NAC@ACK

Michael Thumann
&
Dror-John Roecher

Agenda

- **Part 1 – Introduction (very short)**
  - Some marketing buzz on Cisco NAC
- **Part 2 – NAC Technology**
  - All you need to know about NAC (in order to hack it)
- **Part 3 – Security Analysis**
  - Delving into the security flaws of Ciscos‘ NAC solution
- **Part 4 – Approaching NAC@ACK**
  - The stony road towards a working exploit
- **Part 5 - Showtime**

Part 1 - Introduction

Why is Cisco selling Cisco NAC?

- **Because customers are willing to pay for it ,-)**
- **But why are customers willing to pay for it?**
- **Because Cisco makes some pretty cool promises… see next slide**

From: http://www.cisco.com/go/nac

NAC Business Benefits

Dramatically improves security

- Ensures endpoints (laptops, PCs, PDAs, servers, etc.) conform to security policy
- Proactively protects against worms, viruses, spyware, and malware; focuses operations on prevention, not reaction

Extends existing investment

- Enables broad integration with multivendor security and management software
- Enhances investment in network infrastructure and vendor software
- Combining with Cisco Security Agent enables "trusted QoS" capabilities that classify mission-critical traffic at the endpoint and prioritize it in the network

Increases enterprise resilience

- Comprehensive admission control across all access methods
- Prevents non-compliant and rogue endpoints from impacting network
- Reduces OpEx related to identifying and repairing non-compliant, rogue, and infected systems

Comprehensive span of control

- Assesses all endpoints across all access methods, including LAN, wireless connectivity, remote access, and WAN

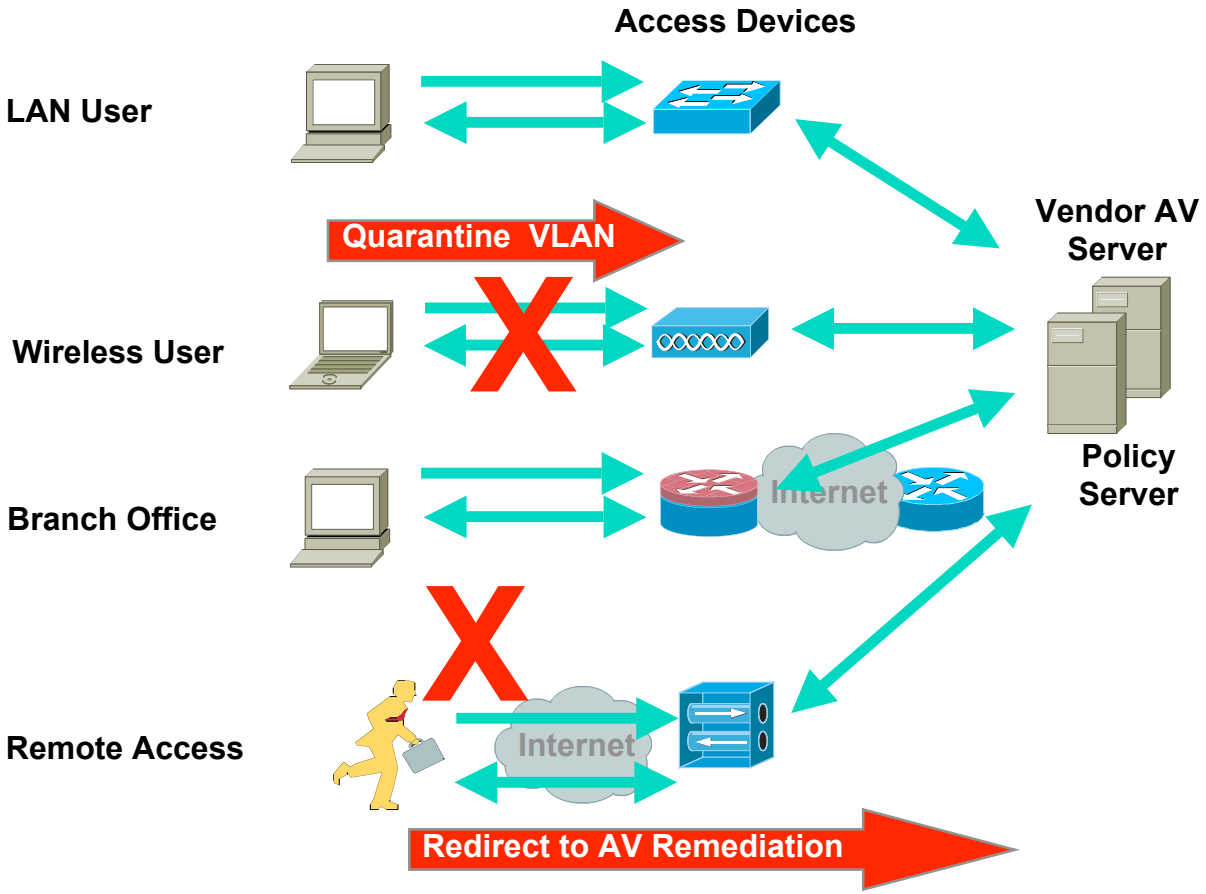
The idea behind Cisco NAC

- **Grant access to the network based on the grade of compliance to a defined (security) policy. So it is first of all a compliance solution and not a security solution.**

- **Security Policy can usually be broken down to:**
  - Patch level (OS & Application)
  - AV signatures & scan engine up to date
  - No „unwanted“ programs (e.g. l33t t00ls)
  - Desktop Firewall up & running

- **If a client is non-compliant to the policy [and is not whitelisted somewhere – think network-printers], restrict access.**

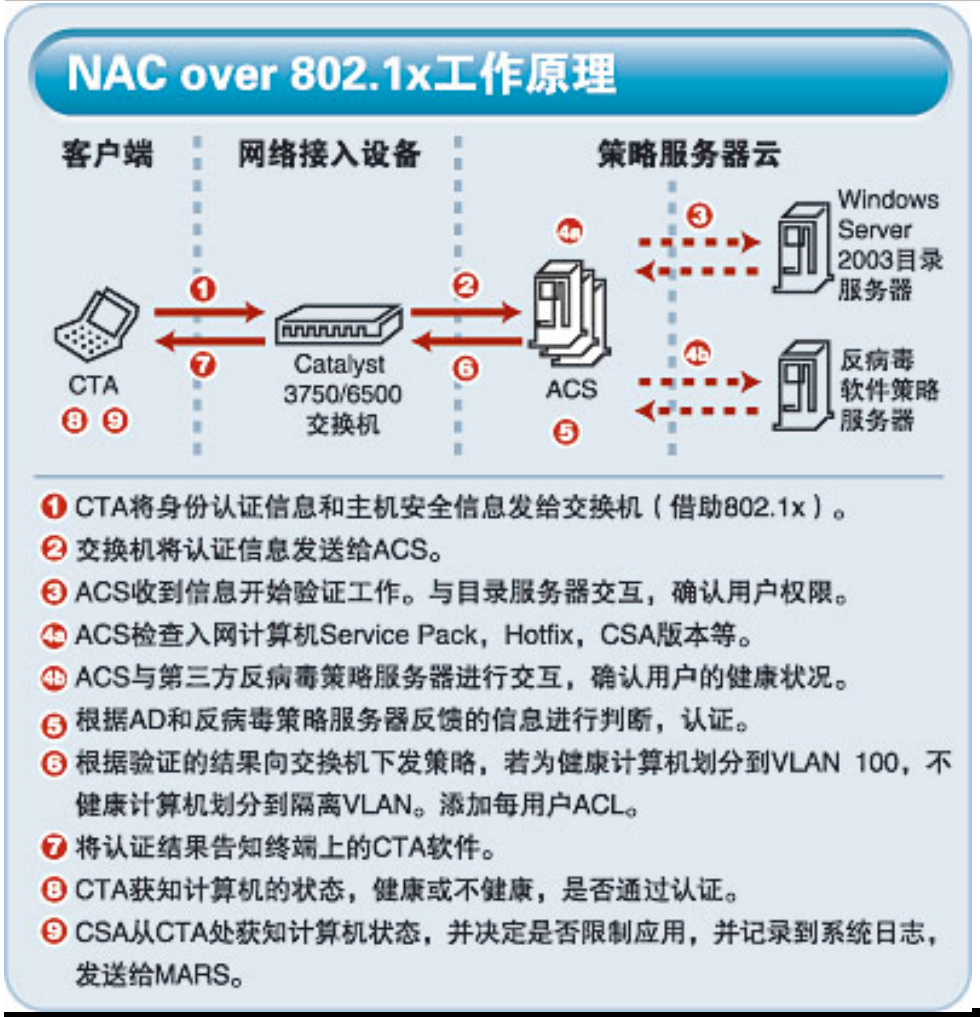
Policy based Access…



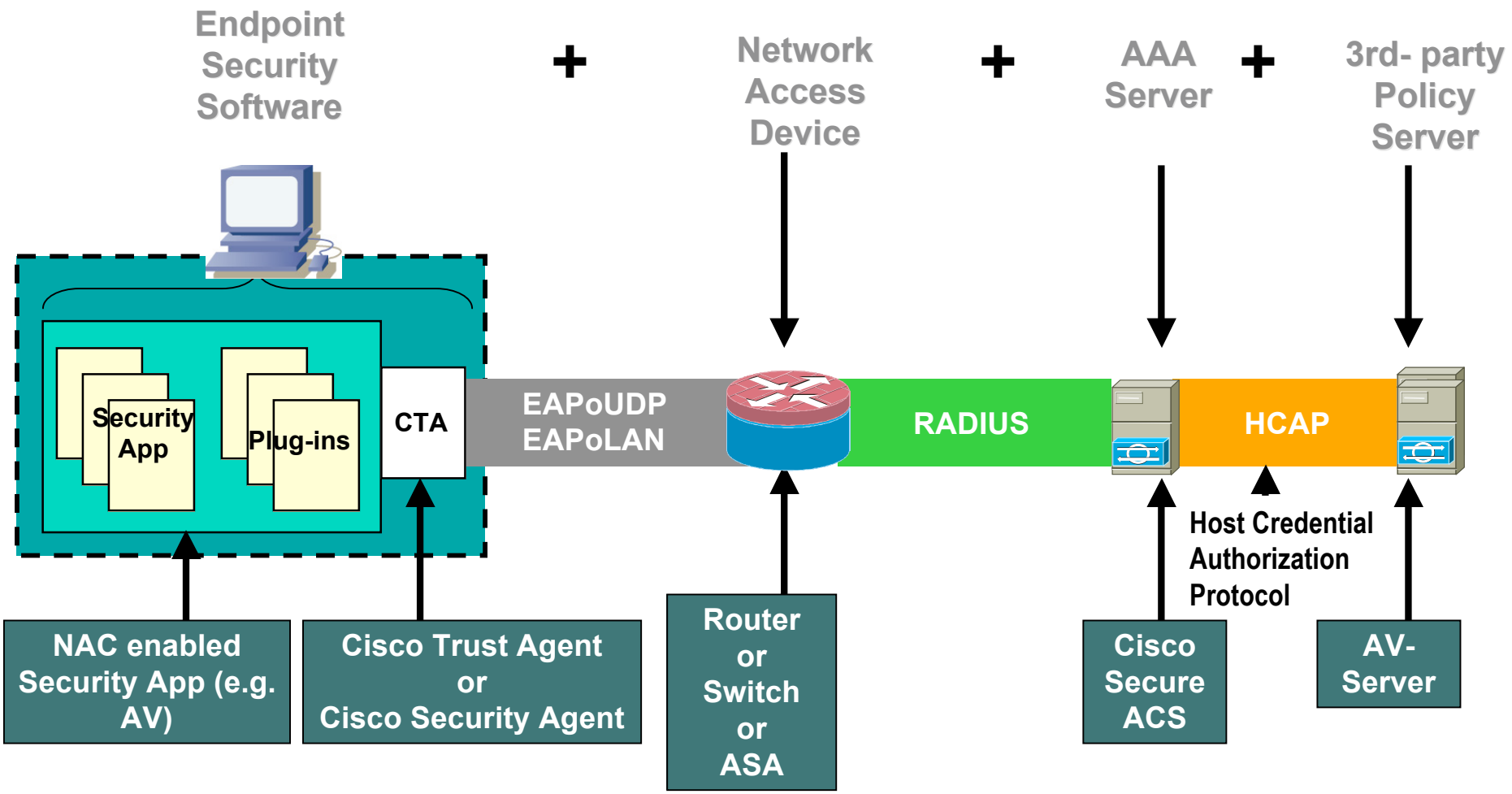
1. Access Device detects new client.
2. Access Device queries the client for an agent and relays information to a backend policy server.
3. Policy Server checks received information against defined rules and derives an appropriate access-level
4. Access-Device enforces restrictions

Part 2 – NAC Technology

What is Cisco NAC?



A „big overview“ picture…



There are 3 different NAC flavours…

- **NAC-Layer3-IP**
  - Access-restrictions are implemented as IP-ACLs
  - NAD is a Layer-3 device (e.g. a Router or a VPN-Concentrator/Firewall).
  - The communication takes place using PEAP over EAP over UDP (EoU).
- **NAC-Layer2-IP**
  - Access-restrictions as IP-ACLs on a VLAN-interface of a switch.
  - The communication takes place using PEAP over EAP over UDP (EoU)
- **NAC-Layer2-802.1x**
  - Uses 802.1x port control to restrict network access
  - Obviously the device enforcing these restrictions is a switch.
  - EAP-FAST is used in conjunction with 802.1x.
  - This is the only NAC flavour where the client is:
    - authenticated before being allowed on the network
    - restricted from communicating with its local subnet

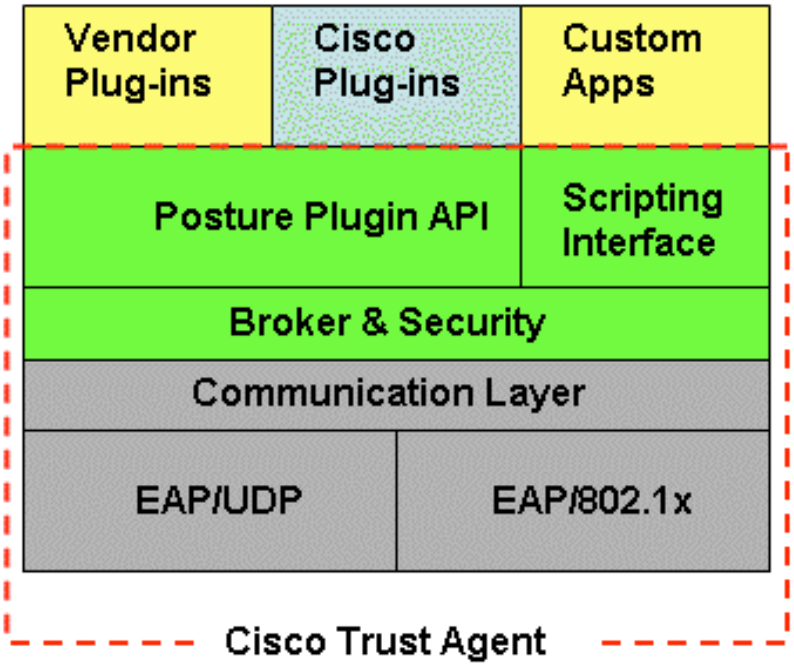
(Some) Features…

Feature	NAC-L2-802.1x	NAC-L2-IP	NAC-L3-IP
Trigger	Data Link / Switchport	DHCP / ARP	Routed Packet
Machine ID	Yes	No	No
User ID	Yes	No	No
Posture	Yes	Yes	Yes
VLAN Assignment	Yes	No	No
URL Redirection	No	Yes	Yes
Downloadable ACLs	Cat65k only	Yes	Yes

Yet another agent: Cisco Trust Agent

- **The Cisco Trust Agent (CTA) is the main component of the NAC framework installed on the clients.**
- **Its‘ tasks are to collect „posture data“ about the client and forward it to the ACS via the NAD.**
- **It has a plug-in interface for 3rd party vendors‘ NAC-enabled applications.**
- **It has a scripting interface for self-written scripts.**

CTA architecture



- **The CTA comes with two plug-ins by default:**
  - Cisco:PA
  - Cisco:Host

Posture Information

- **The information collected are Attribute-Value-pairs categorized by**
  - Vendor: ID based on IANA SMI assignement
  - Application-Type: see next slide
  - Credential Name: e.g. “OS Version”
  - Value-Format: String, Date, etc.
- **For all plug-ins & scripts this information is collected in a plaintext “.inf-file”.**

Application Types in Cisco NAC

Application-Type ID	Application-Type Name	Usage
1	PA	Posture Agent
2	Host / OS	Host information
3	AV	Anti Virus
4	FW	Firewall
5	HIPS	Host IPS
6	Audit	Audit
32768 – 65536		Reserved for “local use” (custom plug-ins or scripts)

Credentials for Cisco:PA & Cisco:Hosts

Application-Type	Attribute Number	Attribute Name	Value-Type
Posture Agent	3	Agent-Name (PA-Name)	String
	4	Agent-Version	Version
	5	OS-Type	String
	6	OS-Version	Version
	7	User-Notification	String
	8	OS-Kernel	String
	9	OS-Kernel-Version	Version
Host	11	Machine-Posture-State	1 – Booting, 2 – Running, 3 – Logged in.
	6	Service Packs	String
	7	Hot Fixes	String
	8	Host-FQDN	String

Posture Tokens…

- **For each plug-in/Application/script an "Application Posture Token" (APT) is derived by the ACS through the configured policy.**
- **This token is one out of:**
  - Healthy, Checkup, Quarantine, Transition, Infected, Unknown (see next slide for definitions of these tokens)
- **From all APTs a "System Posture Token" (SPT) is derived – this corresponds to the APT which will grant the least access on the network to the client.**
- **The SPT is associated with access-restrictions on the ACS (e.g. downloadable ACL, URL-Redirection).**

Posture Tokens – well defined

- **"Healthy": fully compliant with the admission policy for the specified application.**
- **"Checkup": partial but sufficient compliance with the admission policy, no need to restrict access, a warning to the user may be issued.**
- **"Transition": either during boot-time, when not all necessary services have been started or during an audit-process for clientless hosts, temporary access-restrictions may be applied.**
- **"Quarantine": insufficient compliance with the admission policy, network access is usually restricted to a quarantine/remediation segment.**
- **"Infected": active infection detected, usually most restrictive network access even up to complete isolation.**
- **"Unknown": a token can not be determined or no CTA installed on client. This may lead to partial access (guest-vlan & internet-access for example).**

Sample inf-File for Trendmicro AV

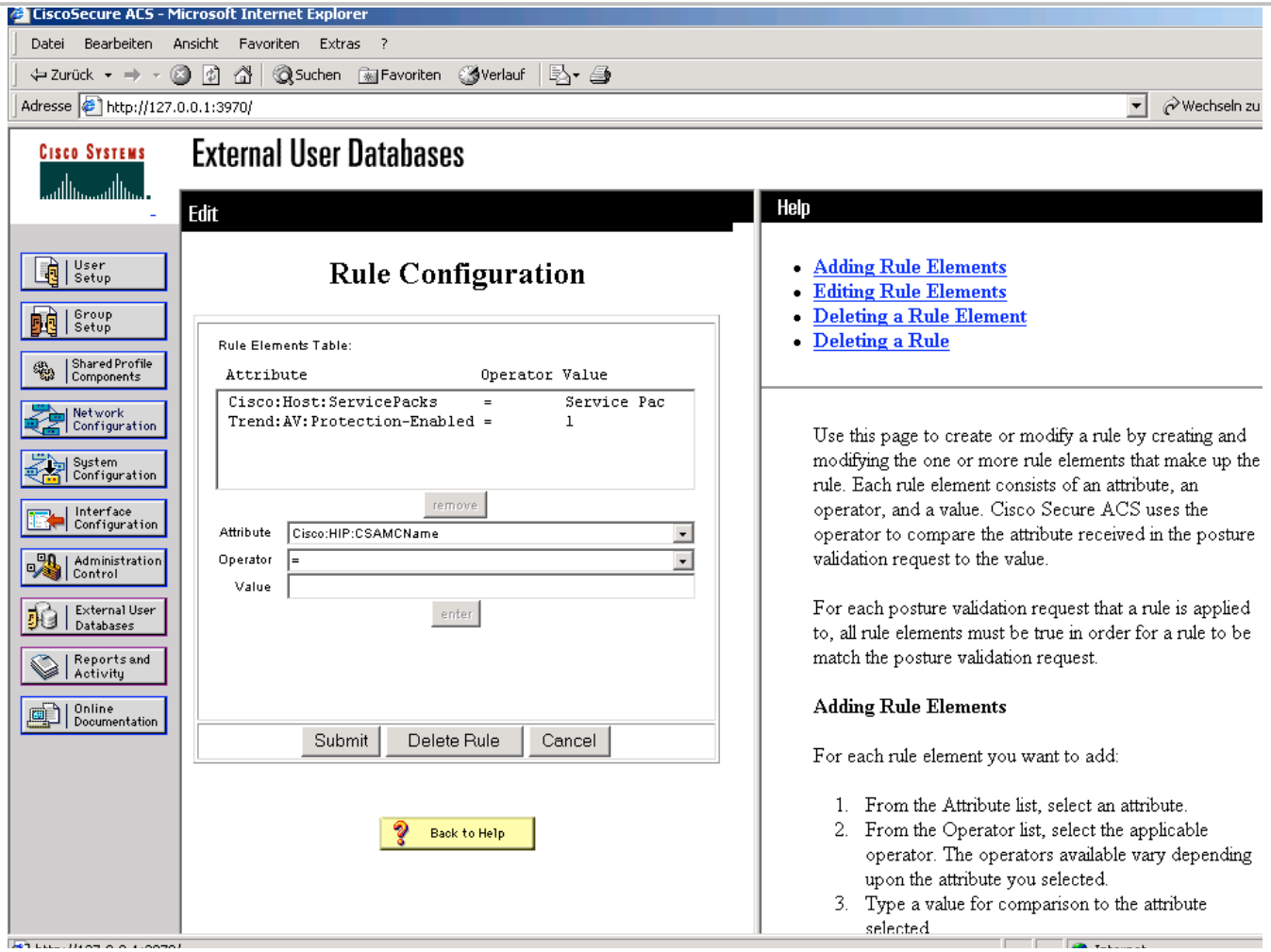
```
[main]

dll=tmabpp.dll
PluginName=tmabpp.dll
VendorID=6101
VendorIDName=TrendMicro, Inc
AppList=av

[av]

AppType=3
AppTypeName=Antivirus
AttributeList=attr1,attr2,attr3,attr4,attr5,attr6,attr7,attr8,attr9,attr10,attr11,attr12,attr13,attr14
attr1= 1, Unsigned32, Application-Posture-Token
attr2=2, Unsigned32, System-Posture-Token
attr3=3, String, Software-Name
attr4=4, Unsigned32, Software-ID
attr5=5, Version, Software-Version
attr6=6, Version, Scan-Engine-Version
attr7=7, Version, Dat-Version
attr8=8, Time, Dat-Date
attr9=9, Unsigned32, Protection-Enabled
attr10=10, String, Action


attr11=32768, String, OSCE-Srv-Hostname
attr12=32769, OctetArray, Client-GUID
attr13=32770, Ipv4Address, Client-IP
attr14=32771, OctetArray,  Client-MACddd
```

The name of the plug-in. In case of a script this would be ctascriptPP.dll and the vendor-id would be "Cisco" for scripts.

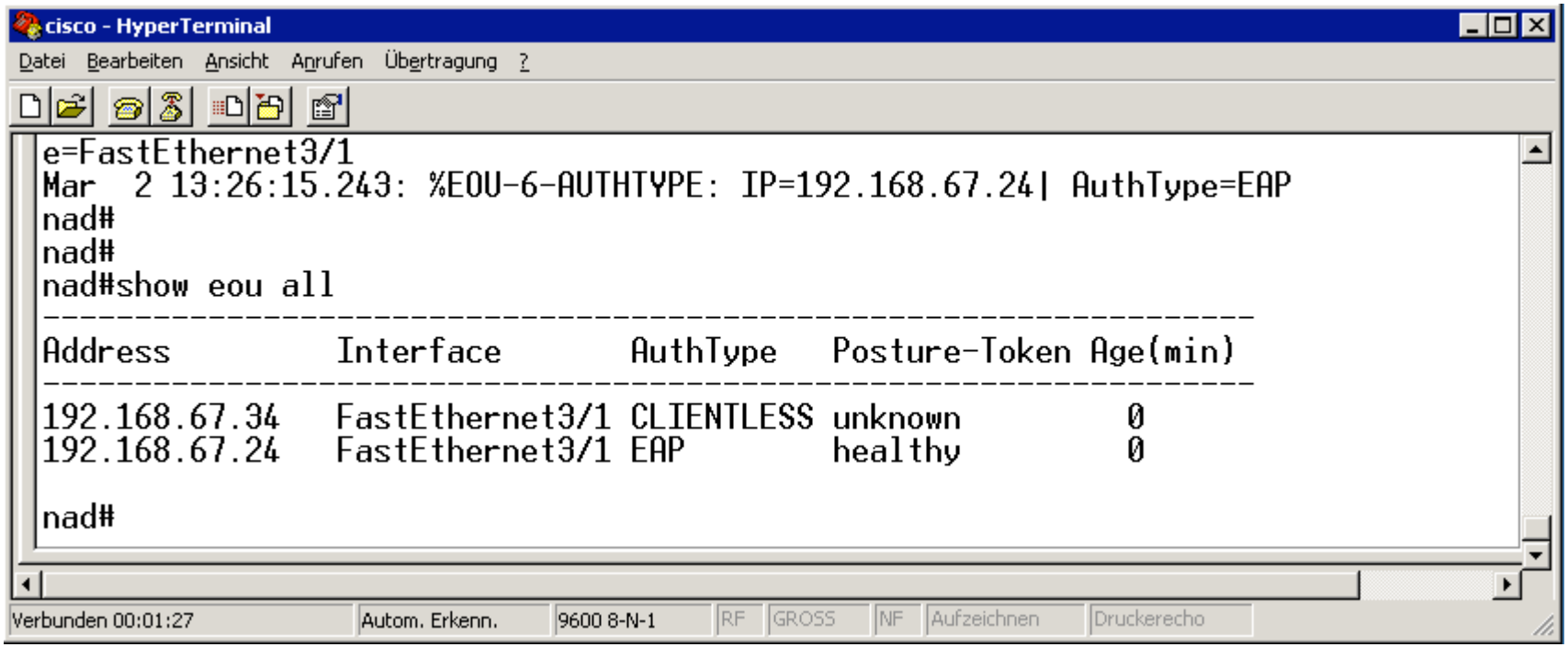
Official Credentials

Private Credentials from the Vendor

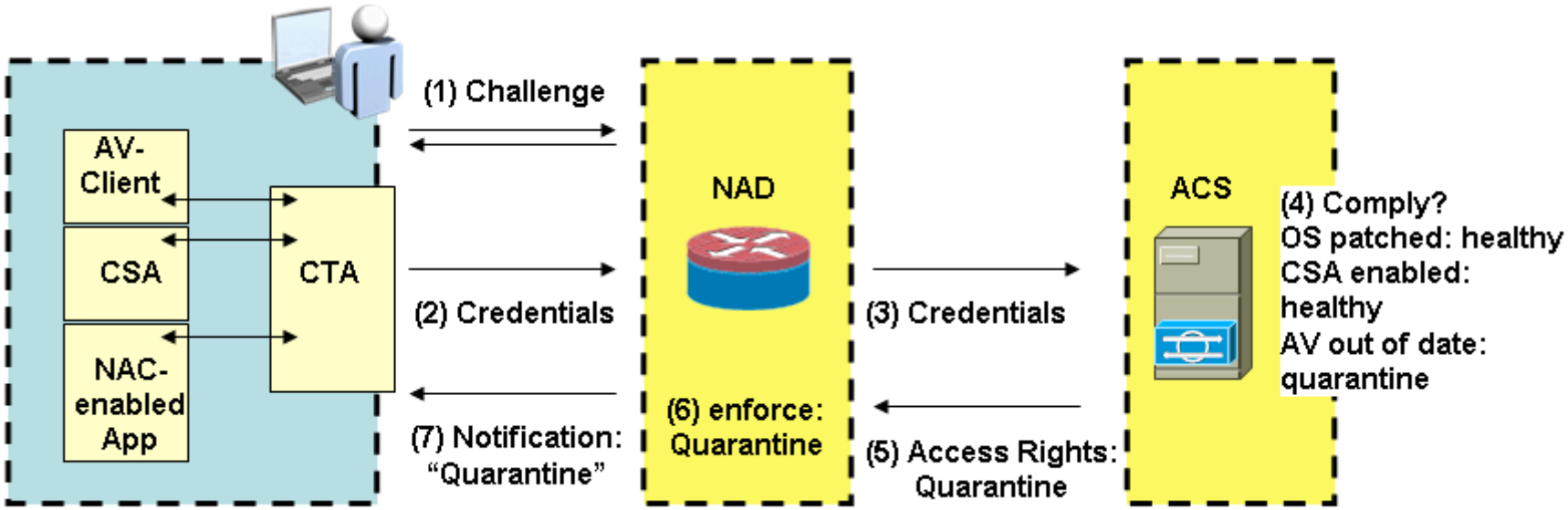
Sample Policy on Cisco ACS



And the resulting SPT on a NAD



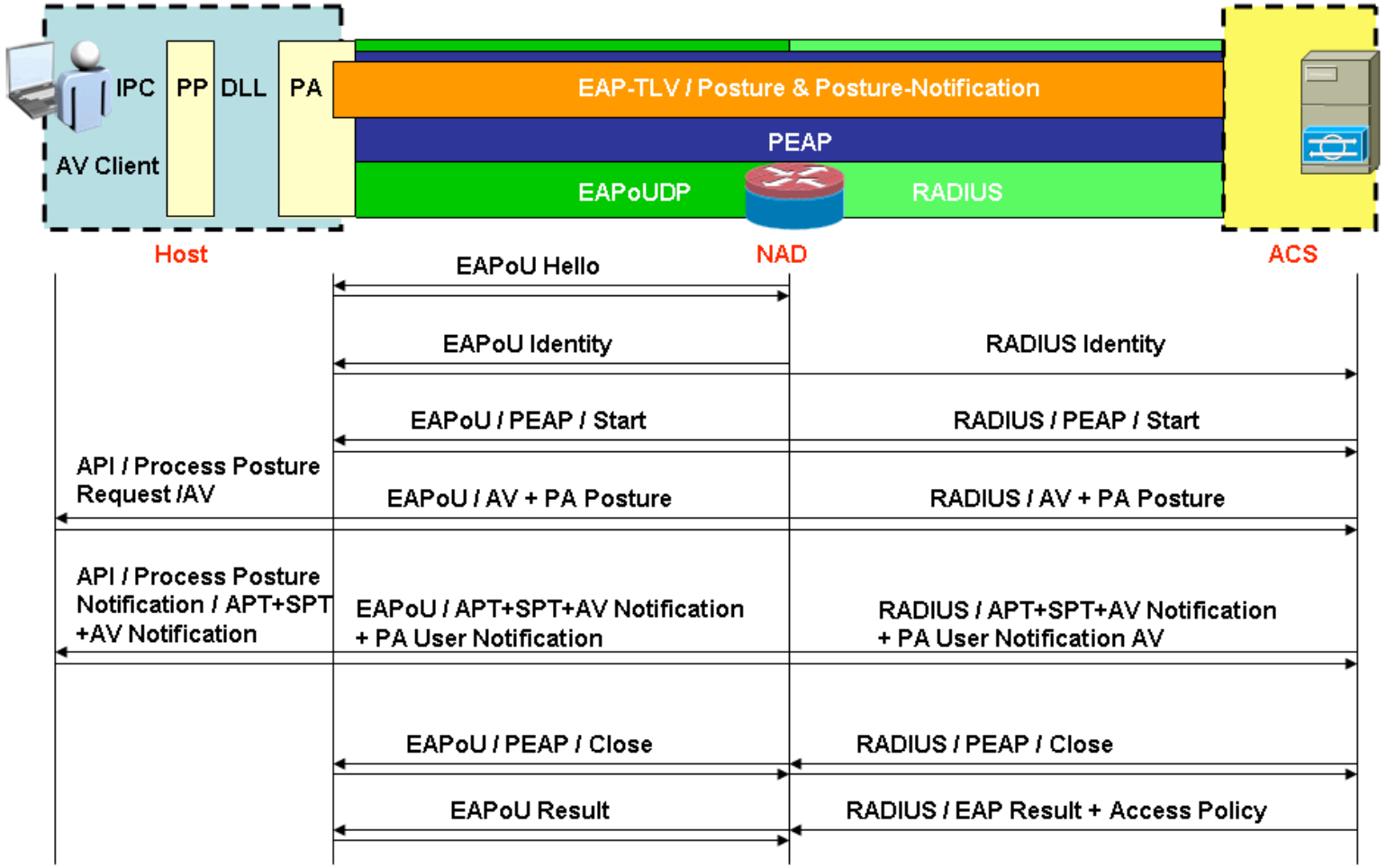
General Communication Flow



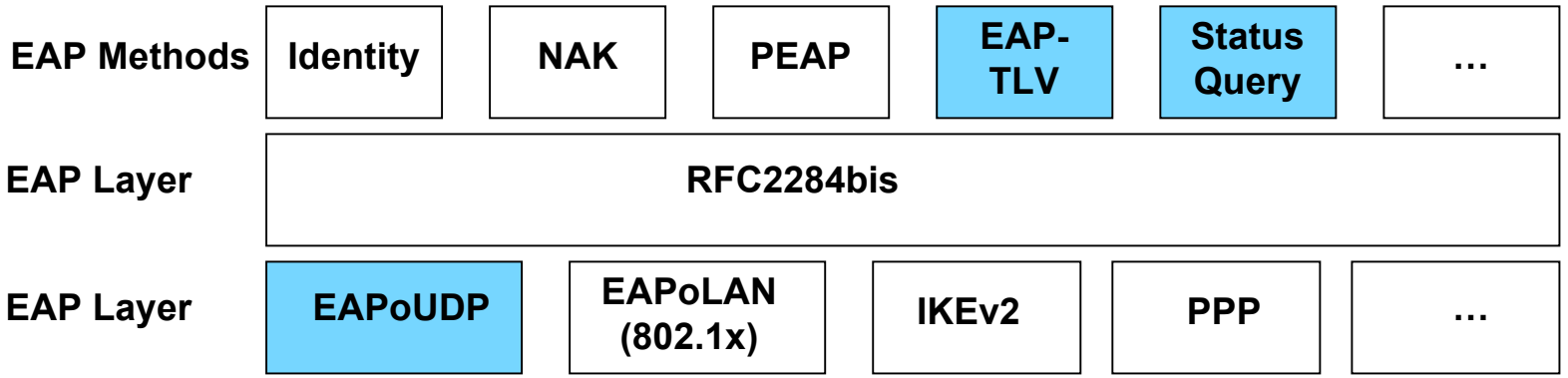
Transport Mechanisms…

- **NAC-Layer2-802.1x**
  - Uses 802.1x
  - Uses EAP-FAST as EAP method
  - Uses EAP-TLV to transport posture information
- **NAC-Layer2-IP**
  - Uses EAP over UDP (Port 21862 on client & NAD)
  - Uses PEAPv1 as EAP method without inner authentication
  - Uses EAP-TLV to transport posture information
- **NAC-Layer3-IP**
  - Uses EAP over UDP (Port 21862 on client & NAD)
  - Uses PEAPv1 as EAP method without inner authentication
  - Uses EAP-TLV to transport posture information

NAC-L3-IP Communication Flow



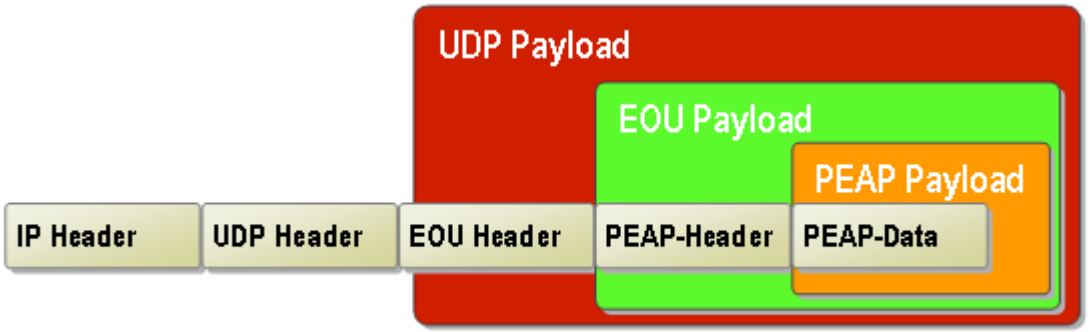
Extensible Authentication Protocol



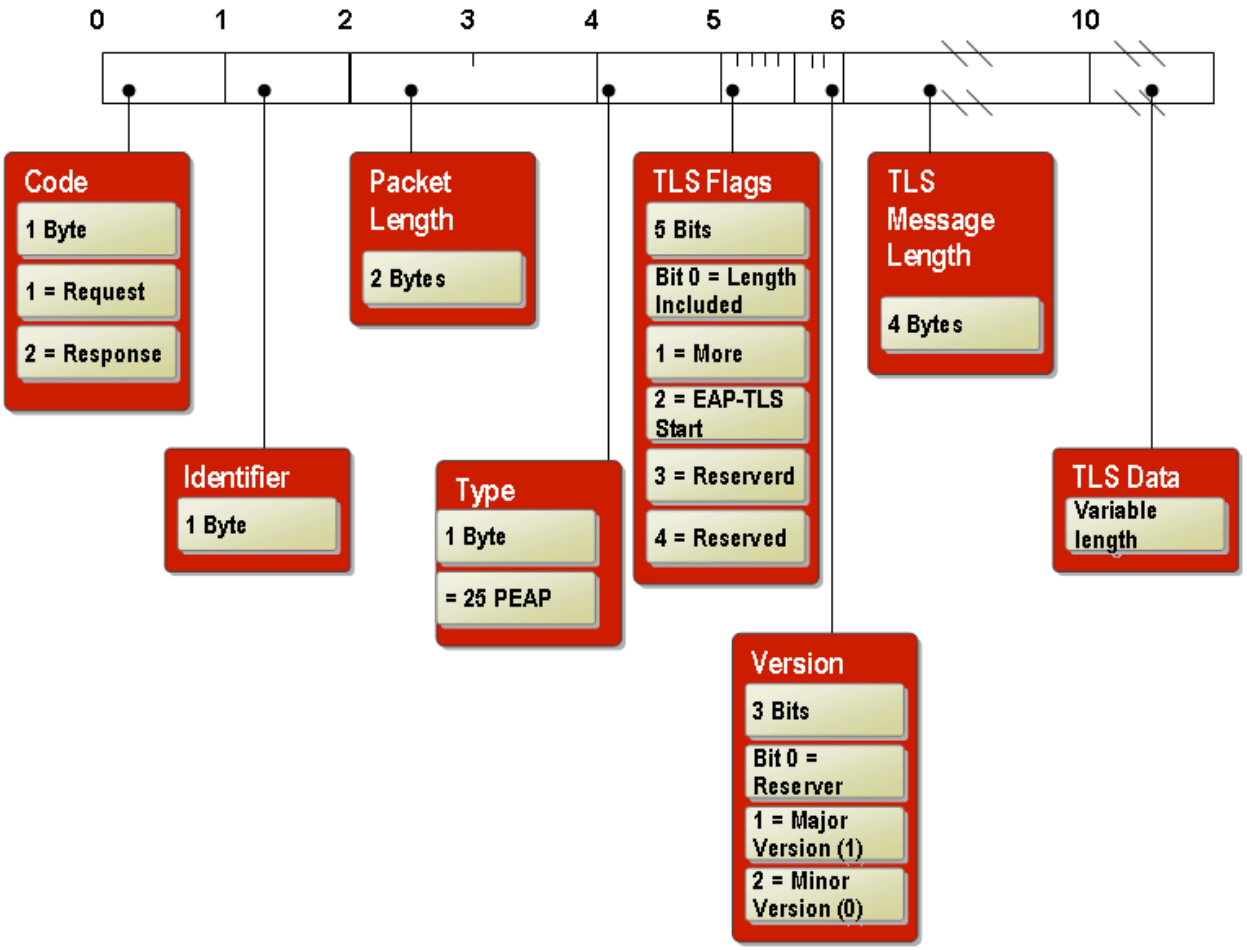
- EAP is a“request-response” Protocol:
  - **Exchange of “identity” and “authentication” information between a supplicant and an AAA server.**
- EAP supports a multitude auf authentication-schemes
  - **EAP-MD5**
  - **EAP-MSCHAP**
  - **…**
- EAP has to be “enhanced” for “policy based access restrictions” (aka NAC)
  - ***EAP-TLV*: Attribute-Type-Length-Value-Pair**
  - ***Status Query*: new method to get query the state of a client**
  - ***EAPoUDP*: EAP Transport over IP (instead of over Layer2 as e.g. 802.1x)**

New Function

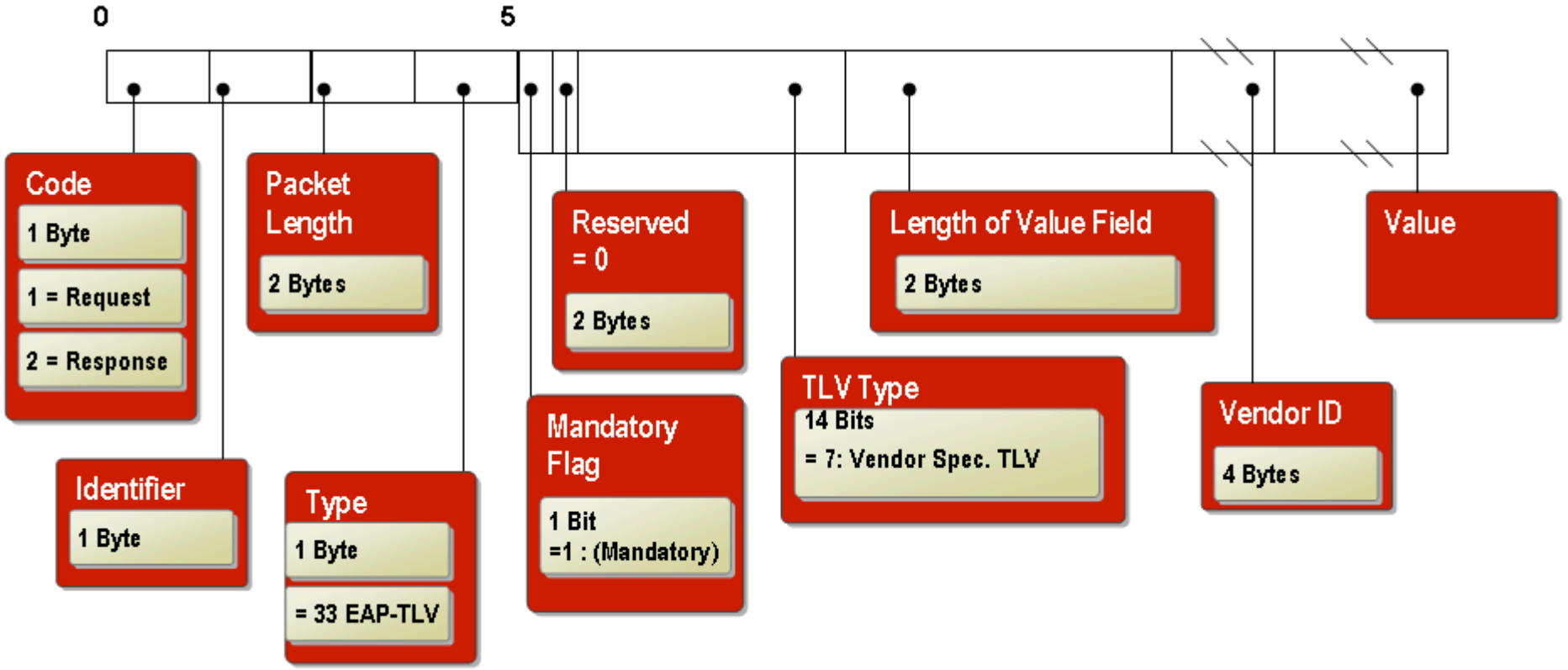
Encapsulation for L2-IP & L3-IP



PEAPv1 Frame Format



EAP-TLV Vendor Frame Format



Part 3 – Security Analysis

Flawed by Design 1:Client Authentication

	NAC-Layer 3 IP	NAC Layer 2 IP	NAC Layer 2 802.1x
Client Authentication	**No intrinsic Client Authentication**. In VPN scenarios there is a "VPN Authentication" which might be considered a "mitigating control".	**No intrinsic Client Authentication** – and no means of "adding" such on top.	Client Authentication based on 802.1x/EAP-FAST
Restriction of access on local subnet.	It is not possible to restrict access to the local subnet via NAC.	It is not possible to restrict access to the local subnet via NAC.	Access to local subnet can be denied through "port shutdown" via NAC.

Flawed by Design

- **So 1st design flaw is :**

Authorization without Authentication

- **This is clearly breaking a “secure by design” approach [for a security product] and is not conforming to “Best Current Practices”**

Flawed by Design 2: Epimenides Paradox

- **Epimenides was a Cretan (philosopher) who made one statement: "All Cretans are liars."**

- **Same paradox applies to Cisco NAC as well:**
  - The goal is to judge the “compliance”-level of (un)known & untrusted clients.
  - This is achieved by asking the (un)known & untrusted client about itself.
  - How can the ACS be sure that the client is a Cretan philosopher (a liar)?

So what? Where is the attack?

Posture Spoofing Attack

- **We define “posture spoofing” as an attack where a legitimate or illegitimate client spoofs “NAC posture credentials” in order to get unrestricted network access.**

Attackers Definition - Insider

- **Insider: An insider is a legitimate user of a NAC-protected network. The client has a working installation of the CTA and valid user/machine-credentials for the network. Additionally the inside attacker has the certificate of the ACS installed in its certificate store and if 802.1x is being used, this attacker has valid EAP-FAST-Credentials (PAC).**
- **The insider simply wants to bypass restrictions placed on his machine (e.g. no “leet tools” allowed and NAC checks list of installed programs).**

Attackers Definition - Outsider

- **Outsider: An outsider is not a legitimate user of the NAC-protected network and wants to get unrestricted access to the network. The outsider has no valid user/machine-credentials and no working CTA installation.**

Attack Vectors

- **Code an “alternative” NAC client**
  - Definitly possible
  - Will not work on 802.1x with EAP-FAST for outsider.
  - Currently “development in process” ☺
- **Replace plug-ins with self-written ones**
  - Definitely possible (be patient for ~50 more slides *just kidding*)
  - Works for the “insider” but not for the “outsider”.
  - Less work than the “alternative client
- **Abuse the scripting interface**
  - Not verified yet – limitations on “Vendor-ID” and “Application-ID” apply and not (yet) known if these are enforced or can be circumvented
  - If possible – the easiest way ☺

Feasible Attack Vectors

	Insider	**Outsider**
NAC-L2-802.1x	DLL/Plug-In replacement Scripting Interface CTA replacement	None as to our current knowledge.
NAC-L2-IP	DLL/Plug-In replacement Scripting Interface CTA replacement	CTA replacement
NACL-L3-IP	DLL/Plug-In replacement Scripting Interface CTA replacement	CTA replacement

Part 4 – Approaching NAC@AK

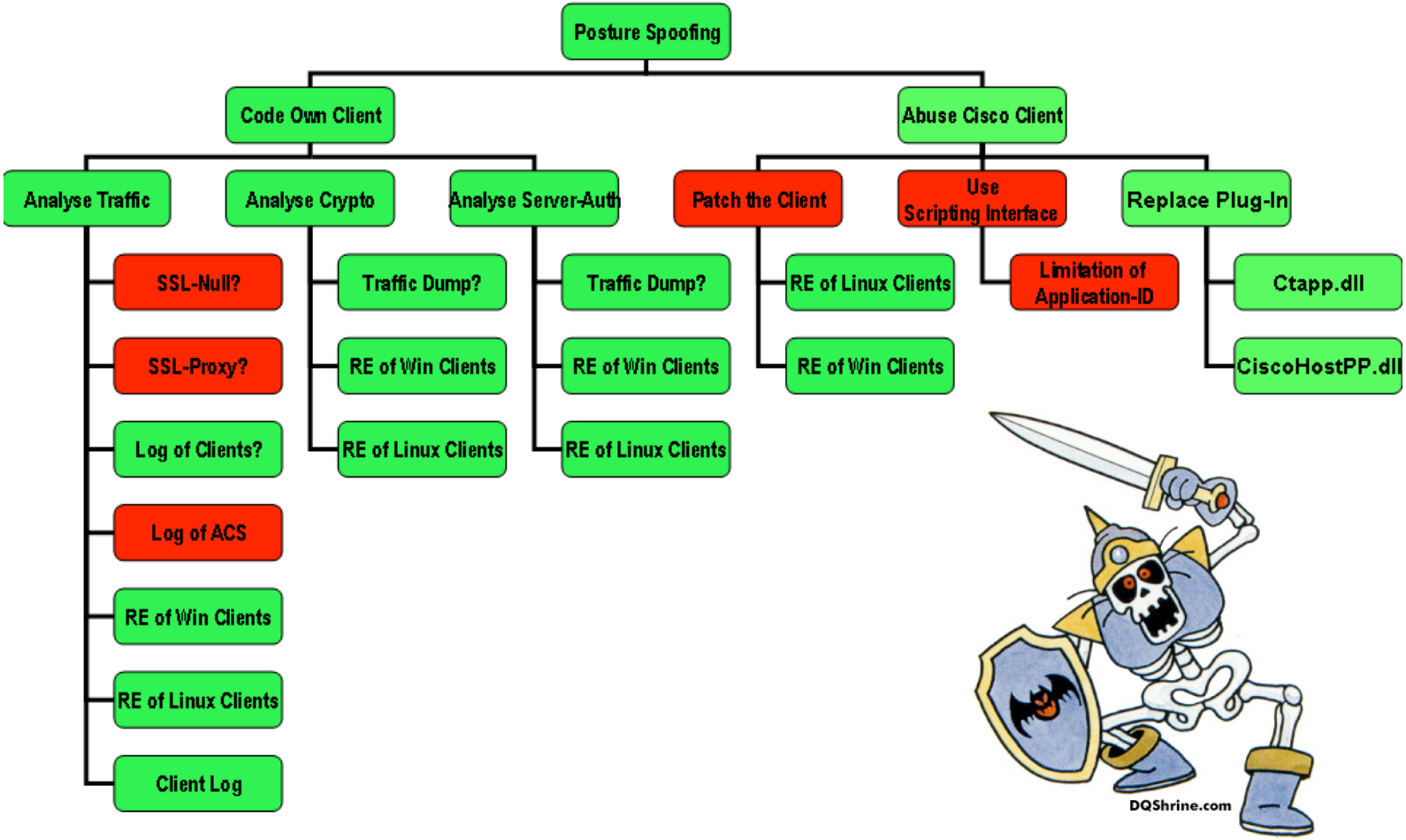
The ugly stuff – working with a structured approach *sigh

- **Step 1: Define what you need to know in order to get it working.**
- **Step 2: Sketch an attack-tree showing steps towards the goal.**
- **Step 3: Evaluate the components of the attack-tree for feasibility. Get the “tools” & know the “techniques” you need.**
- **Step 4: Pursue the feasible steps from step 3.**
- **Step 5: loop to step (1) until you get it working ,-)**

Want to know

- **Everything relating to…**
  - Communication flow
  - Packet format
  - Data-structures
  - Used Crypto
  - Used libraries
  - Existing interfaces
  - Program flow
  - Used Authentication
  - …

Attack Tree


Posture Spoofing
Code Own Client
Abuse Cisco Client
Analyse Traffic
Analyse Crypto
Analyse Server-Auth
Patch the Client
Use Scripting Interface
Replace Plug-In
SSL-Null?
SSL-Proxy?
Log of Clients?
Log of ACS
RE of Win Clients
RE of Linux Clients
Client Log
Traffic Dump?
RE of Win Clients
RE of Linux Clients
Traffic Dump?
RE of Win Clients
RE of Linux Clients
RE of Linux Clients
RE of Win Clients
Limitation of Application-ID
Ctapp.dll
CiscoHostPP.dll
DQShrine.com

Tools & Techniques

- **Reverse Engineering**
  - Reverse Engineering aims at uncovering the constructional elements of a product. IDAPro ☺ … and Hex-Rays
- **Packet Sniffing**
  - You all know that - Wireshark/Ethereal
- **Packet Diffing**
  - Extracting common and differing parts of two packets.
- **Debugging / API-Monitoring / Function-Hooking**
  - Through attaching a debugger or api-monitor to the running process, it is possible to actually see the contents of the stack while the program is running.
- **Built-in capabilities**
  - Logging / Debugging capabilites of the product – Cisco is usually _very_ good at that!
- **RTFM**
  - Read Read Read – often the vendor will tell you a lot about the product.

Big “want to have”: Cleartext Packets…

- **Communication is encrypted using TLS… packet capture shows encrypted packets.**
- **Not possible to get cleartext dump with tools (SSLProxy, etc.) – TLS over UDP not supported by tools.**

- **RTFM: Client Log can be enabled and it can dump cleartext payload of packets *g**

Cleartext Packet Dump in Log

Excerpt from a CTA logfile:

65 16:23:13.343 04/26/2006 Sev=PktDump/13 CTAVSTLV/0x64300016
Request message dump:
000700D100000009800200C900000009000100100001000800000000000000000000000010000200
0800000000000000090001 00A9000700A14865727A6C696368656E20476C7565636B77756E73
6368202D20496872205043206B6F6E6E74652065726F6C6772656963682061757468656E746
966697A6965727420776572646E20756E6420656E7473707269636874206465722053656375
26974792050696F6C6963792E2049687265204E65747A7765726B7A7567616E67207769726420
6E696368742065696E676573636872E46E6B7421800300020001

66 16:23:13.359 04/26/2006 Sev=Info/4 PAPlugin/0x63200001
Application Posture Result = Healthy

67 16:23:13.359 04/26/2006 Sev=PktDump/13
Response message dump: 800300020001

68 16:23:13.359 04/26/2006 Sev=Debug/7 CT
EapHandlePacket exit

[...snipped...]

70 16:23:13.359 04/26/2006 Sev=Info/4 PAPlugin/0x63200002
System Posture Result = Healthy

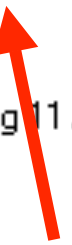
71 16:23:13.359 04/26/2006 Sev=Warning/2 PAPlugin/0xA3200012
CTAPP received UserMsg Notification: Content = Herzlichen Glueckwunsch - Ihr PC konnte erfolgreich authentifiziert werden und entspricht der Security Piolicy. Ihre Netzwerkzugang wird nicht eingeschränkt!

User Notification:
"Herzlichen …"

Convert to Hex:
%48%65%72%7a%6c%69
%63%68%65%6e%20

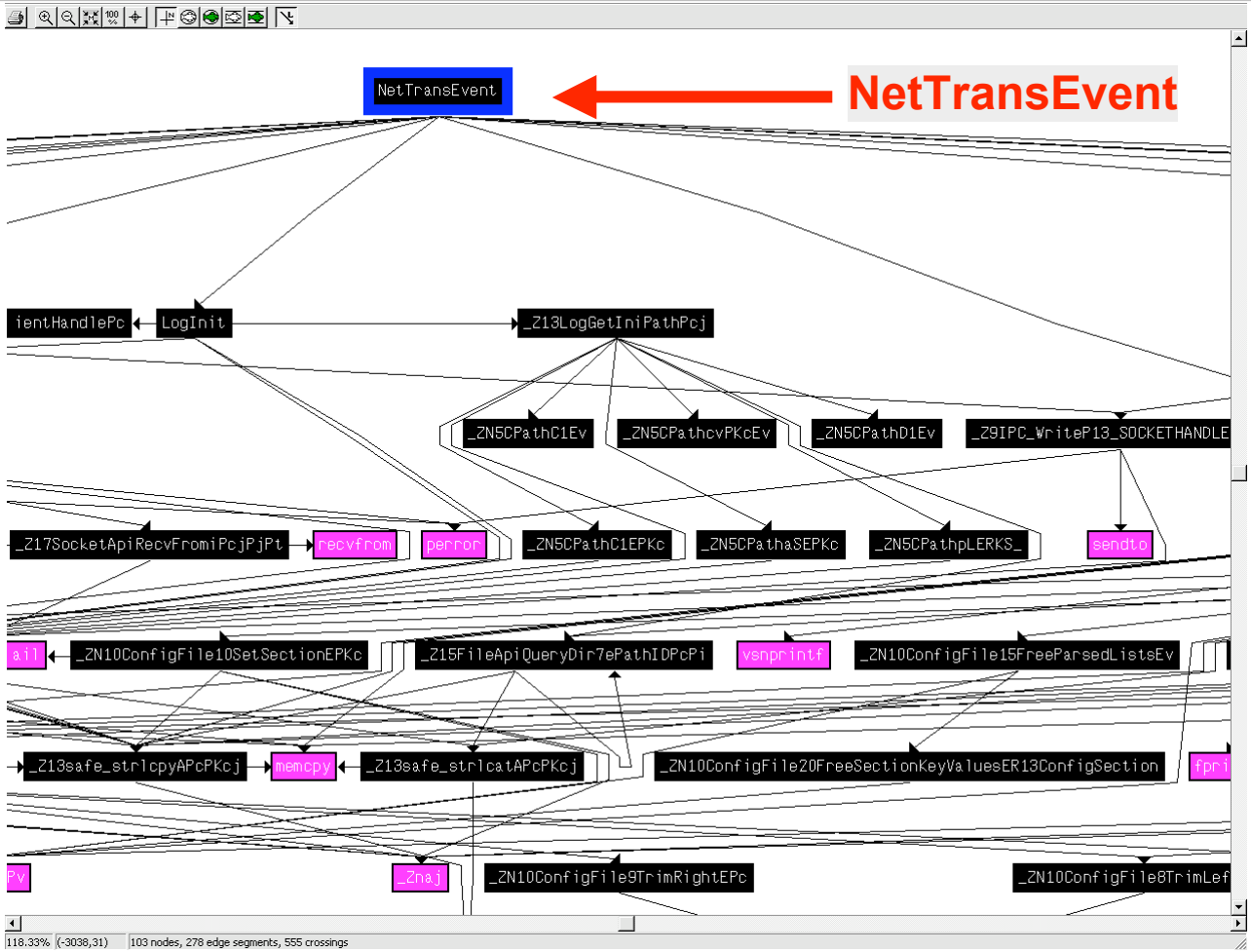
RE of the CTA – 1: Used Crypto

Address	Length	Type	String
"..." .rdata:1...	0000000E	C	FIPS routines
"..." .rdata:1...	0000000E	C	OCSP routines
"..." .rdata:1...	00000010	C	engine routines
"..." .rdata:1...	0000000A	C	func(%lu)
"..." .rdata:1...	00000009	C	lib(%lu)
"..." .rdata:1...	0000001C	C	.\\crypto\\engine\\tb_digest.c
"..." .rdata:1...	0000001B	C	.\\crypto\\engine\\eng_init.c
"..." .rdata:1...	00000029	C	Stack part of OpenSSL 0.9.7g 11 Apr 2005
"..." .rdata:1...	00000017	C	.\\crypto\\stack\\stack.c
"..." .rdata:1...	00000019	C	.\\crypto\\buffer\\buffer.c
"..." .rdata:1...	00000027	C	RSA part of OpenSSL 0.9.7g 11 Apr 2005
"..." .rdata:1...	00000017	C	.\\crypto\\rsa\\rsa_lib.c

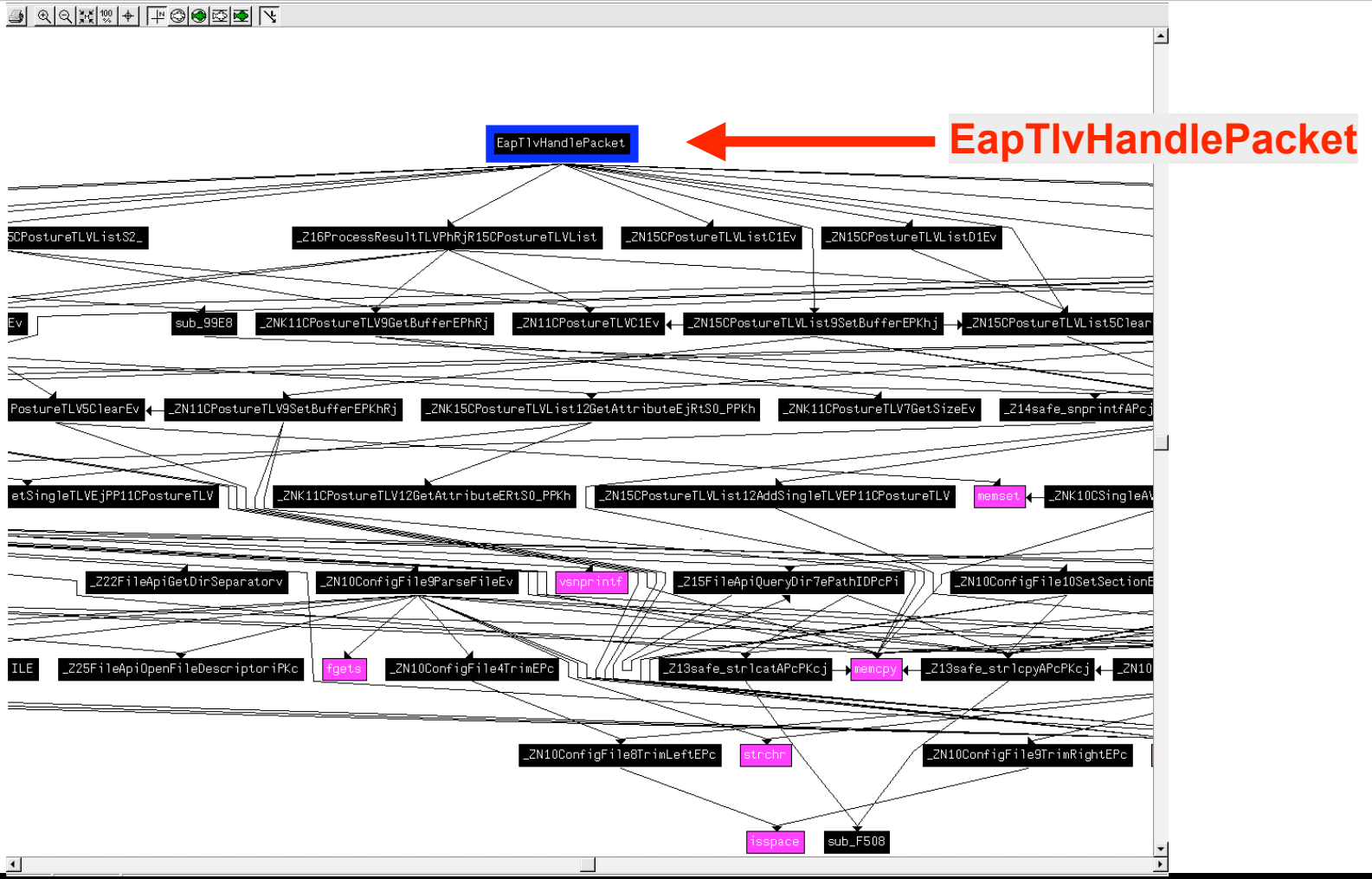


Used crypto (btw: this version is vulnerable)

RE of CTA – 1: Core Function



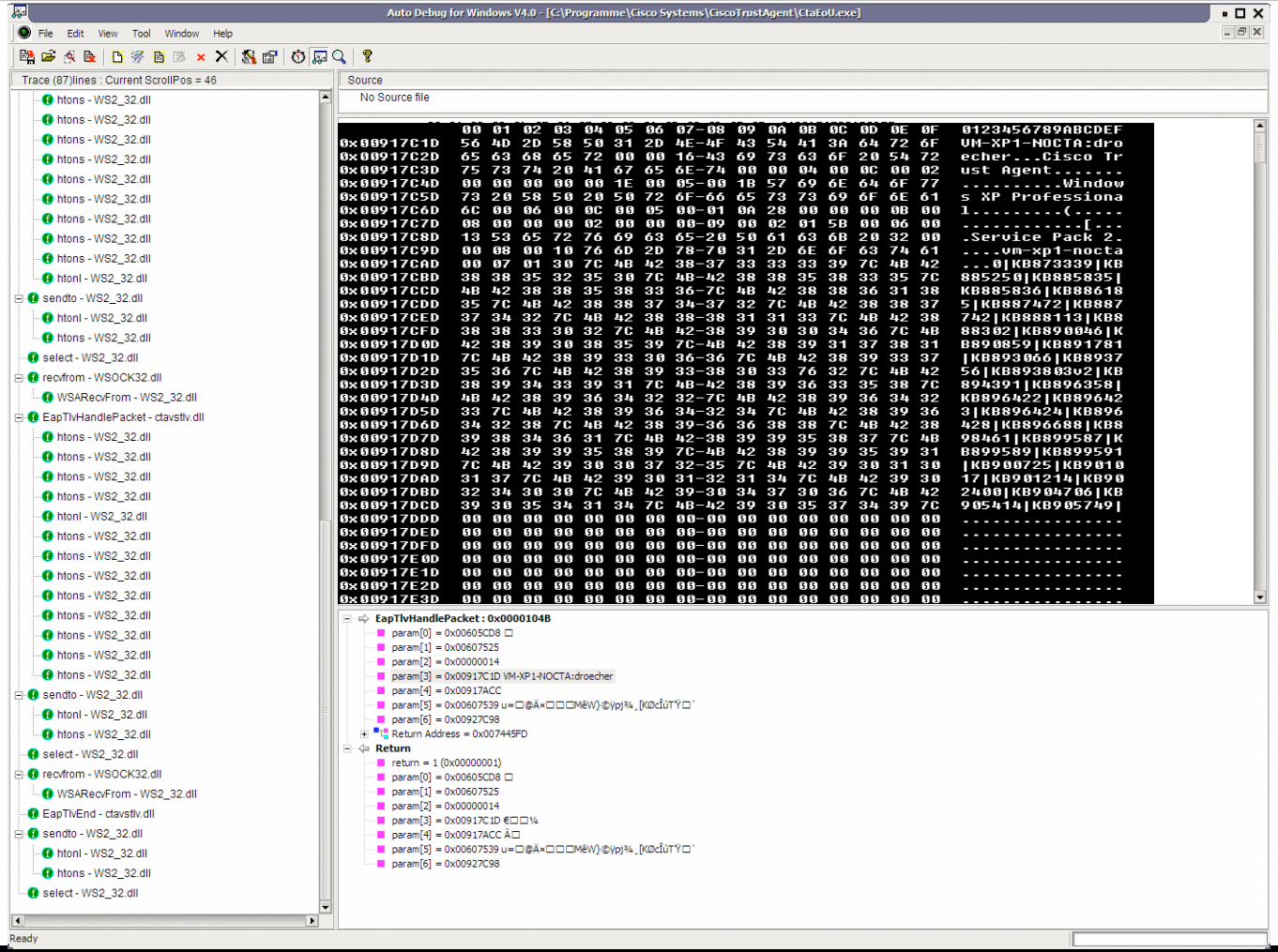
RE of CTA – 2: Core Function



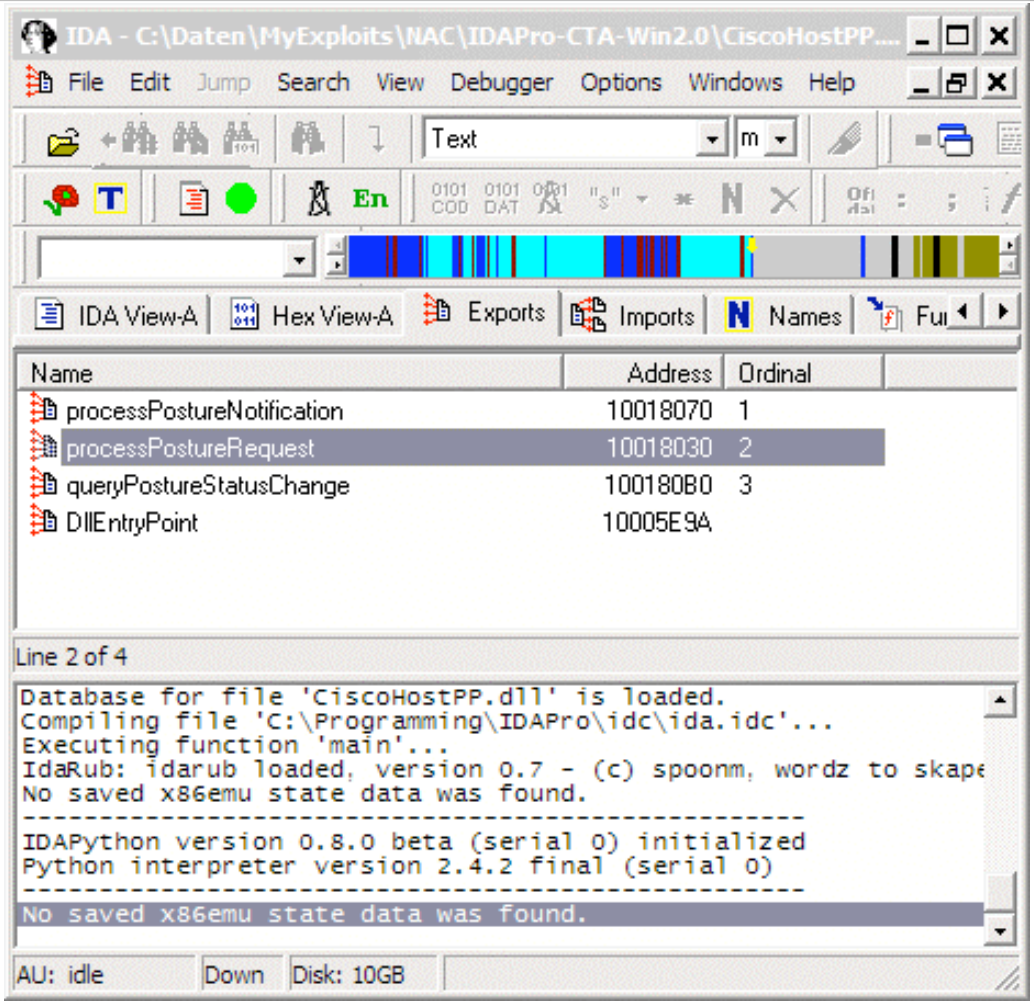
Function Hooking / API Monitoring with Autodebug

- **Step 1: Identify interesting functions with IDAPro**
- **Step 2: Figure out the function prototype (used parameters)**
- **Step 3: Code small C Program with that function prototype**
- **Step 4: Compile with debug symbols**
- **Step 5: Use PDB File (Program Debug Database) with Autodebug (www.autodebug.com)**
- **Step 6: Monitor the function with autodebug and see which parameters are passed to the function ;-)**

Function Hooking into EapTlvHandlePacket



RE of Plug-In 1: Exported Functions



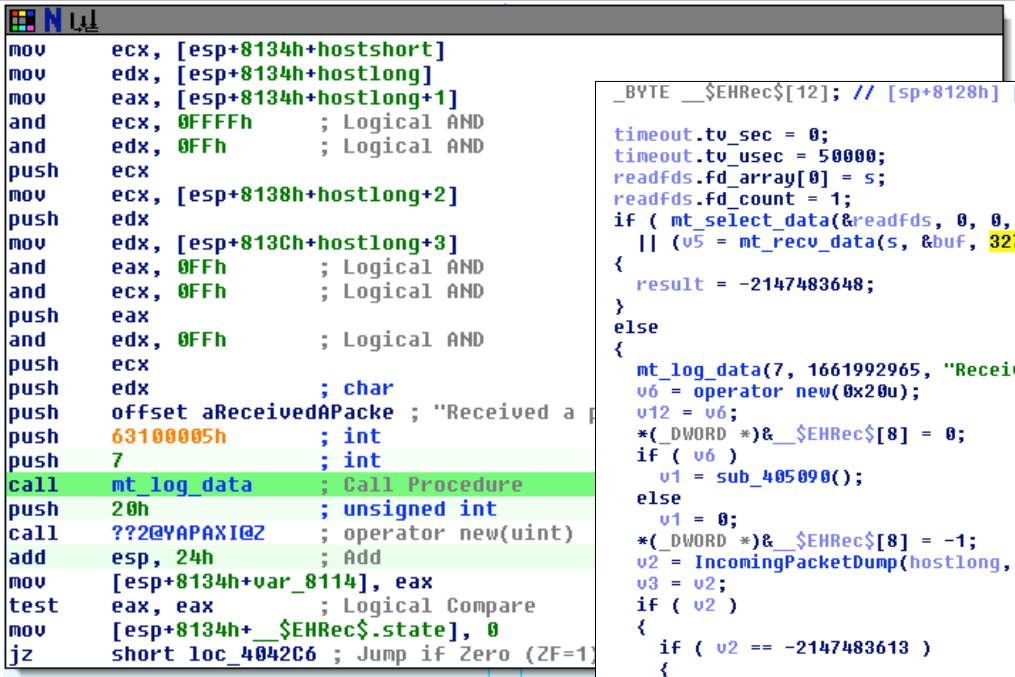
RE of Plug-In 2: Exported Functions

```
; Exported entry   2. processPostureRequest


; int __cdecl processPostureRequest(char *pRequest,int ID,char *pAttributeList,int *pNumber)
public processPostureRequest
processPostureRequest proc near

pRequest= dword ptr  4
ID= dword ptr  8
pAttributeList= dword ptr  0Ch
pNumber= dword ptr  10h

mov     eax, dword_1002788C
push    esi
mov     ecx, [eax+8]
mov     edx, [eax+4]
push    ecx
push    edx
call    sub_10018000
mov     edx, [esp+0Ch+pNumber]
add     esp, 8
mov     ecx, dword_1002788C
push    edx
mov     edx, [esp+8+pAttributeList]
mov     eax, [ecx]
push    edx
mov     edx, [esp+0Ch+ID]
push    edx
mov     edx, [esp+10h+pRequest]
push    edx

; const processPostureRequest::`vftable'
??_7processPostureRequest@@6B@:
call    dword ptr [eax+4]
mov     esi, eax
call    sub_10018020
mov     eax, esi
pop     esi
retn
processPostureRequest endp
```

```
; Exported entry   1. processPostureNotification


; int __cdecl processPostureNotification(char *NotifyBuffer,int Status)
public processPostureNotification
processPostureNotification proc near

NotifyBuffer= dword ptr  4
Status= dword ptr  8

mov     eax, dword_1002788C
push    esi
mov     ecx, [eax+8]
mov     edx, [eax+4]
push    ecx
push    edx
call    sub_10018000
mov     edx, [esp+0Ch+Status]
mov     ecx, dword_1002788C
add     esp, 8
mov     eax, [ecx]
push    edx
mov     edx, [esp+8+NotifyBuffer]
push    edx
call    dword ptr [eax+8]
mov     esi, eax
call    sub_10018020
mov     eax, esi
pop     esi
retn
processPostureNotification endp
```

```
; Exported entry   3. queryPostureStatusChange


; int __cdecl queryPostureStatusChange()
public queryPostureStatusChange
queryPostureStatusChange proc near
mov     eax, dword_1002788C
push    esi
mov     ecx, [eax+8]
mov     edx, [eax+4]
push    ecx
push    edx
call    sub_10018000
mov     ecx, dword_1002788C
add     esp, 8
mov     eax, [ecx]
call    dword ptr [eax+0Ch]
mov     esi, eax
call    sub_10018020
mov     eax, esi
pop     esi
retn
queryPostureStatusChange endp
```

Hex-Rays Decompiler



```
mov     ecx, [esp+8134h+hostshort]
mov     edx, [esp+8134h+hostlong]
mov     eax, [esp+8134h+hostlong+1]
and     ecx, 0FFFFh     ; Logical AND
and     edx, 0FFh       ; Logical AND
push    ecx
mov     ecx, [esp+8138h+hostlong+2]
push    edx
mov     edx, [esp+813Ch+hostlong+3]
and     eax, 0FFh       ; Logical AND
and     ecx, 0FFh       ; Logical AND
push    eax
and     edx, 0FFh       ; Logical AND
push    ecx
push    edx             ; char
push    offset aReceivedAPacke ; "Received a p
push    63100005h       ; int
push    7               ; int
call    mt_log_data     ; Call Procedure
push    20h             ; unsigned int
call    ??2@YAPAXI@Z    ; operator new(uint)
add     esp, 24h        ; Add
mov     [esp+8134h+var_8114], eax
test    eax, eax        ; Logical Compare
mov     [esp+8134h+__$EHRec$.state], 0
jz      short loc_4042C6 ; Jump if Zero (ZF=1)
```

```
_BYTE __$EHRec$[12]; // [sp+8128h] [bp-Ch]@3

timeout.tv_sec = 0;
timeout.tv_usec = 50000;
readfds.fd_array[0] = s;
readfds.fd_count = 1;
if ( mt_select_data(&readfds, 0, 0, &timeout) <= 0
  || (v5 = mt_recv_data(s, &buf, 32768, (int)hostlong, (int)&hostshort), v0 = v5, v5 <= 0) )
{
  result = -2147483648;
}
else
{
  mt_log_data(7, 1661992965, "Received a packet from address %u.%u.%u.%u, port 0x%x", SBYTE3(hostlong[0]));
  v6 = operator new(0x20u);
  v12 = v6;
  *(_DWORD *)&__$EHRec$[8] = 0;
  if ( v6 )
    v1 = sub_405090();
  else
    v1 = 0;
  *(_DWORD *)&__$EHRec$[8] = -1;
  v2 = IncomingPacketDump(hostlong, hostshort, &buf, v0);
  v3 = v2;
  if ( v2 )
  {
    if ( v2 == -2147483613 )
    {
      if ( (*(_BYTE *)(v1 + 1) & 0xF) == 2 )
      {
        sub_405E90((u_long)hostlong, hostshort, v1);
        mt_log_data(3, -1559232487, "Send NAK message to %u.%u.%u.%u (port 0x%x)  ", SBYTE3(hostlong[0]));
      }
    }
  }
  else
  {
    if ( (*(_BYTE *)(v1 + 1) & 0xF) == 2 )
      v3 = sub_405F60((u_long)hostlong, hostshort, v1);
    else
      v3 = sub_404B30(hostlong, hostshort, v1);
  }
```

Hex-Rays Decompiler

- First Decompiler that produces more than crap
- Build by Ilfak Guilfanov (think IDAPro ☺)
- Actually in Beta State (but already impressing)
- Will be released as commercial Addon for IDA
- Planned: API to support Decompiler Plugins like Vulnerability Analyzer and others
- Planned: Type and Function Prototype Recovery
- Planned: Assembler Knowledge not needed anymore
- Further Information at www.hex-rays.com
- Thanks to Ilfak for the Beta Version ☺

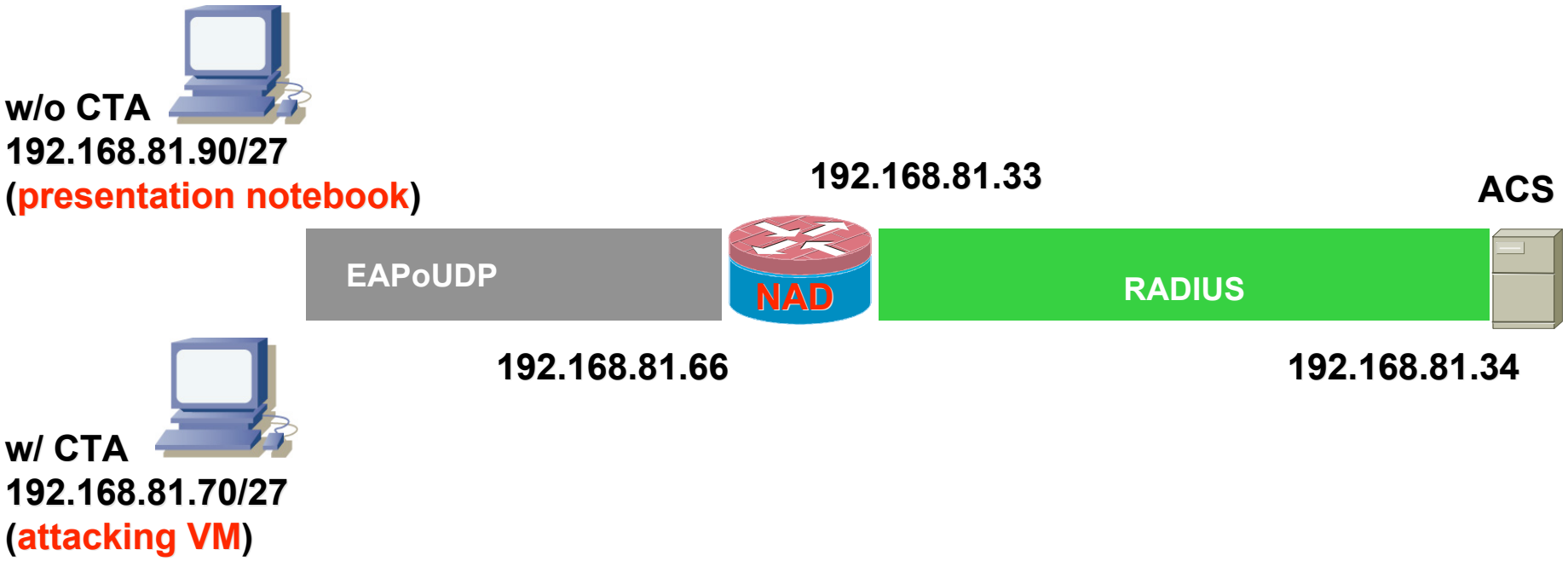
Quick Summary…

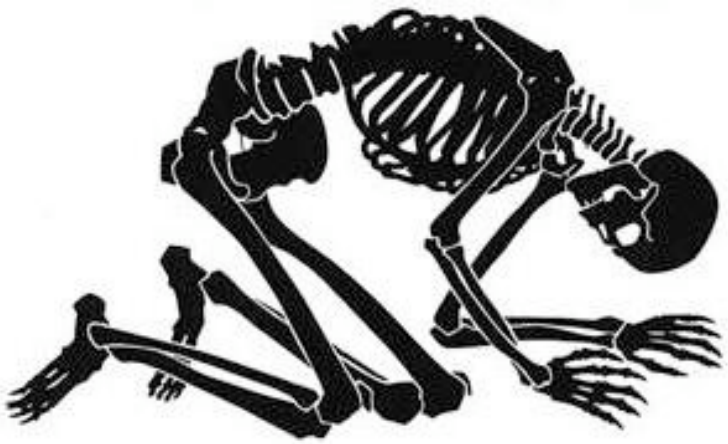
- **A lot of stuff learned so far…**
  - What is used
  - How it works
  - How it interoperates
  - Where to start hacking it

- **So now its…**

SHOWTIME

Showtime Setup





Thank’s for your patience

Time left for `questions & answers` ?

You can always drop us a note at:
droecher@ernw.de
mthumann@ernw.de