ERNW
Wir leben IT-Security.

Hacking VoIP for Fun and Profit

Roland Fiege, Geschäftsführer

ERNW GmbH

Agenda

ERNW
Wir leben IT-Security.

- Kurzvorstellung ERNW
- Grundlagen / Begriffsklärung VoIP
- Sicherheitsziele
- Kleine Historie der VoIP-Vulnerabilities….
- Vorstellung einer neuen Sicherheitslücke in einer SIP-Bibliothek
- Wie werden solche Lücken entdeckt und publiziert?
- Potentielle Auswirkungen der Schwachstelle
- Was lernen wir daraus?

ERNW

ERNW
Wir leben IT-Security.

- Gegründet Sommer 2001
- Sitz in Heidelberg, Deutschland
- unabhängiger IT-Security-Dienstleister
- Aktuell elf Mitarbeiter
- Schwerpunkte:
  - Security Management
  - Audit/Revision (BS7799/ISO27001)
  - Penetrations-Tests
  - Risiko-Bewertung & -Management
  - Security Research
- Kunden: Industrie, Banken, Behörden, Provider

Grundlagen und Begriffe VoIP

ERNW
Wir leben IT-Security.

- Der Transport von ‚paketierten' Telefonverbindungen über IP-Netzwerke. Meist (aber nicht nur) Audio-Daten, teilweise über bestimmte Gateways direkt aus dem *PSTN (Public Switched Telephone Network*).
- Kann innerhalb von abgeschlossenen Netzeinheiten stattfinden (=> dann spricht man oft von *VoIP*) oder über das Internet (=> Terminus ist hier üblicherweise *IP Telephony*)
- Ich verwende im Vortrag beide Begriffe weitgehend austauschbar.
- VoIP ist kein Protokoll oder Standard
  Eher ein Sammelbegriff für verschiedene Technologien.

Bestandteile typischer VoIP-Szenarios

ERNW
Wir leben IT-Security.

- Protokolle (Transport, Signalisierung, Management/Infrastruktur)
- Komponenten (dedizierte VoIP-Devices, Kopplungs-Geräte)
- Verbindungen (öffentliche, nicht-öffentliche Strecken)
- Endgeräte (Hardphones, Softphones, …)
- User (‚Faktor Mensch‘)

Protokolle

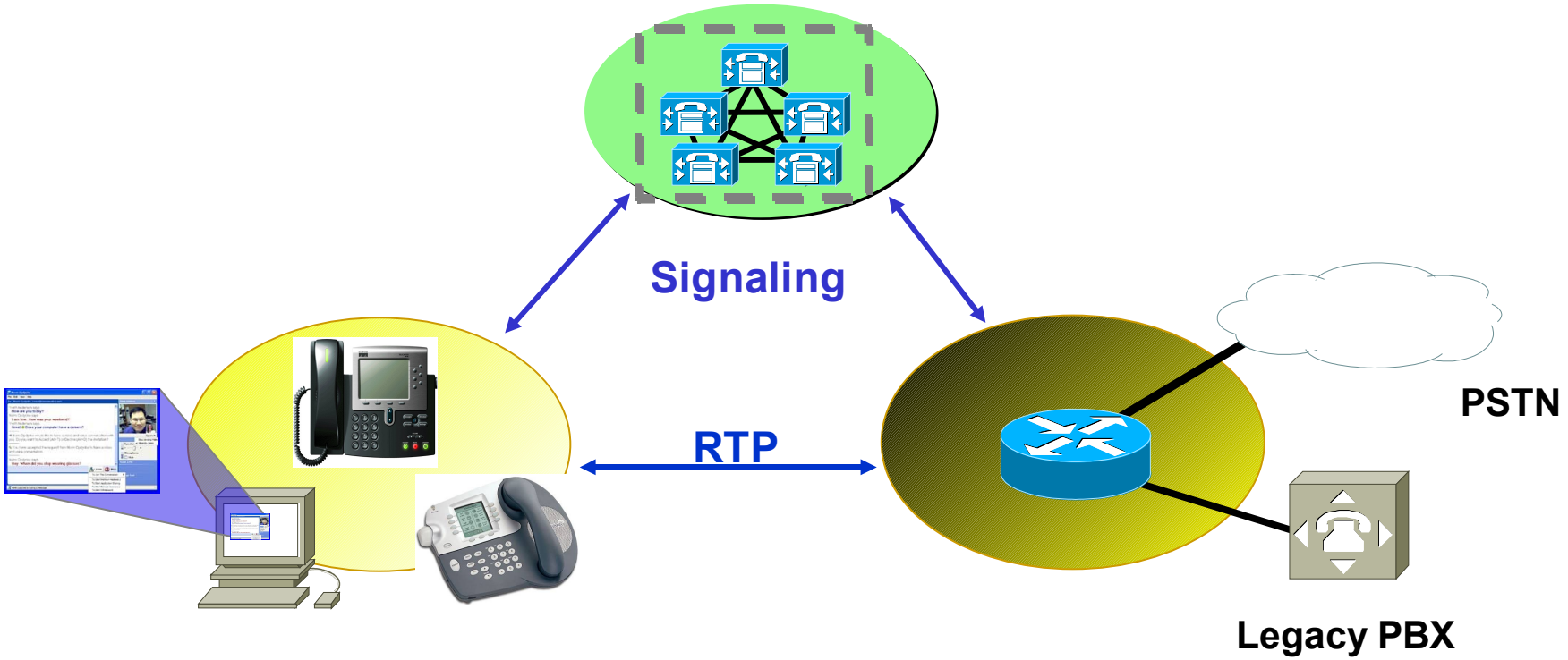
ERNW
Wir leben IT-Security.

- Transport (von Nutzdaten):
  Real-Time Transport Protocol [*RTP*, RFC 1889]
  + ggf. *RTP Control Protocol* [*RTCP*]

- Signalisierung (*Location of Users, Session Setup & Negotiation* etc.):
  - *H.323* [Urheber: ITU]
  - *Session Initiation Protocol* [*SIP*, RFC 2543/3261]
  - *Skinny/SCCP* [Cisco-propietär]
  - *Media Gateway Control Protocol* [*MGCP*, RFC 2705]
  - *Megaco* [RFC 3015, ITU-T H.248]

- Management/Infrastruktur [DNS, SNMP et.al.]

Zusammenwirken der Protokolle



Komponenten

ERNW
Wir leben IT-Security.

- Dedizierte VoIP-Devices, e.g.
  - H.323 Gatekeeper, Multipoint Controller
  - SIP Proxy, Redirector
  - Voice-Mail Server
  - TRIP Location Server
- Kopplungskomponenten zu anderen Netzen, z.B. GGSN/SGSN zu GPRS/UMTS-Netzen
- Infrastruktur-Komponenten (etwa Backbone Router)

Endgeräte

ERNW
Wir leben IT-Security.

- Hardphones: dedizierte Telefone mit VoIP-Fähigkeiten
- Softphones: Software-Applikationen, die auf PCs laufen und VoIP-Funktionalität zur Verfügung stellen.
- Legacy Devices: traditionelle Telefon-/Fax-Geräte, die über geeignete Schnittstellen (etwa Cisco NM-2V+VIC-2FXS) in VoIP-Netze integriert werden.

Sicherheitsziele

ERNW
Wir leben IT-Security.

- Verfügbarkeit
- Vertraulichkeit
- Integrität
- Authentizität
- Non-Repudiation
- Einhaltung gesetzlicher Bestimmungen
  - Datenschutz
  - *Lawful Interception*

Unterschiedliche Fokussierung der Sicherheitsziele

ERNW
Wir leben IT-Security.

- Enduser:
  Vertraulichkeit (das „emotionale Moment“)
- Ihre Organisation:
  Verfügbarkeit, Vertraulichkeit, Datenschutz
- Carrier/Dienst-Anbieter:
  Non-Repudiation (Abrechnungs-Betrug!), Lawful Interception,
  „Kundenbindung durch Vertrauen“

(Main) Threats

ERNW
Wir leben IT-Security.

- Abhören von Verbindungen/Sniffing
  - durch Angreifer
  - ‚Lawful' (⇔ Anwälte, Journalisten, Beratungsinstitutionen)

- Denial-of-Service

- Kompromittierung von Komponenten
  => Abhören
  => Umleiten

- Spoofing
  => Verlust der Authentizität (Telefon-Banking)
  => Abrechnungsbetrug

Vulnerabilities

ERNW
Wir leben IT-Security.

Auf Protokoll-Ebene:

- Implementierungs-Schwächen
- (Hochgradig) Dynamische Kommunikationsbeziehungen
- Fehlende immanente Sicherheitsfeatures

Auf Komponenten-Ebene:

- Unsichere Default-Konfigurationen
- *Design without security in mind*, insbesondere an Kopplungskomponenten
- Seiteneffekte von VoIP Security-Problemen auf ‚Nicht-VoIP Devices' (z.B. bei Softphones)

Auf Endgeräte-Ebene:

- Unsichere Default-Konfigurationen
- (Zu) wenig Sicherheits-Features
- Malicious Code ?

Kleine Historie der VoIP-Vulnerabilities

ERNW
Wir leben IT-Security.

- Vor 18 Monaten:
  - CISCO VoIP-Telefon als Abhöranlage missbraucht

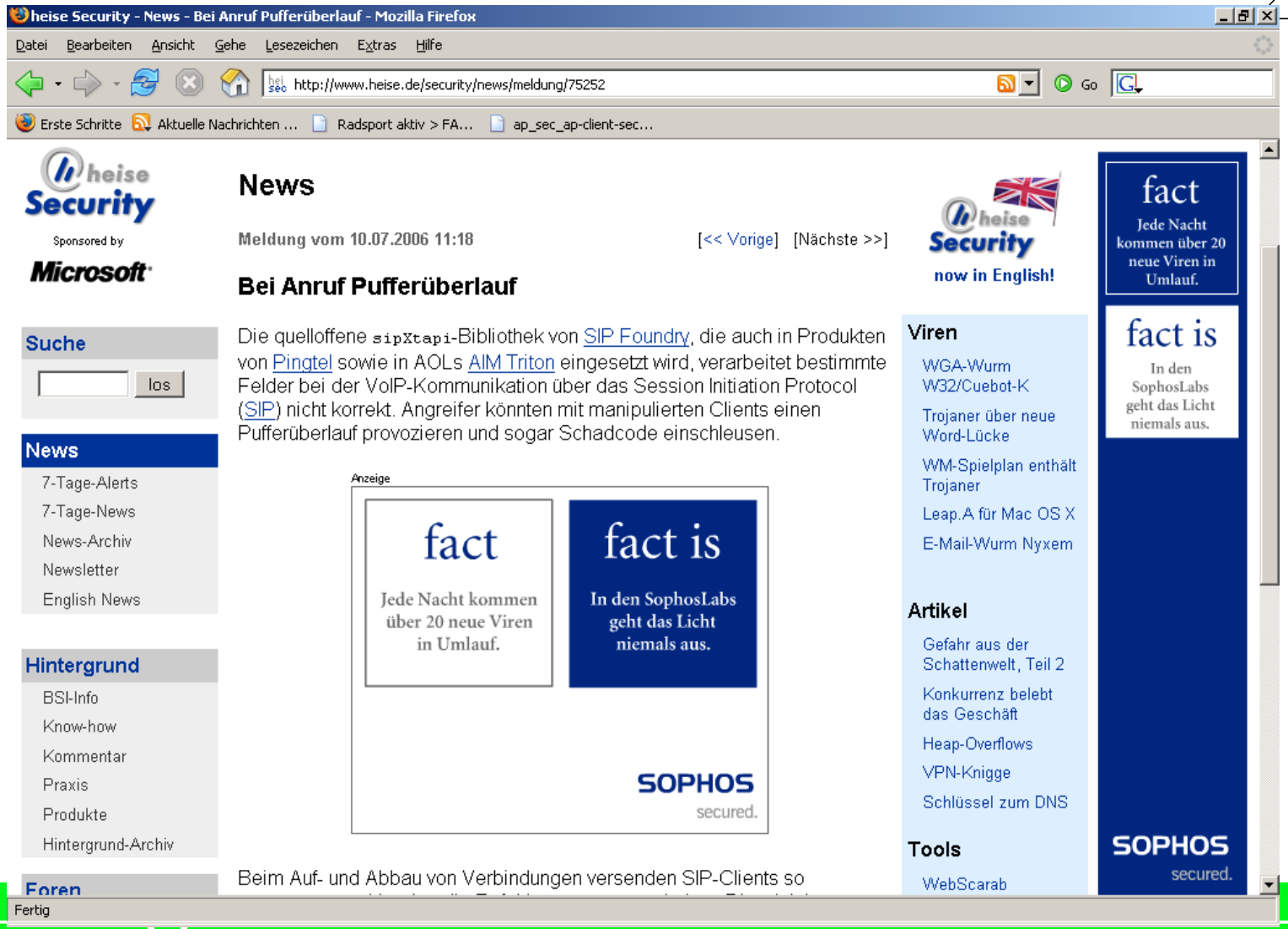
    Aufgrund fehlender Sicherheits-Features (etwa fehlender Authentifizierung), schlechter Default-Konfiguration (e.g. aktiviertem Telnet-Zugang) oder mangelhafter Management-Strukturen (Konfig per TFTP etc.), z.B. gegen ältere Cisco 7960-Modelle.
- Vor 12 Monaten:
  - Sniffing = Abhören von VoIP-Telefonaten

    Da RTP per default unverschlüsselt überträgt, kann ein Angreifer mit Zugriff auf den Netzwerk-Verkehr (üblicherweise im lokalen Netz per ARP-Spoofing) VoIP-Sitzungen abhören. Das bekannteste Tool ist hier *vomit.*
- aktuell
  - der AOL-Triton – Hack (Buffer Overflow)

    ermöglicht die Ausführung von beliebigem Code auf dem „Opfer“-Rechner
- Zukünftig ?

Vor kurzem auf heise.de

ERNW
Wir leben IT-Security.



Wie „entsteht“ eine solche Lücke bzw. ihre Publikation?

ERNW
Wir leben IT-Security.

- Auffälliges Verhalten einer Komponente wird beobachtet
  - zufällig (Absturz)
  - durch gezieltes Suchen
  - im Rahmen von Forschungsarbeit
- Eingrenzung des Problems
- Reproduktion / Proof of Concept
- Kontakt zu Hersteller/Autor/Maintainer
- Publikation

Research Techniken

ERNW
Wir leben IT-Security.

- Fuzzer
- Code Audit
- Reverse Engineering

Fuzzer

ERNW
Wir leben IT-Security.

- Fault Injection / Erzeugung von „ungewöhnlichem Input“
- mithilfe von Protocol Fuzzern (SPIKE, Protos)
- oder SIP Testframeworks (SIPp, SipSak, SIPForumTestFramework)
- Kenntnis der Protokolle, File Formate und API notwendig
- Eher für einfache Fehler geeignet

Fuzzer – Vorgehensweise

ERNW
Wir leben IT-Security.

- Injektion von Fehlern
- Überwachung des Programmverhaltens mit Debugger
- Interessant sind „Programmabstürze“
- Da der injizierte Fehler bekannt ist, kann er auch reproduziert werden (z. B. durch eigenen Code)
- Ggf. Code Audit oder Reverse Engineering

Code Audit

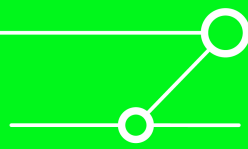
ERNW
Wir leben IT-Security.

- „Read the source Luke“
- Detaillierte Kenntnis der Programmiersprache notwendig
- Typische Probleme der Sprache, z. B. strcpy() und sprintf() in C
- Komplexe Probleme sind nicht trivial zu entdecken
- Beispiel *ERNW Advisory 02/2006 – SipXtapi Library*

Code Audit

ERNW
Wir leben IT-Security.

- Tool-Unterstützung möglich:
- RATS
- Splint
- Flawfinder
- Codescan (kommerziell)



```
#define MAXIMUM_INTEGER_STRING_LENGTH 20
...
...
...
UtlBoolean SipMessage::getCSeqField(int* sequenceNum, UtlString* sequenceMethod) const
{
   const char* value = getHeaderValue(0, SIP_CSEQ_FIELD);
   if(value)
   {
        // Too slow:
       /*UtlString sequenceNumString;
      NameValueTokenizer::getSubField(value, 0,
               SIP_SUBFIELD_SEPARATORS, &sequenceNumString);
      *sequenceNum = atoi(sequenceNumString.data());

      NameValueTokenizer::getSubField(value, 1,
               SIP_SUBFIELD_SEPARATORS, sequenceMethod);*/
        // Ignore white space in the begining
        int valueStart = strspn(value, SIP_SUBFIELD_SEPARATORS);

        // Find the end of the sequence number
        int numStringLen = strcspn(&value[valueStart], SIP_SUBFIELD_SEPARATORS)
            - valueStart;
        // Get the method
        if(sequenceMethod)
        {
            *sequenceMethod = &value[numStringLen + valueStart];
            NameValueTokenizer::frontBackTrim(sequenceMethod, SIP_SUBFIELD_SEPARATORS);

            if(numStringLen > MAXIMUM_INTEGER_STRING_LENGTH)
            {
                osPrintf("WARNING: SipMessage::getCSeqField CSeq number %d characters: %s.\nTruncating to %d\n",
                    numStringLen, &value[valueStart], MAXIMUM_INTEGER_STRING_LENGTH);
                numStringLen = MAXIMUM_INTEGER_STRING_LENGTH;
            }
        }
        if(sequenceNum)
        {
            // Convert the sequence number
            char numBuf[MAXIMUM_INTEGER_STRING_LENGTH + 1];
            memcpy(numBuf, &value[valueStart], numStringLen);
            numBuf[numStringLen] = '\0';
            *sequenceNum = atoi(numBuf);
        }
   }
    else
    {
        if(sequenceNum)
        {
            *sequenceNum = -1;
        }
```

Wo ist hier das Problem? ☺

Reverse Engineering

ERNW
Wir leben IT-Security.

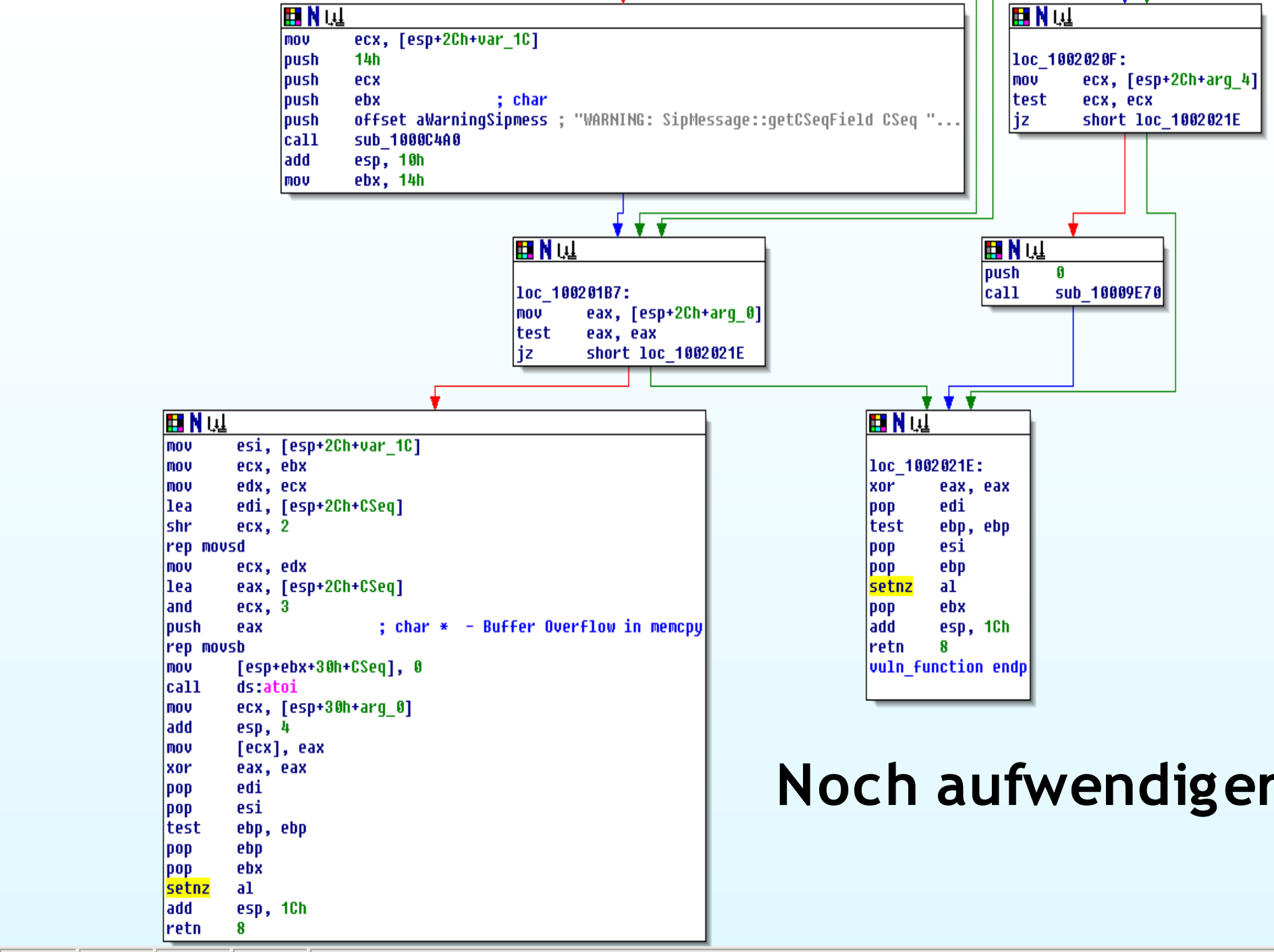
- Binary Audit
- Disassembling
- Prüfen des Assembler Codes
- API Tracing / API Spy
- Aufwendiger Prozess
- Hilfreich: Assembler Know How, OS Architektur und API Know How (Betriebssystemprogrammierung), Prozessor Architektur
- Kenntnisse der eingesetzten Programmiersprache

Reverse Engineering

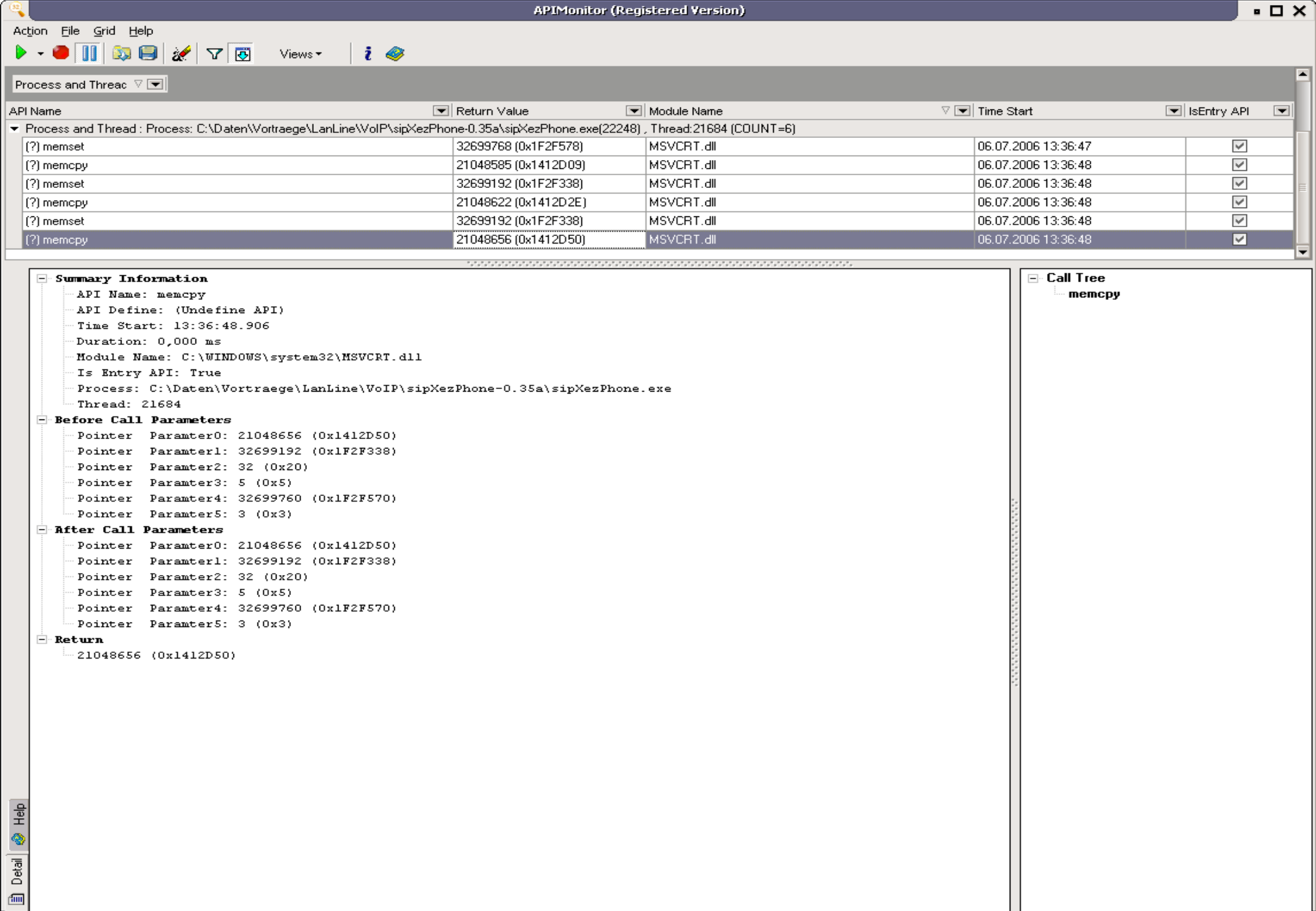
ERNW
Wir leben IT-Security.

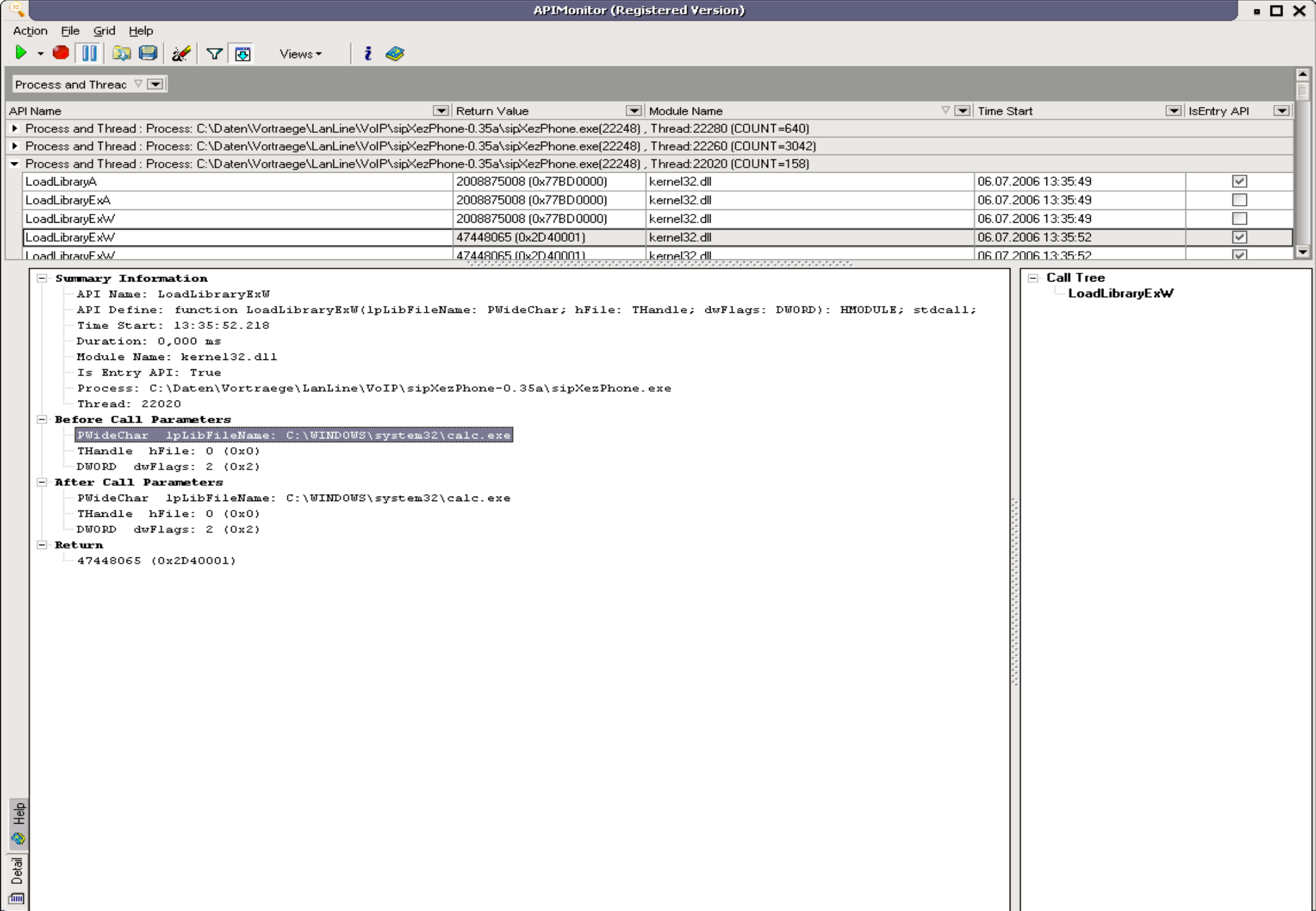
Auch hier Tool-gestütztes Arbeiten möglich

- API Monitor
- IDA Pro (das wohl wichtigste Tool)
- Bugscam
- BinAudit
- BinNavi
- Bindiff



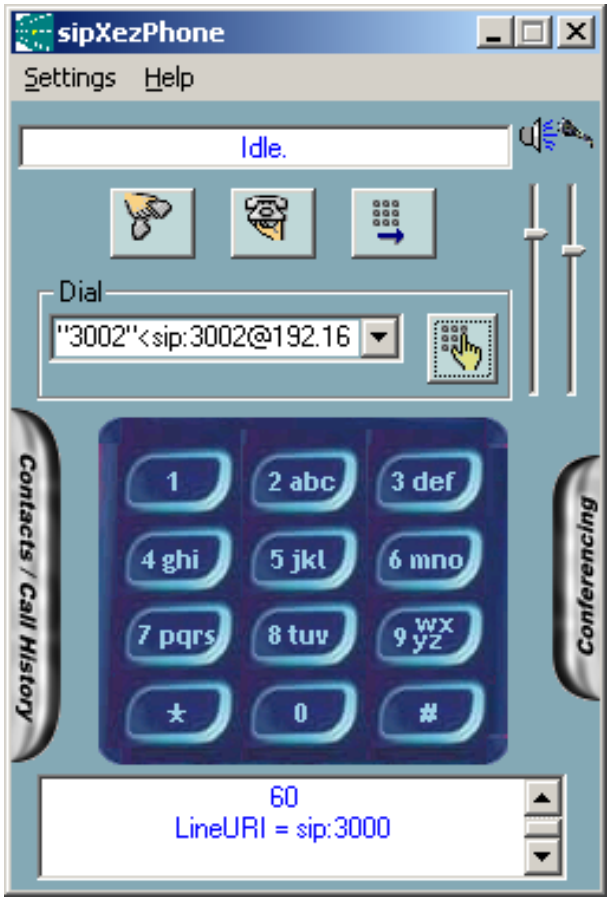
Noch aufwendiger ☺


APIMonitor (Registered Version)
Action File Grid Help
Views
Process and Threac
API Name
Return Value
Module Name
Time Start
IsEntry API
Process and Thread : Process: C:\Daten\Vortraege\LanLine\VoIP\sipXezPhone-0.35a\sipXezPhone.exe(22248) , Thread:21684 (COUNT=6)
(?) memset 32699768 (0x1F2F578) MSVCRT.dll 06.07.2006 13:36:47
(?) memcpy 21048585 (0x1412D09) MSVCRT.dll 06.07.2006 13:36:48
(?) memset 32699192 (0x1F2F338) MSVCRT.dll 06.07.2006 13:36:48
(?) memcpy 21048622 (0x1412D2E) MSVCRT.dll 06.07.2006 13:36:48
(?) memset 32699192 (0x1F2F338) MSVCRT.dll 06.07.2006 13:36:48
(?) memcpy 21048656 (0x1412D50) MSVCRT.dll 06.07.2006 13:36:48
Summary Information
API Name: memcpy
API Define: (Undefine API)
Time Start: 13:36:48.906
Duration: 0,000 ms
Module Name: C:\WINDOWS\system32\MSVCRT.dll
Is Entry API: True
Process: C:\Daten\Vortraege\LanLine\VoIP\sipXezPhone-0.35a\sipXezPhone.exe
Thread: 21684
Before Call Parameters
Pointer Paramter0: 21048656 (0x1412D50)
Pointer Paramter1: 32699192 (0x1F2F338)
Pointer Paramter2: 32 (0x20)
Pointer Paramter3: 5 (0x5)
Pointer Paramter4: 32699760 (0x1F2F570)
Pointer Paramter5: 3 (0x3)
After Call Parameters
Pointer Paramter0: 21048656 (0x1412D50)
Pointer Paramter1: 32699192 (0x1F2F338)
Pointer Paramter2: 32 (0x20)
Pointer Paramter3: 5 (0x5)
Pointer Paramter4: 32699760 (0x1F2F570)
Pointer Paramter5: 3 (0x3)
Return
21048656 (0x1412D50)
Call Tree
memcpy
Help
Detail


APIMonitor (Registered Version)
Action File Grid Help
Views
Process and Threac
API Name
Return Value
Module Name
Time Start
IsEntry API
Process and Thread : Process: C:\Daten\Vortraege\LanLine\VoIP\sipXezPhone-0.35a\sipXezPhone.exe(22248) , Thread:22280 (COUNT=640)
Process and Thread : Process: C:\Daten\Vortraege\LanLine\VoIP\sipXezPhone-0.35a\sipXezPhone.exe(22248) , Thread:22260 (COUNT=3042)
Process and Thread : Process: C:\Daten\Vortraege\LanLine\VoIP\sipXezPhone-0.35a\sipXezPhone.exe(22248) , Thread:22020 (COUNT=158)
LoadLibraryA 2008875008 (0x77BD0000) kernel32.dll 06.07.2006 13:35:49
LoadLibraryExA 2008875008 (0x77BD0000) kernel32.dll 06.07.2006 13:35:49
LoadLibraryExW 2008875008 (0x77BD0000) kernel32.dll 06.07.2006 13:35:49
LoadLibraryExW 47448065 (0x2D40001) kernel32.dll 06.07.2006 13:35:52
Summary Information
API Name: LoadLibraryExW
API Define: function LoadLibraryExW(lpLibFileName: PWideChar; hFile: THandle; dwFlags: DWORD): HMODULE; stdcall;
Time Start: 13:35:52.218
Duration: 0,000 ms
Module Name: kernel32.dll
Is Entry API: True
Process: C:\Daten\Vortraege\LanLine\VoIP\sipXezPhone-0.35a\sipXezPhone.exe
Thread: 22020
Before Call Parameters
PWideChar lpLibFileName: C:\WINDOWS\system32\calc.exe
THandle hFile: 0 (0x0)
DWORD dwFlags: 2 (0x2)
After Call Parameters
PWideChar lpLibFileName: C:\WINDOWS\system32\calc.exe
THandle hFile: 0 (0x0)
DWORD dwFlags: 2 (0x2)
Return
47448065 (0x2D40001)
Call Tree
LoadLibraryExW
Help
Detail

Buffer Overflow in SIP Foundry's SipXtapi Bibliothek

ERNW
Wir leben IT-Security.



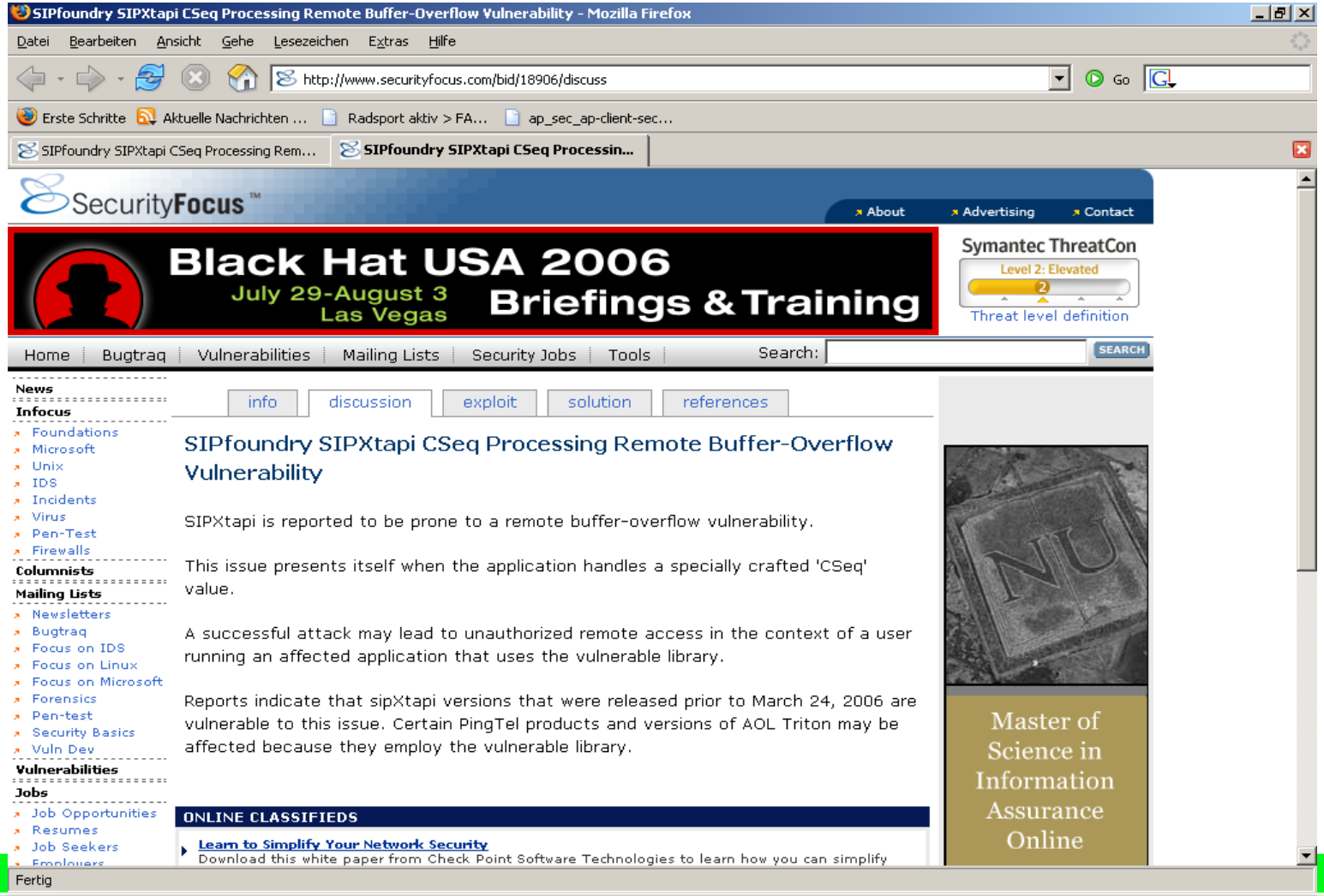
Kommunikation & Publikation

ERNW
Wir leben IT-Security.

- Kommunikation an Hersteller
- Übergabe *sehr* detaillierter Infos und Proof of Concept Tools
- Abstimmung des Zeitfensters für die Problembehebung
- Bereitstellung eines Patches / Hotfixes durch den Hersteller
- Und Veröffentlichung …

Publikation

ERNW
Wir leben IT-Security.



Potentielle Auswirkungen der Schwachstelle

ERNW
Wir leben IT-Security.

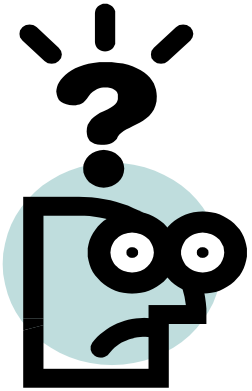
- Anfällig sind (idealerweise: waren) alle Komponenten, die die betroffene Bibliothek nutzen, darunter
  - teilweise Pingtel-Produkte
  - AOL Triton
- Beliebiger Programmcode kann durch Angreifer auf Opfer-System ausgeführt werden (üblicherweise mit den Berechtigungen des Users, der die Komponente gestartet hat)
- Personal Firewall ist nicht zwingend hilfreich (je nach Konfiguration und Kommunikationsverhalten des Users)
- Bei (noch) weiterer Verbreitung von Softphones Potential für Wurm

Was lernen wir daraus?

ERNW
Wir leben IT-Security.

- „VoIP Security“ heisst nicht nur Sicherung des Transports
- VoIP Security heisst auch+gerade Sicherung der Endpunkte sowie Komponenten
- Hardening, Patchen, Administrations-Prozesse
- Wie bei den meisten „neuen Technologien“ werden anscheinend grossflächig vorhandene Bibliotheken eingesetzt

  => weitere Sicherheits-Probleme zu erwarten (das ERNW Research Lab arbeitet ;-)



Fragen?

… und Antworten

Vielen Dank für Ihre Aufmerksamkeit!