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

<table>
<thead>
<tr>
<th>End-user view</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Perfect VoIP software with good quality sound</td>
</tr>
<tr>
<td>- Ease of use and working everywhere and with every OS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network administrator view</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Skype bypasses Firewalls, Nat, Proxies</td>
</tr>
<tr>
<td>- It uses P2P technologies</td>
</tr>
<tr>
<td>- Skype traffic cannot be isolated and is suspicious</td>
</tr>
<tr>
<td>- In a nutshell, the perfect backdoor</td>
</tr>
</tbody>
</table>
Why is Skype seen so suspicious?

The Binary

- Big size (about 12Mo)
- *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
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

```asm
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.

```asm
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 prevents 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>
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.

**Multiple checksums**

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.
Introduction
Skype analysis
Enforcing anti-Skype policies

Binary
Network - Protocol
Skype Authentication

Desclaux Fabrice
Skype uncovered

```
start:
    xor    edi, edi
    add    edi, Ox688E5C
    mov    eax, Ox320E83
    xor    eax, Ox1C4C4
    mov    ebx, eax
    add    ebx, OxFFCC5AFD

loop_start:
    mov    ecx, [edi+Ox10]
    jmp    lbl1
    db     Ox19

lbl1:
    sub    eax, ecx
    sub    edi, 1
    dec    ebx
    jnz    loop_start
    jmp    lbl2
    db     Ox73

lbl2:
    jmp    lbl3
    dd     OxC8528417, OxD8FBBBD1, OxA36CFB2F, OxE8D6E4B7, OxC0B8797A
    db     Ox61, OxBD

lbl3:
    sub    eax, Ox4C49F346
```
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
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

1. Login
2. nsyker
3. password

- MD5 Hash
- User modulus
- Shared Secret
- Cipher (AES 256 based)
- 256 bits key
- Hash (SHA 160 based)
- RSA 1536 bits
- Rand(192 bits)
- Session Key
- Skype modulus
- Encrypted Session_key
- Sent to the Login Server
- Encrypted Shared secret
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.
Skype TCP packet detection

Packet received/send by a Skype client

Seed | Encrypted payload (10 bytes long)

| \x00\x01|\x00\x00|\x00\x00|\x01|\x00|\x00|\x00|\x01 |

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
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?