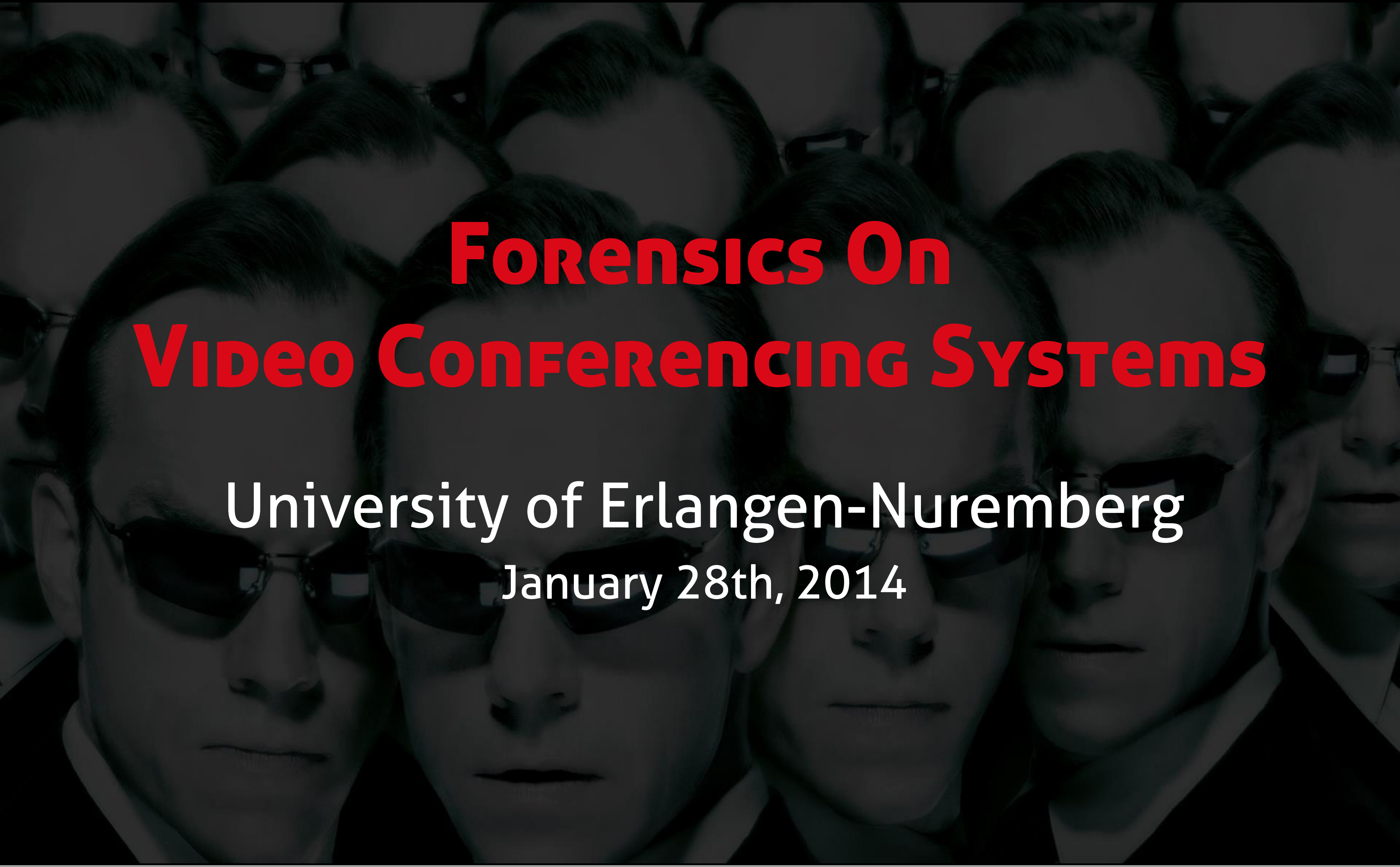


Forensics On Video Conferencing Systems

University of Erlangen-Nuremberg

January 28th, 2014

Agenda

- **Part 1 – Hacking VC Systems**
  - Attack surface
  - Firmware analysis
  - Device rooting
  - Finding and exploiting bugs
- **Part 2 – Forensic Analysis**
  - Challenges
  - Creating forensic copies
  - Finding forensic evidence

Who am I?

- Moritz Jodeit (Twitter: @moritzj)
- Hamburg
- Principal Consultant at n.runs
- Application Security
  - Reversing, bug hunting, writing exploits, …
- Black Hat EU 2013 Talk
  - „Hacking Video Conferencing Systems“

Motivation?

NATIONAL SECURITY AGENCY
UNITED STATES OF AMERICA
ASIA
EUROPE
NORTH AMERICA
AFRICA
SOUTH AMERICA
AUSTRALIA

Hacking Videoconf Systems? yay!

Neue NSA-Dokumente: US-Geheimdienst hörte Zentrale der Vereinten Nationen ab

Demnach ist es der NSA im Sommer 2012 gelungen, in die interne Videokonferenzanlage der Völkergemeinschaft einzudringen und die Verschlüsselung zu knacken. Dies habe für "eine dramatische Verbesserung der Daten aus Video-Telekonferenzen und der Fähigkeit, diesen Datenverkehr zu entschlüsseln" gesorgt, heißt es in einem geheimen NSA-Dokument. "Der Datenverkehr liefert uns die internen Video-Telekonferenzen der Uno (yay!)". Innerhalb von knapp drei Wochen sei die Zahl der entschlüsselten Kommunikationen von 12 auf 458 angestiegen.
In einem Fall habe die NSA zudem den chinesischen Geheimdienst dabei ertappt, ebenfalls zu spionieren. Daraufhin haben die NSA abgefangen, was zuvor die Chinesen abgehört hatten.

Quelle: http://www.spiegel.de/politik/ausland/nsa-hoerte-zentrale-der-vereinte-nationen-in-new-york-ab-a-918421.html

Hacking Videoconf Systems? yay!

Neue NSA-Dokumente: US-Geheimdienst hörte Zentrale der Vereinten Nationen ab

Demnach ist es der NSA im Sommer 2012 gelungen, **in die interne Videokonferenzanlage der Völkergemeinschaft einzudringen und die Verschlüsselung zu knacken**. Dies habe für "eine dramatische Verbesserung der Daten aus Video-Telekonferenzen und der Fähigkeit, diesen Datenverkehr zu entschlüsseln" gesorgt, heißt es in einem geheimen NSA-Dokument. "Der Datenverkehr liefert uns die internen Video-Telekonferenzen der Uno (yay!)". Innerhalb von knapp drei Wochen sei die Zahl der entschlüsselten Kommunikationen von 12 auf 458 angestiegen.
In einem Fall habe die NSA zudem den chinesischen Geheimdienst dabei ertappt, ebenfalls zu spionieren. Daraufhin haben die NSA abgefangen, was zuvor die Chinesen abgehört hatten.

Quelle: http://www.spiegel.de/politik/ausland/nsa-hoerte-zentrale-der-vereinte-nationen-in-new-york-ab-a-918421.html

Hacking Videoconf Systems? yay!

Neue NSA-Dokumente: US-Geheimdienst hörte Zentrale der Vereinten Nationen ab

Demnach ist es der NSA im Sommer 2012 gelungen, in die interne Videokonferenzanlage der Völkergemeinschaft einzudringen und die Verschlüsselung zu knacken. Dies habe für "eine dramatische Verbesserung der Daten aus Video-Telekonferenzen und der Fähigkeit, diesen Datenverkehr zu entschlüsseln" gesorgt, heißt es in einem geheimen NSA-Dokument. "**Der Datenverkehr liefert uns die internen Video-Telekonferenzen der Uno (yay!)**". Innerhalb von knapp drei Wochen sei die Zahl der entschlüsselten Kommunikationen von 12 auf 458 angestiegen.
In einem Fall habe die NSA zudem den chinesischen Geheimdienst dabei ertappt, ebenfalls zu spionieren. Daraufhin haben die NSA abgefangen, was zuvor die Chinesen abgehört hatten.

Quelle: http://www.spiegel.de/politik/ausland/nsa-hoerte-zentrale-der-vereinte-nationen-in-new-york-ab-a-918421.html

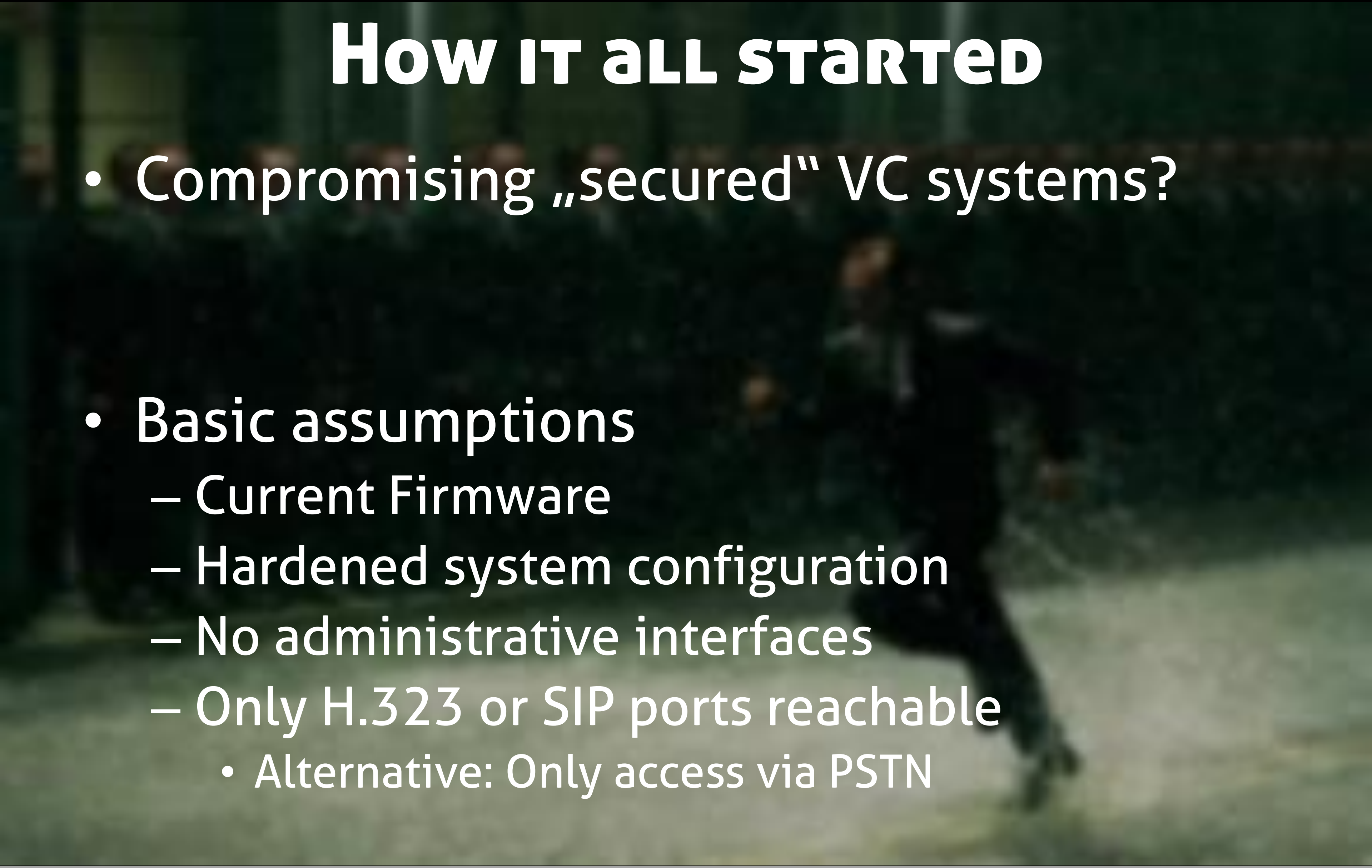
Hacking Videoconf Systems? yay!

Neue NSA-Dokumente: US-Geheimdienst hörte Zentrale der Vereinten Nationen ab

Demnach ist es der NSA im Sommer 2012 gelungen, in die interne Videokonferenzanlage der Völkergemeinschaft einzudringen und die Verschlüsselung zu knacken. Dies habe für "eine dramatische Verbesserung der Daten aus Video-Telekonferenzen und der Fähigkeit, diesen Datenverkehr zu entschlüsseln" gesorgt, heißt es in einem geheimen NSA-Dokument. "Der Datenverkehr liefert uns die internen Video-Telekonferenzen der Uno (yay!)". Innerhalb von knapp drei Wochen sei die Zahl der entschlüsselten Kommunikationen von 12 auf 458 angestiegen.
In einem Fall habe die NSA zudem den **chinesischen Geheimdienst dabei ertappt, ebenfalls zu spionieren**. Daraufhin haben die NSA abgefangen, was zuvor die Chinesen abgehört hatten.

Quelle: http://www.spiegel.de/politik/ausland/nsa-hoerte-zentrale-der-vereinte-nationen-in-new-york-ab-a-918421.html

How it all started

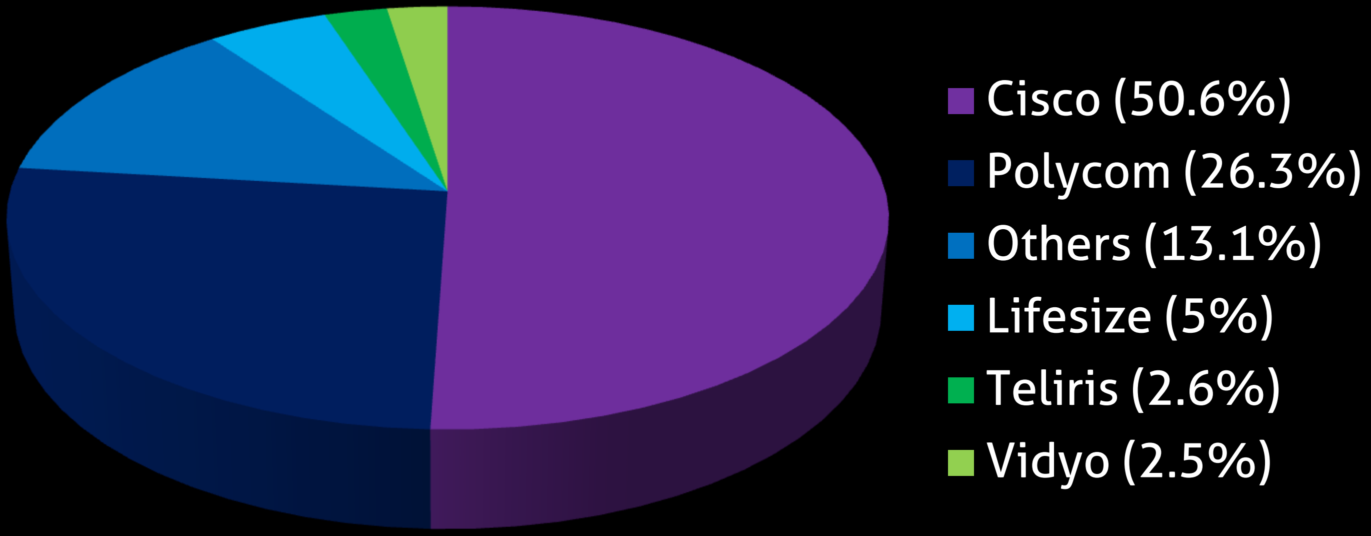


- Compromising „secured“ VC systems?

- Basic assumptions
  - Current Firmware
  - Hardened system configuration
  - No administrative interfaces
  - Only H.323 or SIP ports reachable
    - Alternative: Only access via PSTN

Revenue Market Share

Top Five Enterprise Videoconferencing and Telepresence Vendors



Published by IDC for Q1 2012

Polycom

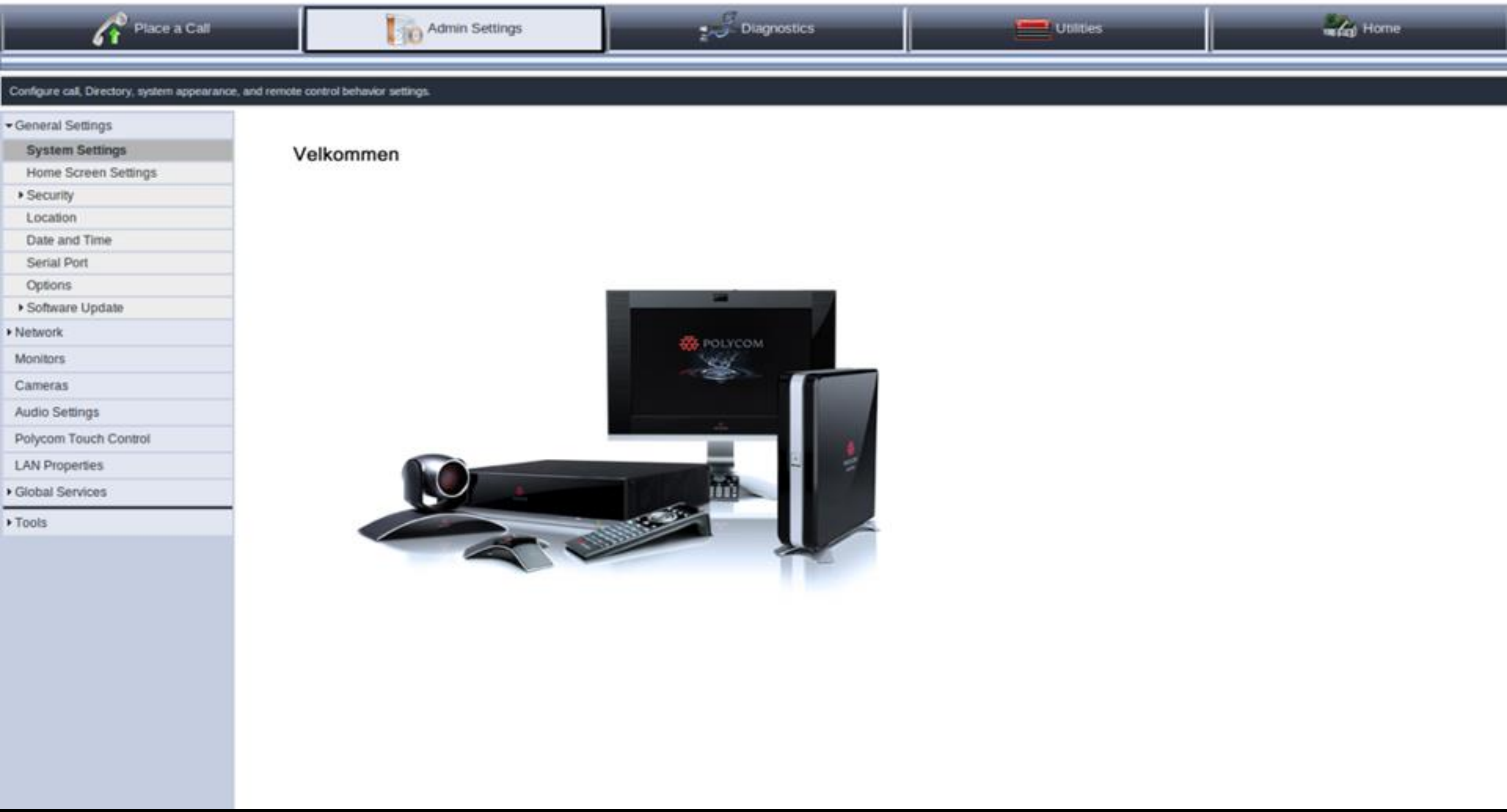
- One of the leading vendors
- Different telepresence solutions
- Most popular units cost up to $25,000
- Polycom customers
  - Government agencies / ministries worldwide
  - World's 10 largest banks
  - 6 largest insurance companies

Polycom HDX Systems

- Popular video conferencing solution
- Different configs (HDX 4000 – 9000)
- HDX 7000 HD (our lab equipment)
  - EagleEye HD camera
  - Mica Microphone array
  - Remote control
  - Connected to ext. display

Attack Surface

Attack Surface



Attack Surface

- Polycom HDX Web Interface
- Provisioning Service
- API Interface (serial console, TCP port 24)
- Polycom Command Shell (TCP port 23)
- SNMP
- Video conferencing protocols
  - H.323 and SIP

Attack Surface

- Polycom HDX Web Interface
- Provisioning Service
- API Interface (serial console, TCP port 24)
- Polycom Command Shell (TCP port 23)
- SNMP
- Video conferencing protocols
  - H.323 and SIP

Firmware Analysis

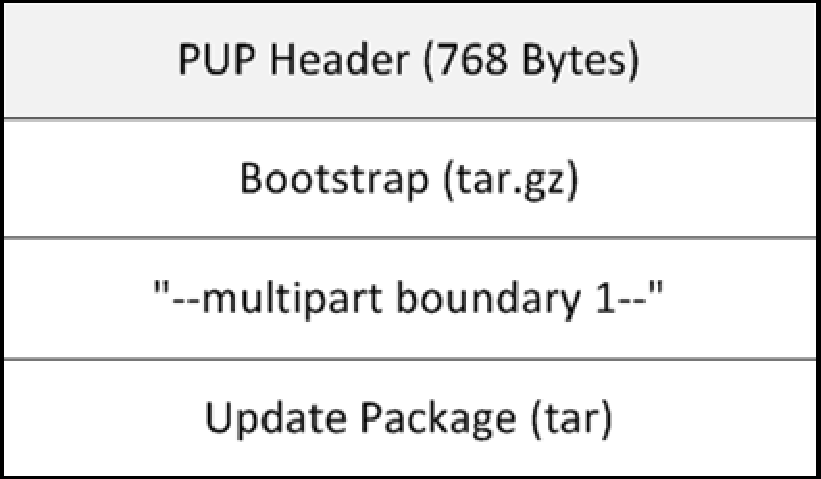
- Software updates (support.polycom.com)
- ZIP archives contain single PUP file
- Manual installation or via provisioning
- Analysis based on version 3.0.5

PUP File Structure

```
$ xxd -g 1 polycom-hdx-release-3.0.5-22695.pup| head -25
0000000: 50 50 55 50 00 30 30 32 00 25 d9 3d 83 e0 b8 a6  PPUP.002.%.=....
0000010: 4c b5 05 cf 41 7f 63 78 0b ae a3 c3 03 47 33 00  L...A.cx.....G3.
0000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 4e 6f 72  .............Nor
0000040: 64 69 63 00 00 00 00 00 00 00 00 00 00 00 00 00  dic.............
0000050: 00 00 00 00 00 00 00 00 00 00 00 00 00 33 2e 30  .............3.0
0000060: 2e 35 00 00 00 00 00 00 00 00 00 00 00 52 65 6c  .5...........Rel
0000070: 65 61 73 65 00 00 00 00 00 00 00 00 00 52 4f 4f  ease.........ROO
0000080: 53 45 56 45 4c 54 00 00 00 00 00 00 00 00 00 00  SEVELT..........
0000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 32 32 36  .............226
00000a0: 39 35 00 00 00 00 00 00 00 00 00 00 00 32 30 31  95...........201
00000b0: 32 2d 30 37 2d 32 33 20 31 39 3a 34 36 3a 34 32  2-07-23 19:46:42
00000c0: 2d 30 35 30 30 00 00 00 00 00 00 00 00 62 75 69  -0500........bui
00000d0: 6c 64 6d 61 73 74 65 72 00 00 00 00 00 31 30 33  ldmaster.....103
00000e0: 37 34 38 34 30 38 00 00 00 00 00 00 00 67 7a 69  748408.......gzi
00000f0: 70 00 48 44 58 20 39 30 30 36 7c 48 44 58 20 39  p.HDX 9006|HDX 9
0000100: 30 30 34 7c 48 44 58 20 39 30 30 32 7c 48 44 58  004|HDX 9002|HDX
0000110: 20 39 30 30 31 7c 48 44 58 20 38 30 30 30 20 48   9001|HDX 8000 H
0000120: 44 7c 48 44 58 20 38 30 30 30 7c 48 44 58 20 37  D|HDX 8000|HDX 7
0000130: 30 30 30 20 48 44 7c 48 44 58 20 37 30 30 30 7c  000 HD|HDX 7000|
0000140: 48 44 58 20 36 30 30 30 20 48 44 7c 48 44 58 20  HDX 6000 HD|HDX 
0000150: 34 30 30 30 20 48 44 7c 48 44 58 20 34 30 30 30  4000 HD|HDX 4000
0000160: 7c 48 44 58 20 34 35 30 30 00 00 00 00 00 00 00  |HDX 4500.......
0000170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

PUP File Structure

- PUP file header
- Bootstrap archive
  - Bootstrap code to install update
  - Main functionality in setup.sh script
- Update package



PUP Header

- Figuring out the PUP header file format
- Found puputils.ppc in extracted firmware
  - Polycom Update Utilities
  - Used to verify and install updates
  - Can be run inside Qemu (Debian on PPC)

```
$ ./puputils.ppc
pc[0]: Welcome to the PUP Utilities.

usage: ./puputils.ppc selftest | genkeys | verify <pup_file>
 <hdx|rabbiteye|diags> | generate <image_file> <pup_file>
 [<supported hw models>] | extract <pup_file> <output_file>

pc[0]: returning PUP_ERR_INVALID_PARAM
```

PUP Header

- Every PUP file starts with fixed PUP file ID
  - „PPUP“ or „PPDP“
- Several fixed-size fields
  - Padded with null bytes

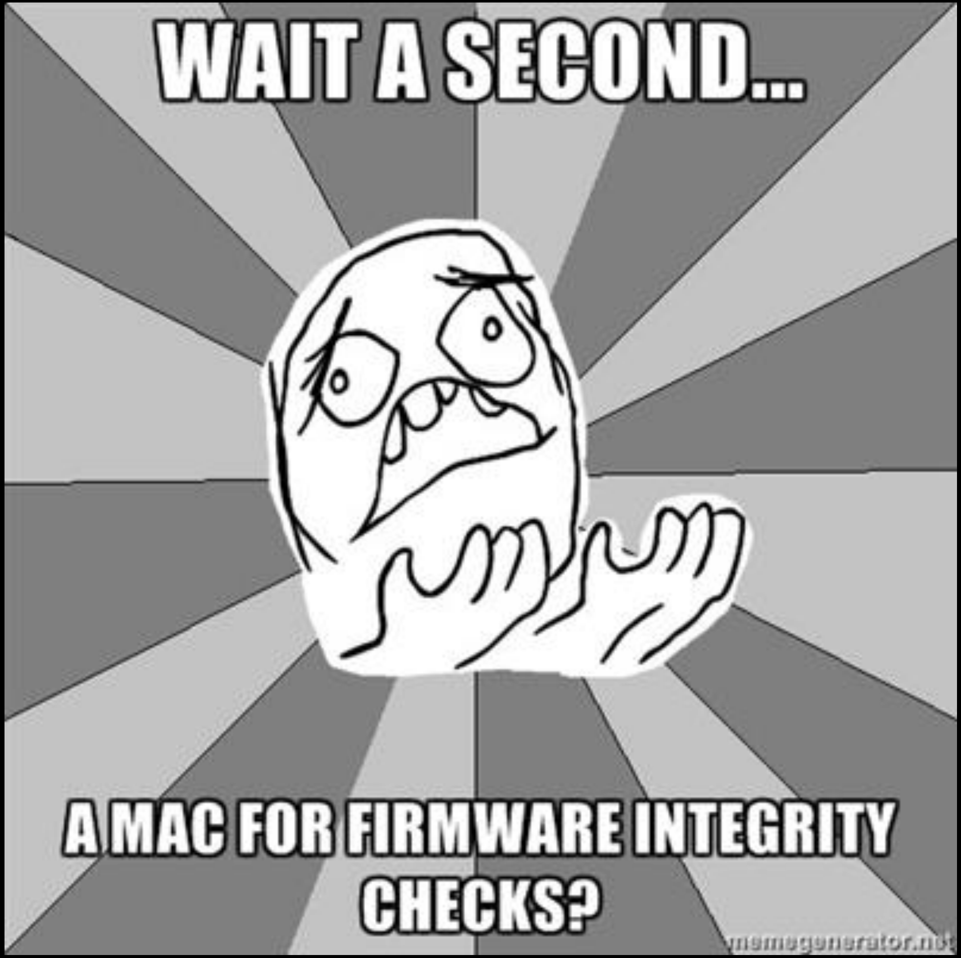
Length (bytes)	Description
5	PUP File ID
4	Header Version
20	Header MAC Signature
32	Processor Type
32	Project Code Name
16	Software Version
16	Type of Software
32	Hardware Model
16	Build Number
32	Build Date
16	Build By
16	File Size (without header)
5	Compression algorithm
445	Supported Hardware
81	Signature (ASN.1 encoded)

Length (bytes)	Description
5	PUP File ID
4	Header Version
20	**Header MAC Signature**
32	Processor Type
32	Project Code Name
16	Software Version
16	Type of Software
32	Hardware Model
16	Build Number
32	Build Date
16	Build By
16	File Size (without header)
5	Compression algorithm
445	Supported Hardware
81	**Signature (ASN.1 encoded)**

Header HMAC

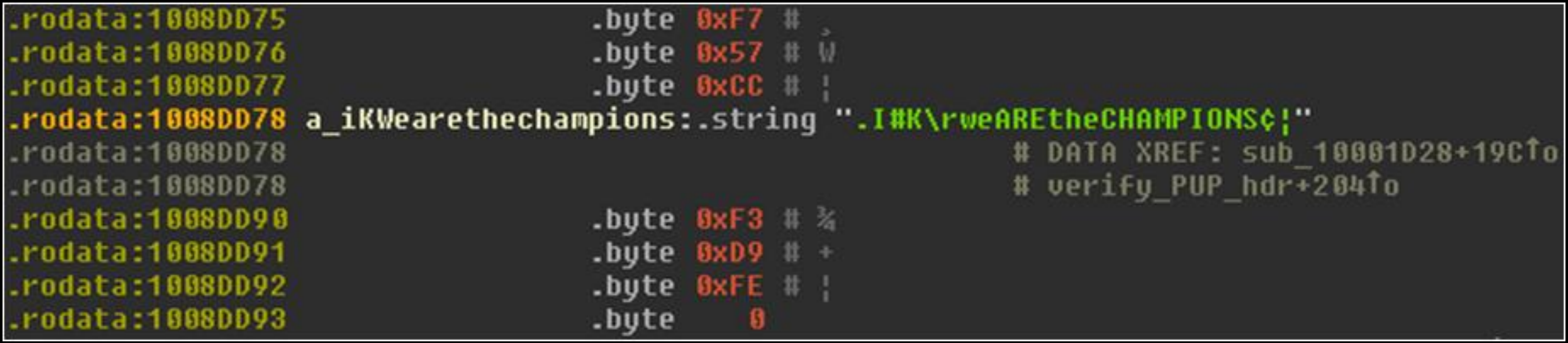
- Header HMAC value stored in PUP header
- Verification process
  1. Set Header HMAC field to zero
  2. Calculate HMAC over PUP header
  3. Compare result with stored value
  4. Abort update if result doesn‘t match

Header HMAC


WAIT A SECOND...
A MAC FOR FIRMWARE INTEGRITY CHECKS?
memegenerator.net

Header HMAC

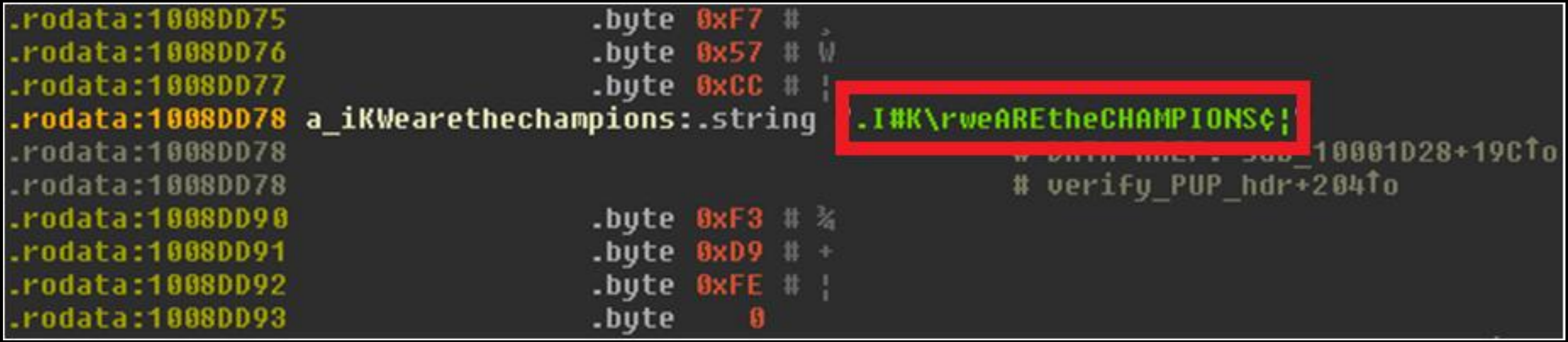
- Secret is required for verification
  - Must be stored on the device
  - Can be extracted :)
- Hardcoded in puputils.ppc binary



```
.rodata:1008DD75                         .byte 0xF7 # ¸
.rodata:1008DD76                         .byte 0x57 # W
.rodata:1008DD77                         .byte 0xCC # ¦
.rodata:1008DD78 a_iKWearethechampions:.string ".I#K\rweAREtheCHAMPIONS¢¦"
.rodata:1008DD78                                                  # DATA XREF: sub_10001D28+19C↑o
.rodata:1008DD78                                                  # verify_PUP_hdr+204↑o
.rodata:1008DD90                         .byte 0xF3 # ¾
.rodata:1008DD91                         .byte 0xD9 # +
.rodata:1008DD92                         .byte 0xFE # ¦
.rodata:1008DD93                         .byte    0
```

Header HMAC

- Secret is required for verification
  - Must be stored on the device
  - Can be extracted :)
- Hardcoded in puputils.ppc binary



```
.rodata:1008DD75                            .byte 0xF7 # ¸
.rodata:1008DD76                            .byte 0x57 # W
.rodata:1008DD77                            .byte 0xCC # ¦
.rodata:1008DD78 a_iKWearethechampions:.string ".I#K\rweAREtheCHAMPIONS¢¦"
.rodata:1008DD78                                          # DATA XREF: sub_10001D28+19C↑o
.rodata:1008DD78                                          # verify_PUP_hdr+204↑o
.rodata:1008DD90                            .byte 0xF3 # ¾
.rodata:1008DD91                            .byte 0xD9 # +
.rodata:1008DD92                            .byte 0xFE # ¦
.rodata:1008DD93                            .byte    0
```

Header HMAC

- Secret allows to calculate valid HMAC
- No reversing of HMAC algorithm required
  - – Correct HMAC is part of the error message!

```
$ ./puputils.ppc verify modified.pup hdx
pc[0]: Welcome to the PUP Utilities.
pc[0]: Verifying the integrity of the PUP file "modified.pup"

pup file SHA-1 Hash: (160-bit)
11876296a8d432841de41526200543caf10ab020
pc[0]:  {1} Verified that we are working with a .pup file.
pc[0]: {2} PUP header version = 002

MAC: (160-bit)
5c3aa27774bd22ff98a1bd95aef09b3b1e11c6f0
pc[0]: The MAC does not match! The PUP header appears to have been tampered with.
pc[0]: returning PUP_ERR_HDR_MAC_MISMATCH
```

Public Key DSA Signature

- 2nd protection to prevent file tampering
  - Used in addition to the header HMAC
- Verifies integrity of the whole file
  - Including the PUP header
- Signature is stored in PUP header
  - ASN.1 encoded form
- No further analysis conducted

Device Rooting

```
# eh?
bash: eh?: command not found
# @%^&!*##!!!!
```

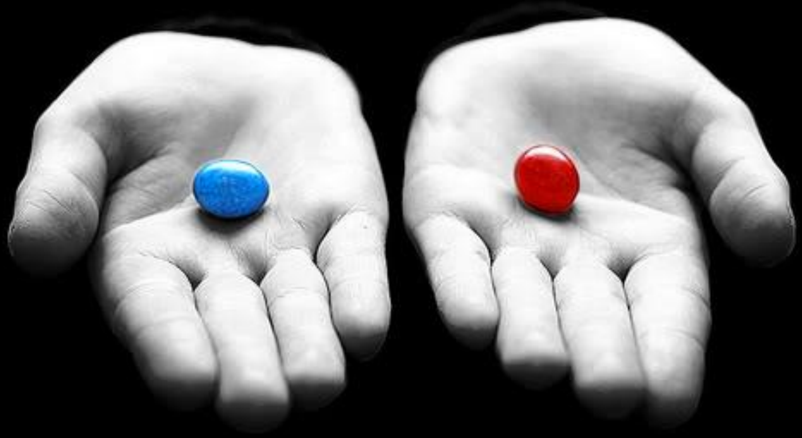
Device Rooting

- No system level access to the device
- Reasons for getting root access
  - Simplifies bug hunting
  - More device control for fuzzing
    - Process monitoring
    - Restarting processes
  - Makes exploit development a lot easier

Device Rooting

- Can be achieved in different ways
  - Exploiting command injection
  - Direct modification of CF card
  - Undocumented Developer Boot Mode
  - …

HDX Boot Modes



- Production vs. Development boot mode
- Development mode enables telnet server
  - – Allows root login without password
  - – For details see my BH 2013 whitepaper

Device Rooting

```
$ cu -l ttyUSB0 -s 9600
-> setenv othbootargs "devboot=bogus"
-> reboot
reboot, are you sure? <y,n> y
```

```
$ telnet 192.168.0.219
Trying 192.168.0.219...
Connected to 192.168.0.219.
Escape character is '^]'.

HDX7000.lan login: root
## Error: "vidoutsize" not defined
# id
uid=0(root) gid=0(root)
```

System Architecture

- MPC8349EMITX SoC
  - Freescale e300c1 PowerPC processor
- Linux-based system
- Kernel 2.6.33.3
- U-Boot boot loader
- Comes with standard binaries
  - busybox, wget, gdbserver, …

Main Processes

- AppMain Java Process
  - GUI
  - Web interface functionality
  - User authentication + crypto functionality

- Polycom AVC
  - H.323
  - SIP

Polycom AVC

- Implemented in /opt/polycom/bin/avc
- Huge non-stripped binary (~ 50 MB)
- Implemented in C
- Running as root
- E.g. implementation of H.323 and SIP
  - and many other complicated protocols…
- What could possibly go wrong? :)

Polycom AVC

- The place to look for bugs in VC protocols
- > 800 xrefs to strcpy()
- > 1400 xrefs to sprintf()
- No exploit mitigations at all
- Easy to reverse engineer due to symbols

Vulndev Environment

- Create debugging environment on device
  - Eases bug hunting
  - Simplifies exploit development process
- GDB remote debugging
  - System already ships with a gdbserver binary
- Disabling Polycom watchdog daemon
  - Create the watchdog_disable.dat config file

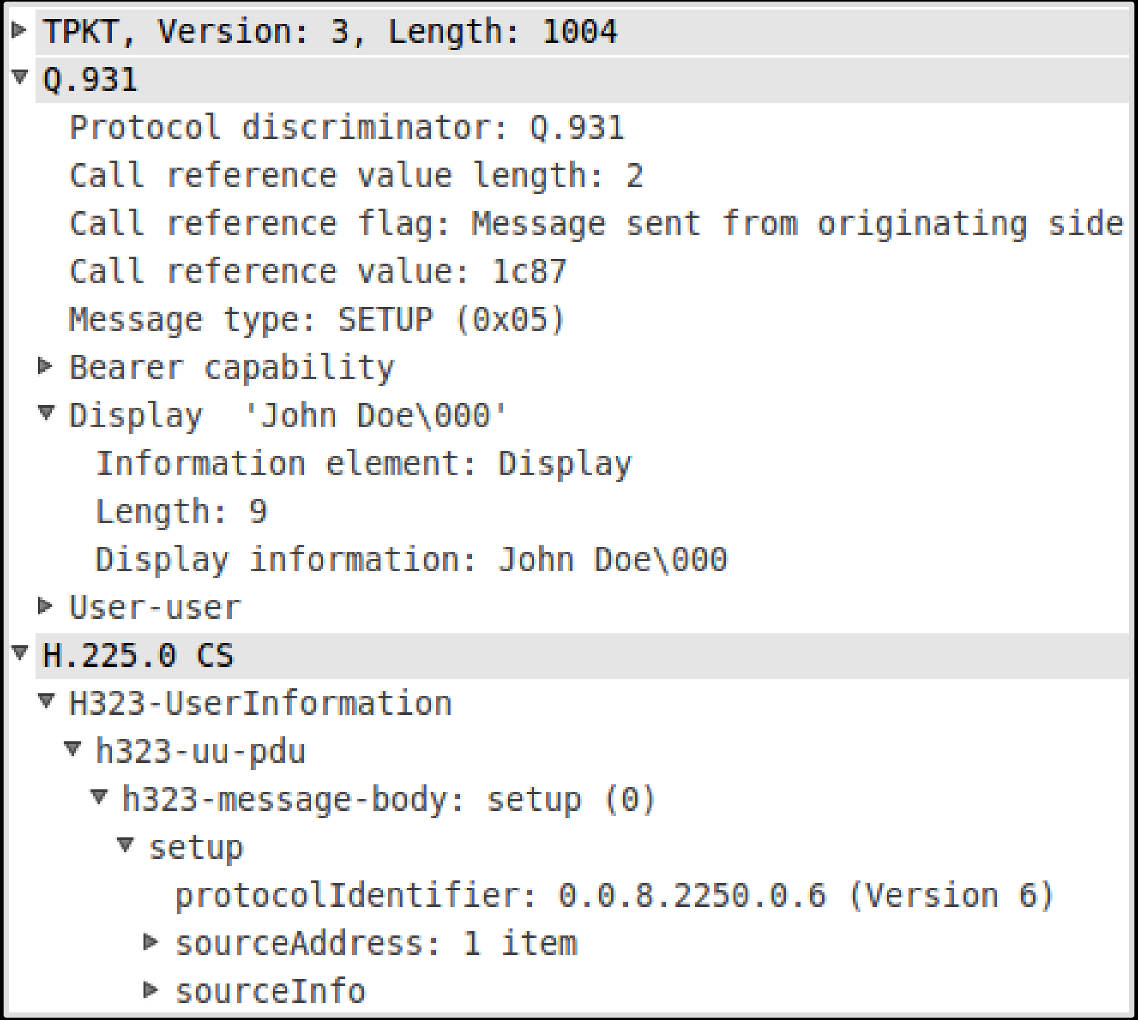
Bug Hunting

- We focused on the H.323 protocol
  – Old and complex protocol
  – Still in use at many locations nowadays
- Many different H.323 signaling protocols
  – We looked at the H.225.0-Q.931 protocol

H.225.0-Q.931

- Consists of binary encoded messages
- Messages consist of Information Elements (IE‘s)
  - Encoded in ASN.1
- Several different IE‘s are defined
- IE‘s provide information to remote site
  - Callers identity, capabilities, …

H.225.0-Q.931



```
▶ TPKT, Version: 3, Length: 1004
▼ Q.931
    Protocol discriminator: Q.931
    Call reference value length: 2
    Call reference flag: Message sent from originating side
    Call reference value: 1c87
    Message type: SETUP (0x05)
  ▶ Bearer capability
  ▼ Display  'John Doe\000'
      Information element: Display
      Length: 9
      Display information: John Doe\000
  ▶ User-user
▼ H.225.0 CS
  ▼ H323-UserInformation
    ▼ h323-uu-pdu
      ▼ h323-message-body: setup (0)
        ▼ setup
            protocolIdentifier: 0.0.8.2250.0.6 (Version 6)
          ▶ sourceAddress: 1 item
          ▶ sourceInfo
```

Call Initiation

- Client connects to TCP port 1720
- Sends SETUP packet
  - Indicates clients desire to start a call
- SETUP packet is parsed even if call fails
  - E.g. call is not accepted by remote site
- Full call establishment requires more msgs
  - But not relevant for this discussion

Call Detail Records

- HDX systems store call detail records (CDRs)
  - Also written for failed calls
  - Every SETUP packet generates a CDR entry
- CDR table stored in SQLite database
  - Written records include
    - Call start/end time
    - Call direction
    - Remote system name ← ***Extracted from Display IE***
    - …

Format String Vulnerability

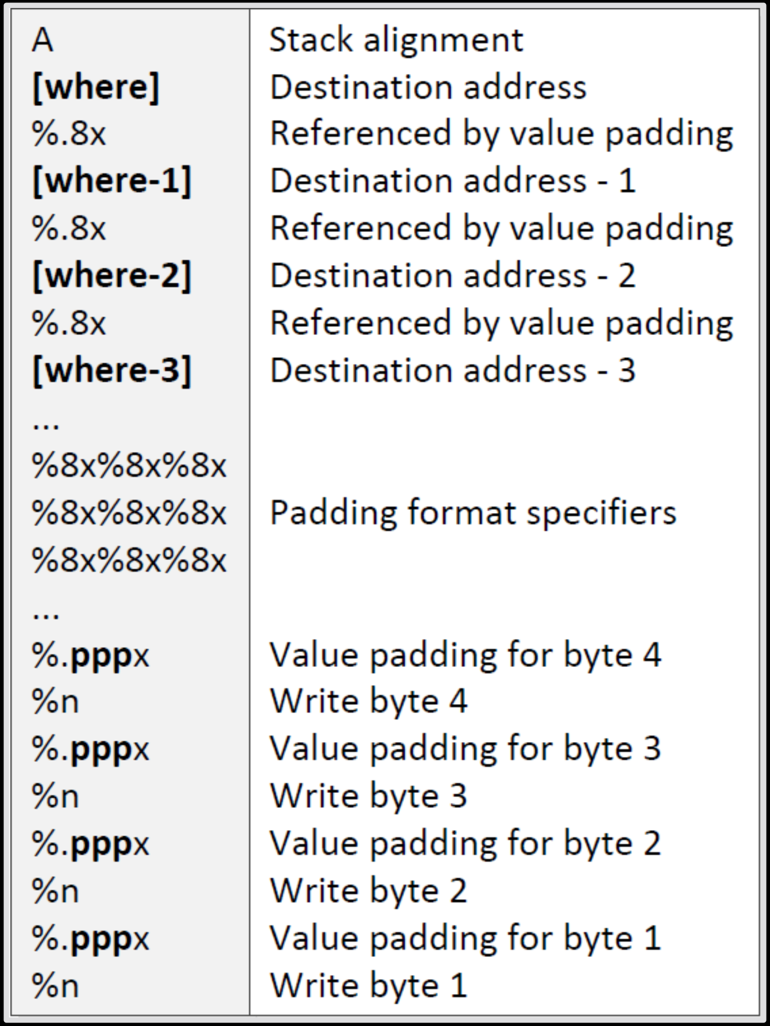
- SQL query string for writing CDR entry
  - Passed as format str to the vsnprintf() function
- We control the embedded Display IE
- Bug triggered with single SETUP packet

```
(gdb) break *0x1032e3ac
Breakpoint 1 at 0x1032e3ac: file ../../../src/Common/OS/logmsg.c, line 747.
(gdb) c
Breakpoint 5, 0x1032e3ac in va_logmsg (ap=0x5e97d298, level=<optimized out>,
    component=<optimized out>, fmt=0x5e97d344 "INSERT into CDR_Table values(
    '23','0','1347451282','1347451282','---','WE CONTROL THIS %n%n%n','',
    '---','h323','0','','1','365','1','0','---','---','terminal','','---',
    '---','---','---','---','---','The call has ended.','16','0','---','---',
    '---','---','---','---','---','---','---','---','---','---','25');")
    at ../../../src/Common/OS/logmsg.c:747
```

Exploit Strategy

1. Turn bug into write4 primitive
   - Write 4 arbitrary bytes at arbitrary address
   - Single SETUP packet writes 4 bytes
2. Use write4 primitive to store shellcode
3. Use write4 to overwrite function ptr
   - And let the code jump into stored shellcode
4. PROFIT!

Format String Stack Layout



A	Stack alignment
[where]	Destination address
%.8x	Referenced by value padding
[where-1]	Destination address - 1
%.8x	Referenced by value padding
[where-2]	Destination address - 2
%.8x	Referenced by value padding
[where-3]	Destination address - 3
...	
%8x%8x%8x	
%8x%8x%8x	Padding format specifiers
%8x%8x%8x	
...	
%.**ppp**x	Value padding for byte 4
%n	Write byte 4
%.**ppp**x	Value padding for byte 3
%n	Write byte 3
%.**ppp**x	Value padding for byte 2
%n	Write byte 2
%.**ppp**x	Value padding for byte 1
%n	Write byte 1

Shellcode

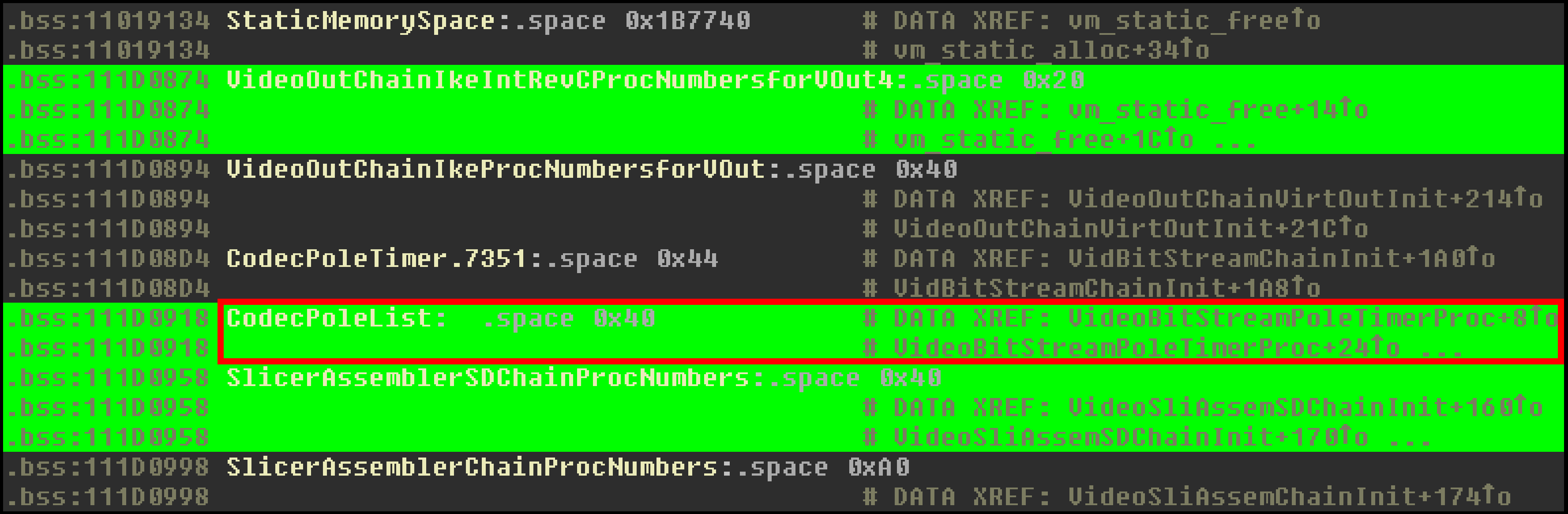
- Simple PowerPC system() shellcode
  - Provides a back-connect shell
  - Executes our HDX payload
- HDX payload
  - Controls the device's peripherals
    - PTZ camera, microphone, display, etc.
    - Based on Polycom's internal IPC mechanism (XCOM)
  - For further details see my BH 2013 whitepaper

Function Pointer Constraints

- The function ptr has a few requirements
  - We need to be able to trigger it remotely
- Restrictions on the format string
  - Bytes in fmt str must be 0x00 < b < 0x80
  - Otherwise logging code is not hit
- Same restriction applies to address of function ptr

Finding Function Pointers

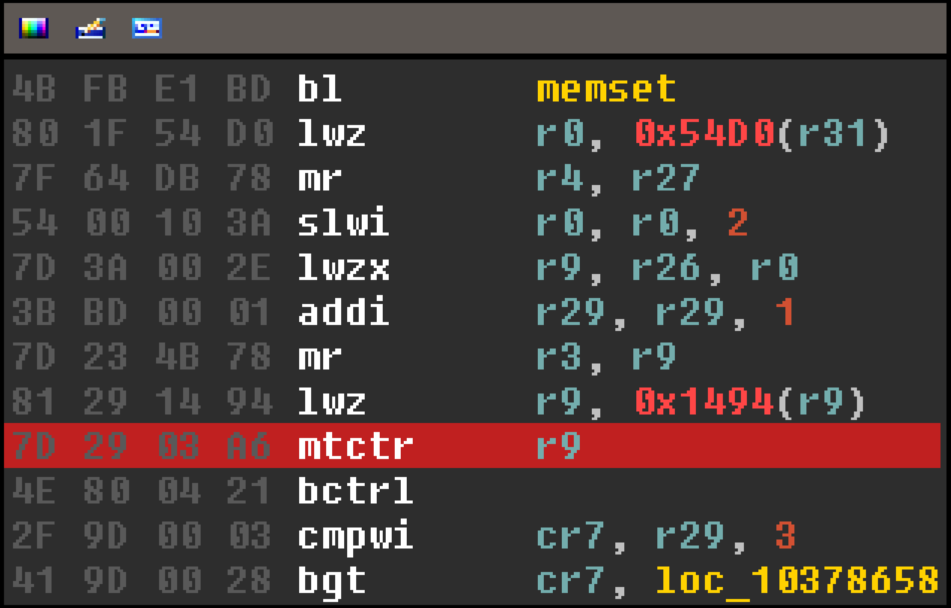
- Highlighted potential addresses in IDA
- Checked xrefs for use of PowerPC mtctr / bctrl instructions



```
.bss:11019134 StaticMemorySpace:.space 0x1B7740        # DATA XREF: vm_static_free↑o
.bss:11019134                                          # vm_static_alloc+34↑o
.bss:111D0874 VideoOutChainIkeIntRevCProcNumbersforVOut4:.space 0x20
.bss:111D0874                                          # DATA XREF: vm_static_free+14↑o
.bss:111D0874                                          # vm_static_free+1C↑o ...
.bss:111D0894 VideoOutChainIkeProcNumbersforVOut:.space 0x40
.bss:111D0894                                          # DATA XREF: VideoOutChainVirtOutInit+214↑o
.bss:111D0894                                          # VideoOutChainVirtOutInit+21C↑o
.bss:111D08D4 CodecPoleTimer.7351:.space 0x44          # DATA XREF: VidBitStreamChainInit+1A0↑o
.bss:111D08D4                                          # VidBitStreamChainInit+1A8↑o
.bss:111D0918 CodecPoleList:   .space 0x40             # DATA XREF: VideoBitStreamPoleTimerProc+8↑o
.bss:111D0918                                          # VideoBitStreamPoleTimerProc+24↑o ...
.bss:111D0958 SlicerAssemblerSDChainProcNumbers:.space 0x40
.bss:111D0958                                          # DATA XREF: VideoSliAssemSDChainInit+160↑o
.bss:111D0958                                          # VideoSliAssemSDChainInit+170↑o ...
.bss:111D0998 SlicerAssemblerChainProcNumbers:.space 0xA0
.bss:111D0998                                          # DATA XREF: VideoSliAssemChainInit+174↑o
```

Function Pointer Overwrite

- Timer thread running in VideoBitsStreamPoleTimerProc()
- Jumps to [CodecPoleList]+0x1494



```
4B FB E1 BD bl      memset
80 1F 54 D0 lwz     r0, 0x54D0(r31)
7F 64 DB 78 mr      r4, r27
54 00 10 3A slwi    r0, r0, 2
7D 3A 00 2E lwzx    r9, r26, r0
3B BD 00 01 addi    r29, r29, 1
7D 23 4B 78 mr      r3, r9
81 29 14 94 lwz     r9, 0x1494(r9)
7D 29 03 A6 mtctr   r9
4E 80 04 21 bctrl
2F 9D 00 03 cmpwi   cr7, r29, 3
41 9D 00 28 bgt     cr7, loc_10378658
```

Remote Root Exploit

Forensic Analyis

Forensic Analysis Challenges

- Requires deep understanding of system
  - Documentation not publicly available
  - Requires extensive research up front
- Every vendor uses their custom firmware
- But even for the same vendor…
  - Different firmware versions
  - Different hardware releases

First Steps

- Disconnect the power supply!
  - HDX systems log a lot of information
  - Use of a pretty small ring buffer
  - Evidence gets overwritten quickly
- Do *not* do a normal shutdow
  - A lot information gets logged in that case!

Creating a Forensic Copy

- We can't work on the system directly
- Forensic copy of the internal memory
- Further analysis only conducted on image

Extracting Memory Cards

- HDX systems use CompactFlash cards
- Various HDX versions have different cases
  - Different ways to get to the CF card
  - HDX 8000 vs. HDX 9000
- Extracting the CF card can be a bit tricky in some cases...

Opening HDX Systems

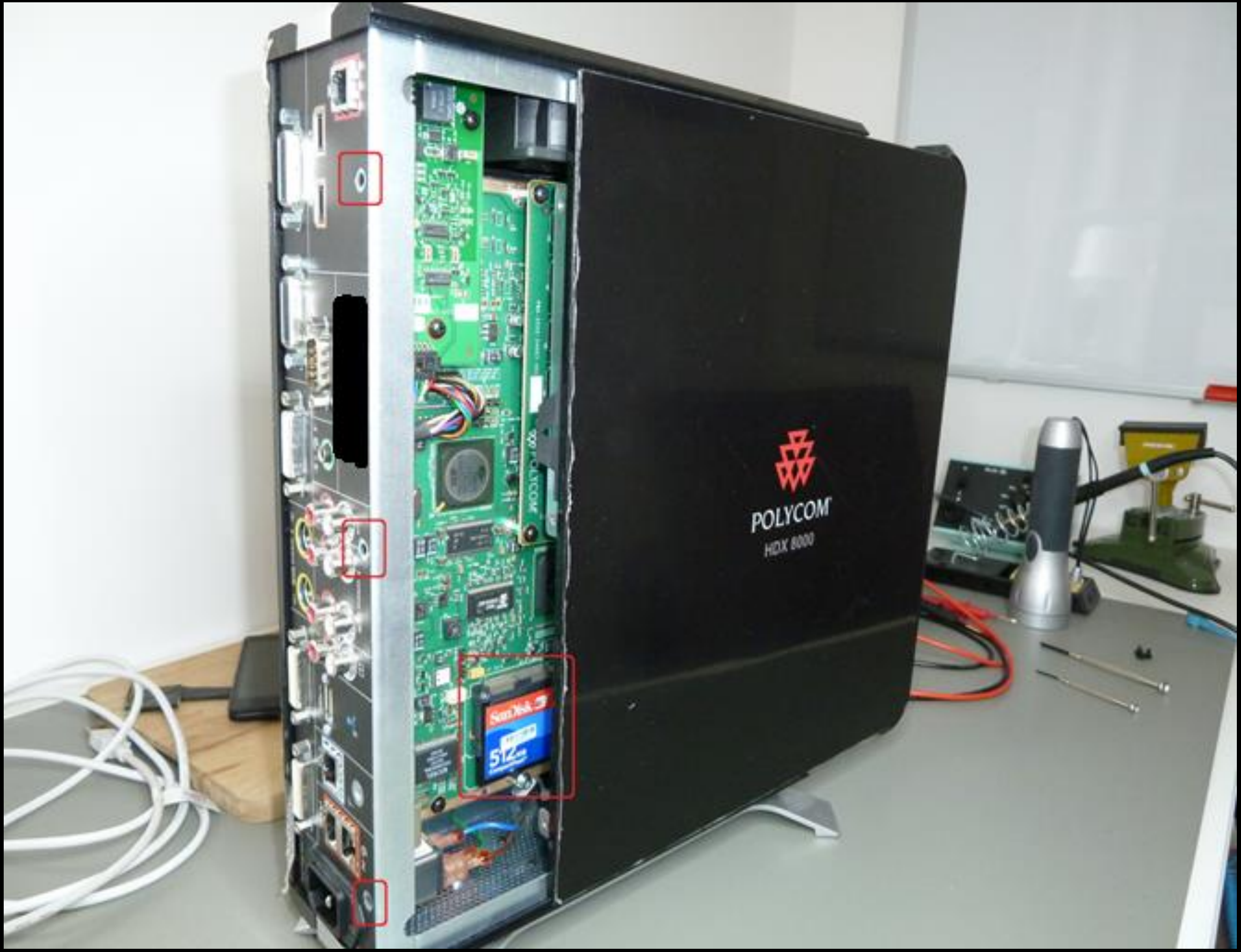
DISCLAIMER

Having the right hardware tools might make the job easier :)

Polycom HDX 8000

- One of the smaller HDX systems
- Can be opened quite easily
  - If you know how to do it ;)
- Three screws need to be removed
- Side of the case can be slided to the front

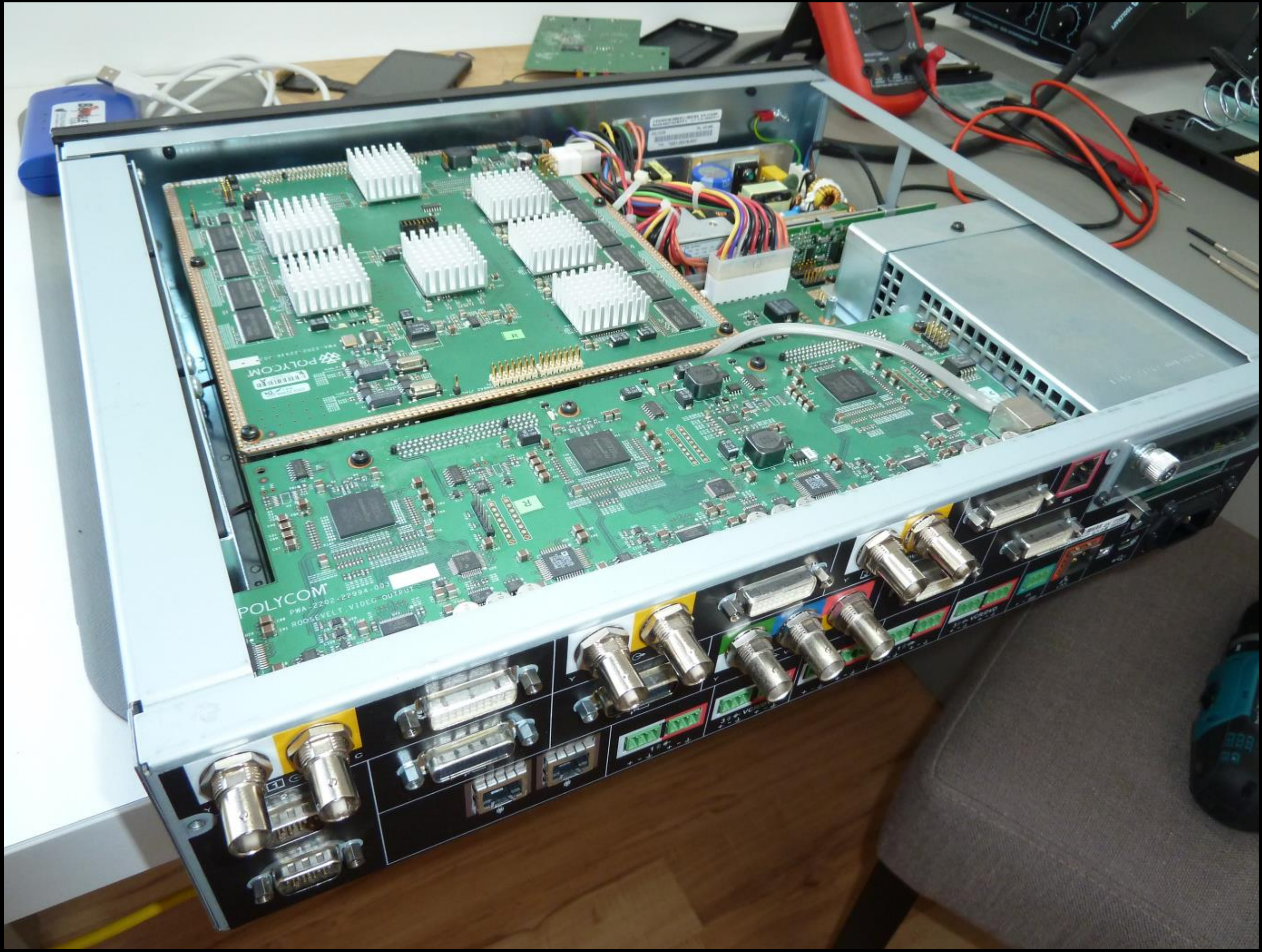
Polycom HDX 8000


[redacted]

Polycom HDX 9000

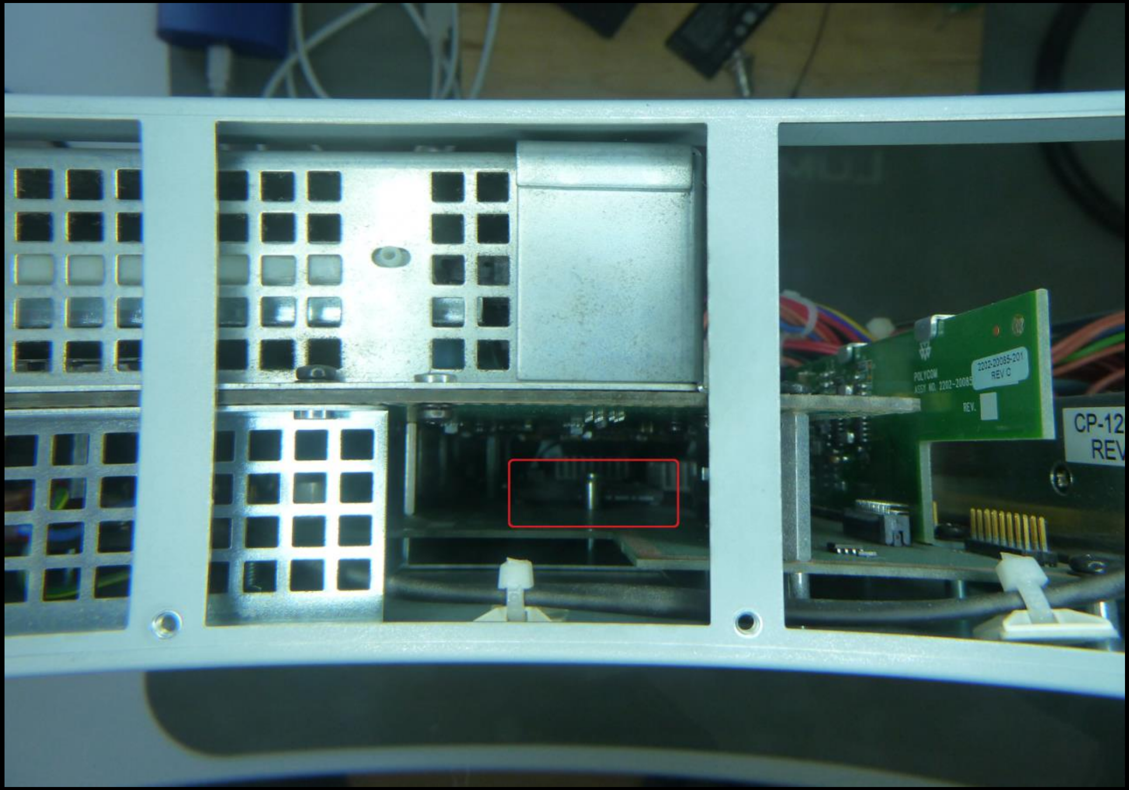
- One of the bigger HDX systems
- Case can be opened quite easily
  - Getting access to the CF card is another story
- Just remove all screws on back and sides

Polycom HDX 9000



Polycom HDX 9000

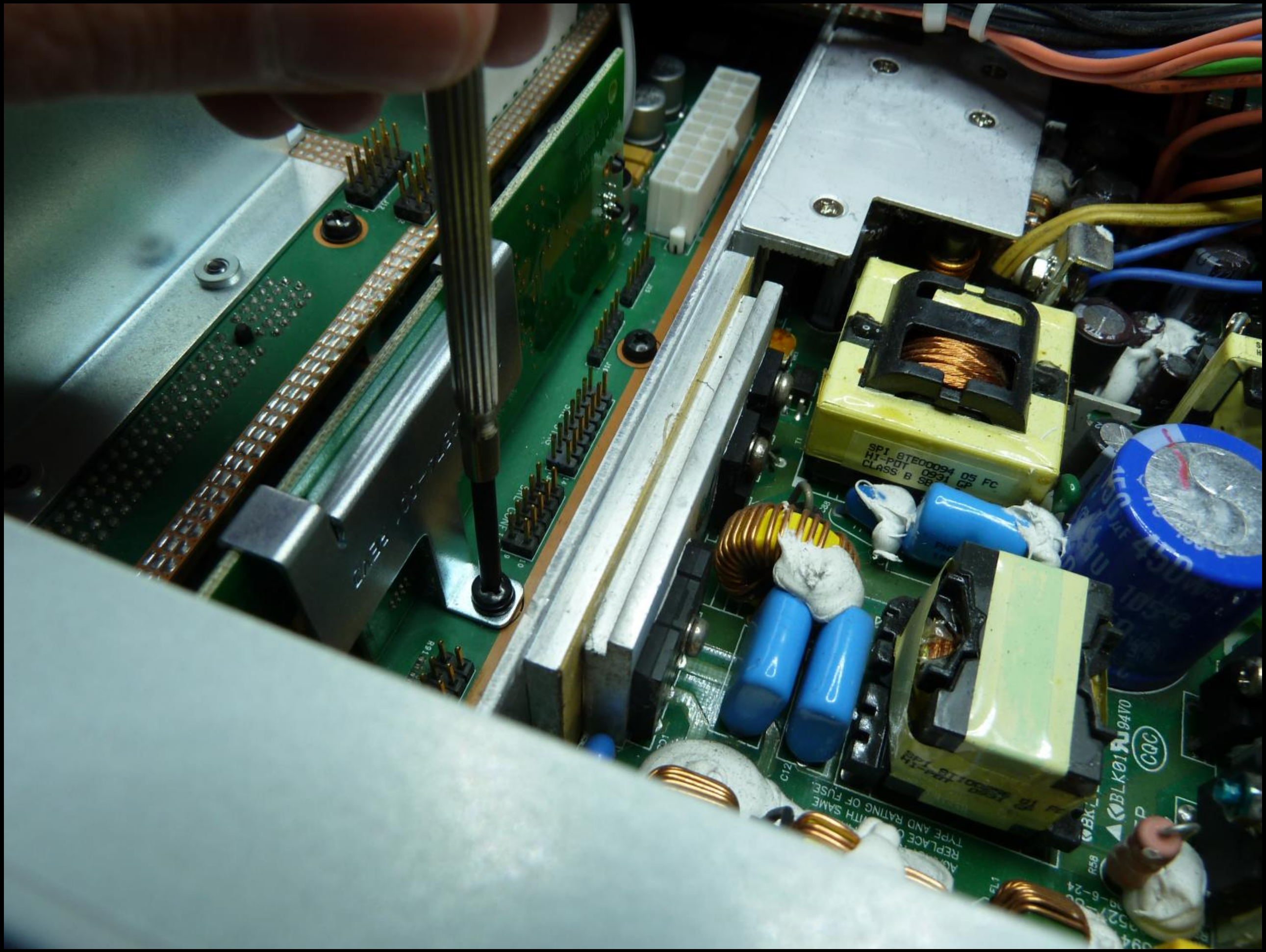
- CF card is hidden beneath several PCBs



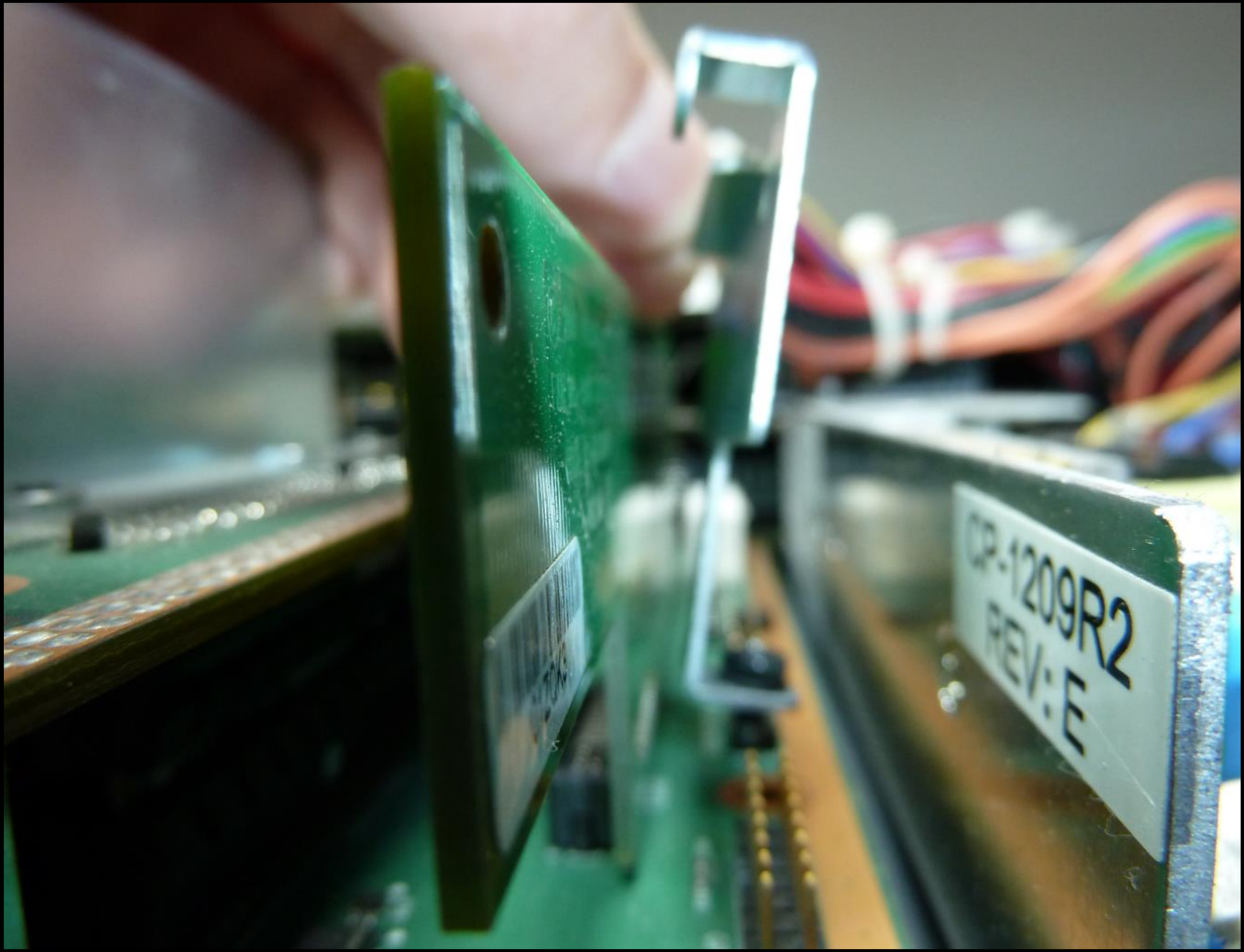
Polycom HDX 9000

- Accessing the CF card is tricky
- Removing all PCBs
  - Would require a complete dismount
  - Could easily damage something :(
- We didn‛t have the right tools
  - We needed to improvise :P

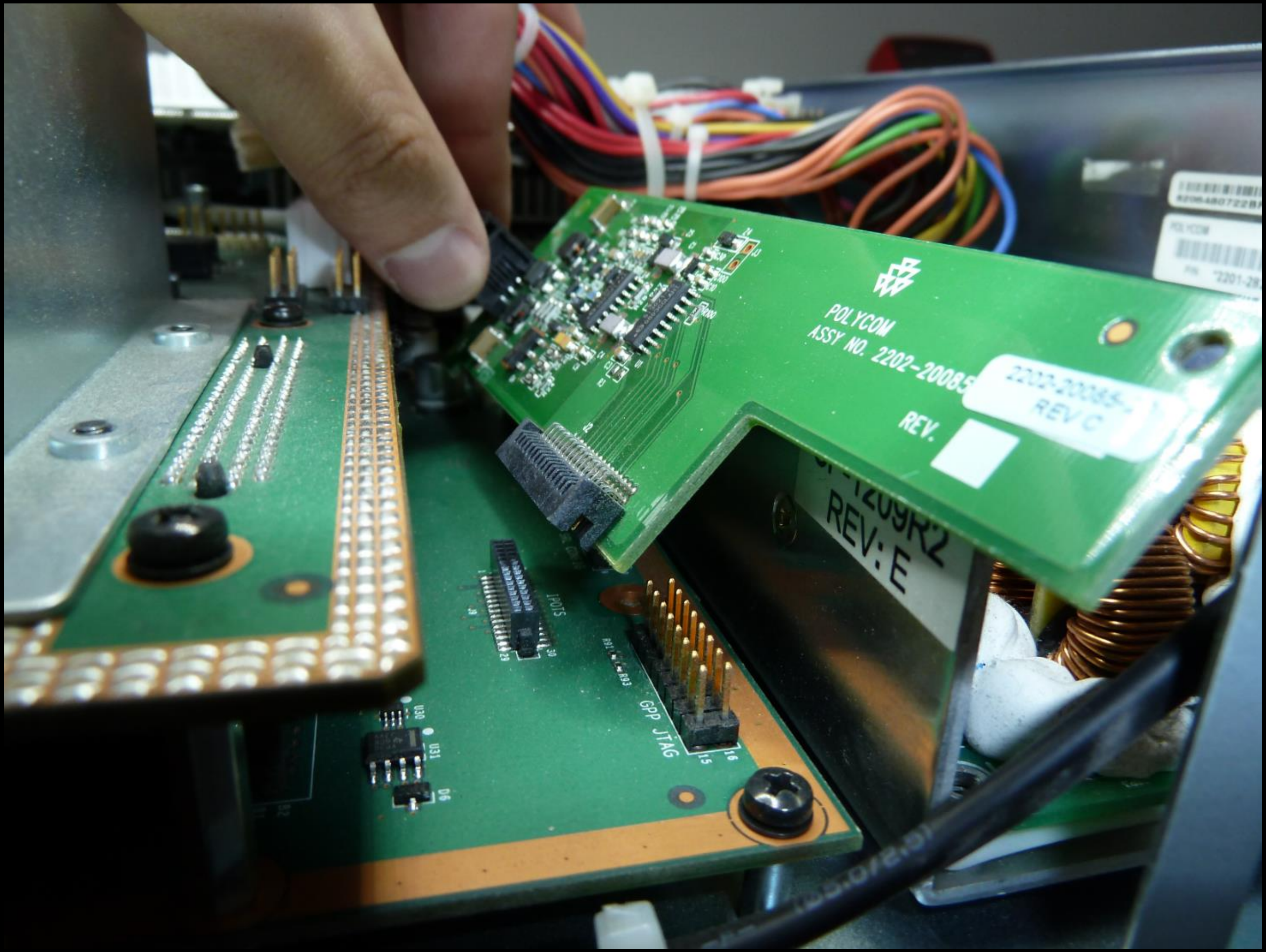
Removing Internal Modem



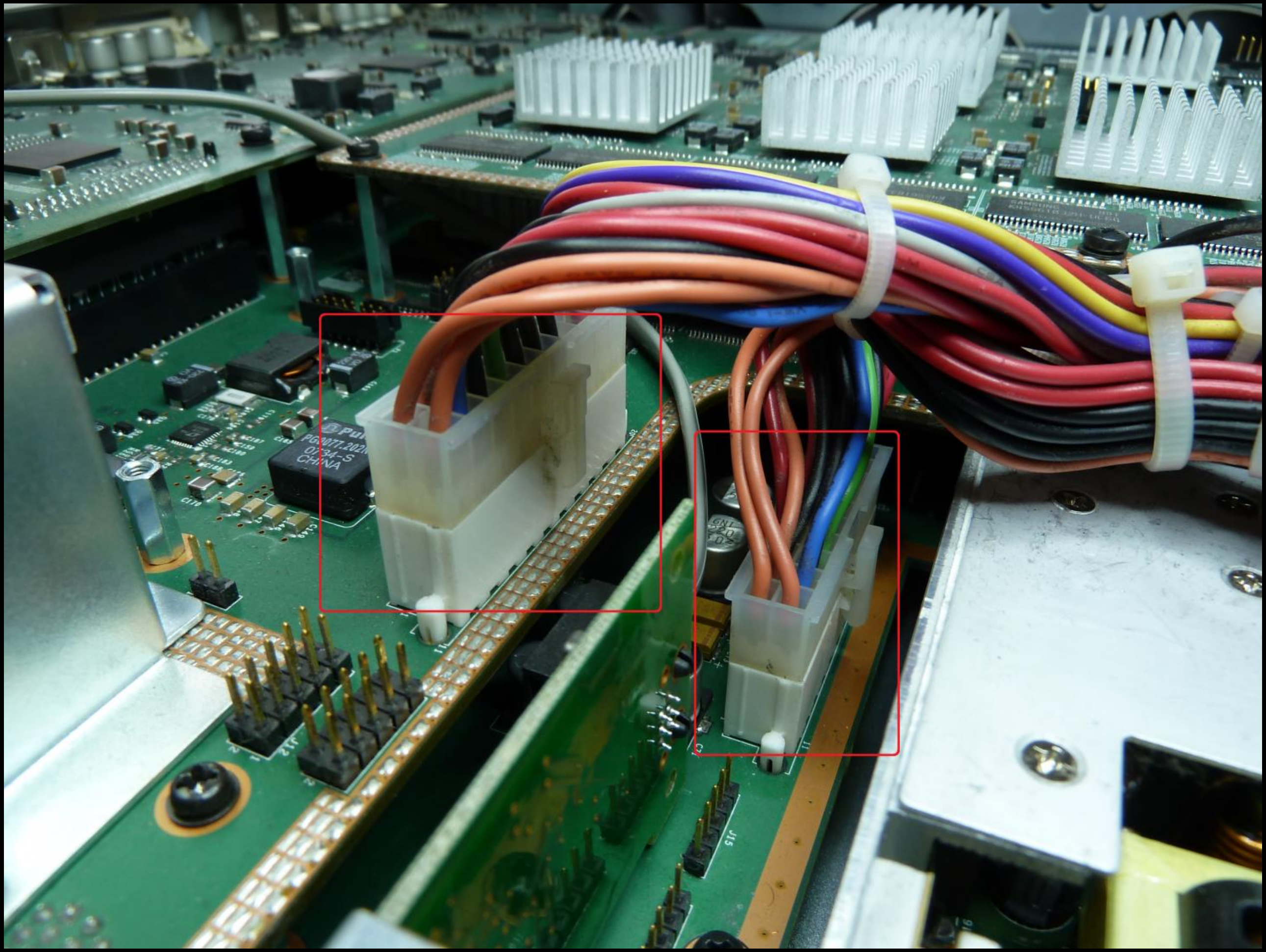
Removing Internal Modem



Removing Internal Modem

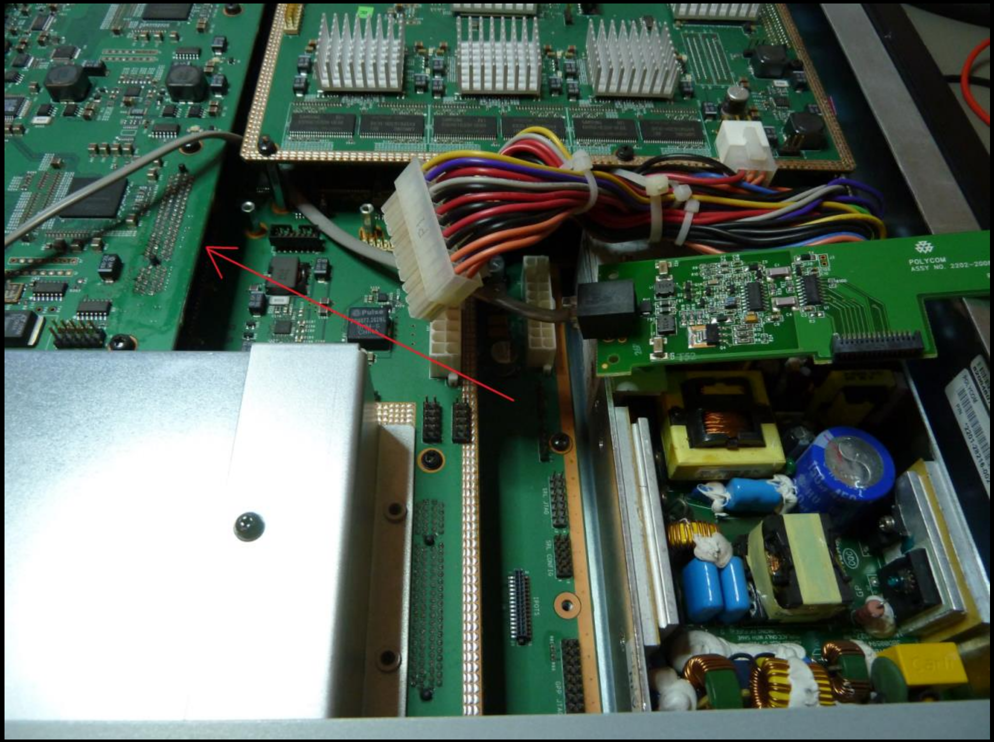


Removing Power Connectors



Removing CF Card Screw

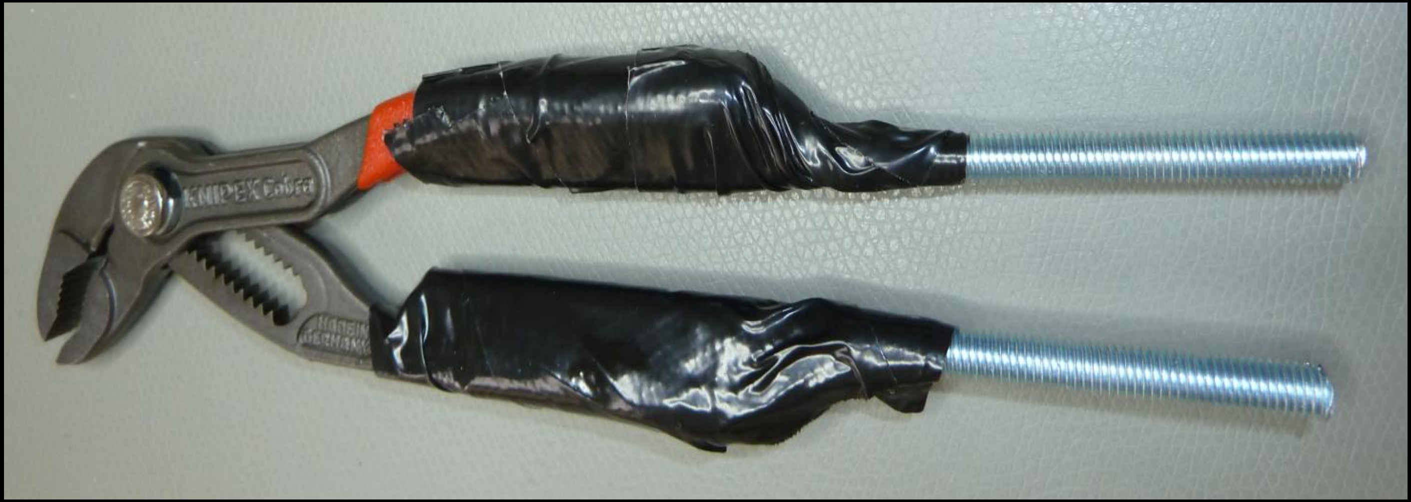
- Touching the screw holding the CF card with a single finger is now possible



Removing CF Card Screw

- Place one hand under covering PCB
- Touching screw with single finger is now possible
- But screw must be loosened first…

Used Tools ;)



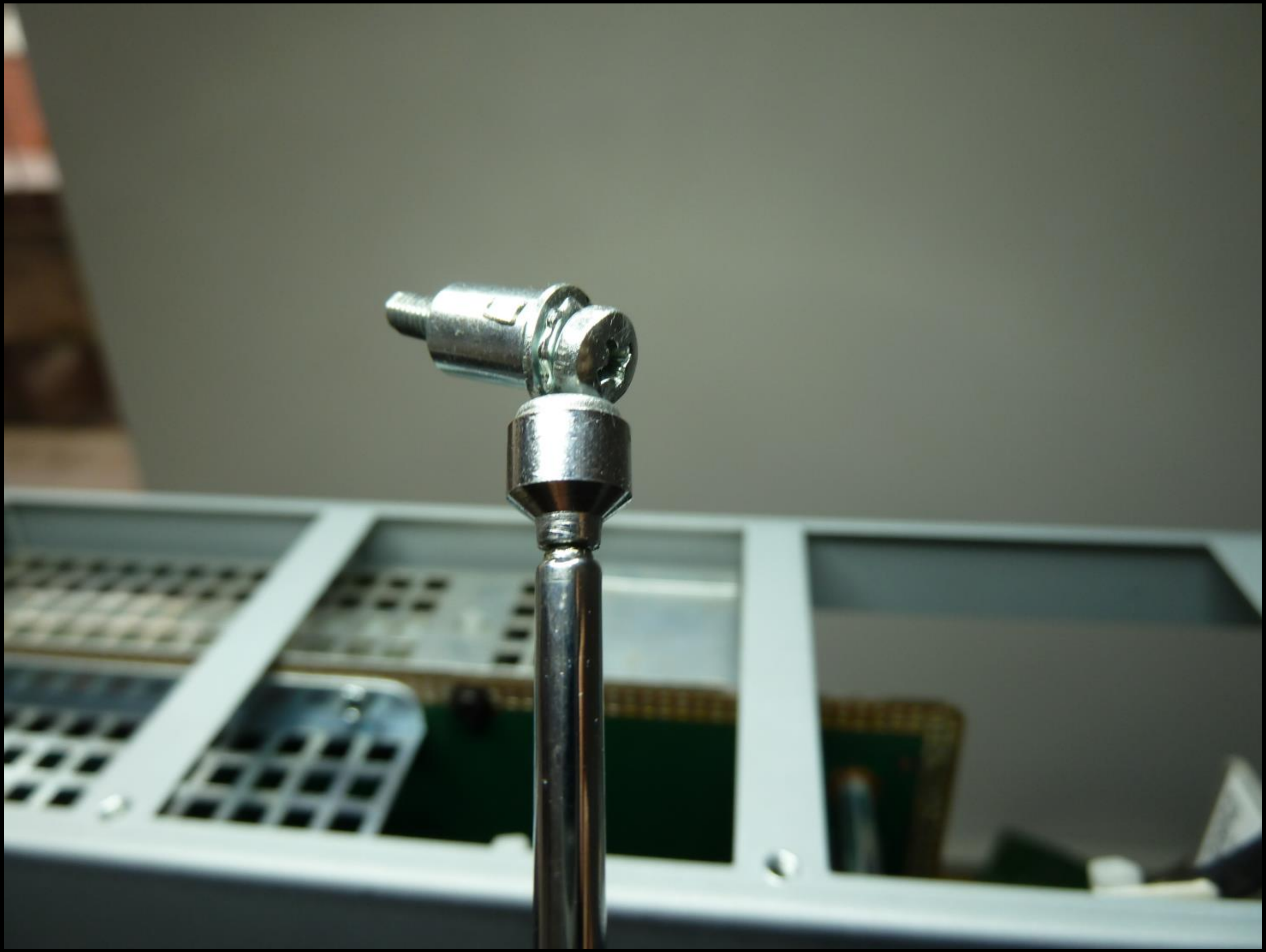
Removing CF Card Screw

- Extended nipper used to loosen screw
- Nipper can't be rotated enough
- Used magnetic stick to turn the screw

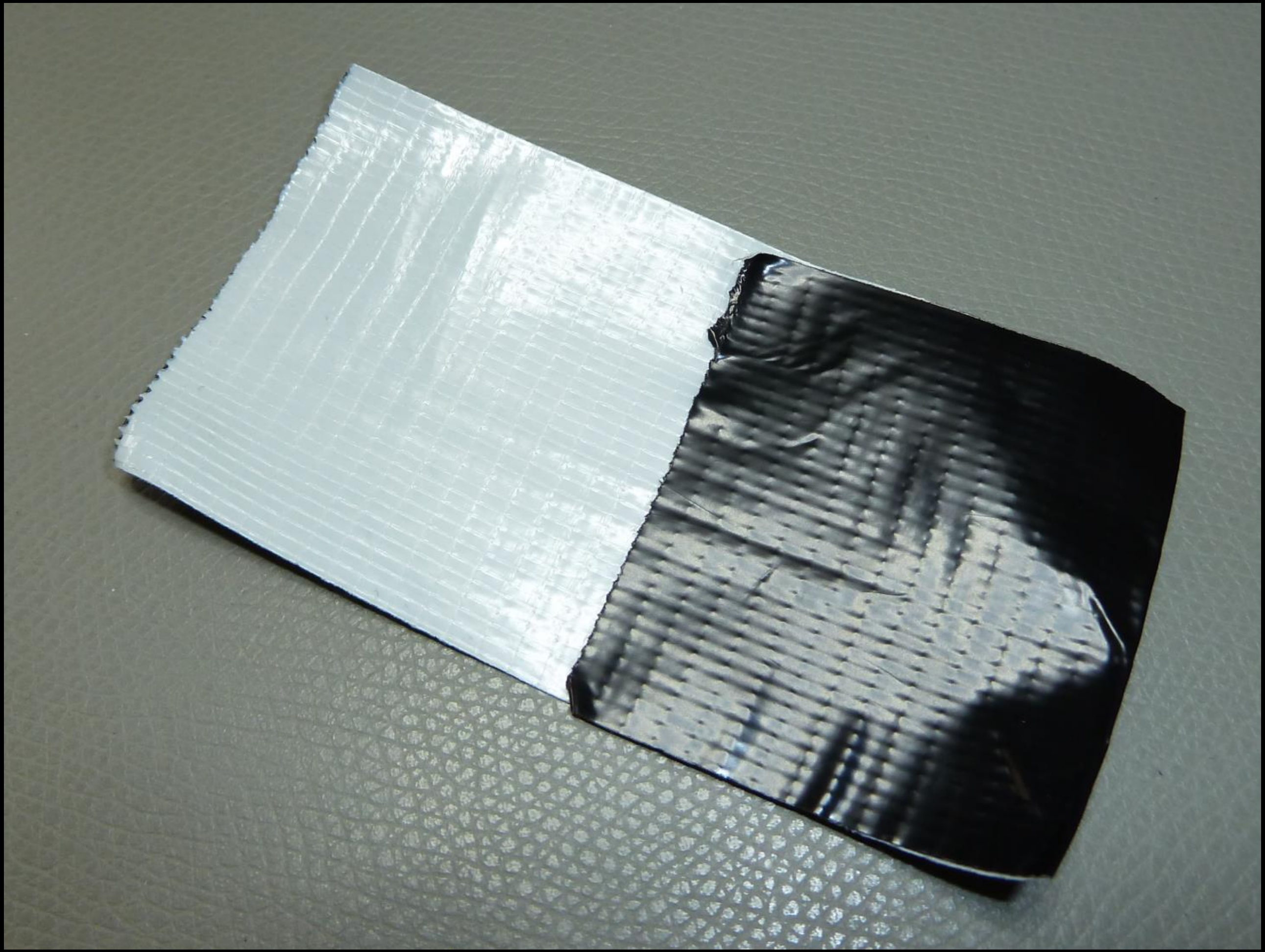
- This was *really* fiddly and required nerves!
  - Probably lost some hair during this operation

Removing CF Card Screw

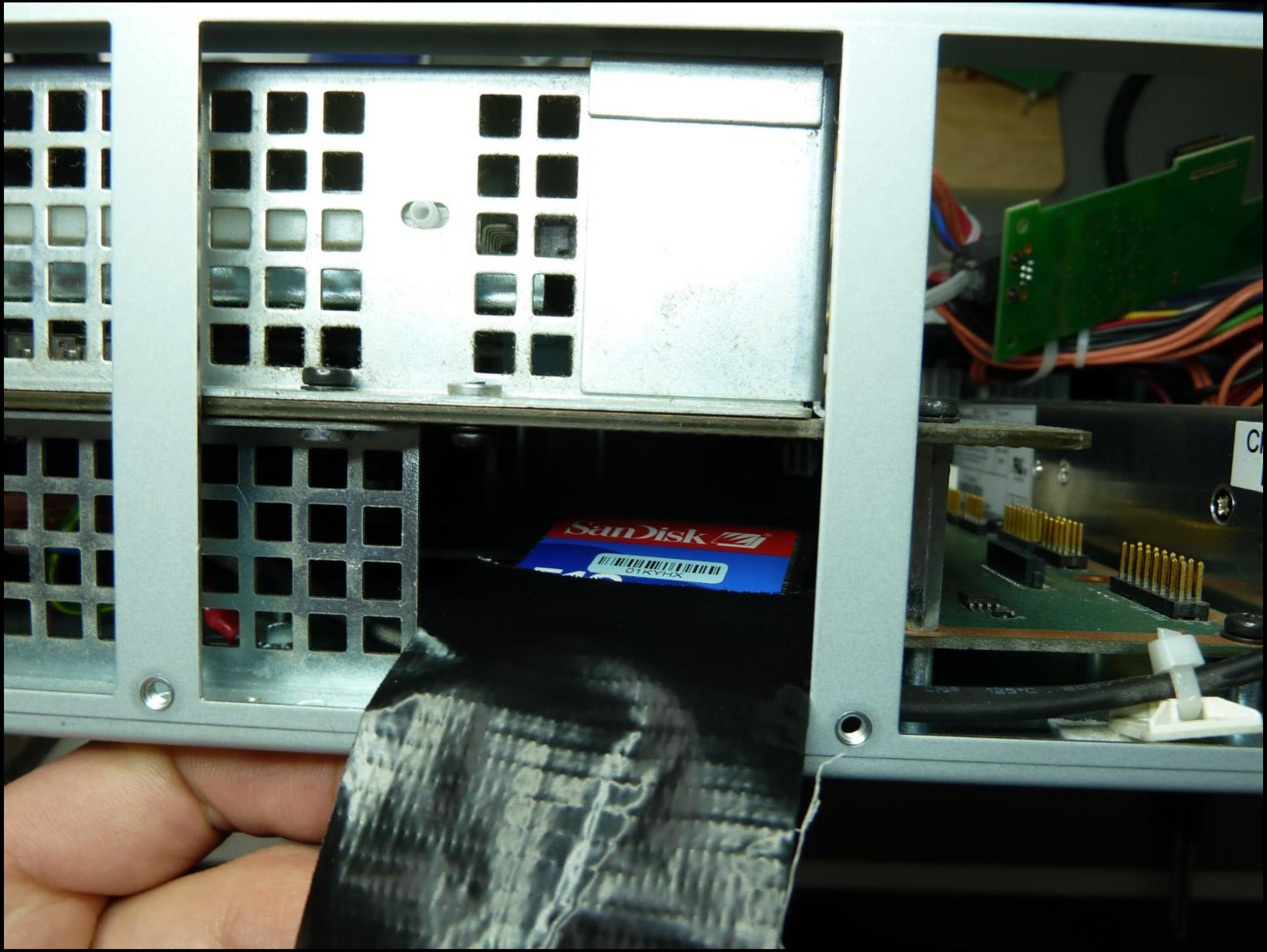
Removing CF Card Screw



Removing CF Card



Removing CF Card



File System Analysis

File System Analysis

- Analysis on created CF card image
- HDX systems have four partitions

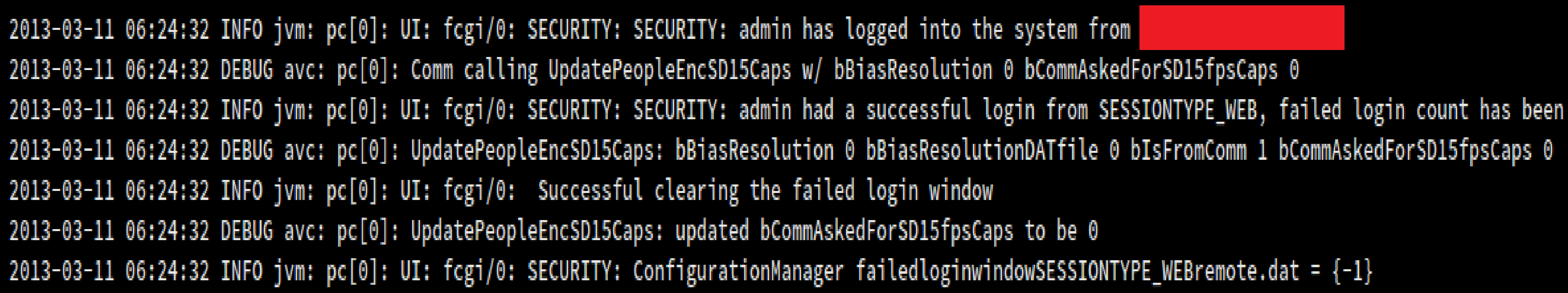
Partition	Description	Type	Mounted
/dev/hda1	Boot related files, Linux kernel image	ext2	ro
/dev/hda2	Root file system	ext2	ro
/dev/hda3	Log and configuration files	ext3	rw
/dev/hda4	Factory restore file system	ext2	--

Log Files

- Stored in /var/log on /dev/hda3
- Pretty extensive logging by default
  - Good for the forensic analysis
  - Bad, because logs get overwritten quickly

Things to look for

- Failed or successful login attempts
- Initiated video calls
- Typical Linux-based forensics stuff
  - Crashed daemons
  - reboots, etc.



```
2013-03-11 06:24:32 INFO jvm: pc[0]: UI: fcgi/0: SECURITY: SECURITY: admin has logged into the system from [redacted]
2013-03-11 06:24:32 DEBUG avc: pc[0]: Comm calling UpdatePeopleEncSD15Caps w/ bBiasResolution 0 bCommAskedForSD15fpsCaps 0
2013-03-11 06:24:32 INFO jvm: pc[0]: UI: fcgi/0: SECURITY: SECURITY: admin had a successful login from SESSIONTYPE_WEB, failed login count has been
2013-03-11 06:24:32 DEBUG avc: pc[0]: UpdatePeopleEncSD15Caps: bBiasResolution 0 bBiasResolutionDATfile 0 bIsFromComm 1 bCommAskedForSD15fpsCaps 0
2013-03-11 06:24:32 INFO jvm: pc[0]: UI: fcgi/0:  Successful clearing the failed login window
2013-03-11 06:24:32 DEBUG avc: pc[0]: UpdatePeopleEncSD15Caps: updated bCommAskedForSD15fpsCaps to be 0
2013-03-11 06:24:32 INFO jvm: pc[0]: UI: fcgi/0: SECURITY: ConfigurationManager failedloginwindowSESSIONTYPE_WEBremote.dat = {-1}
```

Configuration Files

- Stored in /dat directory on /dev/hda3
- Every setting stored in single .dat file
- Text-based files
  - – One or more lines of text

Interesting Config Files

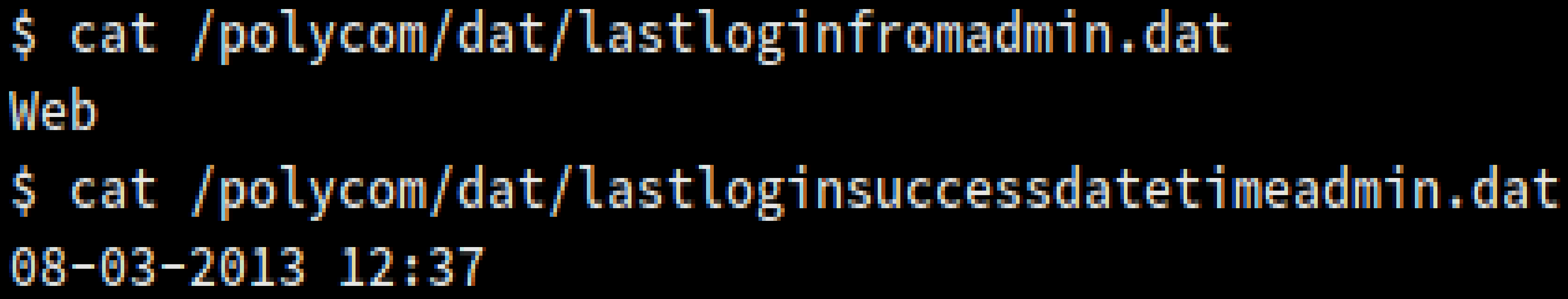
- Version of current firmware
  - Stored in systemsoftwareversion.dat
  - Known vulnerabilities in old versions
- Hashes of previously set passwords
  - historymeetingpassword.dat
  - historyremotepassword.dat
  - historyroomsw.dat

Password Hashes

- Stored to prevent password re-use
- Passwords stored as SHA1 hashes
  - Unsalted of course :)
- Cracking the SHA1 hashes
  - Identifies potentially weak passwords
  - Might give you password set by an attacker
- Timestamps indicate time of PW change

Last Adminstrator Login

- Last admin login is recorded
  - lastloginfromadmin.dat
  - lastloginsuccessdatetimeadmin.dat
- Can be correlated with timestamps



```
$ cat /polycom/dat/lastloginfromadmin.dat
Web
$ cat /polycom/dat/lastloginsuccessdatetimeadmin.dat
08-03-2013 12:37
```

Call Detail Records

- Stored as a SQLite database
  - /data/polycom/cdr/new/localcdr.db
- Included information
  - Start and end date/time
  - Call duration
  - Called number
  - Call direction
  - Used protocols, etc.

Polycom Command Shell

- Was affected by remote vulns in the past
- Check if PSH was enabled
  - telnet_enabled.dat

Root File System

- Always mounted read-only
  - Only mounted read-write for updates
- Check last-modified timestamps
- Match all files against original image

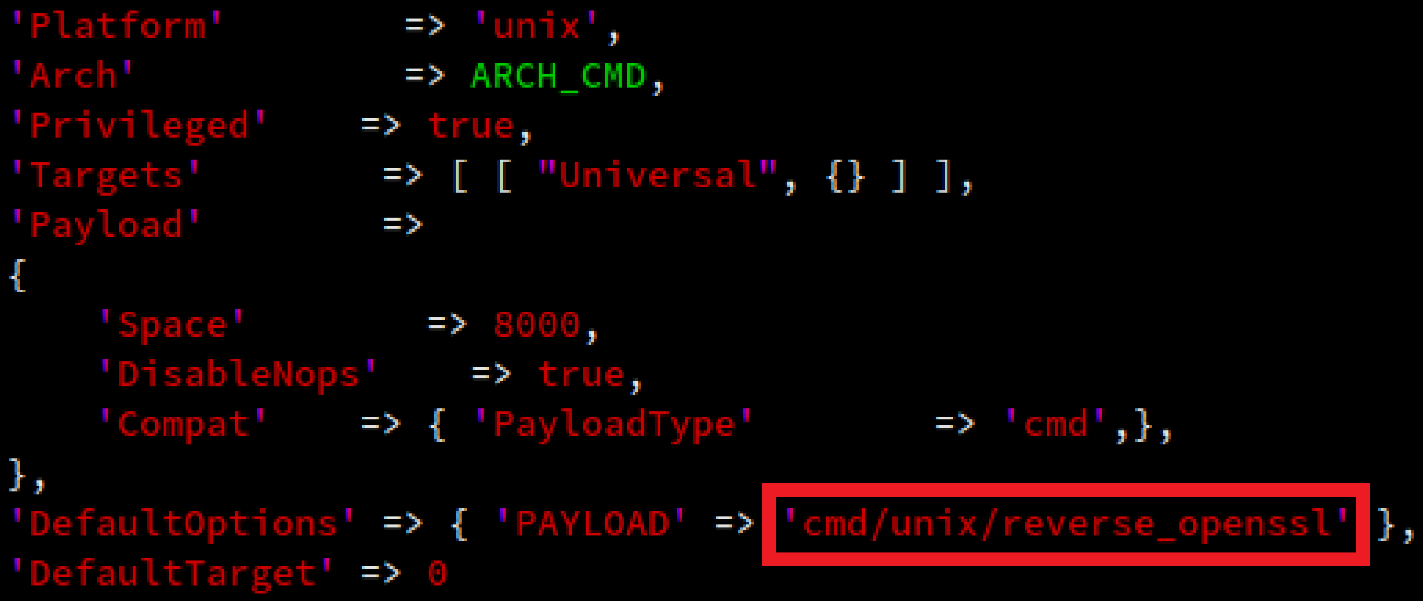
Use of Public Exploits

- Access times might identify use of specific public exploits…

- Metasploit PSH Telnet Auth Bypass
  - Module psh_auth_bypass.rb
  - Exploits auth bypass + command injection
  - Uses OpenSSL reverse connect payload

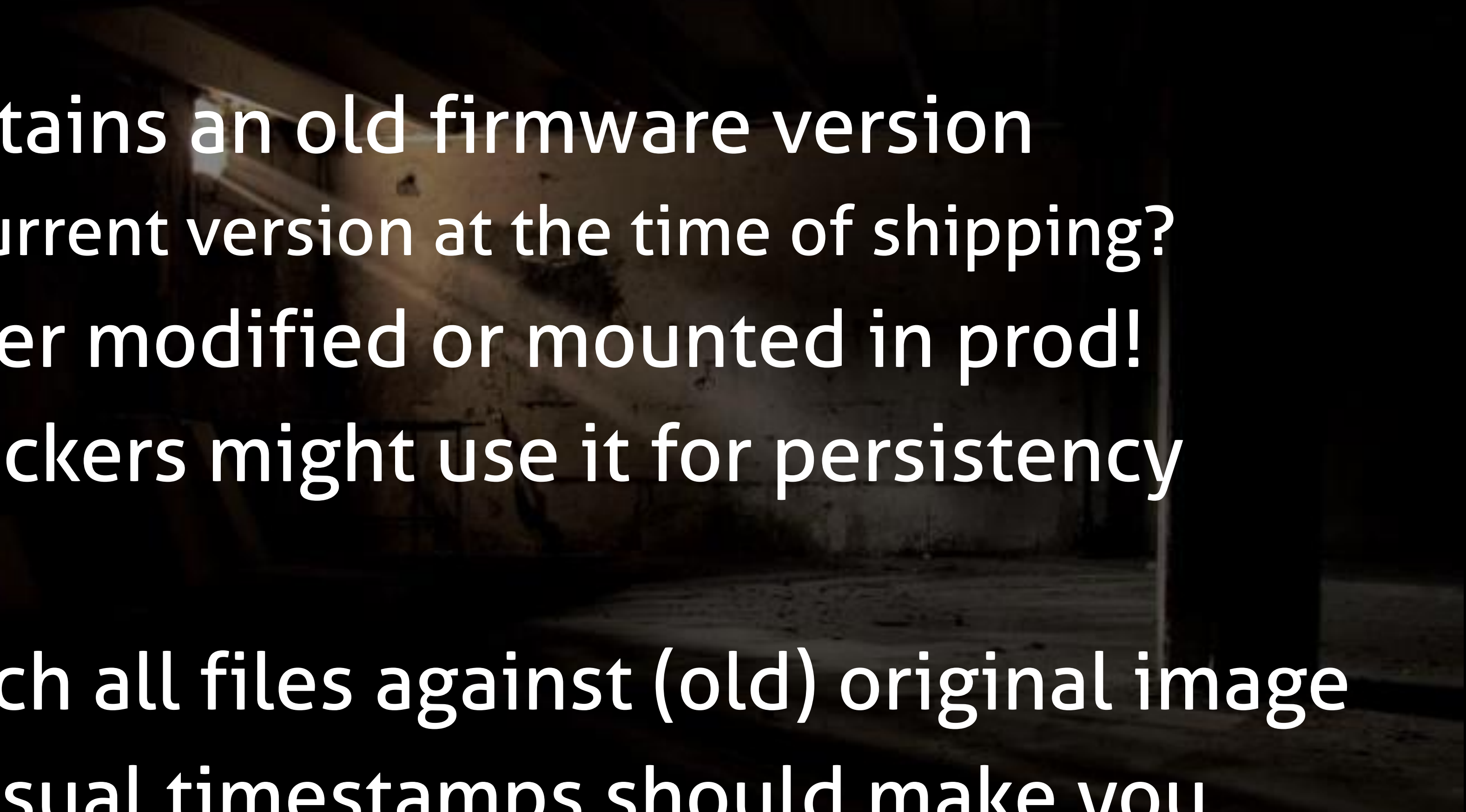
Use of Public Exploits

- cmd/unix/reverse_openssl
  - Uses busybox and openssl binaries
  - Binaries not regularly called in production



```
'Platform'        => 'unix',
'Arch'            => ARCH_CMD,
'Privileged'    => true,
'Targets'         => [ [ "Universal", {} ] ],
'Payload'         =>
{
    'Space'         => 8000,
    'DisableNops'     => true,
    'Compat'    => { 'PayloadType'        => 'cmd',},
},
'DefaultOptions' => { 'PAYLOAD' => 'cmd/unix/reverse_openssl' },
'DefaultTarget' => 0
```

Factory Restore Filesystem



- Contains an old firmware version
  - Current version at the time of shipping?
- Never modified or mounted in prod!
- Attackers might use it for persistency

- Match all files against (old) original image
- Unusual timestamps should make you suspicious

Conclusion

- Forensics on VC systems requires internal system knowledge
- Knowing how to break them helps
- No *advanced* attacks observed yet
  - But they happen! (see NSA hack)
- Having the right hardware tools helps :P

Questions?

Thank You!

Moritz Jodeit
Principal Security Consultant

mobile: +49 170 288 4291
moritz.jodeit@nruns.com

n.runs professionals GmbH
Nassauer Straße 60
D-61440 Oberursel

phone: +49 6171 699-530
fax: +49 6171 699-199

www.nruns.com

it. consulting . infrastructure . security . business