CERIAS Tech Report 2001-04

Penetration Analysis of Windows CE

Jared Crane, Seny Kamara, Pascal Meunier,
Dan Noland, Sofie Nystrom
Center for Education and Research in
Information Assurance and Security
Purdue University, West Lafayette, IN 47907

Penetration Analysis of Windows CE

Progress Report for Fall 2000 (sent to Microsoft, December 2000)

Jared Crane, Seny Kamara, Pascal Meunier, Dan Noland, and Sofie Nystrom

Center for Education and Research in Information Assurance Security
1315 Recitation Building
Purdue University
West Lafayette, IN 47907-1315

winCE@cerias.purdue.edu

December 16, 2000 (CAN numbers added March 6, 2001)

Summary

The vulnerability assessment of Windows CE devices started with 3 Aero 1550 Pocket PC devices by Compaq. Halfway through the semester, the project received the remaining equipment needed for penetration testing: wireless and ethernet cards to be used with two PocketPC iPaq devices by Compaq. Preliminary results implicate the existence of several vulnerabilities (one compromise and several Denial-of-Service vulnerabilities) that the team has not been able to analyze precisely. A problem area is the need to reverse engineer ActiveSync in order to clearly demonstrate the impact of the compromise, and to explore more powerful ways in which it could be exploited. Moreover, the team has identified several areas and hypotheses that should be investigated if this project is continued in the Spring 2001 semester.

I. Introduction

The goal of this project is the vulnerability testing of Windows CE in a wireless environment. Thus, the scope includes wireless cards, base stations, handheld synchronization hosts, and related hardware and protocols. The initial testbed comprised 3 Aero 1550 Pocket PC devices by Compaq, an aironet card and 2 base stations AIR-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 as well 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 (WaveLan). The delays were mostly due to the constrained supply of the desired hardware.

In section II, we considered physical and network attacks on the portable devices. 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. Therefore, we believe that physical attacks on portable

devices are much more likely than against normal computers and are worthy of investigation. Attacks on the portable devices include attacks by foreign synchronization hosts, where we define foreign as not owned nor used by the owner of the portable. Hostile synchronization hosts may or may not be foreign; a synchronization host may be running hostile software due to a virus, worm or trojan. Other attacks would be network-based; we include in that category hostile contents downloaded directly or through a service such as avantgo.com. This includes hostile javascript, Active X and java.

In section III, we considered two kinds of attacks on the synchronization host: a) a hostile handheld being put in the craddle and b) network attacks directed at the services intended for the handheld. A hostile handheld need not be a foreign one; the owner's handheld, if compromised with worms, trojans or viruses, could be hostile.

In section IV, we considered attacks on the Cisco base station. Section V is a list of future work for this project, and section VI presents conclusions.

II. The Handhelds

i. Scripted Network Attacks

Eight pre-written network attacks that were known to work against the Windows 95 and Windows NT TCP/IP stacks were ran as a preliminary test against WinCE. The attacks (Ping Of Death, bloop, coke, flushot, jolt, pong, nestea, killwin, syndrop and teardrop) had no effect on the device.

ii. IP Stack Integrity Check

In order to test the WinCE IP stack, we used a tool named ISIC [1] that generates random IP packets that have tendencies (i.e. a certain percentage will be fragmented, have IP options or a bad IP version) in order to find bugs in IP implementations.

We used ISIC to send 40,000 packets of which:

- 10% had a bad IP version
- 100% were fragmented
- 40% had a random IP header length

We repeated this 8 times and found that it resulted in a successful denial of service attack 5 times out of 8. WinCE's response to our packet stream varied slightly each time but had the same basic characteristics: applications would freeze and the device would need a reset (as opposed to just shutting it off) in order to bring it back up. On one occasion, it would not even turn off. Further investigation is needed to narrow the scope of the testing and find exactly where the vulnerability resides in the IP stack.

iii. Port Scan

Several TCP and UDP port scans using nmap [2] were ran against WinCE and we found that in a default setup (when no servers are running on the device) there were no open ports. This greatly decreases risk since without open ports it is impossible to exploit application level vulnerabilities, it is very difficult to guess initial sequence numbers and it prevents most OS Stack fingerprinting tools from recognizing the OS running on the device.

iv. Vulnerability Scan (CAN-2001-0162)

We also scanned the device for known vulnerabilities using Nessus [3] which did not report anything. However, in probing the TCP stack, 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 (further on referred to as ISN) that they will exchange in order to synchronize their TCP stacks [4]. 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[5] and TCP Session Hijacking[6] attacks.

v. General-use threats to integrity and availability

We found that the hardware design combined with the high drain of an inserted wireless network card was vulnerable in normal use. While carrying the portables in a trench or the jacket of a suit, it was very easy to press buttons and turn on the devices unintentionally due to the bulk of the devices with the cards and adapters. As no protective sleeve was given, we assume that this was normal intended usage. The results were observed incidentally in the course of normal commuting. Most often the batteries were almost completely drained, but the data was intact. However, in one case the iPaq simply crashed, displaying an error message about the network card being inserted. All the files, drivers and data were wiped out by pressing the reset button (which usually leaves the user information and files intact). Thus, basic issues about hardware design and software reliability pose threats against the availability and integrity of data.

III. The synchronization host

A user may wish to automatically send the newest versions of files created on a PC to the device or vice versa. Because of this use case Microsoft has created ActiveSync to compare and continuously update the information stored on a PC and a Windows CE-based device.

An ActiveSync connection may be established in a number of ways. (1) A connection may be established by placing the device into it's charging cradle. The cradle receives power to recharge the device's battery from a wall socket, but it may also be connected to the serial port of a PC. The connection and subsequent file synchronization occur across this serial connection. (2) A connection may be established over a connection between two infrared ports. (3) The connection may be established over the network.

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 pin number.

If the correct pin number 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 broke. A user may then establish a new connection try three more passwords.

i. Vulnerability in ActiveSync Authentication

The Pocket PC password can be compromised using software. The proof for this is the fact the unit does not have to be removed physically from the cradle. The connection can be reset and three more successive tries are granted. This allows a brute force attach of the authentication process to be performed in reasonable time. We estimate that it would take just 3 hours to break by hand.

The claim of an exponential back-off on the physical unit is still valid. However, the current safeguard for remote access consists of three password attempts per connection. There are no controls to restrict the number of remote connections, each allowing three password attempts. Moreover, the exponential back-off is not implemented for remote connections. Since this is not in place it allows access for a quick brute force password attack.

A visual basic script is being developed to utilize a function called SendKeys(). What this function does is send key strokes to a selected application. In this case the application will be activeSync. Using send keys it is possible to establish a connection with a CE device that is reachable by the PC, send it three passwords and reconect to try it again. The algorithm below can be used to break the password by brute force using SendKeys in a simple visual basic srcipt.

```
Algorithm:
 Loop1 i --> 0-9999
   SendKeys("%(F)g")
   SendKeys("n")          'Will cause ActiveSync to get connected
   loop2 3 times
    SendKeys("i{ENTER}")  'Sends the password. Note the digits must be
                           'converted to characters

    if pwd_wrong window exits
      SendKeys{ENTER}
    else
      WshShell.Popup "Output ####"
      exit
   i++
   end loop2
 end loop1
```

There are a number of much more interesting attacks that would be possible if we better understood the ActiveSync protocol. For instance we could possibly program another PDA to masquerade as a PC.

In order to increase our understanding of the ActiveSync protocol we decided to begin observing the communication between the device and the PC. We made the assumption that the ActiveSync protocol is probably the same over all three different possible connection methods (serial, IR, and network). Because of the unreliability of IR and network (as well as the expense involved with capturing IR) communications we decided it would be best to capture the serial communications. We placed a breakout box (a.k.a. a serial line analyzer) between a the cradle and the PC running ActiveSync. We captured many successful connections over which we passed the correct pin number as well as many over which we passed bad passwords. As we expected these communications were encrypted. However we believe that it is worth our time to do some cryptanalysis on these communications because the limited processing power

of the device would not allow it to perform public key encryption operations in the amount of time it takes to make an ActiveSync connection. Thus far we have not gotten very far in our cryptanalysis. A known plaintext attack is possible and that is probably our next step.

The recomended solution to this is a larger domain for the password as well as eponential delay for remote connections. The implementation of exponetial delay for remote access can present a denial of service issue. However a longer password should be enough to make a brute force attack unreasonable.

ii. Denial of Service Attack (CAN-2001-0158, CAN-2001-0159)

In order to allow network syncing, the ActiveSync software running on the desktop listens on port 5679 for any connections. This network syncing service can be enabled or disabled by the user through an option in the file menu.

When the device is synced through the cradle, port 5679 is closed which is sensible since there should be no need to sync via the network if we are already syncing via cradle. When the device is not being synced by cradle however, the port stays open. One inconsistency we found with the ActiveSync software is that the port remains open even when we explicitly disable network syncing.

Furthermore, we were able to remotely close the port by establishing a connection to it and feeding ActiveSync any line that was longer than seven characters. This means that anyone can shut down the port thereby preventing other users from syncing their devices via the network.And the port will stay closed until the network syncing option is locally re-enabled on the desktop machine.

We also found that as long as a connection to port 5679 is open and even when there is no authentication taking place, ActiveSync will not allow any other device to sync by network or by cradle. This lasts for about 20 seconds at which point it closes the connection and becomes available once again to any device wanting to sync. This can also be exploited remotely in order to prevent legitimate users from syncing. By simply establishing a connection to the port, closing it before the 20 seconds elapse and then re-establishing it again, one can keep the port continuously active which will make ActiveSync refuse all other syncing requests.

IV. The Cisco Base Station (CAN-2001-0163)

In probing the Aironet Access Point (need #) we found that it too uses a trivial algorithm to generate its initial sequence numbers. It uses 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.

V. Future Work

The work described below has the potential of revealing additional vulnerabilities:

1. ICMP attacks

2. Remote brute force attack on the encryption key. We had questions about the key, and it is a fixed key in the CISCO implementation.

This value establishes the WEP key the bridge will use to receive packets. The value must match the key used by the Access Point. The key consists of up to 10 hexadecimal characters for 40-bit encryption, or 26 hexadecimal characters for 128-bit encryption. The hexadecimal characters may be any combination of 0-9, a-f, or A-F.

3. Get a trojan horse into Win CE

4. DNS/DHCP attack (?)

5. Hijack TCP stream. Maybe physical location in between base station and hand-held would help override a signal

6. CISCO Base station weak network SSID number – Is it sent in clear text? – small address space ("over 16 million values available to choose" At 1/ms, it would take a few hours to find an SSID. However, they mention 32 characters below, which seems more than 16 million possibilities to me:

"A System Identifier (SSID) that must match the SSID used by the parent Access Point. System ID required, up to 32 ASCII characters"

Establishing an SSID (Ssid)

This string functions as a password for the bridge to join the radio network. The bridge must supply a matching SSID in order to associate to an Access Point.

7. DoS: Send TCP RST signals to reset communication remotely

8. Force transmissions from hand-held to drain battery

9. Find if 128-bit encryption is the entire TCP/IP packet or just the data part. If entire packet, try known plaintext attacks with empty packets by sending packets to the device and listening to what is transmitted on the waves.

10. DHCP DoS attack: request many different DHCP addresses from base station and exhaust the address space, so that other portable devices can't connect

11. We could send lists of MAC addresses to filter from one base station to another. Maybe we can make one adopt *OUR* list...

Release Notes for Cisco Aironet 340 Series Access Points New Software Features in Release 10.13

Address Filters Page Provides MAC Address Filtering

You can use the Address Filters page in version 10.13 of the Access Point management system to restrict users on your Access Point by filtering MAC addresses. MAC addresses entered on the Address Filters page are saved in the Access Point's configuration file under the heading "dot1dStaticAllowedToGoTo" and can be sent to other Access Points using the Distribute Configuration page. Consult the information for the Address Filters page and the Distribute Configuration page in Chapter 3 of the Cisco Aironet 340 Series Access Point User Guide

Limiting Associations to the Access Point

To help prevent unauthorized wireless users from receiving data signals from the Access Point, select no for the Allow Broadcast SSID to Associate setting. You can change this setting through an Internet browser or through the Access Point's console port.

Note To completely block unauthorized users from receiving data signals from the Access Point, set and activate the WEP keys on the Access Point and on all wireless client adapters.

Note You can also restrict users by filtering MAC addresses. Consult the information for the Address Filters page in the Cisco Aironet 340 Series Access Point User Guide to set up a MAC address filter.

12. Attack the Win CE remote registry editor to set different values for the TCP timers. Requires knowing the Win CE sync password.

VI. Conclusions

The networked Windows CE devices are vulnerable to some of the same basic issues as PCs, that are caused by the underlying protocols being used (e.g., the predictable sequence numbers vulnerability). On top of it are issues of trusting a synchronization host, and physical security. We believe that securing Windows CE devices may be a most formidable challenge. It is a good idea to note that our work is not complete, and that even if it was, there might remain vulnerabilities. In fact there is much work to be done just in formalizing and demonstrating what we have presented here. In addition, the future work on potential vulnerabilities in section V demonstrates that we have yet barely scratched the surface of all there is to find out. However, this work is an excellent learning experience for all involved; we hope that the project's duration will be extended to the next semester.

VII. Acknowledgements

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

VIII. References

References

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

[2] Fyodor, Nmap (Network Mapper), http://www.nmap.org .

[3] Renaud Deraison, Nessus, http://www.nessus.org .

[4] RFC 793

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

[6] L. Joncheray, "A Simple Active Attack Against TCP, 1995, Proc. Fifth Usenix UNIX Security Symposium.

[7] S. Bellovin, "Security Problems in the TCP/IP Protocol Suite", April 1989, Computer Communications Review, vol. 19, no. 2, pp. 32-48.

[8] *Microsoft Windows CE Communications Guide*, 1999, Microsoft Press, Redmund, WA

[9] J. Murray, *Inside Microsoft Windows CE*, 1998, Microsoft Press, Redmund, WA

[10] C. Muench, *The Windows CE Technology Tutorial*, 2000, Addison Wesley Longman, ?