Skype and the Flux-Capacitor

Dr. Ralf Schlatterbeck
Open Source Consulting

Email: office@runtux.com
Web: http://www.runtux.com
Tel. +43/650/621 40 17

Contents

Contents

Skype Overview

- „It just works“: no problems with firewalls
- Skype makers known for Spyware-contaminated filesharing software (KazaA)
- built-in software-update function in Skype
- Doesn’t adhere to any standards – no third-party offers
- Closed source – Open Source Skype would be nice and this may be within reach now

→ Whom to entrust with your phone calls?
→ Encryption: Who owns the keys???

VoIP Introduction

- **V**oice **o**ver **I**nternet **P**rotocol (and often also Video)
- Distinction: call signalling and content (voice/video)
- Signalling Protocols e. g. SIP, H.323, . . .
- Content transmission e. g. RTP (Realtime Transfer Protocol)
- Quality: Low latency, low Jitter, low packet loss
- Open: Success of VoIP depends on standards: we want interconnection of different vendor solutions
- If signalling and content are sent via different path we may get problems with firewalls

VoIP Signalling Protocols

- IAX: Inter Asterisk Exchange: use same path for signalling and content – firewall-friendly
- SIP: Session Initiation Protocol
- Jabber/Jingle (Google Talk)
- H.323: ITU-T Standard
- MGCP: Media Gateway Control Protocol
- SCCP/Skinny (proprietary CISCO)
- UNISTIM (proprietary Nortel)
- Skype (proprietary): optionally one signalling path

→ „Asterisk is first PBX in history to natively support proprietary IP terminals from the two biggest players in VoIP“ [MSM05]

VoIP: Bunch of Firewall Problems

- SIP, Jingle, H.323 are signalling protocols
  → only for call-management
- For Content: RTP (Realtime Transfer Protocol)
  RTP-Port is dynamically negotiated

→ Firewall (NAT!) problems

- Remedy (or workaround): NAT-Traversal protocols, or intelligent firewall
- For Linux/Netfilter: SIP+H.323 connection tracking and NAT modules
- or use IAX!

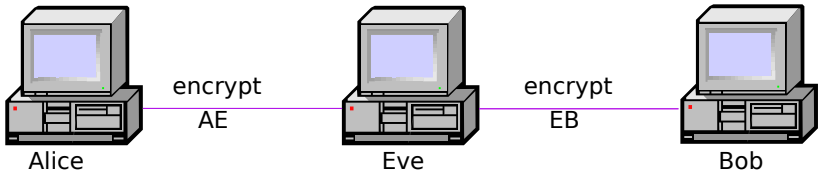
Standards for Encrypted Telephony

- encryption option for IAX with pre-shared keys
- VoIP can be routed via a VPN, e. g. OpenVPN
- SRTP/SRTCP: needs a key management scheme
  Master-Key: call parties need to agree on a key
- Phil Zimmermanns ZRTP: SRTP + Diffie-Hellman
  → call parties need not know each other!
  → uses existing SIP infrastructure
  → approved as informational RFC
- IETF DTLS based on certificates for negotiating SRTP key – will become Standards-Track RFC (!)

Security with Encrypted Telephony

- Traffic Analysis: „Who with whom“
  → nearly conforming to EU Data Retention :-)
- IAX transmits session information in clear
- Even when using a VPN we can determine that there is VoIP-content due to packet frequencies and -sizes
- SRTP without ZRTP, IAX or VPN:
  needs prior agreement of calling/called parties
- ZRTP MITM-Protection: call partners can read key-hash to each other – hard to fake by man in the middle

Excursus: Man in the Middle



- Eve – the woman in the middle – does key exchange with Alice and Bob
- There are two keys, AE und EB
- But: Alice and Bob can compare key checksums by reading them to each other over the phone
  → Authentification via speech!

Existing Implementations

General:

- IAX encryption: no personal experience
- OpenVPN + Asterisk IAX (or SIP/RTP) works well

ZRTP:

- SRTP/ZRTP not yet in Asterisk but ZFone can be used
- Phil Zimmermanns ZFone as add-on to existing RTP-based software
- GNU ccRTP supports ZRTP
- GPL-Client Twinkle with ZRTP

Alternatives to Skype

- SIP-Phones (Ekiga, Twinkle, Linphone)
- Chat: multiprotocol-client Pidgin
- Chat + telephony: Google Talk (Jabber extension „Jingle“ for Voice/Video)
- Open Source telephony: Asterisk: SIP, Jabber/Jingle, and lots of other protocols . . .
- Encryption: Phil Zimmermann's ZRTP + SRTP or IETF's DTLS (certificate-based) + SRTP

History of Publications about Skype

- Biondi, Desclaux: Silver Needle in the Skype, Black-hat Europe 2006 [BD06]
- Desclaux, Kortchinsky: Vanilla Skype [DK06a, DK06b], Recon 2006
- Biondi, Desclaux, Kortchinsky analyzed the network obfuscation and compression algorithms of Skype but didn't publish their work
- they called the network obfuscation function „Flux Capacitor"
- 2010-07-07 Sean O'Neil: Skype's Biggest Secret Revealed

Skype Security Considerations

- Writeable text-segment: Decrypts itself when called
- could load code over the network into memory
- when asking for updates Skype sends its Skype-ID

→ maybe this is used for sending you your own personalized (trojan?) update?

- Skype can find out IPs of other Skype clients [BD06, p. 61] – there is already a service announced to discover Skype IPs by username

→ imagine the possibilities when coupled with Geo-location

Skype Security Considerations

- Remote Scan: We can make another node connect to a specified IP-Adress – even in the network perimeter (inside Firewall) of that host [BD06, p. 82]
- Censor API for chat activated for chinese Skype calls `ContentFilter.exe` – API present in all Skype versions [DK06b, p. 40]
- AP2AP protocol for communication of non-Skype applications via Skype [DK06b, p. 42]
  - Information Exfiltration (file transfer)
  - remote control

⇒ that's why I chose not to use Skype

Skype Network Obfuscation: Flux Capacitor

- A big obfuscated function first described by Desclaux/Kortchinsky: „it is not an improvement of the flux capacitor" [DK06a, p. 52]
- they even describe a method how to generate C-code from the binary [DK06a, p. 55–65]
- obfuscation function takes a 32-bit seed and computes an RC-4 key
- the RC4-key is used to encrypt the UDP-packet or TCP-stream
- for TCP the seed is sent in clear as the first bytes of the stream

Skype Network Obfuscation: Flux Capacitor

- for UDP the CRC32 of source-IP, dest-IP and per packet ID XOR an initialisation vector (IV) is used as the seed
- The packet ID and IV are in clear in every packet
- ... so everything is known about the RC-4 key
- It's just an obfuscation layer, no real encryption because the key is known (security by obscurity)
- But the obfuscation is so "good" that it hasn't been published until now
- I've been able to verify this for TCP, and now also for UDP

Excursus: CRC32

- Silver Needle suggests UDP uses CRC32 – there is example code from Vanilla which has a CRC32
- The well-known crc32 from Ethernet, ZIP, etc. uses the same polynomial but a different finalisation:

```
def zlib_crc (s, seed = 0) :
    state = seed ^ 0xFFFFFFFF
    state = do_crc (s, state)
    return state ^ 0xFFFFFFFF

def skype_crc (s, seed = 0xFFFFFFFF) :
    return do_crc (s, seed)
```


Excursus: CRC32

- zlib crc32 uses initial value 0, skype uses 0xFFFFFFFF

→ effect is the same internal state

⇒ So we can use crc32 from zlib for skype crc:

```
def skype_crc (s, seed = 0xFFFFFFFF) :
    mask = 0xFFFFFFFF
    return (crc32 (s, seed ^ mask)) ^ mask
```

- this implementation allows part-wise computation:

```
x = skype_crc ('abcd')
x = skype_crc ('efgh', x)
y = skype_crc ('abcdefgh')
assert (x == y)
```


Skype Network Obfuscation: Compression

- Before encryption Skype uses an undocumented arithmetic compression algorithm [DK06b, p.7–8]
- Arithmetic compression close to Huffman but using reals
- Sean O'Neil promises to publish the compression, too: "The compression algorithm required to complete the decoding of Skype packets will be published in December this year at 27C3"

Skype and Cryptography

- not everything is just obfuscation
- Skype uses RSA keys signed by Skype master RSA key [DK06b, p. 12-18] Skype public keys are compiled into binary
- uses shared secret: MD5-hash of user information `<login>\nskyper\n<password>`
- Skype trusts Skype master RSA key
- local RSA key used for session key exchange
- when searching for a users you get their public key
- clients authenticate to each other by signing 8-byte challenge with their RSA key

Skype Communication Security

- Voice/Video calls are encrypted with session key
- But: Skype voice engine can generate encrypted trace files if certain variables are set [DK06b, p. 56]
- XOR with 31 byte key – probably breakable
- +AP2AP protocol this may permit eavesdropping

Kurt Sauer, „Leiter der Sicherheitsabteilung von Skype“ im ZDNET Interview: „Wir stellen eine sichere Kommunikationsmöglichkeit zur Verfügung. Ich werde Ihnen nicht sagen, ob wir dabei zuhören können oder nicht.“

Debugging Skype

- All sorts of debugger checks
- but under Linux we can debug with GDB
- . . . as long as we don’t set more than 2 break-points (HW)
- Nice: Wait for Skype to call into lib: `break __libc_start_main`
- Answer y to “Make breakpoint pending on future shared library load?”
- In that state skype has decrypted its text-segment

Skype Task-Force?

- Check how to decrypt UDP packets
- implement compression protocol or wait for 27C3
- Use that to implement a Wireshark dissector for Skype
- Chat
- . . . Voice?
- . . .

Bibliography

[BD06] Philippe Biondi and Fabrice Desclaux. Silver needle in the skype. In *Blackhat Europe*, Netherlands, March 2006.

[DK06a] Fabrice Desclaux and Kostya Kortchinsky. Vanilla skype part 1. In *Recon*, Montreal, Canada, June 2006.

[DK06b] Fabrice Desclaux and Kostya Kortchinsky. Vanilla skype part 2. In *Recon*, Montreal, Canada, June 2006.

Bibliography

[MSM05] Jim Van Meggelen, Jared Smith, and Leif Madsen. *Asterisk – The Future of Telephony*. O'Reilly, September 2005.