An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol

Paper by: Salman Baset and Henning Schulzrinne, Columbia University, New York

Presentation by: Ariane Keller
27.11.06
Introduction

• Skype: peer-to-peer VoIP client developed in 2003 by organisation that created Kazaa
• Skype is not an open protocol
• Analysed Version
  – 1.4.0.84 for windows (current version: 2.5.0.151)
  – 1.2.0.18 for Linux (current version: 1.3.0.53)
Skype Network

- SC: Skype client
- SN: Super node
  - Skype client that has further responsibilities, each SC has to connect to an SN for a successful login
- Login server
  - Central entity responsible for user authentication
Key components

- **Ports**
  - Random TCP and UDP listening port
  - TCP listening ports at port 80 (http) and 443 (https)

- **HC: Host cache**
  - Each SC maintains a list of available SN (IP addresses with ports)

- **Buddy list**
  - Contact list, stored in a xml file on the local computer and on a central server
Key components

- **Encryption** (explanation by Skype)
  - Each packet encrypted
  - AES, 256-bit encryption
  - RSA to negotiate AES keys
  - User public key certified by RSA certificates at login

- **NAT, Firewall**
  - Skype determines and stores whether it is behind a NAT or firewall
  - Variation of STUN and TURN Protocols
Experimental Setup

- 3 different Network setups:
  - 2 SC on machines with public IP addresses and no firewalls
  - 1 SC behind port-restricted NAT, 1 SC with public IP address
  - Both SC behind port-restricted NAT and UDP restricted firewall

- Ethereal is used to monitor network traffic

- Shared library and system call redirection on Linux
Skype Function Analysis: Start up

• The first time after installation an http 1.1 GET message is sent to skype.com with keyword 'installed'.
Skype Function Analysis: Login

- Experiment 1: clear the HC and override connect() and sendto() to return always false
- Experiment 2: fill HC with one invalid entry and observe login process
- Experiment 3: clear HC and allow all traffic
Skype Function Analysis: Login

Send UDP packet to 7 bootstrap nodes to port 33033

Send TCP packet to 7 bootstrap nodes to port 33033

Receive IP address/port pairs of SN

Exchange some data with some SN, obtain IP of login server

Authentication on login server

Receive IP address/port pairs of SN

HC full

HC empty
Login Server

- 2 IP addresses
- Only central component of Skype network
- Ensures that Skype user names are unique
- Located in Denmark and Netherlands
- Buddy list is hosted on login server
- SC receives IP of Login Server from a SN
Login Process Time

• Measure login time for the 3 network setups
  – public IP addresses and port restricted NAT: 3 - 7 seconds
  – UDP restricted firewall: 35 seconds (after sending UDP packets to 20 SN a TCP connection is established)

• Analysis of subsequent logins
  – Login time for UDP restricted firewall decreased to 5 to 10 seconds -> Skype stores its last connectivity information
Login Messages

• First and second message always identical
  – Payload 22 3 1 0 0 0 for the SC -> server packet
  – Payload 23 3 1 0 0 0 for the server -> SC packet
  – SSL uses similar patterns

• Messages 3 and 4 different for each login attempt
  – 4 byte common header
  – Length fields to indicate message length and location of next headers
Blocking Skype Login

• Experiment 1: block IP of login server
  - Login attempt succeeded

• Experiment 2: block alternative IP addresses
  - Login attempt succeeded

• Experiment 3: block all packets which include byte sequence 22 3 1 0 (this pattern occurs in every login attempt)
  - Login attempt failed
Skype Function Analysis: User Search

• Analysis-Problem: Packets are encrypted and cannot be traced behind a SN

• Global Index technology (Information of users are stored in a distributed way on the SN)

• Skype claims that it will find each user if it has logged in during the last 72 hours

• No details could be found on how search is performed
Skype Function Analysis: User Search

- For SC behind a UDP restricted Firewall, the search is performed by the SN.
- Search results are cashed on intermediate nodes.
- Login server used as fall back (for non-existent user-names, login server was always contacted).
- Wildcard searches do not return identical results on different machines.

Skype client → network
- tcp to SN
- udp to 8 nodes
- tcp to SN
- udp to 16 nodes
Skype Function Analysis: Call Establishment

- Caller and callee on public IP addresses
  - TCP signalling between caller and callee SC
  - Caller sends also some UDP packets to other online Skype nodes
- Caller behind port restricted NAT, callee on public IP address
  - TCP signalling packets routed over other Skype node
  - UDP media packets routed directly
- Caller and callee behind UDP restricted firewall
  - TCP signalling over other Skype node
  - Voice packets are transmitted over TCP
Media Transfer and Codecs

- Media transfer preferably over UDP
- Roughly 85 voice packets exchanged both ways in 1 second
- Payload size of voice packet: 40 to 120 bytes
- No silence suppression (if no data needs to be transferred packets are still sent)
  - Maintains UDP binding at NAT
  - Avoid to drop TCP congestion window size
- Calls on hold: send UDP voice packets and TCP signalling packets in regular intervals
- Frequency range: 50Hz - 8000Hz
- Congestion: minimum bandwidth required: 2 kilobytes/second
Conferencing

- Experimental setup:
  - 3-way conferencing
  - “A” always public IP
  - Different connection setup scenarios

- Results:
  - No full mesh conferencing
  - “A” always mixed the packets

- Exception
  - “B” and “C” communicate via relay “D” and then “A” joins the conference: Data flows still over “D”
Other experiments

- Skype allows user to log in from multiple machines simultaneously
  - Calls are routed to all locations and upon picking a call they are immediately cancelled at the other locations
- Skype super nodes cannot be "generated"
  - If an ordinary Skype node “A” is the only entry in the HC, a connection will be established to the super node of “A”

- If two SC are behind same NAT voice traffic flows directly over the private network
Comparison of Skype, Yahoo, MSN, Google Talk voIP Applications

| Application | Memory Usage before call (caller, callee) | Memory Usage during call (caller, callee) | Process priority before call | Process priority during call | Mouth-to-ear latency | Latency Standard Deviation |
|-------------|-----------------------------------------|------------------------------------------|----------------------------|----------------------------|----------------------|---------------------------|
| Skype       | 19 MB, 19 MB                            | 21 MB, 27 MB                             | Normal                     | High                       | 96 ms                | 4                         |
| Yahoo       | 38 MB, 34 MB                            | 43 MB, 42 MB                             | Normal                     | Normal                     | 152 ms               | 12                        |
| MSN         | 25 MB, 22 MB                            | 34 MB, 31 MB                             | Normal                     | Normal                     | 184 ms               | 16                        |
| G-Talk      | 9 MB, 9 MB                              | 13 MB, 13 MB                             | Normal                     | Normal                     | 109 ms               | 10                        |

TABLE III
SKYPE, YAHOO, MSN AND GOOGLE TALK COMPARISON
Skype Super Node Map

- Approximately 8000 logins were performed
- Each Super Node (approx. 900) involved was registered
- Using MaxMind the position of each SN was determined
Summary

• Analysis of the Skype protocol with the following approaches:
  – Change and observation of the entries in the Host Cache
  – Observation of the network traffic generated by Skype using Ethereal
  – Shared library and system call interception techniques

• Results:
  – Some insight in the login process
  – Comparison of VoIP applications
  – Super Node map
Conclusion

Since Skype relays on super nodes and since they need to have a certain bandwidth, the hole system would collapse if all Skype clients decided to put a bandwidth limitation on their Skype application.

Observing the network traffic is not enough to get good knowledge of the Skype protocol, since it is an encrypted and proprietary protocol.