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
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…

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?

NAC over 802.1x工作原理

1. CTA将身份认证信息和主机安全信息发给交换机（借助802.1x）。
2. 交换机将认证信息发送给ACS。
3. ACS收到信息开始验证工作。与目录服务器交互，确认用户权限。
4. ACS检查入网计算机Service Pack，Hotfix，CSA版本等。
5. ACS与第三方反病毒策略服务器进行交互，确认计算机的健康状况。
6. 根据AD和反病毒策略服务器反馈的信息进行判断，认证。
7. 根据验证的结果向交换机下发策略，若为健康计算机划分到VLAN 100，不健康计算机划分到隔离VLAN。添加每用户ACL。
8. 将认证结果告知终端上的CTA软件。
9. CTA获知计算机的状态，健康或不健康，是否通过认证。
10. CSA从CTA处获知计算机状态，并决定是否限制应用，并记录到系统日志，发送给MARS。

NAC @ACK by Michael Thumann & Dror-John Roecher

September 5th 2007
A „big overview“ picture…

Endpoint Security Software

NAC enabled Security App (e.g. AV)
Cisco Trust Agent or Cisco Security Agent

CTA

EAPoUDP EAPoLAN

Network Access Device

RADIUS

AAA Server

HCAP

3rd-party Policy Server

Host Credential Authorization Protocol

Router or Switch or ASA

Cisco Secure ACS

AV-Server

Endpoint Security Software

NAC enabled Security App (e.g. AV)
Cisco Trust Agent or Cisco Security Agent

CTA

EAPoUDP EAPoLAN

Network Access Device

RADIUS

AAA Server

HCAP

3rd-party Policy Server

Host Credential Authorization Protocol

Router or Switch or ASA

Cisco Secure ACS

AV-Server
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

<table>
<thead>
<tr>
<th>Feature</th>
<th>NAC-L2-802.1x</th>
<th>NAC-L2-IP</th>
<th>NAC-L3-IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger</td>
<td>Data Link / Switchport</td>
<td>DHCP / ARP</td>
<td>Routed Packet</td>
</tr>
<tr>
<td>Machine ID</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>User ID</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Posture</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>VLAN Assignment</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>URL Redirection</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Downloadable ACLs</td>
<td>Cat65k only</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Application-Type ID</th>
<th>Application-Type Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PA</td>
<td>Posture Agent</td>
</tr>
<tr>
<td>2</td>
<td>Host / OS</td>
<td>Host information</td>
</tr>
<tr>
<td>3</td>
<td>AV</td>
<td>Anti Virus</td>
</tr>
<tr>
<td>4</td>
<td>FW</td>
<td>Firewall</td>
</tr>
<tr>
<td>5</td>
<td>HIPS</td>
<td>Host IPS</td>
</tr>
<tr>
<td>6</td>
<td>Audit</td>
<td>Audit</td>
</tr>
<tr>
<td>32768 – 65536</td>
<td></td>
<td>Reserved for “local use” (custom plug-ins or scripts)</td>
</tr>
</tbody>
</table>
## Credentials for Cisco:PA & Cisco:Hosts

<table>
<thead>
<tr>
<th>Application-Type</th>
<th>Attribute Number</th>
<th>Attribute Name</th>
<th>Value-Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Posture Agent</td>
<td>3</td>
<td>Agent-Name (PA-Name)</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Agent-Version</td>
<td>Version</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>OS-Type</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>OS-Version</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>User-Notification</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>OS-Kernel</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>OS-Kernel-Version</td>
<td>Version</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Service Packs</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Hot Fixes</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Host-FQDN</td>
<td>String</td>
</tr>
</tbody>
</table>
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=tnabpp.dll
PluginName=tnabpp.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, Date-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-MAC
```

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

Official Credentials

Private Credentials from the Vendor
Sample Policy on Cisco ACS

External User Databases

Rule Configuration

Rule Elements Table:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Operator</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CiscoSecureServicePack</td>
<td>Service</td>
<td>Pack</td>
</tr>
<tr>
<td>Monday/FridayProtection-Enabled</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Attribute</td>
<td>Operator</td>
<td>Value</td>
</tr>
<tr>
<td>Entry</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Submit | Delete Rule | Cancel

Adding Rule Elements

Editing Rule Elements

Deleting a Rule Element

Deleting a Rule

Use this page to create or modify a rule by creating and modifying the one or more rule elements that make up the rule. Each rule element consists of an attribute, an operator, and a value. Cisco Secure ACS uses the operator to compare the attribute received in the posture validation request to the value.

For each posture validation request that a rule is applied to, all rule elements must be true in order for a rule to be matched the posture validation request.

Adding Rule Elements

For each rule element you want to add:

1. From the Attribute list, select an attribute.
2. From the Operator list, select the applicable operator. The operators available vary depending upon the attribute you selected.
3. Type a value for comparison to the attribute selected.
And the resulting SPT on a NAD

```
 Mar  2 13:26:15.243: %E0U-6-AUTHTYPE: IP=192.168.67.241 AuthType=EAP
 nad#
 nad# show eou all

<table>
<thead>
<tr>
<th>Address</th>
<th>Interface</th>
<th>AuthType</th>
<th>Posture-Token</th>
<th>Age(min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.67.34</td>
<td>FastEthernet3/1</td>
<td>CLIENTLESS</td>
<td>unknown</td>
<td>0</td>
</tr>
<tr>
<td>192.168.67.24</td>
<td>FastEthernet3/1</td>
<td>EAP</td>
<td>healthy</td>
<td>0</td>
</tr>
</tbody>
</table>
```

NAD @ACK by Michael Thumann & Dror-John Roecher  September 5th 2007  22
General Communication Flow

1. Challenge
2. Credentials
3. Credentials
4. Comply?
   - OS patched: healthy
   - CSA enabled: healthy
   - AV out of date: quarantine

(3) NAD
(5) Access Rights: Quarantine
(7) Notification: “Quarantine”

AV-Client
CSA
NAC-enabled App
CTA

NAD

ACS

NAC @ACK by Michael Thumann & Dror-John Roecher

September 5th 2007
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

API / Process Posture Request / AV
- EAPoU Hello
- EAPoU Identity
- EAPoU / PEAP / Start
- EAPoU / AV + PA Posture
- EAPoU / APT+SPT+AV Notification + PA User Notification
- EAPoU / PEAP / Close
- EAPoU Result

API / Process Posture Notification / APT+SPT + AV Notification
- EAPoU Identity
- RADIUS Identity
- RADIUS / PEAP / Start
- RADIUS / AV + PA Posture
- RADIUS / APT+SPT+AV Notification + PA User Notification
- RADIUS / PEAP / Close
- RADIUS / EAP Result + Access Policy
### Extensible Authentication Protocol

<table>
<thead>
<tr>
<th>EAP Methods</th>
<th>Identity</th>
<th>NAK</th>
<th>PEAP</th>
<th>EAP-TLV</th>
<th>Status Query</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAP Layer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RFC2284bis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EAP Layer</td>
<td>EAPoUDP</td>
<td>EAPoLAN (802.1x)</td>
<td>IKEv2</td>
<td>PPP</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

- EAP is a “request-response” Protocol:
  - Exchange of “identity” and “authentication” information between a supplicant and an AAA server.
- EAP supports a multitude of 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 query the state of a client
  - **EAPoUDP**: EAP Transport over IP (instead of over Layer2 as e.g. 802.1x)
Encapsulation for L2-IP & L3-IP

![Diagram showing encapsulation process]
PEAPv1 Frame Format

- **Code**: 1 Byte, 1 = Request, 2 = Response
- **Packet Length**: 2 Bytes
- **TLS Flags**: 5 Bits, Bit 0 = Length Included, 1 = More, 2 = EAP-TLS Start
- **TLS Message Length**: 4 Bytes
- **Identifier**: 1 Byte
- **Type**: 1 Byte, = 25 PEAP
- **Version**: 3 Bits, Bit 0 = Reserved, 1 = Major Version (1), 2 = Minor Version (0)
- **TLS Data**: Variable length
EAP-TLV Vendor Frame Format

- Code: 1 Byte, 1 = Request, 2 = Response
- Packet Length: 2 Bytes
- Identifier: 1 Byte
- Type: 1 Byte, = 33 EAP-TLV
- Reserved: 2 Bytes, = 0
- Mandatory Flag: 1 Bit, = 1: (Mandatory)
- Length of Value Field: 2 Bytes
- TLV Type: 14 Bits, = 7: Vendor Spec. TLV
- Vendor ID: 4 Bytes
- Value
Part 3 – Security Analysis
# Flawed by Design 1: Client Authentication

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

<table>
<thead>
<tr>
<th></th>
<th><strong>Insider</strong></th>
<th><strong>Outsider</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NAC-L2-802.1x</strong></td>
<td>DLL/Plug-In replacement</td>
<td>None as to our current knowledge.</td>
</tr>
<tr>
<td></td>
<td>Scripting Interface</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CTA replacement</td>
<td></td>
</tr>
<tr>
<td><strong>NAC-L2-IP</strong></td>
<td>DLL/Plug-In replacement</td>
<td>CTA replacement</td>
</tr>
<tr>
<td></td>
<td>Scripting Interface</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CTA replacement</td>
<td></td>
</tr>
<tr>
<td><strong>NAACL-L3-IP</strong></td>
<td>DLL/Plug-In replacement</td>
<td>CTA replacement</td>
</tr>
<tr>
<td></td>
<td>Scripting Interface</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CTA replacement</td>
<td></td>
</tr>
</tbody>
</table>
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
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 Differing**
  - 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 capabilities 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:
000700D100000009800200C900000009000100100001000080000000000000000000010000200
080000000000000000000000A9000700A114865727A6C696368656E20476C7565636B77756E73
6368202D20496872205043206B6F6E674A2065726666F6C677265636862061757468656E746
966697A69657274207657264656E20756E6420656E74737072696368742064657220536563757
26974792050696F6C6963792E20496872265204E65677A7765726B7A7566616E67207769726240
6E69666742065696E6765736862E46E6B7421806300020001

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

Response message dump: 800300020001

68  16:23:13.359  04/26/2006  Sev=Debug/7  CT  Convert to Hex:
EapHandlePacket exit
\%48\%65\%72\%7a\%6c\%69 
\%63\%68\%65\%6e\%20

69  16:23:13.359  04/26/2006  Sev=Info/4  PAPplugin/0x63200002
System Posture Result = Healthy

70  16:23:13.359  04/26/2006  Sev=Warning/2  PAPplugin/0xA3200012
CTAPP received UserMsg Notification: Content = Herrlichen Glueckwunsch - Ihr PC konnte erfolgreich authentifiziert werden und entspricht der Security Policy. Ihre Netzwerkzugang wird nicht eingeschrankt!
RE of the CTA – 1: Used Crypto

<table>
<thead>
<tr>
<th>Address</th>
<th>Length</th>
<th>Type</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;....&quot;.rdata:1...</td>
<td>0000000E</td>
<td>C</td>
<td>FIPS routines</td>
</tr>
<tr>
<td>&quot;....&quot;.rdata:1...</td>
<td>0000000E</td>
<td>C</td>
<td>OCSP routines</td>
</tr>
<tr>
<td>&quot;....&quot;.rdata:1...</td>
<td>00000010</td>
<td>C</td>
<td>engine routines</td>
</tr>
</tbody>
</table>

| "....".rdata:1... | 0000000A | C    | func(%lu) |
| "....".rdata:1... | 00000009 | C    | lib(%lu) |
| "....".rdata:1... | 0000001C | C    | .\crypto\engine\tb_digest.c |
| "....".rdata:1... | 00000018 | 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

EapTlvHandlePacket
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 ;-)

www.autodebug.com
Function Hooking into EapTlvHandlePacket
**RE of Plug-In 1: Exported Functions**

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Ordinal</th>
</tr>
</thead>
<tbody>
<tr>
<td>processPostureNotification</td>
<td>10018070</td>
<td>1</td>
</tr>
<tr>
<td>processPostureRequest</td>
<td>10018030</td>
<td>2</td>
</tr>
<tr>
<td>queryPostureStatusChange</td>
<td>10018080</td>
<td>3</td>
</tr>
<tr>
<td>DIIEntryPoint</td>
<td>10005E9A</td>
<td></td>
</tr>
</tbody>
</table>

IDAPython version 0.8.0 beta (serial 0) initialized
Python interpreter version 2.4.2 final (serial 0)

No Saved x86emu state data was Found.
RE of Plug-In 2: Exported Functions

; 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_100278BC
push esi
mov ecx, [eax+8]
mov edx, [eax+4]
push ecx
push edx
call sub_10018000
mov edx, [esp+1Ch+PNumber]
add esp, 8
mov ecx, dword_100278BC
push edx
mov edx, [esp+8+PAttributeList]
mov eax, [ecx]
push edx
mov edx, [esp+8Ch+10]
push edx
mov edx, [esp+8+PRequest]
push edx
; const processPostureRequest::'uTable'
?? _processPostureRequest0M13:
call dword ptr [eax+4]
mov esi, eax
call sub_10018020
mov eax, esi
pop esi
ret
processPostureNotification endp

; 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_100278BC
push esi
mov ecx, [eax+8]
mov edx, [eax+4]
push ecx
push edx
call sub_10018000
mov edx, [esp+1Ch+PNumber]
add esp, 8
mov ecx, dword_100278BC
push edx
mov edx, [esp+8+PAttributeList]
mov eax, [ecx]
push edx
mov edx, [esp+8Ch+10]
push edx
mov edx, [esp+8+PRequest]
push edx
; const processPostureRequest::'uTable'
?? _processPostureRequest0M13:
call dword ptr [eax+4]
mov esi, eax
call sub_10018020
mov eax, esi
pop esi
ret
processPostureRequest endp

; Exported entry 3. queryPostureStatusChange

; int cdecl queryPostureStatusChange()
public queryPostureStatusChange
queryPostureStatusChange proc near
mov eax, dword_100278BC
push esi
mov ecx, [eax+8]
mov edx, [eax+4]
push ecx
push edx
call sub_10018000
mov edx, [esp+1Ch+Status]
mov ecx, dword_100278BC
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
ret
queryPostureStatusChange endp
Hex-Rays Decompiler

```assembly
mov ecx, [esp+8134h+hostshort]
mov edx, [esp+8134h+hostlong]
add ecx, [esp+8134h+hostlong*1]
add edx, [esp+8134h+hostlong*2]
push ecx
push edx
mov edx, [esp+813ch+hostlong*9]
add eax, [eax]; Logical AND
add edx, [eax]; Logical AND
push eax
and edx, [eax]; Logical AND
push edx
push eax
push edx
push eax
push edx
; char
push offset aReceiveAPacket; "Received a packet"
push 0x41000000h; int
push 7; int
call nt_log_data; Send Procedure
push 2ah; unsigned int
add esp, 2ah; Add
new [esp+813ah+var_8114h], eax
test eax, eax; Logical Compare
new [esp+8138h+ $EHRec$state], 0
jz short loc_404266; Jump if Zero (ZF=1)

BYTE __$EHRec$[12]; // [sp+0128h] [bp-Ch]63

timeout.tv_usec = 0;
timeout.tv_sec = 500000;
readfds_fd_array[0] = s;
readfds_fd_count = 1;
if (nt_select_data(&readfds, 0, 0, &timeout) == 0)

|| (v5 = nt_recv_data(s, &buf, 0x2768, (int)hostlong, (int)hostshort), v0 = v5, v5 <= 0)

{ result = -2147483648;
}
else

nt_log_data(7, 1661992962, "Received a packet from address %u.%u.%u.%u, port %d", SBVIE3(hostlong[8]));

v5 = operator new(Rx28h);

v12 = v6;

*(DWORD *)&$EHRec$[8] = 0;

if (v6)

v1 = sub_14559999h();
else

v1 = 0;

*(DWORD *)&$EHRec$[8] = -1;

v2 = IncomingPacketBmp(hostlong, hostshort, &buf, v0);

v3 = v0;

if (v2)

if (v2 == -2147483613)

if ((*(BYTE *)&v1 + 1) & 0xf) == 2)

\sub_405f90((u_long)hostlong, hostshort, v1);
nt_log_data(2, -1509222067, "Send MH message to %u.%u.%u.%u (port 0x%3)", SBVIE3(hostlong[8]));

}
else

if ((*(BYTE *)&v1 + 1) & 0xf) == 2)

v3 = sub_405f90((u_long)hostlong, hostshort, v1);
else

v3 = sub_404830(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 Setup

w/o CTA
192.168.81.90/27
(presentation notebook)

w/ CTA
192.168.81.70/27
(attack VM)

192.168.81.66
EAPoUDP

192.168.81.33
NAD

192.168.81.34
RADIUS

ACS
Thank’s for your patience

Time left for `questions & answers`?

You can always drop us a note at:

droecher@ernw.de
mthumann@ernw.de