Attacking Automatic Wireless Network Selection

Dino A. Dai Zovi, Shane A. Macaulay
ddz@theta44.org, ktwo@ktwo.ca

March 18, 2005

Abstract

Wireless 802.11 networking is becoming so prevalent that many users have become accustomed to having available wireless networks in their workplace, home, and many public places such as airports and coffee shops. Modern client operating systems implement automatic wireless network discovery and known network identification to facilitate wireless networking for the end-user. In order to implement known network discovery, client operating systems remember past wireless networks that have been joined and automatically look for these networks (referred to as *Preferred* or *Trusted* Networks) whenever the wireless network adapter is enabled. By examining these implementations in detail, we have discovered previously undisclosed vulnerabilities in the implementation of these algorithms under the two most prevalent client operating systems, Windows XP and MacOS X. With custom base station software, an attacker may cause clients within wireless radio range to associate to the attacker's wireless network without user interaction or notification. This will occur even if the user has never connected to a wireless network before or they have an empty Preferred/Trusted Networks List. We describe these vulnerabilities as well as their implementation and impact.

1 Introduction

IEEE 802.11 wireless networking has demonstrated explosive growth and popularity, especially in dense urban areas. This has resulted in commercial offerings of public access wireless networks (*hotspots*) in many airports, hotels, coffee shops, and even some parks. Large hotspot providers include T-Mobile and Verizon. There are even community-based projects to provide free hotspots in community areas like Manhattan parks [1].

The prevalence of these hotspots has had an unanticipated effect on the mechanisms in client operating systems for selecting wireless networks. It has been a known problem that an attacker can provide a rogue access point with a common name (such as the default SSID of a popular home-office access point, such as *linksys*). If a nearby wireless client has associated to a similarly-named access point in the past, they may mistake the rogue access point for their trusted access point. The prescribed solution to this is to ensure that all networks connected to are encrypted. While this is possible when the only networks connected to are at the home or workplace, the use of hotspots (which must be unencrypted to provide public access) means that users are more likely to have connected to unencrypted networks in the past.

According to a Gartner research report (quoted in [2]), Microsoft Windows and Apple MacOS are the two most prevalent desktop operating systems, with 96% and 2.8% market share, respectively. Our research examined the latest releases of these desktop operating system families, Windows XP Service Pack 2 and MacOS X 10.3.8. We describe in detail the mechanisms used by the latest releases of these operating systems for automatic network selection and how they may be attacked. Our research in this area has also uncovered vulnerabilities in the implementation of these algorithms. Namely, the implementations set the wireless network adapter in a "parked" mode when the user is not associated to a network. In this mode, the card's "desired SSID" setting is set to a value that the implementor has expected to never exist as an available network. When a network by this name does exist (or at least appears to), however, the wireless network card will automatically associate. This occurs without user interaction or notification.

This paper is organized as follows. Section 2 provides an overview of the concepts involved in 802.11 wireless networking and describes related work in 802.11 client security research. Section 3 documents

in detail how Windows XP and MacOS X implement automatic wireless network selection. Section 4 describes vulnerabilities in the implementations of automatic wireless network selection and weaknesses in the algorithms they use. In section 5, we describe the implementation of a customized software access point driver to exploit the previously described vulnerabilities. Section 6 discusses the ramifications of these vulnerabilities and describes future work in this area of research.

2 Background

The IEEE 802.11 standard [3] specifies medium access control (MAC) and physical layer (PHY) operation for local area wireless networking. It is now the most common standard for wireless networking and most laptops now include integrated wireless network cards using the 802.11 standard.

The 802.11 standard defines two entities in a wireless network, the Station (STA) and Access Point (AP). A wireless network, or Basic Service Set (BSS) as it is referred to in the standard, may be created in two configurations: the Independent Basic Service Set (IBSS) and the Extended Service Set (ESS). Every IBSS or ESS is named by a Service Set Identifier (SSID), usually a 7-bit ASCII string with a length not exceeding 32 characters. An IBSS, or *Ad-Hoc network*, is created by any number of stations without requiring any Access Points. The more common network configuration, an ESS or *Infrastructure network*, is created with one or more Access Points creating a Distribution System (DS) that clients may join. The most common configuration is an Infrastructure network with one Access Point.

In order to understand the attacks introduced in this paper, we must detail the 802.11 frame types used for locating and joining networks. In order to announce its presence, each AP in an ESS broadcasts *Beacon* frames containing the SSID and various characteristics of the ESS, including supported data rates as well as whether encryption is enabled. Stations may locate nearby networks by observing these Beacon frames or by sending *Probe Request* frames. The Probe Request frame contains the SSID the STA is looking for as well as the transfer rates supported by the STA. The SSID may be an empty string indicating that the frame is a broadcast Probe Request. Access Points within signal range typically respond to both broadcast Probe Requests and Probe Requests containing their SSID with a *Probe Response* frame containing the network's SSID[1], supported rates, and whether the network is encrypted. If encryption is enabled, the STA must authenticate prior to associating to the network. This is performed through a sequence of *Authentication* frames. If the STA has properly authenticated itself or the network does not require authentication, the STA will send an *Association Request* frame to which the AP responds with an *Association Response* frame. At this point the STA may begin to participate on the wireless network.

It is important to point out that while the standard specifies how a STA joins a ESS, how the ESS is chosen is unspecified. The specification allows for roaming between base stations with the same SSID, but there is no mention of whether the base station should be authenticated or simply trusted. The specification also does not address how a wireless client is to select an available network, and this has been left up to implementation by hardware and operating system software vendors.

Although the security of wireless networks has been the subject of much research, the security of wireless clients has not had the same level of focus. There has been some level of research and related work that we have been able to build upon which we will briefly describe.

Vulnerabilities in the 802.11 MAC layer have been discovered that allow nearby attackers to launch a denial-of-service attack against nearby wireless clients, effectively "jamming" the wireless network [5].

The Wireless LAN 802.11b Security FAQ [6] describes several attacks against wireless clients. The FAQ mentions how a cloned base station (*Evil Twin*) can produce a stronger signal than the legitimate base station and divert unsuspecting clients away from the original base station. The FAQ also mentions that nearby wireless attacker can target vulnerable TCP/IP services or perform denial-of-service attacks against a nearby wireless client.

Max Moser's Hotspotter [7] is an automated wireless client penetration tool for Linux. Hotspotter places the attacker's wireless network card in monitor mode to passively listen for Probe Request frames. For each Probe Request frame received, the requested SSID is compared to a list of known hotspot names. If there is a match, the wireless network card is reconfigured to act as an access point with that name.

[1] In response to a well-publicized practice called *Wardriving* [4] many Access Points support an option to ignore broadcast Probe Requests and not broadcast the SSID in Beacon frames. Such access points are referred to as a *closed* or *hidden*.

3 Wireless Network Selection

3.1 Microsoft Windows XP Wireless Auto Configuration

Microsoft Windows XP and Windows Server 2003 were the first Microsoft operating systems to fully support 802.11 wireless networking on the client and server, respectively. Under these operating systems, wireless network configuration and detection is performed through a process called Wireless Auto Configuration.

Central to the configuration and operation of Wireless Auto Configuration are the *Preferred Networks List* (PNL) and *Available Networks List* (ANL). The PNL is an ordered list of the networks that the user has connected to in the past. The ANL is an ordered list of all the Access Points that responded to a broadcast Probe Request in the last wireless network scan. Whenever the user connects to a new network, that network's name (SSID) is added to the head of the PNL. Windows provides user interfaces for viewing the Available Networks List, managing the Preferred Networks List, and configuring the behavior of the Wireless Auto Configuration algorithm, described below.

The Windows XP and Server 2003 Wireless Auto Configuration algorithm [8] is presented in a pseudocode notation in Figure 1 but will also be presented in narrative and a network trace in order to fully illustrate its behavior.

The algorithm begins by building the Available Networks List by sending a scan request to the wireless network card. This results in a broadcast 802.11 Probe Request with an empty SSID being sent over every channel. Probe Responses are collected by the card in the order they are received and this list of networks along with configuration details such as encryption status and available rates is returned to the operating system. The algorithm then traverses the Preferred Networks List in order and if a preferred network is found in the Available Networks List, Windows attempts to connect to the wireless network. A connection is attempted to each preferred network found in the Available Networks List until there is a successful connection.

If no preferred networks were found in the Available Networks List or no connection attempts were successful, Windows attempts a second pass of the Preferred Networks List trying to connect to each network in the Preferred Networks List in order regardless of whether the network exists in the Avail-

```
Begin:
State = Unconnected
// Build list of visible networks (ANL)
AvailableNetworks = ScanForAvailableNetworks()

// Step through PNL in order until a network
// from the ANL is found and connected to
foreach n in PreferredNetworks
  if AvailableNetworks contains n
  then ConnectToWirelessNetwork(n)
  if State == Connected then return

// If unable to connect to any networks in the
// intersection of the PNL and ANL, check for
// closed networks by stepping through PNL in
// order and attempting each network explicitly
foreach n in PreferredNetworks
  ConnectToWirelessNetwork(n)
  if State == Connected then return

// If unable to connect to any network in the
// PNL and the PNL contains an Ad-Hoc network,
// configure card for the first Ad-Hoc network
// in the PNL.  Otherwise, if the configuration
// setting "Connect to Non-preferred Networks"
// is enabled, step through ANL in order and
// attempt to connect to each one.
if PreferredNetworks contains an Ad-Hoc network
  then ConfigureAdHocNetwork
else
  if ConnectToNonPreferredNetworks == True
  then foreach n in AvailableNetorks
         ConnectToWirelessNetwork(n)
         if State == Connected then return

  // If not connected thus far, generate a
  // random SSID, wait, and restart algorithm
  SetSSID(GenerateRandomSSID())
  SleepForOneMinute()
  Goto Begin
```

Figure 1: Pseudocode for Wireless Auto Configuration Algorithm

```
1) 03:11:39.266743 BSSID:ff:ff:ff:ff:ff:ff
  DA:ff:ff:ff:ff:ff:ff SA:00:02:2d:2b:a5:35
  Probe Request (^]^V^K^A^T^B^T^A^X^E^Y^V...)
    [1.0 2.0 5.5 11.0 Mbit]
2) 03:11:39.400194 BSSID:ff:ff:ff:ff:ff:ff
  DA:ff:ff:ff:ff:ff:ff SA:00:02:2d:2b:a5:35
  Probe Request () [1.0 2.0 5.5 11.0 Mbit]
3) 03:11:40.426391 BSSID:ff:ff:ff:ff:ff:ff
  DA:ff:ff:ff:ff:ff:ff SA:00:02:2d:2b:a5:35
  Probe Request (^]^V^K^A^T^B^T^A^X^E^Y^V...)
  [1.0 2.0 5.5 11.0 Mbit]
4) 03:11:42.377495 BSSID:ff:ff:ff:ff:ff:ff
  DA:ff:ff:ff:ff:ff:ff SA:00:02:2d:2b:a5:35
  Probe Request (aye) [1.0 2.0 5.5 11.0 Mbit]
5) 03:11:44.432180 BSSID:ff:ff:ff:ff:ff:ff
  DA:ff:ff:ff:ff:ff:ff SA:00:02:2d:2b:a5:35
  Probe Request (bee) [1.0 2.0 5.5 11.0 Mbit]
6) 03:11:46.485148 BSSID:ff:ff:ff:ff:ff:ff
  DA:ff:ff:ff:ff:ff:ff SA:00:02:2d:2b:a5:35
  Probe Request (sea) [1.0 2.0 5.5 11.0 Mbit]
7) 03:11:48.500975 BSSID:ff:ff:ff:ff:ff:ff
  DA:ff:ff:ff:ff:ff:ff SA:00:02:2d:2b:a5:35
  Probe Request (^D^F^R^_^]^R^^^\^G^H^J^F...)
  [1.0 2.0 5.5 11.0 Mbit]
```

Figure 2: Wireless Auto Configuration Algorithm packet trace

able Networks List. This second pass is performed in case any of the networks are "closed networks", a common deviation from the standard where the SSID is not placed in 802.11 Beacon frames and broadcast Probe Requests are not responded to.

If the wireless network card is not connected and there are one or more Ad-Hoc networks in the Preferred Networks List, Wireless Auto Configuration will configure the card for the most preferred Ad-Hoc network available. If none are available, Wireless Auto Configuration will create the most preferred Ad-Hoc network and the algorithm terminates.

If there have been no connections thus far and there are no Ad-Hoc networks in the Preferred Networks List, the algorithm examines the *Connect To Non-preferred Networks* flag. If the flag is enabled (it is disabled by default), Windows will attempt to connect to each network in the Available Networks List in order. If the flag is not set, the network card is "parked" in Infrastructure mode with a randomly generated SSID for 60 seconds at which point the algorithm restarts.

When analyzing the algorithm from an attacker's point of view, it is most helpful to examine it at the 802.11 protocol layer. Figure 2 is a packet trace of a single iteration of the Wireless Auto Configuration algorithm running on a laptop with a fresh install of Windows XP with Service Pack 2. The network trace was recorded on a nearby laptop with its wireless network card in "monitor mode", a special operating mode where all received 802.11 frames are returned to the operating system. The output has been edited slightly for cleaner presentation by numbering frames, removing duplicate frames, wrapping lines cleanly, and abbreviating long random SSIDs.

Frames 1-3 show the first phase of the algorithm where the network card is "parked" with a random SSID, but sends a broadcast Probe Request frame when Wireless Auto Configuration initiates a scan request. Frames 4-6 result from the second phase of the algorithm where a connection to each of the networks in the Preferred Networks List is attempted in order. We observe that the networks in the Preferred Networks List are (in order): "aye", "bee", and "sea". Since none of these networks were found, Frame 7 shows the wireless card being parked with another random SSID. In the full trace, 16 Probe Requests for the last random SSID were observed over one minute before the algorithm on the observed wireless client restarted.

3.2 Apple MacOS X AirPort

Apple's MacOS X operating system supports wireless networking with Apple's AirPort and AirPort Extreme 802.11 wireless networking products for 802.11b and 802.11g, respectively.

MacOS X allows the user to select three wireless network selection modes: always connecting to a specific wireless network, connecting to the most recently associated network or automatically connecting to the unencrypted network with the strongest signal. The user may also manually select an alternate available network or enter the name of a closed network. In December 2003, a vulnerability was published whereby if an attacker can get a MacOS X user to join their wireless network, the attacker's DHCP server may provide an DHCP option specifying a directory server that the user's machine will use as an authentication server. MacOS X 10.3.3 addressed this vulnerability by modifying the system to maintain a list of trusted wireless networks.

The MacOS X trusted wireless networks is a system-wide list of networks the user has connected to

and opted to add to the list. No user interface is provided to view or modify this list. The list, in fact, is even fairly difficult to find. It is stored as an XML file which is base-64 encoded and stored within another XML file containing basic wireless network adapter settings. Cursory Internet searches reveal that the existence of this list is completely undocumented, yet many have discovered that they can delete the entire file to clear the list of trusted wireless networks.

MacOS X begins the search for trusted wireless networks when a user logs in or the machine awakes from sleep. The search begins with the network the machine was most recently associated with. If this network is not found, each network in the trusted network is attempted in order. If none of these networks are found, a dialog is presented to the user stating that none of their trusted wireless networks could be found and asking whether they would like to join the unencrypted network with the strongest signal and optionally remember the network as a trusted wireless network. If the user opts not to connect to the selected network, the wireless network card is "parked" awaiting user interaction. In this state, the wireless network card remains in Infrastructure mode, however, it is assigned a either a constant "dummy" SSID or a dynamic SSID that is chosen when the driver is initialized (system boot or resume). Both settings have been seen, although the dummy SSID appears to be set at system boot and when awakening from sleep, but the dynamic SSID is only set when the user logs in. Broadcast Probe Requests are sent on each channel roughly every two minutes or when the available networks need to be presented to the user.

4 Attacking Wireless Network Selection

Through the detailed examination and documentation of the processes used by the two most prevalent desktop operating systems, a number of vulnerabilities in these processes were uncovered. The specific vulnerabilities in each implementation is detailed below and the attacks are summarized in terms of which configurations are vulnerable in section 4.3.

4.1 Microsoft Windows Wireless Auto Configuration

Windows' Wireless Auto Configuration has several serious weaknesses: all networks and their precedence in the Preferred Networks List are revealed, ad-hoc networks are automatically created, and "parking" with a random SSID does not prevent associating to an Access Point with custom firmware modifications. Moreover, this association may occur without user interaction or notification. Through these weaknesses, the wireless client is most vulnerable when it is not associated to a surrounding network.

When no preferred networks are discovered in the Available Networks List built in the first phase of the Wireless Auto Configuration algorithm, the algorithm attempts to connect to each network in the Preferred Networks List in order. An attacker within signal range (potentially assisted by high-gain antennae and signal boosters) may passively monitor a single wireless channel and observe the Probe Requests for each network connection attempted. The attacker may use this information to recreate the victim's Preferred Networks List. However, only the names and precedence of the networks are revealed, their encryption status is not. With this knowledge, the attacker may create a software Access Point with the SSID of one of the networks in the victim's Preferred Network List. If the client is expecting the network to be encrypted, the connection will fail and the attacker may simply attempt the next network in their recreated copy of the victim's Preferred Networks List. If *any* of the networks in the Preferred Networks List are not encrypted, the attacker will have an opportunity to create a look-alike network that the client will join.

If there are any Ad-Hoc networks in the Preferred Networks List and none are found in the Available Networks List, the client will become the first node in the Ad-Hoc network. If this network is not configured to be encrypted, any other client within wireless signal range may join this network. If the network is configured with WEP encryption, any of the known WEP attacks may be performed against it. Several of these attacks are facilitated by the fact that the network interface is configured using Windows' Automatic Private IP Addressing whereby the interface is given an automatically selected IP address from the link local IP address space 169.254.0.0/16 documented in RFC 3330 [9]. Because the interface is configured, Windows will periodically send NetBIOS broadcasts, continually supplying the attacker with WEP encrypted data packets. Once the network is joined by the attacker, they can proceed to discover the self-assigned IP address used by the victim's wireless network interface. This may be done by sniffing

the network for the previously mentioned NetBIOS broadcasts or brute-forcing the link local IP address space with Address Resolution Protocol (ARP) requests for each possible IP address.

As described above, when the Wireless Auto Configuration algorithm sleeps for 60 seconds before restarting, the wireless network card is "parked" by placing the card in Infrastructure mode with a random SSID. Other card settings such as authentication mode or encryption are not changed. When the card is placed in this state, it assumes its normal behavior attempting to connect to a network with that name. If there are no entries in the Preferred Networks List or the last entry is an unencrypted network, the network adapter will attempt to connect to an unencrypted network with this random SSID. If an attacker can provide a network with that name, the wireless client will automatically associate to the attacker's wireless network.

This attack may be performed by modifying the firmware of an Access Point to respond with Probe Responses to Probe Requests for *any SSID*. This attack is facilitated by using the newer firmware-less wireless network cards (such as those using Atheros chipsets). We describe the necessary driver modifications and implementation of this attack in Section 5. As this network is joined without the Wireless Auto Configuration Service's knowledge, the user interface does not inform the user that they have associated to a network. In this case, the client may associate, receive an IP address from a DHCP server, and become reachable on the network all while the interface informs the user that they are not currently connected to a network.

4.2 MacOS X AirPort

MacOS X's AirPort implementation shares several of the pitfalls identified in the Windows Wireless Auto Configuration implementation but avoids others. Similar to the vulnerabilities described above, MacOS X AirPort reveals the list of trusted wireless networks and machines using the 802.11b AirPort hardware may associate to a specially-configured access point without user interaction or notification.

Like Microsoft Windows' Wireless Auto Configuration, MacOS X AirPort reveals the list of trusted wireless networks when the system is looking for a network to associate to. This is, however, an important difference because MacOS X AirPort does not continually look for the user's trusted wireless networks, only when a user logs in or the system resumes from sleep. This makes it more difficult for an attacker to cause the user to join their rogue access point as it adds temporal constraints, requiring the attacker to be in the right place at the right time.

The drivers for the older 802.11b AirPort hardware, like Wireless Auto Configuration, also keep the wireless card up in a "parked" state when not actively associated. As described above, the card's desired SSID is set to either a dynamic value or a static "dummy SSID". In order to further prevent unintended associates, the card is set with WEP enabled, requiring a rogue Access Point to know this WEP key in order to cause the client to join automatically. This WEP key, however, is the hard-coded, static 40-bit key (in hexadecimal) 0x0102030405. Our custom Access Point software configured with this WEP key in Shared Key authentication mode successfully causes nearby AirPort cards to automatically associate without requiring any user interaction or providing any user notification. The AirPort menu applet will, however, light up, and if the user clicks on it, it will indicate that the user is connected to a network. It should be noted that the newer AirPort Extreme hardware and drivers do not leave the card in a "parked" state vulnerable to this attack. When the card is not associated no wireless traffic is sent unless the user requests a scan for visible networks.

4.3 Summary of Attacks

The attacks described above focus on the ability of the attacker to provide a network that the wireless client will automatically join. The attacker may perform this by either discovering the networks that the client prefers or by using a special software access point that masquerades as any SSID. We describe the construction of such a software AP in section 5. Either way, the attacker must make this network present when the victim is looking for a wireless network to join.

If the victim is running MacOS X, the attacker must be present when the user logs in or the system wakes from sleep. Regardless of the operating system, however, if the user is currently associated to a nearby network, the attacker may forcibly cause the victim to restart the search for available networks. The attacker may achieve this by spoofing 802.11 Disassociation frames from the base station that the victim is associated to. These frames are always sent in the clear, even if the network is encrypted, and the only information the attacker requires to forge them is the hardware address of the base station, which is read-

ily available from the Beacon frames the base station is required to continuously transmit. If the targeted client is currently configured in Ad-Hoc mode, there is no known way to cause it to restart the search for a preferred network.

At this point, the attacker may learn the victim's preferred or trusted networks as they attempt to rejoin an available network or respond that they are the base station for every requested SSID. If the targeted client looks for one or more unencrypted networks, they will automatically join the attacker's access point.

As a special case that was mentioned above, if a nearby Windows XP client is not currently associated to a network, it may still associate to a network with a random SSID. When this is the case, this connection will occur without notifying the user and the user interface will still report that the machine is not currently connected to any wireless networks.

The attacks detailed above have been observed and verified against a Windows XP laptop with PCMCIA PrismII and Orinoco Hermes-based 802.11b wireless network cards as well as against a G4 Macintosh with an internal AirPort 802.11b wireless card. We also tested a Windows XP SP2 laptop with an internal 802.11a/b/g card based on the Atheros chipset and found that while it still did send out Probe Requests for randomly-generated SSIDs, it would not join these networks automatically. By testing a newer G4 Powerbook, we found that the newer Apple AirPort Extreme 802.11b/g cards did not probe for the dynamic SSID and dummy SSIDs described above. We hypothesize that the newer generation of wireless network cards that perform more of the 802.11 handling in software are more flexible and robust than their firmware-based predecessors.

```
1) 00:49:04.007115 BSSID:ff:ff:ff:ff:ff:ff
   DA:ff:ff:ff:ff:ff:ff SA:00:e0:29:91:8e:fd
   Probe Request (^J^S^V^K^U^L^R^E^H^V^U...)
   [1.0* 2.0* 5.5* 11.0* Mbit]
2) 00:49:04.008125 BSSID:00:05:4e:43:81:e8
   DA:00:e0:29:91:8e:fd SA:00:05:4e:43:81:e8
   Probe Response (^J^S^V^K^U^L^R^E^H^V^U...)
   [1.0* 2.0* 5.5 11.0 Mbit] CH: 1
3) 00:49:04.336328 BSSID:00:05:4e:43:81:e8
   DA:00:05:4e:43:81:e8 SA:00:e0:29:91:8e:fd
   Authentication (Open System)-1: Succesful
4) 00:49:04.337052 BSSID:00:05:4e:43:81:e8
   DA:00:e0:29:91:8e:fd SA:00:05:4e:43:81:e8
   Authentication (Open System)-2:
5) 00:49:04.338102 BSSID:00:05:4e:43:81:e8
   DA:00:05:4e:43:81:e8 SA:00:e0:29:91:8e:fd
   Assoc Request (^J^S^V^K^U^L^R^E^H^V^U...)
   [1.0* 2.0* 5.5* 11.0* Mbit]
6) 00:49:04.338856 BSSID:00:05:4e:43:81:e8
   DA:00:e0:29:91:8e:fd SA:00:05:4e:43:81:e8
   Assoc Response AID(1) :: Succesful
```

Figure 3: Windows XP host associating to random SSID

5 Attack Implementation

Implementation of attacks against 802.11 networks and clients are often hampered by the limitations imposed by wireless network card firmware. For example, certain frame types may be handled directly by the firmware and will not be made available to the host operating system. This behavior prevents many commercial off-the-shelf 802.11 wireless network cards from being used as an arbitrary attack platform. Initial implementations of the attacks presented in this paper were attempted using wireless network cards based on the PrismII chipset. These chipsets feature a *HostAP* operating mode allowing the card to act as an Access Point. Other cards based on the Orinoco Hermes chipset, for example, require an alternate firmware to provide this functionality. This operating mode makes management frames, including Authentication and Association Request frames, available to the operating system to allow the operating system or a running user process to provide station authentication and management services. Notably missing, however, are the Probe Request frames necessary to implement our attacks. Due to the real-time requirements on Probe Request handling, Probe Request frames are handled internally by the card firmware. This limited our attack to sniffing for Probe Requests and immediately reconfiguring the card to serve as an access point for the requested SSID, serializing the attack to targeting one wireless client at a time.

The newer generation of firmware-less wireless network cards allow significantly more flexibility in the implementation of attacks against IEEE 802.11. Wireless network cards based on chipsets manufactured by Atheros Communications, for example, do not include a traditional firmware providing station and/or access point functionality. Instead, all of this

```
1) 14:14:18.778980 BSSID:ff:ff:ff:ff:ff:ff
   DA:ff:ff:ff:ff:ff:ff SA:00:30:65:00:e9:65
   Probe Request (00-30-1e-37-7a-44-7f49c08d)
   [1.0 2.0 5.5 11.0 Mbit]
2) 14:14:18.779845 BSSID:00:05:4e:43:81:e8
   DA:00:30:65:00:e9:65 SA:00:05:4e:43:81:e8
   Probe Response (00-30-1e-37-7a-44-7f49c08d)
   [1.0* 2.0* 5.5 11.0 Mbit] CH: 1, PRIVACY
3) 14:14:18.837813 BSSID:ff:ff:ff:ff:ff:ff
   DA:ff:ff:ff:ff:ff:ff SA:00:30:65:00:e9:65
   Probe Request (00-30-1e-37-7a-44-7f49c08d)
   [1.0 2.0 5.5 11.0 Mbit]
4) 14:14:18.906312 BSSID:00:05:4e:43:81:e8
   DA:00:05:4e:43:81:e8 SA:00:30:65:00:e9:65
   Authentication (Shared Key)-1: Succesful
5) 14:14:18.907962 BSSID:00:05:4e:43:81:e8
   DA:00:30:65:00:e9:65 SA:00:05:4e:43:81:e8
   Authentication (Shared Key)-2 [Challenge]
6) 14:14:18.909513 BSSID:00:05:4e:43:81:e8
   DA:00:05:4e:43:81:e8 SA:00:30:65:00:e9:65
   AuthenticationAuthentication (Shared-Key)-3
   Data IV:  0 Pad 0 KeyID 0
7) 14:14:18.910320 BSSID:00:05:4e:43:81:e8
   DA:00:30:65:00:e9:65 SA:00:05:4e:43:81:e8
   Authentication (Shared Key)-4:
8) 14:14:18.911565 BSSID:00:05:4e:43:81:e8
   DA:00:05:4e:43:81:e8 SA:00:30:65:00:e9:65
   Assoc Request (00-30-1e-37-7a-44-7f49c08d)
   [1.0 2.0 5.5 11.0 Mbit]
9) 14:14:18.912575 BSSID:00:05:4e:43:81:e8
   DA:00:30:65:00:e9:65 SA:00:05:4e:43:81:e8
   Assoc Response AID(1) : PRIVACY : Succes
```

Figure 4: AirPort client parked with dynamic SSID associating

functionality is handled in software on the host and a binary-only Hardware Abstraction Layer (HAL) module is shipped to vendors and driver authors to provide low-level functionality. This HAL module enforces communications regulation compliance and provides a consistent interface to the different implementations manufactured by Atheros. This binary-only HAL is also used by the open-source drivers for this family of wireless network cards.

To implement our attacks, we have taken the open-source MADWiFi driver for Linux [10] and made several modifications. We implemented several trivial modifications including disabling SSID validation and rewriting the SSID field of incoming Probe Requests

```
1) 14:25:05.926870 BSSID:ff:ff:ff:ff:ff:ff
   DA:ff:ff:ff:ff:ff:ff SA:00:30:65:00:e9:65
   Probe Request (dummy SSID *{M-^R^LIM-^O...)
     [1.0 2.0 5.5 11.0 Mbit]
2) 14:25:05.927916 BSSID:00:05:4e:43:81:e8
   DA:00:30:65:00:e9:65 SA:00:05:4e:43:81:e8
   Probe Response (dummy SSID *{M-^R^LIM-^O...)
   [1.0* 2.0* 5.5 11.0 Mbit] CH: 1
3) 14:25:06.095809 BSSID:00:05:4e:43:81:e8
   DA:00:05:4e:43:81:e8 SA:00:30:65:00:e9:65
   Authentication (Open System)-1: Succesful
4) 14:25:06.096565 BSSID:00:05:4e:43:81:e8
   DA:00:30:65:00:e9:65 SA:00:05:4e:43:81:e8
   Authentication (Open System)-2:
5) 14:25:06.098862 BSSID:00:05:4e:43:81:e8
   DA:00:05:4e:43:81:e8 SA:00:30:65:00:e9:65
   Assoc Request (dummy SSID *{M-^R^LIM-^O...)
   [1.0 2.0 5.5 11.0 Mbit]
6) 14:25:06.099773 BSSID:00:05:4e:43:81:e8
   DA:00:30:65:00:e9:65 SA:00:05:4e:43:81:e8
   Assoc Response AID(1) :: Succesful
```

Figure 5: AirPort client parked with dummy SSID associating

and Association Request frames with the configured SSID of the software access point. Outgoing frames required no modification. This allowed us to easily fool the rest of the driver into serving any SSID requested. This implementation allows us to create a wireless network that masquerades as any network requested by a client within range.

Figures 3, 4, and 5 show 802.11 frame traces of our attack driver exercising the previously detailed vulnerabilities in wireless network selection. In each of the three cases, the wireless client associated to a network name that the implementation did not expect to exist but was responded to by our attack driver. This allowed us to cause the client to associate to our network without any user interaction or notification.

Further work on the attack driver component will focus on creating a network interface that will enable reception and transmission of raw frames, even while serving as an access point. This functionality will allow a user to utilize existing tools requiring passive sniffing capabilities while also enabling active attacks. In addition, it may enable true 802.11 MAC-layer man-in-the-middle attacks.

6 Conclusion

We have shown that there are both architectural and implementation vulnerabilities in common wireless network selection algorithms. The broadcasting of specific Probe Request frames containing the SSID of a user's desired network allows a nearby attacker to learn the contents and precedence of the user's desired networks. With this knowledge, the attacker may create a rogue access point with this SSID that the user will automatically associate to.

In addition, both implementations also leave the wireless network cards in a vulnerable state when they are not currently associated to a network. Windows XP configured the network card with a randomly generated SSID while MacOS X AirPort uses a dynamic, but not random, SSID. This is done assuming that no access points will ever serve these SSIDs. We created a special access point that responded to any Probe Request, allowing us to serve any SSID requested by a client within range. Experiments with this driver revealed serious vulnerabilities in the previously mentioned driver behavior. We discovered that when our access point responded to probes for these special SSIDs, we could cause clients to associate to our network without any user interaction and little or no user notification. Under Windows XP, the wireless configuration user interface will report that the user is not associated to any wireless networks. Network interface configuration elements, however, will report that the interface is connected and configured with an IP address. MacOS X AirPort will not request permission to join a previously unknown wireless network, but will correctly report that it is connected.

While there is a known growth of client-side attacks, there has been a lack of knowledge regarding how they may be realistically used by an attacker to achieve their objectives. Frequently, client-side attacks are described as being a danger if the attacker can cause the victim to view a malicious web page or e-mail. While it is evident that an adversary may send a large number of e-mails to harvested addresses within an organization to increase the chances of an internal user viewing the message and possibly viewing the malicious content within, this is in no way a stealthy attack. Similarly, coercing internal users to view a malicious web site requires an element of social engineering. We believe that a wirelessly connected attacker is in the best position to deploy attacks against client-side applications.

We have shown that an attacker can force wireless clients within signal range to join an attacker-controlled network. This opens up a very viable and dangerous avenue for passively exploiting client-side vulnerabilities using man-in-the-middle techniques. Drawing from existing cyber-warfare principles ([11]), we are calling this scenario *medium-range cyber-warfare*. By exploiting the vulnerabilities described in this paper, we may cause our opponent to join a wireless network where we control the entire network environment. Since we control the entire network, we may leverage this to control the opponent. We may use this approach to attack any wireless-enabled clients within range, and we may employ strong wireless antennas and transmitters to increase this range.

Many attacks can be easily implemented as fake services responding to requests by client-side applications. Bare-bones emulation servers can often be developed with ease [12] or existing packages may be modified to exercise vulnerabilities in client code. For example, while implementing a malicious POP3 server is quite simple due to the simplicity of the POP3 protocol, implementing a malicious SMB server may require modifying an existing implementation such as the open-source Samba SMB/CIFS server[13].

With a client associated to the rogue network, credentials may be captured if the client's software attempts to automatically connect or re-connect to a network service. For example, rogue mail servers can capture credentials from clients connecting to clear-text mail services such as POP3 and IMAP.

In some cases, the client's credentials may be used against the client itself. Older variants of the Microsoft Networking NTLM authentication mechanism are vulnerable to a man-in-the-middle attack where the attacker may proxy the challenge-response protocol in order to authenticate as the client to the server.

If a host is compromised, an agent may be placed on it that yields remote control to the attacker. The agent will attempt to establish communication with the attacker whenever a network connection is present. This effectively gives the attacker access to all the networks the wireless client has access to, exploiting the inherent mobility of wireless clients. This effectively makes the security of every network the client connects to dependent upon the security of all other networks they connect to.

Our future work will further explore the vulnerabilities in client-side functionality and client mobility and investigate the construction of a client-side attack toolkit to demonstrate these risks.

References

[1] "NYC wireless." http://www.nycwireless.net.

[2] S. Lohr, "One small step in uphill fight as linux adds a media player," *The New York Times*, June 28, 2004.

[3] IEEE Computer Society LAN/MAN Standards Committee, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," tech. rep., ANSI/IEEE, 1999.

[4] P. Shipley, "Open WLANs: The early results of wardriving." http://www.dis.org/filez/openlans.pdf.

[5] P. Nobles and P. A. Horrocks, "Vulnerability of IEEE802.11 WLANs to MAC layer DoS attacks," in *Proceedings of The 2nd IEE Secure Mobile Communications Forum*, Instituion of Electrical Engineers, 2005.

[6] C. W. Klaus, "Wireless LAN security FAQ." http://www.iss.net/wireless/.

[7] M. Moser, "Hotspotter: Automatic wireless client penetration." http://new.remote-exploit.org/index.php/Hotspotter_main.

[8] The Cable Guy, "Windows XP wireless auto configuration," *Microsoft TechNet*, November 2002.

[9] Internet Assigned Numbers Authority, "RFC 3330: Special-use IPv4 addresses." ftp://ftp.internic.net/rfc/rfc3330.txt, September 2002.

[10] S. Leffler, "Multimode atheros driver for WiFi on linux." http://madwifi.sourceforge.net.

[11] R. C. Parks and D. P. Duggan, "Principles of cyber-warfare," *Proceedings of the IEEE Workshop on Information Assurance and Security*, pp. 122–125, June 2001.

[12] N. Provos, "A virtual honeypot framework," in *Proceedings of The 13th USENIX Security Symposium*, (San Fransisco, CA), August 2004.

[13] The Samba Team, "Samba." http://www.samba.org.