应。最终系统会将我们伪造的 `http://WPAD/wpad.dat` 存入WEB缓存。之后当系统真正试图获取并解析 `http://WPAD/wpad.dat` 来设置代理服务器时，会使用WEB缓存中的这个。而至少对Windows 7来说，伪造的 `http://WPAD/wpad.dat` 会像其它被缓存的WEB资源一样，即使关机重启动，仍然有效。

即使不考虑WEB缓存，NBNS也有自己的缓存机制。只要成功实现一次NBNS Spoofing，伪造的结果会被NBNS缓存10分钟：

此后10分钟内系统本身也会试图去解析“WPAD”进而访问 `http://WPAD/wpad.dat` 来设置代理，但获得的将会是缓存中这个伪造的结果。而攻击者在一旦通过WPAD劫持到用户的流量，可以定时对某些HTTP请求返回302重定向，实现循环BadTunnel攻击，保持劫持状态：

```
HTTP/1.1 302 Found
Content-Type: text/html
Location: file://10.10.10.10/BadTunnel
Content-Length: 0
```

0x03 总结

本文所描述的BadTunnel攻击，是一个严重的安全问题。但当我们试图寻找问题根源时，却发现这并不容易。BadTunnel攻击能得以实现，至少依赖于以下这些特性：

1、 UDP协议无会话；

2、 广播请求可接受网段外回应。

3、 Windows默认开启WPAD。

4、 Windows文件处理API默认支持UNC path。

5、 Windows访问UNC path时，连接139和445端口失败后会发起NBNS NBSTAT query。

6、 NBNS无论作为服务端还是客户端，都使用同一个端口号。

7、 NBNS Transaction ID递增而不是随机。

8、 NBNS NBSTAT query和NBNS NB query共享同一个计数器。

9、 系统在实现WPAD时也使用WEB缓存机制和NBNS缓存机制。

以上所有设计特性，单独来看，几乎都没问题，甚至是必需的。我们当然不能认为UDP协议无会话是个漏洞。即使NBNS Transaction ID非随机这一点，也很难说是安全问题。因为NBNS NB这套机制原本设计用于内网，NBNS NB query以广播包形式发出，内网任何机器都能收到。但是，所有这些单独看起来都没问题的特性，在协同工作时，就形成了一个巨大的安全问题。那么，我们应该如何去发现下一个BadTunnel?

0x04 防御建议

即使不能及时安装MS16-063和MS16-077补丁，也有一些其它方法可以阻止BadTunnel攻击。

对企业来说，可以在边界防火墙上关闭内部网络和互联网之间的137/UDP通信。

对无需访问Windows网络共享服务的个人用户来说，可以考虑禁用NetBIOS over TCP/IP：

对兼容性影响最小的方式可能是在 `%SystemRoot%/System32/drivers/etc/hosts` 中添加固定的WPAD解析，或关闭自动检查代理配置，来防止“WPAD”这个名称被劫持：

不过要注意的是，这样并不能阻止对其它名称的劫持。而BadTunnel，不只是WPAD。

0x05 一点遗憾

利用BadTunnel劫持WPAD可能是历史上影响范围最广、触发途径最多的Windows漏洞，更可能是绝无仅有的写一个Exploit即可攻击所有版本Windows的漏洞。而实际上还可能更有趣。

MAC OS系统也实现了NBNS，并在某些场合支持UNC path，理论上也可以手工开启WPAD，但由于MAC OS的NBNS实现细节和Windows有所不同，并且系统自身默认使用mDNS而不是NBNS去解析名称，所以这个问题并不影响MAC OS——要不然就太酷了。

0x06 参考链接

【1】 Evilgrade

https://github.com/infobyte/evilgrade/

【2】 10Places to Stick Your UNC Path

https://blog.netspi.com/10-places-to-stick-your-unc-path/

【3】 WebProxy Auto-Discovery Protocol

http://tools.ietf.org/html/draft-ietf-wrec-wpad-01

【4】 NetBIOS Over TCP/IP

https://technet.microsoft.com/en-us/library/cc940063.aspx

【5】 Disable WINS/NetBT name resolution

https://technet.microsoft.com/en-us/library/cc782733(v=ws.10).aspx

【6】 MS99-054, CVE-1999-0858

https://technet.microsoft.com/en-us/library/security/ms99-054.aspx

【7】 MS09-008, CVE-2009-0093, CVE-2009-0094

https://technet.microsoft.com/en-us/library/security/ms09-008.aspx

【8】 MS12-074, CVE-2012-4776

https://technet.microsoft.com/en-us/library/security/ms12-074.aspx

【9】 MS16-063,CVE-2016-3213

https://technet.microsoft.com/en-us/library/security/ms16-063.aspx

【10】 MS16-077,CVE-2016-3213, CVE-2016-3236

https://technet.microsoft.com/en-us/library/security/ms16-077.aspx