# Student Research Project 1: HomePlug Security

Axel Puppe, Jeroen Vanderauwera

February 2, 2010

# Outline

# How do the homeplugs work?

- Network over the power lines
- Traffic is broadcasted (200m range)
- Plug & Play due to default password 'HomePlug'

# How are they secured?

- NEK defines logical network
  - MD5(MD5(password + salt)) * 998
  - Salt: `0x08 0x85 0x6D 0xAF 0x7C 0xF5 0x81 0x85`
  - Size: 8 bytes
- 56-bit DES encryption
- Security through obscurity

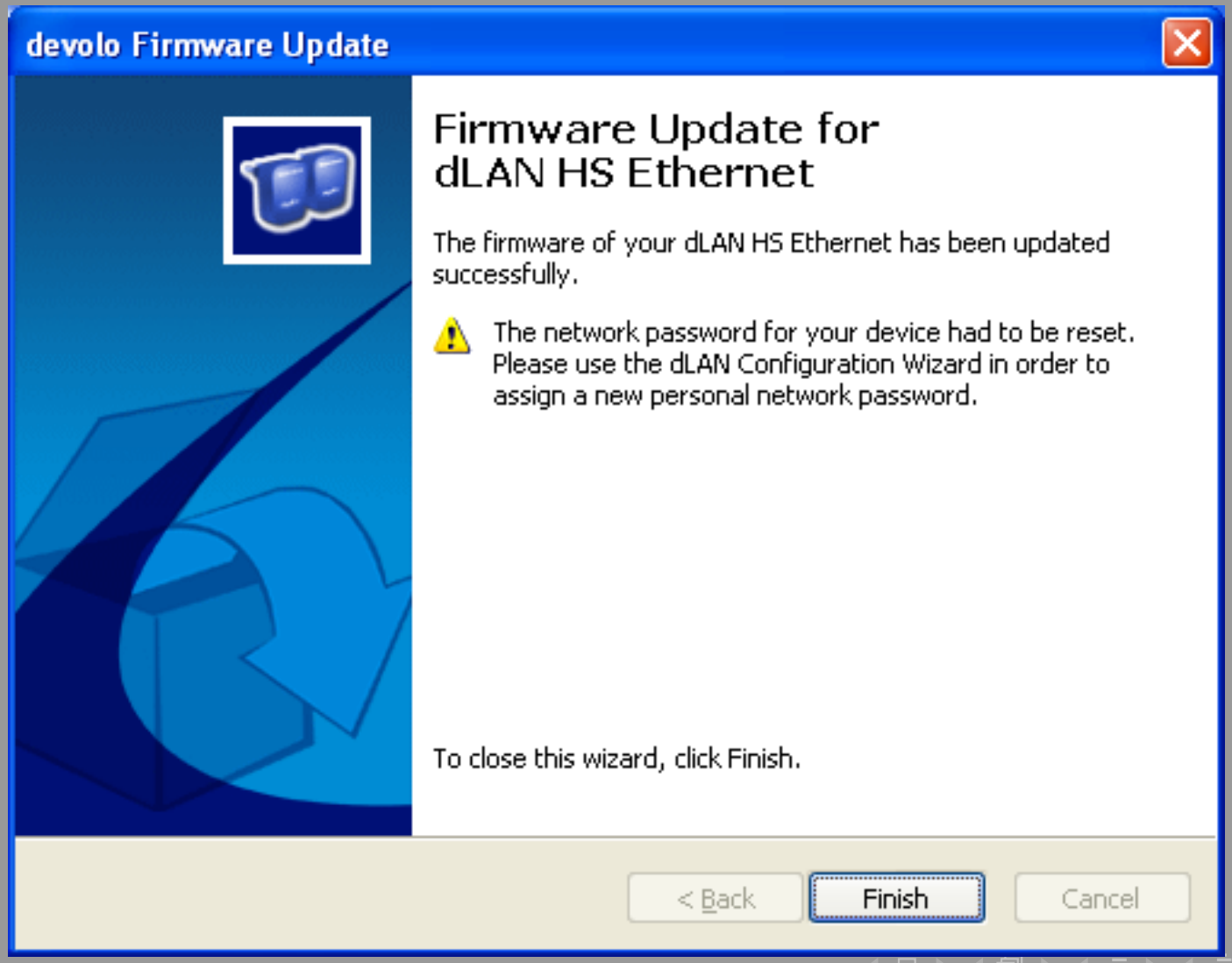
# Our research questions

- Can we reverse-engineer the homeplug firmware to enable promiscuous mode?
  - If successful:

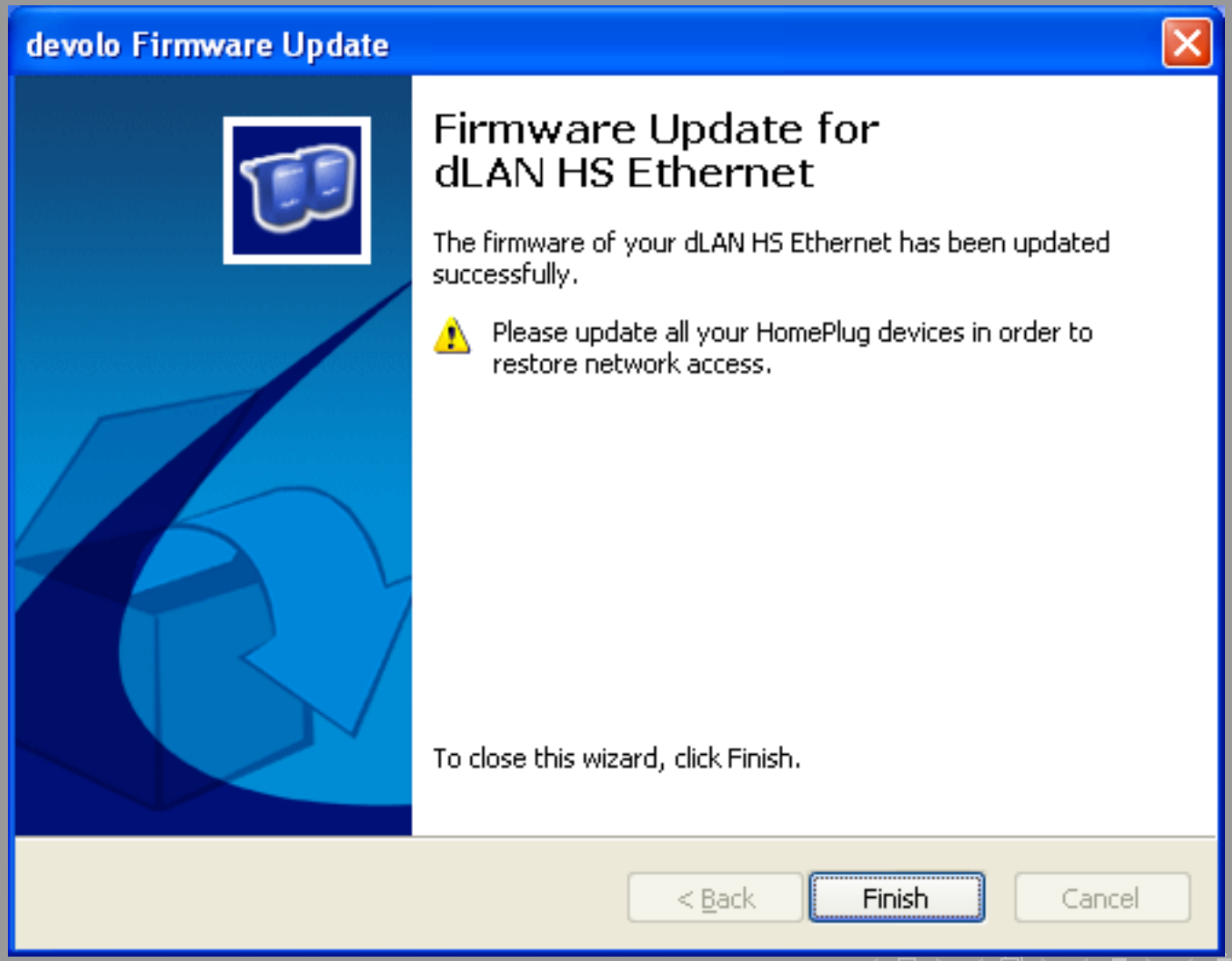
    Can we decrypt the encryption within a reasonable time frame with consumer hardware?
  - If unsuccessful:

    Are there other attack vectors to join or disrupt a target homeplug network?

# Before...


devolo Firmware Update
Firmware Update for
dLAN HS Ethernet
The firmware of your dLAN HS Ethernet has been updated successfully.
The network password for your device had to be reset. Please use the dLAN Configuration Wizard in order to assign a new personal network password.
To close this wizard, click Finish.
< Back
Finish
Cancel

# ...and after


devolo Firmware Update
Firmware Update for
dLAN HS Ethernet
The firmware of your dLAN HS Ethernet has been updated successfully.
Please update all your HomePlug devices in order to restore network access.
To close this wizard, click Finish.
< Back
Finish
Cancel

# Attempts

Firmware image: int5500cs-mac-firmware-zip.img

| | |
|---|---|
| Linux 'file' | No known magic numbers |
| Linux 'mount' | No known file system |
| Windows Daemon tools | Could not mount it |
| Windows Magic ISO | Could not mount it |
| Disassembling in IDA-Pro | Failed to load it |
| Looked for strings | No plain text |
| Testing randomness | True random |
| Scanning for magic numbers | Only false positives |
| Looked at other firmwares | Did not help us understand the firmware image |

Atheros did not provide any information, unless we signed an NDA.

# Scripting

- Bash: 5.8 keys per second
- Python/Scapy: 40 keys per second
- Python/Scapy optimised: 65 keys per second

# Covering the entire 8 byte keyspace

- Size: $256^8 = 1.8 \cdot 10^{19}$ (18 billion billion!)
- Speed: 65 keys per second
- Time: $8.9 \cdot 10^9 = 8.900.000.000$ years
- Obviously not feasible...

# Alternative to bruteforce

- English dictionary: 80.000 words
- Speed: 65 keys per second
- Time required: 20 minutes
- Drawbacks:
  - Success rate is not 100%
  - Only works if people picked a weak password

# If we can't hack it, can we break it?

Yes we can!

| | Without DoS | DoS with correct NEK | DoS without correct NEK |
|---|---|---|---|
| **Minimum** | 2ms | 2ms | 61ms |
| **Average** | 2ms | 271ms | 462ms |
| **Maximum** | 5ms | 1184ms | 1300ms |
| **Packetloss** | 0% | 2% | 30% |
| **Download speed** | 731KBps | 3KBps | 10Bps |

# Step-by-step plan

1. Reverse engineer the firmware updater
2. Set up the sniffing machine
3. Initiate denial-of-service attack
4. Hand over the malicious firmware to the victim
5. Terminate denial-of-service attack

# And the results are...

- Can we reverse-engineer the homeplug firmware to enable promiscuous mode? No.
  - If successful:

    Can we decrypt the encryption within a reasonable time frame with consumer hardware? No.
  - If unsuccessful:

    Are there other attack vectors to join or disrupt a target homeplug network? Yes.
  - Can we conclude that it's safe?

# And the results are...

- Can we reverse-engineer the homeplug firmware to enable promiscuous mode? No.
  - If successful:

    Can we decrypt the encryption within a reasonable time frame with consumer hardware? No.
  - If unsuccessful:

    Are there other attack vectors to join or disrupt a target homeplug network? Yes.
  - Can we conclude that it's safe? No!

# Any questions?