Internet Security

Prof. Anja Feldmann, Ph.D.
anja@net.t-labs.tu-berlin.de
http://www.net.t-labs.tu-berlin.de/

Prof. Dr. Jean-Pierre Seifert
jpseifert@sec.t-labs.tu-berlin.de
http://www.sec.t-labs.tu-berlin.de/
SIP and VoIP

Skype
an example for a VoIP client
SIP / VoIP: what are these?

- Voice over IP (VoIP)
- Session Initiation Protocol (SIP)
  - Control channel
    - Known in telephone world as signaling channel
    - Does call setup:
      - locates other end point
      - Determines if it’s available
      - Asks endpoint to alert called party
      - Passes status back to caller, ...
    - Needed even in pure IP world, e.g., to interfaces with PSTN (Public Switched Telephone Network)
  - Other control channels exist: e.g., H.323 and Skype
History of signaling channels

- Telephone signaling in the past: „in-band“ pulses or tones were sent over same circuit as used for carrying the voice traffic for call
  - „Blue boxes“ – telephone fraud devices – worked by simulating some of the control tones used to setup free calls

- Solution: „out-of-band“ signaling
  - Separate data network, known today as CCIS (Common Channel Interoffice Signaling)
  - Advantages
    - More efficient
    - Allows creation of fancier services
VoIP challenges

- What address to use? DNS name, IP address?
  - Many endpoints do not have stable, easily-memorized domain names
  - IP addresses change frequently, e.g., dialup, hotspot users
  - NAT: many endpoints have only a few IP addresses
  - What about unreachable hosts?

- Firewalls?

- PSTN interconnection?
  - Who pay’s?
  - Mapping between "phone number" and IP address?
  - Business arrangements between companies
  - What about fancy phone features?
Basic SIP architecture

- SIP endpoints speak IP
- Ideally: end-to-end conversations (SIP-to-SIP)
- Yet, each node can use a SIP proxy for call setup
Example: simple SIP call
Example: simple SIP call (2.)

- Alice uses VoIP provider 1 (VP1) as proxy
- Bob uses VoIP provider 2 (VP2) as proxy
- Alice sends SIP URI to VP1 via TCP
- VP1 determines that URI points to VP2
  Relays call setup request to VP2 via TCP
- VP2 tells Bob about call via TCP
  If Bob wants to he can accept it
- Notification is send back to Alice via VP1
- Alice establishes UDP data connection to Bob for voice call
SIP details

- SIP URIs
  - URIs are converted by means of DNS magic (NAPTR records) to an IP address
    (Not important how, just that it is)
  - tel: URIs are used for ordinary phones

- Firewalls and NATs
  - If Alice and/or Bob are behind firewalls or NATs direct end-to-end data connections may not be possible
  - Data traffic can be relayed through the proxy for one or both parties

- Multiple proxies
  - Sometimes necessary
  - How to ensure trust?
Attacking SIP

Information at risk
  • Voice content itself
    • Main concern: confidentiality
      – VoIP easier to wiretap than traditional phone service...
      – Only endpoints should see that info; use encryption for proxies
      – Relatively hard to spoof VoIP in real-time ) authenticity not that much of a concern
  • Caller and called party
    • Of great interest (see HP case)
    • Useful even after the call
    • Must be kept confidential – but proxies need it to route call
    • Must be authentic, or call can be misrouted maliciously
Attacking SIP (2.)

- Billing information
  - Derived in part from caller / called party information
  - May use other information from call routing process
  - Must be confidential – but there is no need for other parties to see any of it
  - Integrity failures can lead to billing errors, in either direction
  - (Can be a major privacy concern after the fact, e.g. HP)
Attacking via eavesdropping

- On link
  - e.g., listening at WiFi hotspot
  - ...

- On call
  - Simplest approach:
    - Listen on some link, e.g., their access link
  - What for mobile ones? Harder – they could be from anywhere
  - At proxy? What about encryption?
  - At provider? What if VoIP provider is in unfriendly country?
Attacking: other

- Registration hijacking: diverting calls
  - Attacker can try to register with VP2 as Bob
  - If attacker succeeds, all calls destined for Bob are routed to the attacker
  - Man in the middle attack ...

- Registration hijacking: tearing down sessions
  - Violates availability

- Abusing DNS
  - Call routing is partially controlled by DNS
  - Corrupt DNS answers?
    - Create fake DNS entries and reroute call via interception station
Caller/Called party information

- Easier: proxies do not move 😊 via link eavesdropping and DNS attacks

- VoIP providers are high-value targets
  - Hack the proxy
  - Conventional phone switches have been hacked
  - SIP switch speaks a much more complex protocol than PSTN switch

- IP address
  - Hard to hide
  - Legitimate recipient sees sender address, leaks location data
  - Rerouting via proxy can thus be a privacy feature
Billing system

- Similar in nature to old-style ones
- SIP billing systems are more likely to be Internet connected
  - Need strong defenses and firewalls
  - ...

...
Protecting SIP

- Use crypto to guard against eavesdropping
- Alice to VP1
  - Alice has trust relationship with her proxy
  - Authentication is relatively easy, e.g., use TLS to protect TCP session from Alice to proxy
  - Alice must verify VP1’s certificate
  - Alice can use passwords or client-side certificates to authenticate herself
- Why not IPsec?
  - Tough to protect a specific service
  - But good for organizational SIP gateway
Protecting SIP (2.)

- Proxy to proxy traffic
  - VP1 may not have a trust relationship with VP2
  - Use PKI or Web of trust
  - Use appropriate security protocol, e.g., TLS

- Proxy to Bob
  - See Alice to proxy

- End-to-end signaling traffic
  - Some information must be secure end-to-end, e.g., Bob needs to know, authoritatively, that it is Alice who has called him
  - Digitally sign the data (e.g., S/MIME) but no encryption (Intermediate nodes may need to see this!)
Key management for VoIP

- How to establish a shared key for voice traffic encryption?
  - Alice uses S/MIME to send Bob an encrypted traffic key
  - But – how does Alice get Bob’s certificate?
    - No general PKI for SIP users
    - True end-to-end confidentiality can only happen by prearrangement...
Complex scenarios / features

- Complexity causes problems
  - In this case: complex trust patterns!

- Scenario A:
  - Alice tries to call Carol – reaches Bob, Carol's secretary
  - Bob decides the call is worthy of Carol's attention – wants to transfer the call to Carol
  - Bob's phone sends Alice's phone a message saying „Call Carol, you are authorized“
  - Carol's phone has to verify that Bob authorized it
Complex scenarios (2.)

- **Scenario A: solution 1**
  - Bob uses authenticated identity body (AIB) with his name and the time
  - He sends that to Alice along with Carol’s SIP URI
  - Alice presents the AIB to Carol
  - ?

- **Scenario A: problem?**
  - Nothing linking the AIB to referral
  - Alice can give the AIB to someone else
  - Good: timestamp defends against replay
Complex scenarios (3.)

- Scenario A: solution
  - AIB sent by Bob needs to include Alice’s identity
  - Carol’s phone needs to check the certificate used in Alice’s call setup message, to verify that it is really from Alice
  - Alice’s identity in AIB must match identity in certificate
Complex scenarios (4.)

Scenario B:
- Suppose SIP call is relayed to the PSTN
- Where does the CallerID information came from?
- Can it be spoofed?

Phone network design
- Based on trust – only “real” telephone companies had phone switches
- No authentication was done on information from other switches, including CallerID
- Today: anyone can run a phone switch....
**CallerID and VoIP**

- Run Asterisk, an open source PBX program, on some machine
- Get a leased line to a VoIP-to-PSTN gateway company
- Configure Asterisk to send whatever information you want
- This is happening now, e.g.,
State of practice

- Most vendors do not implement fancy crypto
- VoIP is thus not as secure as it could be
  (But note Skype does do a lot of crypto)
- Beyond that, SIP phones tend to boot themselves
  over the network – is that connection secure?
- NIST recommends great care in using VoIP – see
Skype a P2P VoIP application
P2P: what is it?

- 1999 Napster 1. widely used P2P application
Definition of Peer-to-peer (or P2P)

- Network that relies primarily on computing power and bandwidth of participants rather than on a small number of servers
- No notion of clients or servers (client-server model), only equal peer nodes (these function simultaneously as "clients" and "servers" to other nodes)
Lots of applications

- P2P-File download
  - Napster, Gnutella, KaZaa, eDonkey, ...
- P2P-Communication
  - VoIP, Skype, Messaging, ...
- P2P-Video-on-Demand
- P2P-Computation
  - seti@home
- P2P-Streaming
  - PPLive, ESM, ...
- P2P-Gaming
- ...

...
...
...
...
Why is P2P so successful?

- Scalable – it is all about sharing resources
  - No need to provision servers or bandwidth
  - Each user brings its own resource
  - E.g., resistant to flash crowds
    (a large number of users all arriving at the same time)

Resources could be:

- Files to share;
- Upload bandwidth;
- Disk storage;
- …
Why is P2P so successful? (2.)

- Cheap – No infrastructure needed
- Everybody can bring its own content (at no cost)
  - Homemade content
  - Ethnic content
  - Illegal content
  - But also *legal* content
  - ...

- High availability – Content accessible most of time
P2P-Overlay

- Build network at application layer
- Forward packet at the application layer
- Network is *virtual*
  - Underlying physical graph is transparent to the user
  - Edges are TCP connection or an entry of a neighboring node’s IP address
- Network has to be continuously maintained (e.g., check if nodes are still alive)
P2P-Overlay (cont’d)
The P2P enabling technologies

- Unstructured p2p-overlays
  - Generally random overlay
  - Used for content download, telephony, streaming

- Structured p2p-overlays
  - Distributed Hash Tables (DHTs)
  - Used for node localization, content download, streaming
P2P techniques

- Unstructured p2p-overlays
  - Generally random overlay
  - Used for content download, telephony, streaming

- Structured p2p-overlays
  - Distributed Hash Tables (DHTs)
  - Used for node localization, content download, streaming
Skype overlay

- Protocol not fully understood
  - Proprietary protocol
  - Content and control messages are encrypted
- Protocol reuses concepts of the FastTrack overlay used by KaZaA
- Builds upon an unstructured overlay
  - Combines
    - Distributed index servers
    - A flat unstructured network between index servers
  - Two tier hierarchy
Skype overlay (cont’d)

- Super nodes (SN)
  - Connect to each other
  - Flat unstructured overlay (similar to Gnutella)

- Ordinary nodes (ON)
  - Connect to super nodes that act as a directory server (similar to index server in Napster, Gnutella clients)

- Skype login server
  - Central component
  - Stores and verifies usernames and passwords
  - Stores the buddy list
Skype Overlay (cont’ d)

Message exchange during login for authentication

Skype login server

SN ON neighbor relationship
How is overlay constructed?

- How to connect? == Find Super node
  - Use Super Node list implemented as host cache
  - Needs at least one valid entry!
  - Up to 200 entries
  - Some Super Nodes IP-addresses are hard-coded
    - Super Nodes provided by Skype

- Login:
  - Contact login server and authenticate
  - Advertise your presence to other peers: contact
    - Super Node
    - Your buddies (through Super Node), and notify presence
Super Nodes – Index servers

- Index servers
  - I.e. index of locally connected Skype users (and their IP addresses)
  - If buddy is not found in local index of Super Node
    - Spread search to neighboring Super Nodes
    - Not clear how this is implemented (flood the request similar to Gnutella?)

- Relay nodes
  - Enables NAT traversals
  - Avoid congested or faulty paths
Super Nodes – Relay nodes

- Alice would like to call Bob (or inversely)
Super Nodes – Relay nodes

- Alice would like to call Bob (or inversely)
Super Node election

- When does an ordinary node become super node?
  - High bandwidth, public IP address, details unclear
  - Highly dynamic
    - Super Node Churn, Short Super Node session time
Super Node election

- A world map of Skype Super Nodes
Skype's use of ports

- One TCP and one UDP listening port
  - As configured in connection dialog box
  - Or randomly chooses one upon installation
  - Default 80 (HTTP), 443 (HTTPS)
Skype features

- Encryption
  - 1536 to 2048 bit RSA
    - User public key is certified by login server during login
  - AES (Rijndel) to protect sensitive information
    - (256-bit encryption: $1.1 \times 10^{77}$ possible keys)
    - RSA to negotiate symmetric AES keys

- NAT and firewall
  - Conjecture use of STUN (Simple Traversal of UDP through NATs) and TURN (Traversal Using Relay NAT) to determine the type of NAT and firewall
  - Information is stored in the Windows registry
  - Use TCP to bypass UDP-restricted NAT/firewall
Skype – Functional summary

- VoIP has other requirements than file download
  - Delay
  - Jitter
- Skype network seems to handle these well in spite of
  - High node churn
- Protocol not fully understood
Skype analysis
“Silver Needle in the Skype”

(Philippe Biondi and) Fabrice Desclaux
BlackHat Europe, March 2006
Skype uncovered
Security study of Skype

Desclaux Fabrice\textsuperscript{1}

\textsuperscript{1}EADS CCR/STI/C
1. **Introduction**
   - Should we be afraid of Skype?

2. **Skype analysis**
   - Binary
   - Network - Protocol
   - Skype Authentication

3. **Enforcing anti-Skype policies**
   - Skype detection
Quick overview of Skype

End-user view
- Perfect VoIP software with good quality sound
- Ease of use and working everywhere and with every OS

Network administrator view
- Skype bypasses Firewalls, Nat, Proxies
- It uses P2P technologies
- Skype traffic cannot be isolated and is suspicious
- In a nutshell, the perfect backdoor
Why is Skype seen so suspicious?

The Binary
- Big size (about 12 Mo)
- `strings` doesn’t reveal interesting things
- Few functions in the binary import table
- The binary doesn’t want to launch if the `Soft-ice` debugger is present

The network
- Protocol is proprietary and not obvious to observe
- The number of boxes contacted by a client is very important

Conclusion
⇒ Skype is a total black box.
1 Introduction
   - Should we be afraid of Skype?

2 Skype analysis
   - Binary
     - Network - Protocol
     - Skype Authentication

3 Enforcing anti-Skype policies
   - Skype detection
Binary analysis: Encryption

Encryption layers

- Some parts of the binary are *xored* by a hard-coded key in the code.
- In memory, Skype is fully decrypted.

Decryption Procedure:
Each encrypted part of the binary will be decrypted at run time.
Binary protection: Anti debuggers

Anti Softice
- Some tests are done in order to detect the Softice debugger
- First tests are easy to detect
- The others are hidden in the binary
Binary protection: Anti debuggers

Example

First Softice test

```assembly
mov eax, offset str_Siwvid ; "\\\\.\\\\Siwvid"
call test_driver
test al, al
```

Example

Hidden test: It checks if Softice is not in the Driver list.

```assembly
call EnumDeviceDrivers
...
call GetDeviceDriverBaseNameA
...
cmp eax, 'ntic'
jnz next_
cmp ebx, 'e.sy'
jnz next_
cmp ecx, 's\x00\x00\x00'
jnz next_
```
Binary protection: Import functions

Hidden imports
- In a common binary, imported libraries and functions are described in a structure
- In Skype only some functions are present
- The other part is dynamically loaded after decryption
- This prevent disassemblers from watching interesting functions

Example

<table>
<thead>
<tr>
<th>Libraries used in hidden imports:</th>
<th>Number of total hidden imports:</th>
</tr>
</thead>
<tbody>
<tr>
<td>KERNEL32.dll</td>
<td>169/843</td>
</tr>
<tr>
<td>WINMM.dll</td>
<td></td>
</tr>
<tr>
<td>WS2_32.dll</td>
<td></td>
</tr>
<tr>
<td>RPCRT4.dll</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Binary analysis: Integrity

Multiple checksums

Skype checks its own integrity by implementing thousands of code checkers. If a software breakpoint is installed, or a modification is done in the binary, Skype will stop/crash randomly.

Main scheme of Skype code checkers
Binary analysis: Obfuscation

Code obfuscation

- Some parts of the binary are obfuscated. This may be used in order to avoid *Skype light remakes*.
- The next code represents a code checker that is generated to avoid being detected by IDA.
- Pointers are calculated, junk code is inserted in the real code.
```
start:
    xor   edi, edi
    add   edi, 0x688E5C
    mov   eax, 0x320E83
    xor   eax, 0x1C4C4
    mov   ebx, eax
    add   ebx, 0xFFCC5AFD
loop_start:
    mov   ecx, [edi+0x10]
    jmp   lbl1
    db    0x19
lbl1:
    sub   eax, ecx
    sub   edi, 1
    dec   ebx
    jnz   loop_start
    jmp   lbl2
    db    0x73
lbl2:
    jmp   lbl3
    dd    0xC8528417, 0xD8FBBBD1, 0xA36CFB2F, 0xE8D6E4B7, 0xC0B8797A
    db    0x61, 0xBD
lbl3:
    sub   eax, 0x4C49F346
```
1 Introduction
   • Should we be afraid of Skype?

2 Skype analysis
   • Binary
     • Network - Protocol
     • Skype Authentication

3 Enforcing anti-Skype policies
   • Skype detection
Protocol analysis

Indication Packets
Most packets are compounded in two parts:
- A clear header
- A ciphered payload. The payload is ciphered with a RC4 stream

Signalling Packets
- The RC4 is only used to obfuscate the packet payload
- That’s why a simple tcpdump doesn’t reveal interesting things
- RC4 key can be recovered from the packet (UDP)

VoIP Packets
This encryption is different. Skype uses AES and only the sender/receiver can decrypt them. This is not a simple obfuscation.
Packet dissection

```
IP
version: 4L
ihl: 5L
tos: 0x0
len: 128
id: 9288
flags: 0
frag: 0
ttl: 128
proto: UDP
chksum: 0x462d
src: 172.16.13.13
dst: 84.30.194.243
options:
UDP
sport: 40036
dport: 2142
len: 53
chksum: 0x18cd

Skype SoF
id: 0x18e9
func: 0x2

Skype Crypted Data
iv: 0xB5E06D9DL
crc32: 0x8E154F85L
crypted: 5...99.x98a.x[...]
```
**UDP packet deciphering**

- The RC4 key is generated using src/dst IP plus packet ID.
- The clear payload is composed by objects containers, in which data are stored.
- Those data will be received by an object manager.
1. Introduction
   - Should we be afraid of Skype?

2. Skype analysis
   - Binary
   - Network - Protocol
   - Skype Authentication

3. Enforcing anti-Skype policies
   - Skype detection
Client authentication

Authority public key

13 trusted moduli (RSA). Size is between 1536 and 2048 bits.

Client public key

- Each client generates its private/public key (RSA 1024 bits) at login time. It’s a session RSA key
- A secret is shared between clients and the authority: the hashed password

Login mechanism

- The client generates a session key
- Encrypts the shared secret with it
- Then encrypts the session key with RSA (using a trusted modulus)
- If the authority passes the test, it signs the couple login/public key and sends it to Supernodes
Client authentication

- **Skype modulus**
- **RSA 1536 bits**
- **Rand(192 bits) Session Key**
- **Hash (SHA 160 based)**
- **User modulus**
- **Shared Secret**
- **256 bits key**
- **Cipher (AES 256 based)**
- **Encrypted Session key**
- **Sent to the Login Server**
- **Encrypted Shared secret**

- **Login**
- **\nskype\n**
- **password**
- **MD5 Hash**
1 Introduction
   • Should we be afraid of Skype?

2 Skype analysis
   • Binary
   • Network - Protocol
   • Skype Authentication

3 Enforcing anti-Skype policies
   • Skype detection
TCP Skype packet detection

When a TCP session is established:
- Each machine sends its seed key to the other
- This seed will be used to generate a continues RC4 stream
- Except for the two first packets

This can be used to detect Skype connection by deciphering TCP packet without using internal decryption mechanism.

Packet sent by a Skype client and by a supernode

Seed

Bytes xored with a RC4 stream (generated by Seed)

Deencription of the packet

\x00\x01 \x00\x00\x00\x01 \x00\x00\x00\x01

The First bytes of the clear payload are always the same
Skype TCP packet detection

Packet received/send by a Skype client

Seed

Encrypted payload (10 bytes long)

1x00x01x00x00x00x01x00x00x01

RC4 Stream recovered (10 bytes)

Second packet: Encrypted payload

First cleared 10 bytes of the second packet

Check Skype packet properties

This packet is not a Skype packet

This packet is a Skype packet:
We need to take counter measures

Desclaux Fabrice
Skype uncovered
Conclusion

Proprietary protocol

- Proprietary and obfuscated protocols don’t prevent flaws
- It can only slow down the exploitation of it
- Worse, it may protect a 0-day


Questions?
Thank you!

Questions?