To be covered...

- Windows kernel debugging
- Basics of the serial debug protocol
- An implementation of the protocol in Perl
- All of the above in less than 20 minutes, hopefully will have time for a demo
Windows Kernel Debugging

- To debug a live system (target) you need another system (host) to run the debugger.
- Windows achieves this via serial connection (latest version also via USB 2.0 or IEEE1394).
- Add `/DEBUG` to boot.ini, plug in a null-modem cable and away we go!
windbg

- Microsoft provides its own debugger, windbg
- Available in the Windows DDK
- Full-featured, if a little less-than-user-friendly
- Extension DLLs can add functionality, API available
- But the host system has to run Windows... what fun is that?
Windows Serial Debug Protocol

- Windows uses a packet-based protocol for communication between the host and the target
- Not officially documented
- But not terribly complex, either
- Best reference is available from Albert Almeida:

http://www.vsj.co.uk/articles/display.asp?id=265
Packet Classes

- Three classes of packets
  - Normal packets: used for debug commands or data exchange
  - Control packets: used to govern the protocol
  - Break-in packet: a special packet used to interrupt system execution and pass control to the debugger

- Normal and control packets have types, which describe their specific function
Control Packet Types

- **PACKET_TYPE_KD_ACKNOWLEDGE**
  - used to ACK packet received from remote side
- **PACKET_TYPE_KD_RESEND**
  - used to request resend of packet from remote side
- **PACKET_TYPE_KD_RESET**
  - used to resynchronize the communication between the two peers
Normal Packet Types

- **PACKET_TYPE_KD_STATE_CHANGE32**
  - Reports when the target has changed from one state to another

- **PACKET_TYPE_KD_STATE_MANIPULATE**
  - Used by debugger to send command/data
  - Used by target to send results of command

- **PACKET_TYPE_KD_DEBUG_IO**
  - Used to handle debug string print IO

- **PACKET_TYPE_KD_STATE_CHANGE64**
  - 64-bit version of state change packet
# Packet Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet Leader</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Packet Type</td>
<td>2 bytes</td>
</tr>
<tr>
<td>Byte Count</td>
<td>2 bytes</td>
</tr>
<tr>
<td>Packet ID</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Checksum</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>
Packet Exchange

• Typical sequence
  - Host sends break-in packet
  - Target replies with state change packet
  - Host ACKs state change
  - Host sends command in state_manipulate packet
  - Target ACKs state manipulate
  - Target replies with data in state_manipulate packet
Debug API

- API is accessed using state manipulate packets
- \_DBGKD\_MANIPULATE\_STATE32 is the payload of the packet, first element is API number
- Each API number corresponds to a specific structure which is appended to the state manipulate struct
- See ReactOS project windbgkd.h for all API structures
• Uses Device::SerialPort module to implement the Windows debug protocol
• Should work on any *nix system where the Device::SerialPort module is supported
• Now we can debug the Windows kernel from almost any system
• Using a scripting language makes it easy to hack in new functionality
Questions?