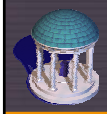
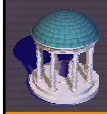


DS Download Play Protocol

Reverse engineering the Wireless MultiBoot (WMB) transport protocol

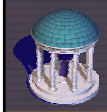
Michael Noland



Nintendo DS

- ARM9 @ 66 MHz
- ARM7 @ 33 MHz
- 4 MB main RAM
- 656 KB video RAM
- 802.11b wireless
- Two screens @ 256x192
  - One is touch sensitive
  - 2D: 128 sprites, 4 layers ea.
  - 3D: 120,000 triangles / sec
- 16 stereo audio channels

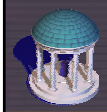
Michael Noland



Overview

- Rationale
- Methodology
  - Introduction to 802.11 standard
- Wireless Multiboot (WMB)
  - Game listings
  - Association
  - Transport protocol
  - Verification
- PictoChat
- Conclusions
- Future Work

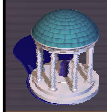
Michael Noland



Motivation

- No good documentation available for the protocol (people know, but nothing public)
- First method to run homebrew required custom hardware
- WMB allows for a ‘soft passme’ (WifiMe)
- On modified firmware, allows for rapid development of software (10 s turnaround)

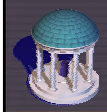
Michael Noland



Methodology

- Studied IEEE 802.11 standard and existing work on GBAdev.org forum
- Acquired hardware and capture software
- Analyzed packets transmitting data corresponding to a known game.
- Wrote a program to process packet logs or live caps
- Compared differences in headers between different game transmissions
- Analyzed packets for clear-text (e.g. descriptive text in UCS-2 transmitted during advertisement stage)
- Tested theories by retransmitting captured binary

Michael Noland



Tools

- capture.exe – Creates libpcap format captures using raw drivers on Ralink cards
- wmb.exe – Implements WMB protocol; mechanism not documented, not open source
- Ethereal – Packet analysis (*)
- XVI32 – Hex Editor
- dsgrok – My NDS parser / verifier
- MappyVM – Disassembler
- Peanuts – My RE framework

Michael Noland

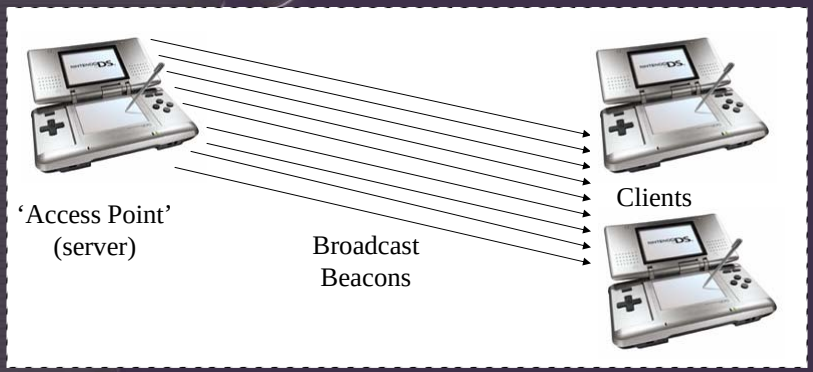
Download process

- Host advertises available download
- User selects download on client
- Client authenticates with host (authentication 1)
- Host accepts the authentication (authentication 2)
- Client tries to associate with host (association request)
- Host accepts association (association response)
- Host sends 0x01 format packets (keep-alive?)
- Host sends 0x03 format packet (RSA signature, size and destination list)
- Host sends 0x04 format packets (game data)

Michael Noland

Advertisement of a download

- WMB (ab)uses beacon frames to advertise available games for download.



Michael Noland

WMB beacon format

WMB beacon format:

- Management frame (24)
- Fixed beacon header (12)
- Tagged elements (*)
  - 0x01: Supported rates (advertises 1 Mbit, 2 Mbit)
  - 0x03: DS parameters
  - 0x05: TIM bit vector
  - 0xDD: Nintendo extension
- FCS (4)

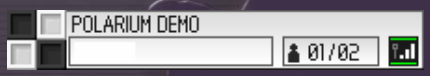
Information element 0xDD format:

Offset	Size	Description
0x00	3	Manufacturer tag (00 09 BF)
0x03	21	Flags and tags (see report for more details)
0x18	2	Game specific
0x1A	2	Stream ID (advertisements mix if identical)
0x1C	1	End-of-advertisement marker
0x1D	1	Maximum number of players
0x1E	1	Players currently connected
0x1F	1	Sequence number
0x20	2	Checksum (custom algorithm)
0x22	1	Sequence number*
0x23	1	Advertisement length (in beacons – 1)
0x24	2	Payload size (in bytes)
0x26	*	Payload

Michael Noland

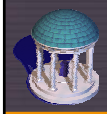
Advertisement format

- Normally transmitted as payload of 10 beacon frames
- Includes title, description, and icon shown on the client selection screen



Offset	Size	Description
0x000	32	Icon palette (16 RGB555 entries)
0x020	512	Icon tiles (4x4 8x8 4 bit palletized tiles)
0x220	1	Unknown (0x0B)
0x221	1	Length of hosting name
0x222	20	Name of hosting DS (10 UCS-2)
0x236	1	Max number of players
0x237	1	Unknown (0x00)
0x238	96	Game name (48 UCS-2)
0x298	192	Game description (96 UCS-2)
0x358	64	00's if no users are connected
0x398	0	End of data if no users are connected

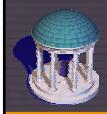
Michael Noland



Download process (recap)

- Host advertises available download
- User selects download on client
- Client authenticates with host (authentication 1)
- Host accepts the authentication (authentication 2)
- Client tries to associate with host (association request)
- Host accepts association (association response)
- Host sends 0x01 format packets (keep-alive?)
- Host sends 0x03 format packet (RSA signature, size and destination list)
- Host sends 0x04 format packets (game data)

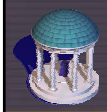
Michael Noland



WMB Transport Protocol

- Uses special MAC addresses to indicate different data flows
- 03:09:BF:00:00:xx, where xx indicates:
  - 00: Host to client data ('broadcast'y)
  - 10: Client to host replies
  - 03: Host to client replies
- Data frames sent on 00 flow outward are both link-layer ACKed and transport-layer acknowledged on 10 flow back with a small data frame.

Michael Noland



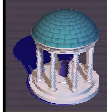
WMB Transport Protocol (2)

Host to client data (flow 0x00)

0x0	Marker (06 01 02 00)
0x4	Payload size (/2)
0x5	Flags
0x6	Payload
*	End marker (00 02 00)

- If flags is 0x11
  - command byte before the payload
- Observed Commands:
  - 0x00: ?
  - 0x01: Idle (ping)
  - 0x01: name request?
  - 0x03: RSA frame
  - 0x04: data frame

Michael Noland

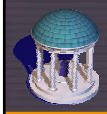


Acknowledgements

- Transmitted over client to host response flow (0x10)
- Always 10 bytes of data
- Never observed acknowledge types 01..06

- Idle reply (pong)
  - 04 81 00 00 00 00 00 00 00 00
- Client name reply
  - 04 81 07 01 [U1] [U2] [U3]
  - 04 81 07 02 [U4] [U5] [U6]
  - 04 81 07 03 [U7] [U8] [U9]
  - 04 81 07 04 [U10] 01 00 00 00 (each [Ux] is a UCS-2 character)
- RSA frame acknowledge
  - 04 81 08 02 xx xx xx xx xx xx
- Data packet acknowledge
  - 04 81 09 [last] [best] 00 00 00
  - [last] is packet being ACKed
  - [best] is highest continuous # seen (2 byte LE packet IDs)

Michael Noland



RSA signature frame

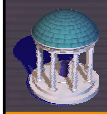
Contents:

- Header destination and size
- ARM9 binary destination, execute address, and size
- ARM7 binary destination, execute address, and size
- RSA signature (public key, SHA-1 message digest)
- Frame itself is not signed, a foolish security flaw :)

Frame payload format:

Offset	Size	Description
0x00	4	ARM9 execute address
0x04	4	ARM7 execute address
0x08	4	0x00
0x0C	4	Header destination[1]
0x10	4	Header destination[1]
0x14	4	Header size (0x160)
0x18	4	0x00
0x1C	4	ARM9 destination address[2]
0x20	4	ARM9 destination address[2]
0x24	4	ARM9 binary size
0x28	4	0x00
0x2C	4	0x022C0000
0x30	4	ARM7 destination address
0x34	4	ARM7 binary size
0x38	4	0x01
0x3C	136	Signature block
0xC4	36	0x00's
0xE8	0	End of frame payload

Michael Noland



RSA signature frame

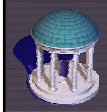
Contents:

- Header destination and size
- ARM9 binary destination, execute address, and size
- ARM7 binary destination, execute address, and size
- RSA signature (public key, SHA-1 message digest)
- Frame itself is not signed, a foolish security flaw :)

Frame payload format:

Offset	Size	Description
0x00	4	ARM9 execute address
0x04	4	ARM7 execute address
0x08	4	0x00
0x0C	4	Header destination[1]
0x10	4	Header destination[1]
0x14	4	Header size (0x160)
0x18	4	0x00
0x1C	4	ARM9 destination address[2]
0x20	4	ARM9 destination address[2]
0x24	4	ARM9 binary size
0x28	4	0x00
0x2C	4	0x022C0000
0x30	4	ARM7 destination address
0x34	4	ARM7 binary size
0x38	4	0x01
0x3C	136	Signature block
0xC4	36	0x00's
0xE8	0	End of frame payload

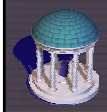
Michael Noland



WMB game data packets

- Type 0x04 frame payload format:
  - 0x00: Pad byte
  - 0x01: Two-byte packet sequence number
  - 0x03: True payload
- Sends NDS header first (0x160 bytes)
- Sends ARM9 binary next (varies)
- Sends ARM7 binary last (varies)

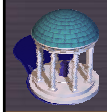
Michael Noland



Verification of theories

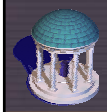
- Captured the Polarium demo in real-time with Peanut
- Verified file integrity using dsgrok
- Used wmb.exe to retransmit to an unmodified DS (RSA checks intact)
- Success! The DS accepted the transfer and played the demo perfectly.

Michael Noland



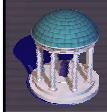
PictoChat

- Analyzed some captures of PictoChat traffic to determine similarities
- Each room has a fixed 802.11 channel:
  - Room A – Channel 1
  - Room B – Channel 7
  - Room C – Channel 13
  - Room D – Channel 7 again
- Room beacon work similarly, using an extension 0xDD tag to transmit information like the current room capacity



Conclusions

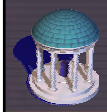
- The NiFi protocol, simply put, isn't
- Each game / app uses it's own protocol, although there are clear similarities, probably due to common base code distributed by Nintendo
- Does not violate the letter of the 802.11 standard
- Passive is sufficient with multiple captures, but moving to an active system will allow flawless captures in a single download.



Future Work

- Add documentation for protocols to NDSTech Wiki
- Continue implementing 802.11 state machine and authentication system
- Active capture and open source WMB.exe re-implementation
- Continue disassembling firmware to discern purpose of flag&tag fields

Michael Noland



References

- NDSTech Wiki: http://www.bottledlight.com/ds
- Firefly's utilities: http://users.belgacom.net/bn967347/
- GBADev.org Forums: http://forums.gbadev.org
- IEEE 802.11 standard: http://grouper.ieee.org/groups/802/11/
- Protocol information will appear on the Wiki.

Michael Noland

Acknowledgements

- Darkain, Gladius, Crazyc, and all the others who have had a crack at it
- Tim Schuerewegen (Firefly) for raw Ralink drivers and WMB.exe
- Stephen Stair for putting up with my barrage of 802.11 questions