CERIAS Tech Report 2002-17

ActiveSync, TCP/IP and 802.11b Wireless Vulnerabilities of WinCE-based PDAs

by Pascal Meunier, Sofie Nystrom, Seny Kamara,
Scott Yost, Kyle Alexander, Dan Noland, Jared Crane

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907

ActiveSync, TCP/IP and 802.11b Wireless Vulnerabilities of WinCE-based PDAs

Pascal Meunier, Sofie Nystrom, Seny Kamara, Scott Yost, Kyle Alexander, Dan Noland, Jared Crane
Center for Education and Research in Information Assurance Security (CERIAS), 1315 Recitation Building, Purdue University, West Lafayette, IN 47907-1315; winCE@cerias.purdue.edu

Abstract

Researching the vulnerabilities and security concerns of WinCE-based Personal Digital Assistants (PDAs) in an 802.11 wireless environment resulted in identifying CAN-2001-{0158 to 0163}. The full understanding and demonstration of some vulnerabilities would have required reverse engineering ActiveSync, which was beyond the scope of this research. Moreover, the WinCE IP stack demonstrated unstabilities under a number of attacks, one of which produced symptoms in hardware. The inaccessibility of the 802.11b standard documentation was a source of delays in the research; however, we created three proof-of-concept applications to defeat 802.11b security. One collects valid MAC addresses on the network, which defeats MAC-address-based restrictions. Another builds a code book using known-plaintext attacks, and the third decrypts 802.11b traffic on-the-fly using the code book.

Keywords:WinCE, WEP, ActiveSync, wireless, security, 802.11b, vulnerability

1. Introduction

Personal Digital Assistants (PDAs) may be vulnerable to accidents during transport or attacks against the services and protocols used. Networked PDAs may be vulnerable to a number of TCP/IP attacks, especially resource exhaustion attacks due to the limited resources provided by the PDA. Other attacks against networked PDAs would include malicious content downloaded directly or through a service such as avantgo.com, hostile JavaScript, ActiveX and Java. An aspect particular to WinCE-based PDAs is the availability of synchronization services through the ActiveSync protocol over a serial link, infrared or over TCP/IP; therefore ActiveSync can also be attacked. The convenience of PDAs is enhanced by wireless access, which makes them vulnerable to attacks against the wireless standard 802.11b.

Whereas normal computers may be locked in a trusted area, portables may be carried into unsafe areas, and may also be whisked away by an attacker in a moment of distraction or during a break and replaced after installing a keylogger or other trojan program. Therefore, we believe that physical attacks on portable devices are much more likely than against normal computers; in particular, we are concerned about the speed of the attacks which may enable "tainting" the portable device.

ActiveSync was created to provide file synchronization capabilities for WinCE-based PDAs. It provides both network-based and local (USB or serial) synchronization capabilities. The impersonation of a host or PDA, as well as attacks on the host providing the service were considered. However, ActiveSync specifications are proprietary (closed) and were unavailable to us. Therefore, we were limited to probing the outside of this black box without being able to provide a comprehensive analysis.

In an 802.11b wireless environment, several measures and features are supposed to provide some measure of security. These are the SSID (Service Set Identifier), MAC address restriction, and the wireless encryption protocol (WEP). Networks are identified and segregated by an SSID. The SSID can be broadcast using "beacon" frames. So-called "closed" networks do not broadcast SSIDs, so the users have to know the name of the network. However, finding valid SSIDs is trivial with any network sniffer. Therefore, we focused on MAC address restriction, and the wireless encryption protocol (WEP). Our methods of attack have been found independently by other teams and announced first through unrefereed channels [1, 2], even though we had independently identified those we used before their announcements. However, we implemented these attacks and found them effective and practical, contrary to claims by the 802.11b chairperson [6]. We also found implementation errors in 802.11b hardware.

2. Materials and Methods

The initial testbed comprised 3 Aero 1550 Pocket PC devices by Compaq, an Aironet card and 2 base stations AP340 by Cisco (gifts of Microsoft) as well as two PCs belonging to CERIAS. One PC ran Windows 2000 and served as a synchronization host for the handhelds. Another was able to boot in either Windows 2000 or Linux Red Hat 6.2; this PC had a PCMCIA card reader and was used to sniff wireless traffic and as a source of attacks against the synchronization host and the Cisco base station. It was found that the Aero 1550 devices were incapable of powering the Aironet card due to incompatible voltages. We later received two PocketPC iPaq devices by Compaq, two Ethernet cards and two more wireless cards (Orinoco WaveLan). The iPaqs were running WinCE 3.0.9348 (Build 9456).

3. ActiveSync

a) Synchronization over a serial link

The most direct ActiveSync connection may be established by placing the device into a charging cradle which is connected to the serial port of a PC. After the initial connection the user at the PC will be prompted to authenticate himself/herself if and only if the device is set to require authentication. Authentication for ActiveSync comes in the form of a 4 decimal digit personal identification number (PIN) . Should a PDA be left briefly unattended, it is possible to try to synchronize it in the cradle of a hostile computer.

If the correct PIN is supplied then the device and the PC compare the files that are to be synchronized and transfer them as necessary. If an incorrect PIN number is supplied, then an error message is displayed and the user may guess twice more before the connection is broken. To re-establish a new connection, with three more attempts, seemed at first to require removing and replacing the PDA in the cradle. However, we discovered that the unit does not have to be removed physically from the cradle. The connection can be reset programatically and three more successive tries are granted.

We were successful in making a proof-of-concept (unoptimized and slow) brute force attack in the cradle by resetting the connection every three password attempts. We communicated our findings to Microsoft and suggested that an exponential delay be implemented, which would deny for a while new connection attempts to the PDA. The delay would be increased by a hard coded factor after every failed connection attempt.

The brute force program was a Visual Basic script that utilized a function called SendKeys(), which sends keystrokes through the GUI interface. By sending key strokes to the ActiveSync connection application on the PC, it was possible to establish a connection, try three passwords and reconect to try it again. Obviously, anyone who understands the internals of ActiveSync can directly brute force the 4 required digits in a fraction of the time by avoiding the GUI entirely. The key to this vulnerability is the trust relationship between the PDA and the host PC. Because the PDA acts as the authentication server, it should not be trusting the client PC and the software running on it. The PDA should be controlling the exponential backoff, and not the PC.

b) ActiveSync synchronization over a Network

(DoS Attacks CAN-2001-0158, CAN-2001-0159)

ActiveSync listens on port 5679 of a host PC for a PDA attempting a network synchronization. If a PDA is synchronized through the cradle, port 5679 is closed. The availability of the network synchronization function can be enabled or disabled by the user through an option in the file menu of the host.

We were able to enact a denial of service (DoS) attack and remotely close the port by establishing a connection to it and feeding ActiveSync any line that was longer than seven characters. The port stayed closed until the network synchronization option was locally re-enabled on the desktop machine. Anyone with TCP/IP connectivity to the PC could shut down the service, which is potentially anyone on the internet.

Once open, port 5679 stayed open even if network synchronization was disabled. Another DoS attack was that as long as a connection to port 5679 was established, even if there were no authentication taking place (no traffic), ActiveSync would not allow any other device to synchronize by network or by cradle. This lasted for about 20 seconds at which point the connection was closed and became available once again. This could also be exploited remotely in order to prevent legitimate users from synchronizing. By simply establishing a connection to the port, closing it before the 20 seconds elapsed and then re-establishing it again, we could keep the port continuously busy, thus making ActiveSync refuse all other synchronization requests.

If the protocol specifications and the code had been available to us, we would have been able to assess whether the vulnerabilities were protocol flaws or implementation problems.

4. TCP/IP Vulnerabilities

a) Random IP packets and DoS attacks

In order to test the WinCE IP stack, we used a tool named ISIC [4] that generates random IP packets. ISIC is a suite of utilities to exercise the stability of an IP Stack and its component stacks (TCP, UDP, ICMP et. al.). It generates pseudo-random packets of the target protocol. The packets can be given characteristics, e.g., 40% of the packets generated can have IP Options. The percentages are arbitrary and most of the packet fields have configurable parameters. We sent packets to the IP stack of an iPaq using both an Ethernet card and a wireless (Orinoco Silver 802.11b) card.

We were surprised to find that which pseudo-random number generator was used was an important consideration. Some observed WinCE severe crashes could not be obtained with a different random number generator. This is most likely due to the fact that a specific sequence of numbers from a generator is needed in order to produce a sequence of packets. Other pseudo-random generators may be completely unable to produce a sequence that another generator can. Finally, we used a hardware random number generator embedded into an Intel Celeron motherboard, which seemed to be able to produce the entire range of observed effects. No application was running on the iPaq while the packets were sent to it. Thereafter, the functionality of the TCP/IP stack was tested by running Internet Explorer, if possible.

Five test results were randomly observed from sending hundreds of thousands of packets:

1)Loss of Internet capability: the iPaq was no longer able to connect to any outside source, however no local programs seemed affected. Shutting off the iPaq did not restore Internet capability, and a soft reset (pressing the iPaq's reset button) was necessary to restore functionality.

2)Needed hard reset: All programs froze on the iPaq and the unit gave no responses. A soft reset was insufficient to restore functionality, and disconnecting the iPaq's battery via the unit's switch was necessary. This caused a total data loss.

3)Internet Explorer application error: Trying to run I.E. produced a "fatal application error" and a soft reset was necessary to restore the program's functionality.

4)Memory Critically Low error: This error message was present in a window at the end of the ISIC run, most often with several instances of it on the screen. It stated that one or more programs should be closed; yet there were no programs running at the time. A soft reset was necessary to free the allocated memory.

5)No effect: No visible effects were found after running ISIC.

However, the effects were highly unreproducible, even after resending the same sequence of packets, or a subset. The longer we would try to test the iPaq (not necessarily time but attempts), the less frequently we would observe problems. After much testing and grief, we found that there was a memory problem related to when the iPaq was turned "off" (sleep) for long periods of time. The most common time for the Memory Critically Low error was just after the iPaq was turned on after a long period of being off (several hours); about 27MB (out of 32 MB) were in use after sending the random packets. As the iPaq was reset during tests, the memory reported as being used slowly went down to about 5MB, until effects 1-4 could rarely be observed.

Due to the irreproducibility of the results, we were unable to specify the conditions causing each effect. Microsoft was notified of this during summer of 2001. While we were unable to pinpoint a specific vulnerability, these results suggest a lack of robustness that could be exploited to cause data losses.

b) Known exploits

A number of known attacks against Windows operating systems were run against the iPaq (see Appendix A) on a wireless network, with and without using encryption (WEP, see part 3). In order to have an open port, the application "vxWeb" (by Cambridge Computer Corp) was running on the iPaq, as needed. It provided a web server on port 80. The attacks were compiled on and sent from Red Hat Linux and OpenBSD machines. Of these, Kod was very effective, disabling the IP stack every time. There was a complete loss of internet connectivity until the iPaq was given a soft reset. This is caused by an IGMP vulnerability known since at least June 1999 (CVE-1999-0918). However it was unknown that it affected WinCE, as this information is absent from the CVE and databases like ISS's X-Force at the time of this writing.

c) Vulnerability Scan

We scanned the WinCE iPAQ for known vulnerabilities using Nessus [3]. We found that WinCE uses a common trivial time dependency algorithm to generate its initial sequence numbers. To achieve reliability, TCP uses sequence numbers that keep track of the data exchanged during its sessions. During the setup of a TCP connection (the three way handshake), each

side generates an initial sequence number (ISN) that they will exchange in order to synchronize their TCP stacks [RFC 793]. If the ISN can be guessed, then a TCP connection can be setup without needing to receive any packets from the other side, which enables IP spoofing [7] and TCP Session Hijacking [5] attacks. This was given ID CAN-2001-0162, as there was no report that we could find of this WinCE vulnerability at the time of the discovery.

The Cisco Wireless Base Station also used a trivial algorithm to generate its initial sequence numbers (CAN-2001-0163). It used the 64K rule which increases a sequence counter by a constant (usually 128000) every second and by 64000 for each new connection. The same attacks that were discussed earlier could be used against the access point.

5. The 802.11b Standard

a) MAC addresses

Restricting association and access to an 802.11b network based on the MAC (hardware) address of the wireless cards is one security mechanism that we attacked. We created a program, wmacs, that sniffs a wireless interface and gives the MAC addresses that are currently used on the network. Once a collection of allowed MAC addresses has been obtained, using them is a matter of convincing the wireless card to use it. We studied whether a wireless card could be used to spoof any MAC address, or whether it was restricted to MAC addresses of the same manufacturer, etc... Whereas this is not highly original research, it needed to be verified because the wireless hardware is significantly different from regular Ethernet cards.

We setup the base station to restrict incoming MAC addresses and allow only one representative Ethernet address from each range associated to a manufacturer (Appendix B). The first 3 bytes represent the company that the product is registered to; for simplicity's sake we used "11:11:11" as the last three bytes. This produced an address different from the one given to our wireless card at the factory. Therefore, our wireless card should not have been able to associate and communicate with the wireless network.

Then we setup our Aironet card on our OpenBSD system to connect to the base station:

```
ifconfig an0 down
```
(brings an0, the Aironet card, offline)
```
ancontrol an0 -n [3bears]
```
(sets an0 to use the SSID [3bears])
```
ancontrol an0 -o 1
```
(sets the operating mode of the Aironet interface from ad-hoc mode to infrastructure mode)
```
ifconfig an0 up
```
(brings an0 back online)
```
ancontrol -S
```
(displays settings and associations)

The last command showed us that the card was configured correctly for the wireless network `[3bears]`, but could not get associated. The Cisco base station logs showed that the wireless card had been denied authorization, proving that the MAC address filters were working correctly. Then we changed the MAC address to each one of the test MAC addresses (e.g., Intel below):

```
ancontrol an0 -m 00:02:B3:11:11:11
```

We then brought up the information screen on the Aironet card (`ancontrol -S`) and it showed the card as associated. We also checked the station's log, and it showed that the corresponding MAC address (`00:02:B3:11:11:11`) was authorized and associated. In this manner we successfully spoofed a broad range of MAC addresses on our Aironet card. During the process we would occasionally change the address to one that we knew wasn't allowed to connect (i.e., `00:02:B3:11:11:21`), and the base station would not allow these addresses to connect.

We conclude that our Aironet card could be used to impersonate any valid MAC address, and could connect to the restricted base station as that address. Some WinCE drivers restricted the capability to change the MAC address; e.g., the WaveLan (Lucent) cards would allow only multicast addresses to be specified successfully. However, freely available operating systems had drivers compatible with standard Ethernet functionality. Therefore, defeating MAC wireless authentication was relatively easy.

b) The Wireless Encryption Protocol (WEP)

WEP struck us as exploitable due to the combination of a limited space IV (24-bits) with the use of an XOR operation with a bit stream. The bit stream is uniquely determined by the IV. Therefore, it is possible to build a code book (array) of the encryption bit stream for every IV by doing a known plaintext attack.

In our active attack, one does not need to have a sniffer on the LAN to which the wireless network is connected. Any internet connection may be used by the attacker to send packets to an IP address (such as a

broadcast address) on the wireless network. Knowing the plain text and the encrypted one by sniffing the wireless network (which can be done at quite a distance with directional antennas) allows deducing the encryption bit stream and building a code book.

The completion of the code book is easier when hardware such as the Lucent cards increment the IV by one for every packet sent, rather than using random IVs (CAN-2001-0160; Lucent was notified on 3/6/01). Building a complete code book takes longer because of random IV repetitions ("collisions") that do not provide new information. We realize that random IVs allow collisions to happen more frequently [8], so the defender is vulnerable no matter how the IVs are determined. However, we believe that "collisions" are more difficult to exploit than a complete code book, hence the vulnerability.

We built two applications, one to collect and build a code book, and one to decrypt packets on the fly. Both are available upon request to other accredited security researchers. The one that collects pads from an WEP network and stores them in a database (pads.db by default) is named cpads. The collection speed was several times slower than the maximal theoretical rate; however it demonstrated that building a complete code book, even with random IVs, was practical within the limitations of the year 2001 hard drive sizes (20-60 GB, depending on the desired length of the bit stream in the code book) and on a time scale of approximately a day. A harder to detect passive attack that would sniff incoming traffic and compare it to the WEP-encrypted one is also possible, although more difficult.

The second application was dwep, and sniffed traffic from an encrypted wireless network and performed on-the-fly decryption of the traffic and output it either to the screen or to a file. It performed flawlessly, demonstrating that such attacks are practical.

c) Vulnerabilities in 802.11b implementations

We found that Cisco 340-series Aironet access points used a subset of the available IV space for WEP encryption. Cisco access points using firmware 11.01 do not use 6 bits out of 24 (3 bytes) IV for WEP encryption. The result is a 64 times weaker protection of communications, and makes the collection of pads through plaintext attacks fairly easy. This was communicated to Cisco (CAN-2001-0161, Bugtraq ID 2418) on 3/6/01, along with the TCP initial sequence number vulnerability.

6. Conclusions

Networked Windows CE devices are vulnerable to some of the same issues as PCs, i.e., flaws in protocols and implementation problems. Closed source and proprietary protocols make it difficult to distinguish which was a protocol flaw and which was an implementation problem. Security through obscurity delays the finding of vulnerabilities and makes their analysis more difficult. However, the delays result in a larger installed base of vulnerable devices once the vulnerabilities are found. Moreover, malicious discoverers of vulnerabilities have longer windows during which to exploit them -- perhaps they will even be the only ones finding them, because they were not limited by the economic goals that productive workers must meet.

In addition, there are issues specific to PDAs in a wireless environment. PDAs have proprietary synchronization protocols which require their counterpart on PCs, and which can therefore be attacked on both ends. Moreover, wireless transport exposes both the PDA and the infrastructure (including PCs) to anonymous ranged attacks. Our experiments provided proof that some of the theoretical wireless attacks were practical.

7. Acknowledgements

We are grateful for Kent Wert's advice and expertise with Windows CE, and for Microsoft's funding of this research.

8. Appendix A: Exploits attempted against WinCE

1) Pingflood.c had no effect. However, giving the following command: ping -f [dest_address] caused the iPaq to slow considerably when in conjunction with running internet explorer, but the effects only lasted as long as the flood was executing.

2) Killwin (CVE-1999-0153, a.k.a. Winnuke - send out of band data to port 139) had no visible effects.

3-4) Flushot (Invalid ICMP fragments, Microsoft Q154174) and Pong (spoofed ICMP broadcast flood) seemed to connect and communicate, but had no visible effect on the iPaq.

5) Jolt (CAN-1999-0345). This attack usually locks up a Windows 95 or NT machine or causes it to reboot.

To recover from the Jolt attack, it is usually necessary to reboot. Jolt, given a high enough number of packets to send, slowed the system while the packets were being received, but had no other effects (it worked basically like the ping flood above.) However, when wireless encryption was activated, Jolt used 100% of the iPaq's CPU, causing the iPaq to freeze completely for as long as jolt packets were being sent (this occurred even without running any programs on the iPaq, including vxWeb and internet explorer). It did seem to log at least the last command given during its frozen state (tapping the start menu while frozen caused the menu to open once the jolt attack stopped). We conclude that whereas processing the fragmented ICMP packets was onerous to the iPaq especially with wireless encryption (WEP), WinCE was resistant to the attack.

6) Nestea (CAN-1999-0257) had no visible effect.

7-10) Teardrop, Octopus, Fawx and Jolt2 (respectively CAN-1999-0015, opening a large number of connections, oversized/fragmented IGMP flood and CVE-2000-0305) had no visible effects.

11) Kod (Kiss of Death). Kod disabled the IP stack every time, exploiting an IGMP vulnerability (CVE-1999-0918).

9. Appendix B: Tested MAC addresses by company

The company codes below were found at http://standards.ieee.org/regauth/oui/index.shtml:

`00:02:2D:11:11:11` Lucent Tech WCND
`00:30:6D:11:11:11` Lucent Technologies
`00:00:0C:11:11:11` Cisco Systems, Inc.
`00:30:78:11:11:11` Cisco Systems, Inc.
`00:E0:FE:11:11:11` Cisco Systems, Inc.
`08:00:07:11:11:11` Apple Computer, Inc.
`00:0A:27:11:11:11` Apple Computer, Inc.
`10:00:5A:11:11:11` IBM Corporation
`00:50:76:11:11:11` IBM Corporation
`00:10:D9:11:11:11` IBM Japan, Fujisawa
`00:04:BD:11:11:11` Motorola BCS
`08:00:6C:11:11:11` Suntek Technology, Int'l
`00:03:BA:11:11:11` Sun Microsystems
`00:50:F2:11:11:11` Microsoft Corp.
`00:02:B3:11:11:11` Intel

10. References

[1] Arbaugh, W. A., Shankar, N., Wan, Y.C. J. (2001) Your 802.11 Wireless Network has No Clothes. Department of Computer Science, University of Maryland College Park, Maryland 20742

[2] Borisov N., Goldberg, I. , Wagner, D. (2001) Intercepting mobile communications: the insecurity of 802.11 In: Proceedings of the seventh annual international conference on Mobile computing and networking. pp. 180 - 189. ACM Press, New York, NY, USA.

[3] Deraison, R. (2000) Nessus, http://www.nessus.org

[4] Frantzen M. (2000) ISIC (IP Stack Integrity Checker), http://expert.cc.purdue.edu/~frantzen

[5] L. Joncheray (1995) A Simple Active Attack Against TCP. Proc. Fifth Usenix UNIX Security Symposium.

[6] Miller, S.K. (2001) Facing the challenge of wireless security. Computer, Vol. 34 Issue: 7 , July 2001, pp. 16 -18

[7] Morris, R.T. (1985) A Weakness in the 4.2BSD UNIX TCP/IP Software, CSTR 117, 1985, AT\&T Bell Laboratories, Murray Hill, NJ.

[8] Walker J. R. "Unsafe at Any Key Size; An Analysis of the WEP Encapsulation," (2000) http://grouper.ieee.org/groups/802/11/Documents/DocumentHolder/0-362.zip