Advanced Attacks Against PocketPC Phones

Collin Mulliner
collin[at]trifinite.org

Reliable Software Group, UC Santa Barbara
&
the trifinite group

defcon-14, August 2006

Advanced Attacks Against PocketPC Phones

0wnd by an MMS

About Myself

- Collin Mulliner
  - Handheld computing freak
  - Bluetooth hacker
  - Security researcher (mostly PDA/smart phone stuff)
  - Contact:
    - email: collin[at]trifinite.org
    - web: http://www.mulliner.org/collin/

What this Talk is about

- Attacking and exploiting PocketPC smart phones
- Vulnerability analysis of smart phones
  - Applying fuzzing to smart phones
- The Multimedia Messaging Service (MMS)
  - The User Agent/client side of MMS
- **Analyzing and attacking the PocketPC MMS User Agent**

Agenda

- Mobile Phone Attacks - State of the Art
- PocketPC Overview
- The Multimedia Messaging Service
- Mobile Phone Vulnerability Testing
- 0wnd by an MMS
- Conclusions

Mobile Phone Attacks - State of the Art

- Bluetooth-based attacks
  - Take over phone, initiate calls and send text messages
  - Steal phonebook and/or other files
  - Denial-of-service
- Third-party application vulnerabilities
  - Code injection/execution
  - Denial-of-service

Mobile Phone Attacks - SMS/MMS

- Symbian MMS worms
  - Don't use vulnerabilities in applications or the OS
  - Require user interaction in order to infect a target
  - Examples: CommWarrior and Mabir
- SMS-based denial-of-service attacks
  - Nokia 6210: vCard format string vulnerability
  - Siemens 3568i: crash because of “unusual characters”

PocketPC Attacks

- Third-party applications vulnerabilities
  - For example: FTP servers
- Bluetooth stack remote code execution exploit
  - Non-public, because it will never be fixed (Tim Hurman)
- Bluetooth OBEX push vulnerability (bypass authentication)
  - Full access to all files on device
- ActiveSync denial-of-service
- Some “local” attacks

PocketPC

- PocketPC is WindowsCE version for PDAs and phones
  - WindowsCE: Windows for Consumer Electronics
- Many platforms supported (x86, SH, ARM)
  - Most PocketPC PDAs and phones are ARM-based
- Current version of WinCE is 5.0, we are looking at 4.2x
  - Still many WinCE 4.2x devices out there

PocketPC Phones

- i-mate PDA2k
  - PocketPC 2003 SE (WinCE 4.21)
  - GSM, WLAN, Bluetooth, IrDA

- HP iPAQ h6315
  - PocketPC 2003 (WinCE 4.2)
  - GSM, WLAN, Bluetooth, IrDA
- Many more...

The WindowsCE 4.2x OS

- One single 4GB virtual address space
  - Divided into so-called slots (each slot is 32MB)
  - All processes share the virtual address space
    - Memory protection exits
- Limited to 32 concurrent processes
  - Basically no thread limit

WindowsCE OS Security

- Single user OS
  - No real login, just an optional 'device lock'
- Any process can access everything
  - Once you are in ... you are in
- Full access to everything...
  - For example: Bluetooth API, Mobile Phone API, etc...

WindowsCE Exploitation

- Buffer overflow/stack smashing
  - Overwrite return address, take control of program flow
- WindowsCE/ARM shellcode and exploit development
  - Covered quite well by now
    - 2004 Seth Fogie (at Defcon-12)
    - 2005 San (in Phrack #63)
    - 2005 Collin Mulliner (at WhatTheHack!)
    - 2005 Tim Hurman (pentest.co.uk)

WindowsCE Exploit Issues

- No “command shell”
  - Need to hard code everything you want to do
- Return address depends on slot used by process
  - Slots are dynamically assigned
  - Need to “guess” return address

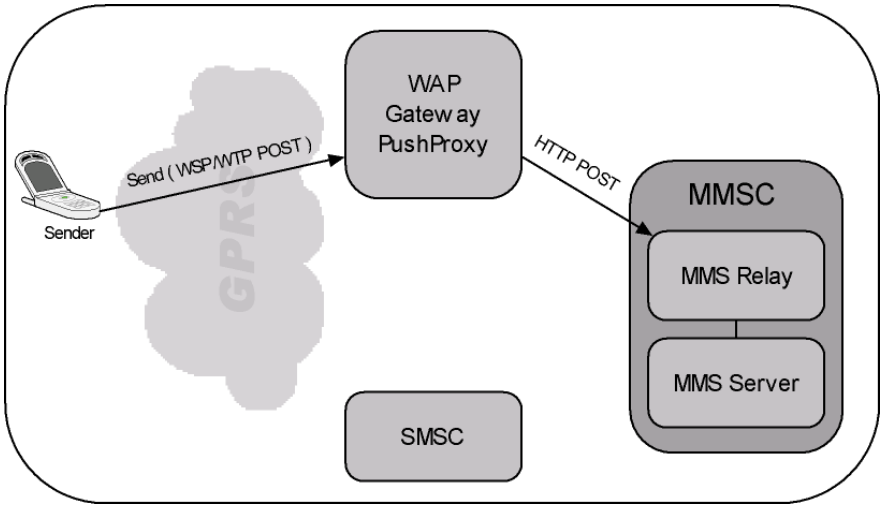
The Multimedia Messaging Service

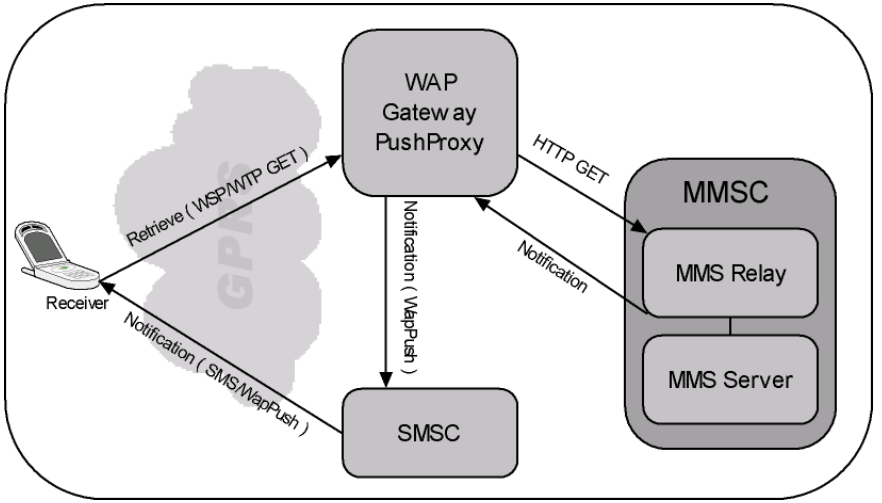
- Messaging service for mobile phones
  - Commonly known as MMS and/or Picture Messaging
- Designed for multimedia content like: pictures, audio, video
  - But basically supports any kind of data
- Messages are sent in a store and forward manner
  - Service requires infrastructure to function
- Pay-per-use service
  - Per message fee

The MMS Architecture

- IP-based service
  - Utilizes HTTP and WAP (Wireless Application Protocol)
  - Transported by GSM/GPRS/etc.
- Message delivery is carried out by four components
  - MMS Server
  - MMS Relay
  - WAP Gateway/PushProxy
  - SMSC (Short Message Service Center)

MMS Message Delivery





- Sender submits message to MMS Relay
  - Sender (WTP/WSP) → WAP Gateway (HTTP) → MMS Relay
- Recipient retrieves message after notification
  - MMS Relay → WAP PushProxy/SMSC (SMS) → Recipient
  - Recipient (WTP/WSP) → WAP Gateway (HTTP) → MMS Relay

MMS Messages

- Structured like Internet email messages
  - Messages are split into header and body
    - Header contains control information
    - Body contains message content (MIME multi-part)
- Messages are in binary form when transfered to and from mobile phone user agents
  - Reduce size for over-the-air transport

MMS Message Types

Transaction	Request Type	Result Type
Sending a message	`M-Send.req`	`M-Send.conf`
Receiving a message	WTP/WSP/HTTP `Get.req`	`M-Retrieve.conf`
New message notification	`M-Notification.ind`	`M-NotifyResp.ind`
Delivery Report	`M-Delivery.ind`	
Acknowledgment	`M-Acknowledge.ind`	

The MMS User Agent

- Represents the sending and receiving end-point
- Handles multiple different network types and protocols
  - SMS-based WAPPush
  - IP-based WAP GET/POST
- Processes and displays multiple media types
  - SMIL and WML for the presentation part
    - SMIL (Synchronized Multimedia Integration Language)
  - GIF, JPEG, BMP, AMR, MV4, ... for the content part

The PocketPC MMS User Agent

- MMS Composer by ArcSoft
  - Version 1.5.5.6 (HP iPAQ h6315 WinCE4.2)
  - Version 2.0.0.13 (i-mate PDA2K WinCE 4.21)
- Application binary: tmail.exe

Analyzing the MMS User Agent

- Identify the inputs to the user agent
  - Possible attack vectors
- Determine message sanitization by the MMS infrastructure
  - Avoid testing sanitized parts of a message
- Implement a virtual MMS infrastructure
  - Testing is done using fuzzing

MMS User Agent Inputs

- New message notification (*M-Notification.ind*)
  - Header fields
  - Delivered via WAPPush (SMS)
- Message header (*M-Retrieve.conf*)
  - Delivered via WAP/HTTP GET
- Message body (*M-Retrieve.conf*)
  - MIME multipart
  - Delivered via WAP/HTTP GET

Sanitization in the MMS Infrastructure

- MMS messages are sanitized by the MMS Relay
  - Sanitization is performed during message submission
  - Messages failing the checks are rejected
- Sanitization has to be avoided
  - Vulnerabilities may not be exploitable if the message part used to deliver the attack is sanitized
  - Need to determine sanitization rules of MMS Relay

Testing the Sanitization Rules

- Fuzzing-like testing procedure
  - Test each message part to determine if it is affected by the sanitization
- Findings:
  - Message header is heavily sanitized
    - Most header fields not usable for attacks
  - Message body is not sanitized
    - Body parts are suitable for attacks

A Closer Look at MMS Delivery

- New Notification is sent to receiver as WAPPush via SMS
  - Binary SMS from port 9200 to port 2948 (SMS ports!)
- PushRouter forwards WAPPush to MMS User Agent
  - If content-type is: application/vnd.wap.mms-message
- MMS User Agent retrieves message via WAP/HTTP
  - The message URL is part of the notification message
- PocketPC also accepts WAPPush via UDP port 2948
  - Also on the wireless LAN interface

MMS New Message Notification

- Encapsulated in a WAPPush message

```
pos   hex                                                ascii
0000  0006 2261 7070 6C69 6361 7469 6F6E 2F76   ..."application/v
0010  6E64 2E77 6170 2E6D 6D73 2D6D 6573 7361   nd.wap.mms-messa
0020  6765 00AF 848C 8298 3233 3432 3235 3437   ge......23422547
0030  3839 3233 008D 9089 1080 0E83 2B31 3535   8923........+155
0040  3531 3233 3435 3637 0097 1083 2B31 3535   51234567....+155
0050  3534 3232 3334 3232 3335 0096 1E83 4772   5422342235....Gr
0060  6565 7469 6E67 7320 746F 2074 6865 2044   eetings to the D
0070  4546 434F 4E20 6372 6577 008E 0202 9A83   EFCON crew......
0080  6874 7470 3A2F 2F79 6F75 726D 6D73 7365   http://yourmmsse
0090  7276 6572 2E63 6F6D 2F6D 6D73 3F72 6566   rver.com/mms?ref
00A0  3D34 3232 3330 3831 3500                  =42230815.
```

wappush, transaction id, subject, message url

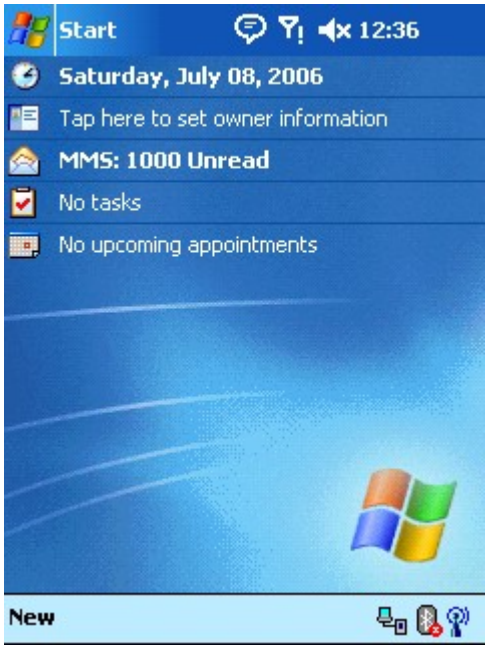
Notification Attack

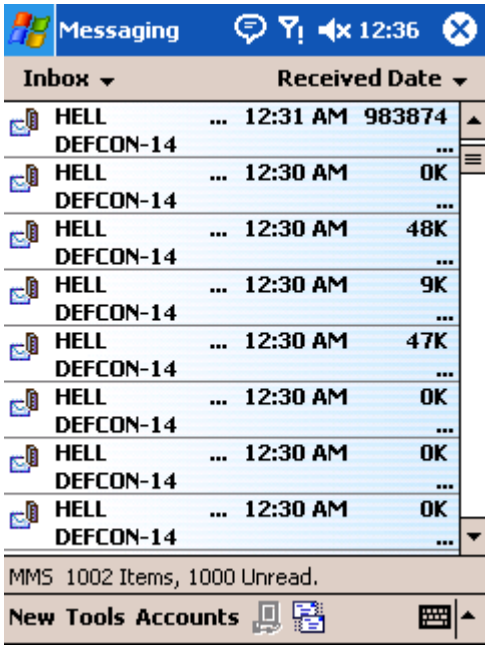
- Flood phone with notifications over WLAN (UDP port 2948)
  - Phone tries to dial-up GPRS to retrieve message
  - New message “sound” is very annoying
- Sending hundreds of messages DoSs the phone
  - Phone becomes slow (lots of memory is used)
- Messages fill up MMS inbox and filesystem
  - Messages have to be deleted one by one
    - It's not fun to delete +1000 messages

Proof-of-Concept: NotiFlood

- PocketPC accepts notifications sent to broadcast address
  - We don't even need to scan for devices!
- Each notification needs to be unique
  - This means unique *Transaction ID* and *ContentLocation*
- Tool available online after presentation!
  - http://www.mulliner.org/pocketpc/

You have 1000 New Messages





Fuzzing

- Feed target application half way valid input in order to find bugs and exploitable vulnerabilities
  - Fuzzing is highly effective with only access to application binary
- Fuzzing requires sending a lot of messages
  - Sending thousands of messages is expensive
  - ➔ Use own MMS infrastructure for fuzzing

Virtual MMS System

- MMS infrastructure
  - MMS Relay/Server (Apache Web server)
  - WAP Gateway (Kannel WAP Gateway)
  - MMS Message Generator (customized MMSLib)
- GSM infrastructure simulated using wireless LAN
  - New message notification sent via UDP (port 2948)
- Configure User Agent to use virtual infrastructure
  - WAP Gateway and MMS Relay and Server

Fuzzing the User Agent

- Focused on triggering standard buffer overflows
  - Mainly modified string length or replaced variable size binary data with string
- The fuzzing process
  - Attach debugger to tmail.exe
  - Generate message and dump into dir. accessible by web server
  - Send notification to phone (phone get message from web server)
  - Watch out for exceptions caught by the debugger

MMS Message

- Message type is: *M-Retrieve.conf*

```
pos  hex                                                ascii
000  8C84 9838 3135 3437 3131 3432 3335 008D  ...81547114235..
010  9089 1080 0E83 2B31 3830 3532 3539 3233  ......+180525923
020  3432 0097 0E83 2B31 3830 3532 3539 3432  42....+180525942
030  3233 0096 0783 4865 6C6C 6F00 8A80 841B  23....Hello.....
040  B38A 3C53 4D49 4C3E 0089 6170 706C 6963  ..<SMIL>..applic
050  6174 696F 6E2F 736D 696C 0002 1017 83C0  ation/smil......
060  223C 7465 7874 3E00 8E74 7874 3100 4869  "<text>..txt1.Hi
070  204A 6F68 6E2C 2068 6F77 2061 7265 2079   John, how are y
080  6F75 3F20 0A21 8267 6170 706C 6963 6174  ou? ....applicat
090  696F 6E2F 736D 696C 00C0 223C 534D 494C  ion/smil.."<SMIL
0A0  3E00 8E73 6D69 6C31 003C 736D 696C 3E0A  >..smil1.<smil>.
0B0  3C68 6561 643E 0A3C 6C61 796F 7574 3E3C  <head>.<layout><
0C0  726F 6F74 2D6C 6179 6F75 742F 3E3C 7265  root-layout/><re
...
200  3C2F 626F 6479 3E0A 3C2F 736D 696C 3E0A  </body>.</smil>.
```

subject, multi-part entry header, text file, SMIL file

Advantages of Simulated Testing

- Full control over all parts of the delivery process
  - Deterministic testing
  - More possibilities for testing
    - For example, message parts that would otherwise be sanitized
- Increased testing speed
  - Testing is much faster (about 10 times)
- Avoidance of usage fees
  - Extensive testing not possible otherwise

Bugs Found 1/3

- **M-Notification.ind**
  - Buffer overflows in parsers for:
    - Transaction ID
    - Subject
    - ContentLocation
  - Non of these are exploitable for code injection
  - NotiFlood now can also crash tmail.exe
    - Actively prevent victim from reading and writing SMS/MMS while using wireless LAN

Bugs Found 2/3

- **M-Retrieve.conf** (header)
  - Buffer overflows in parsers for:
    - Subject (crash only; non-exploitable)
    - Content-Type (can overwrite return address; exploitable)
    - Start-info parameter of Content-Type (non-exploitable)

Bugs Found 3/3

- **M-Retrieve.conf** (body - in the Multipart Entry Header)
  - Buffer overflows in parsers for:
    - Content-Type
    - Content-ID
    - ContentLocation
  - All allow overwriting the return address (exploitable)
- M-Retrieve.conf bugs are not exploitable in the real-world due to sanitization by MMS infrastructure
  - ➔ Avoid sanitization through running our own MMS Server

Rogue MMS Server

- Use setup like the *Virtual MMS System*
- Send notification via SMS to target devices
  - ContentLocation in notification points to rogue MMS Server
- Unfortunately not possible with mobile phone service provider that operates closed WAP Gateway
  - Major US and German mobile phone service providers block this!
  - **Test your service provider!**

SMIL

- Synchronized Multimedia Integration Language
  - XML-based presentation language
  - Specifies how MMS content is displayed to user
  - Basically the HTML for MMS
- SMIL files are transported in the message body and therefore are not sanitized
  - ➔ Perfect attack vector!

SMIL File

```
<smil>
 <head>
  <meta name="title" content="mms"/>
  <layout>
   <root-layout width="229" height="226" />
   <region id="Image" left="4%" top="2%" width="92%" height="80%" fit="hidden" />
   <region id="Text" left="4%" top="81%" width="87%" height="16%" fit="hidden" />
  </layout>
 </head>
 <body>
  <par dur="5000ms" >
   <text src="1.txt" region="Text"/>
  </par>
 </body>
</smil>
```

id parameter of region tag, region parameter of text tag

SMIL Parser Vulnerabilities

- REGION tag, buffer overflow for ID parameter
  - Exploitable (can be used to overwrite return address)
- TEXT tag, buffer overflow for REGION parameter
  - Exploitable (can be used to overwrite return address)
- In both cases the content enclosed in the double-quotes is just copied to the stack (probably the same parser)

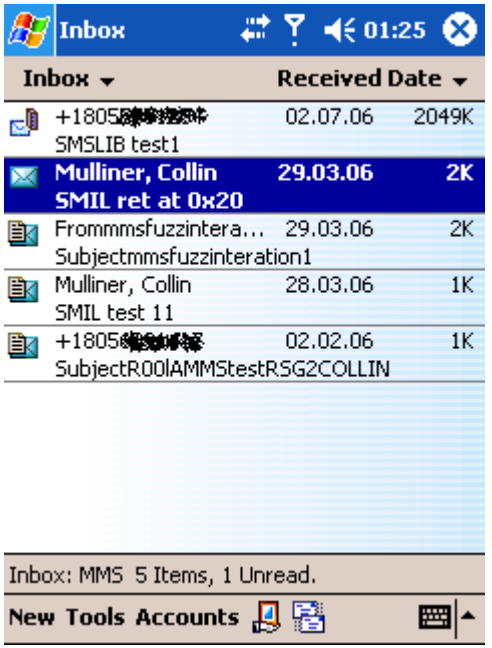
Building an MMS User Agent

- Need our own User Agent in order to send exploit to target
- The User Agent basically is:
  - Message generator (based on MMSLib)
  - WAP Client to send message (based on JWAP)
- MMS Message type to send is: *M-Send.req*
- Use mobile phone for GPRS dial-up
  - MMS Relay is not reachable from the Internet or is on private IP-range

MMS / SMIL Exploit (0-day)

- The first mobile phone remote code execution exploit
  - MMS as the attack vector
- Real code injection/execution
  - User only needs to view the message to trigger exploit
- WindowsCE exploit complications apply
  - Return address guessing is tricky ...but works!

MMS g0t Y0u 0wnd



Exploit Details

- Return address and stack size
  - i-mate PDA2k
    - Return address 0x??05EE**40** (?? -> slot address prefix)
    - Stack size 400 bytes
  - iPAQ h6315
    - Return address 0x??05EE**9C**
    - Stack size 300 bytes
- Common slots used by tmail.exe: 14, 16, 20, 24
- Have fun!!!

Defense

- WLAN notification flooding denial-of-service
  - Packet filter / firewall on phone
- MMS message based attacks (the SMIL exploit)
  - IDS / “Anti-Virus” on phone
  - Mobile phone service provider based IDS / “Anti-Virus”

Conclusions

- Security analysis of mobile phones is more complicated
  - One has to deal with the service infrastructure
- Found +10 bugs in the PocketPC MMS implementation
  - Full advisory will be published soon after defcon
- First code injection against a mobile/smart phone
  - Stuff like this will become a major problem in the future!

Future Work

- Attack WindowsCE 5.0 devices
  - They are supposed to be “super secure”
- Find bugs in other MMS User Agents
  - Symbian, PalmOS, Linux

Questions

Thank you for your attation,
any questions?

References

- MMS http://www.wapforum.com (Documents WAP-[205,206,209,210,230]-WSP)
- JWAP http://jwap.sourceforge.net (Java WAP Library)
- SMSLib http://smslib.sourcefore.net (Java SMS Library)
- MMS Lib http://www.hellkvist.org/software/ (PHP MMS Library)
- Kannel http://www.kannel.org (Free/OpenSource WAP Gateway)
- Reliable Software Group http://www.cs.ucsb.edu/~rsg/
- The trifinite group http://www.trifinite.org
- My PocketPc stuff http://www.mulliner.org/pocketpc/