SKYPE VOICE OVER IP - SOFTWARE VULNERABILITIES

TECHNIQUES & METHODS – ZERO DAY EXPLOITATION 2011

1. AUTHORS OF THE SECURITY SKYPE WHITE-PAPER

1.1 Pim J. F. Campers (24) has worked around five years in the IT Security sector. It began as a hobby, but after high school, he decided to expand his experience in the area of IT Security. His specialties are security checks on web applications, server and client applications, underground economy, bypass/crack filters or walls & risk/threat analysis. He currently works closely with academia and high class software manufacturers and companies.

Pim has joined the "Global Evolution" Research Team 2007. From 2010 to 2011, Pim J.C. and Benjamin M. (Research Team) identified over 300 zero day vulnerabilities in well known products from companies such as DELL, Mozilla, Kaspersky, McAfee, Google, Cyberoam, Safari, Bitdefender, Asterisk, Telecom, PBX & SonicWall. In 2010 he founded the company "Evolution Security" with Benjamin K.M.. After the firm's establishment arose the european Vulnerability Lab as the legal european initiative for vulnerability researchers, analysts, penetration testers, and serious hacker groups. Pim is also the co-leader of the european Wargaming + Vulnerability-Lab Research Team & have a lot of stable references by solved events or contests like ePost SecCup, SCS2, EH2008, HAR2009, Da-op3n & exclusive zero-day exploitation sessions/releases.

1.2 Benjamin Kunz M.(28) is active as a penetration tester and security analyst for private and public security firms, hosting entities, banks, isp(telecom) and ips. His specialties are security checks(penetrationtests) on services, software, applications, malware analysis, underground economy, military intelligence/cyberwar, reverse engineering, lectures and workshops about IT Security. During his work as a penetration tester and vulnerability researcher, many open- or closed source applications, software and services were formed more secure. In 1997, Benjamin K.M. founded a non-commercial and independent security research group called, "Global Evolution - Security Research Group" which is still active today. From 2010 to 2011, Benjamin M. and Pim C. (Research Team) identified over 300 zero day vulnerabilities in well known products from companies such as DELL, Barracuda, Mozilla, Kaspersky, McAfee, Google, Cyberoam, Safari, Bitdefender, Asterisk, Telecom, PBX & SonicWall. In 2010 he founded the company "Evolution Security". After the firm's establishment arose the Vulnerability Lab as the legal european initiative for vulnerability researchers, analysts, penetration testers, and serious hacker groups. Ben is also the leader of the Contest + Vulnerability-Lab Research Team. He have a lot of stable references by solved events or contests like ePost SecCup, SCS2, EH2008, Har2009, Da-op3n & exclusive zero-day exploitation sessions/releases.

2. INFOMERCIAL ON SKYPE

Skype is a software application that allows users to make voice and video calls and chats over the Internet. Calls to other users within the Skype service are free, while calls to both traditional landline telephones and mobile phones can be made for a fee using a debit-based user account system. Skype has also become popular for its additional features which include instant messaging, file transfer, and video conferencing. Skype has 663 million registered users as of 2010. The network is operated by Skype Limited, which has its headquarters in Luxembourg. Most of the development team and 44% of the overall employees of Skype are situated in the offices of Tallinn and Tartu, Estonia. In April 2011 Microsoft bought the skype company. Skype is now a official Microsoft Company.

The Skype Team provids an own security team for special software issues. "Vulnerability-lab.com is in no way affiliated with Skype, does not work with Skype and cannot provide any conduit to them. Any vulnerabilities to be reported to Skype should follow the process as detailed on their website.

www.skype.com

3. OVERVIEW PUBLIC SKYPE VULNERABILITIES

Let's have a look on the last Microsoft Skype software vulnerabilities. We split the categories into 4 classes: URI Handler Bugs, Creation + Deletion Misconfigurations, Denial of Service & Buffer Overflows to give an overview of older bugs/vulnerabilities.

3.1 URI Handler Vulnerabilities on Skype

Skype URI Handler Input Validation Vulnerability - 2010
Skype File URI Security Bypass Code Execution Vulnerability - 2008
Skype skype4com URI Handler Remote Heap Corruption Vulnerability - 2007
Skype "callto:" URI Handler Buffer Overflow – 2004

Example: HACK
Description: Allows an attacker to execute malicious content via skype message board on clicks.

Example: http:/www.example.com/?foo="><script>document.='http://11.133.0.7';</script>
Description: Client-Side XSS attack that allows an attacker to redirect a MacOS user via message board.

Bypass Example: %20%20%20%20%20%20%20?foo=%22%3E%22%3C%69%66%72%61%6D%65%20%73%72%63%3D%68%74%74%70%3A%2F%2F%76%75%6C%6E%2D%6C%61%62%2E%63%6F%6D%3E
Description: Client-Side XSS attack that allows to bypass the older software filter with HEX URL Value.

Example: callto:"><script>document.='http://11.133.0.7';</script>
Description: Client-Side link spoofing attack allows to redirect victims, execute malware & script code

3.2 Denial of Service Vulnerabilities on Skype

Skype Client for Mac Chat Unicode Denial of Service Vulnerability 2010

Example: Mathematical Alphanumeric Symbols (1D400-1D7FF)
Description: After recieving a message the client freeze itself & the user in unable to get new messages.

3.3 Creation & Deletion Vulnerabilities on Skype

Skype URI Processing Arbitrary XML File Deletion Vulnerability – 2009
Skype Linux Insecure Temporary File Creation – 2005

Create Example:

strace -e trace=open skype
open("/home/vulnerability-lab/image.jpg", O_RDONLY|O_LARGEFILE) = 21 // Picture by User
open("/tmp/skype_profile.jpg", O_WRONLY|O_CREAT|O_TRUNC|O_LARGEFILE, 0666) = 23 // Insecure temporary file creation (it should use O_EXCL or O_NOFOLLOW flag)
Log: ln -s file_to_overwrite /tmp/skype_profile.jpg
Description: Bug located in multi-user environment because usually /tmp directory is "world-writable"

Delete Example:

Description: undocumented 'save_pxml' command that upon clicking will trigger the deletion of an arbitrary attacker specified XML file. (Extras Plugin!)

3.4 Parse Buffer Overflow Vulnerabilities on Skype

Skype M/VLD Parse Integer Buffer Overflow Vulnerability - 2005

Example:

Object Counter*	**M objects**
M (VLD)	(VLD)

* The first number in the packet is the amount of forthcoming objects!

Description: The overflow occurs when *M* is greater than **0x40000000:** e. g. when **M=0x40000010, HeapAlloc(0x40)** is called, but up to **0x40000010** objects are effectively read in the packet and written into memory.

4. HOW 2 EXPLOIT & Detect?

On this section of the white paper we will explain "how to find a way to exploit the software". Everybody knows it's not that easy to declare vulnerabilitiess & bugs on the skype voip software ... or everybody keeps his material private on his own. Like you can see on the securityfocus list there are not much or tricky vulnerabilities listed ... http://www.securiteam.com/products/S/Skype.html

4.1 How to detect own Skype Vulnerabilities?

Skype is not easy to hack as the most people out there know. After our research tour through the Skype VOIP software & modules our group discovered 6 new security vulnerabilities. The most detected bugs are on software nodes like internal modules & overall levels/forms. To detect this type of vulnerabilities is not that easy, the attacker mostly needs to find a specific node on the software were script-code or commands are executed. After detection of such a vulnerability the attacker also needs to verify that the bug is remotely exploitable because a lot of them just locally exploitable.

1. Find/Detect/Identify Bug/Vulnerability
2. Analyse Bug/Vulnerability
3. Verify Vulnerability (Remote/Local)
4. Exploitation (PoC)

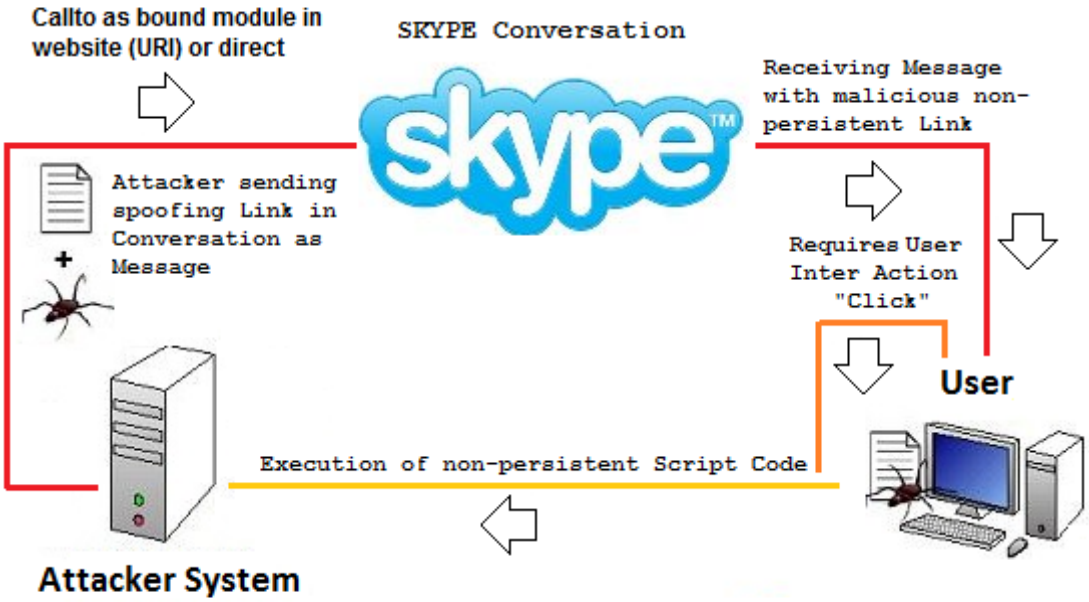
From the search, find/detect & verify to the ending exploitation is a long way. In the next section we will show the attack models & how the exploitation out of the box works. Attackers have to think innovatively & creatively on the identification process. Very often the bugs are located on in a specific module but just get displayed in a third, to trigger/exploit this kind of bugs is very hard.

4.2 How to exploit own Skype Vulnerabilities (Out of the Box!)?

The following attack model describes the client site attacks on the famous Skype software.

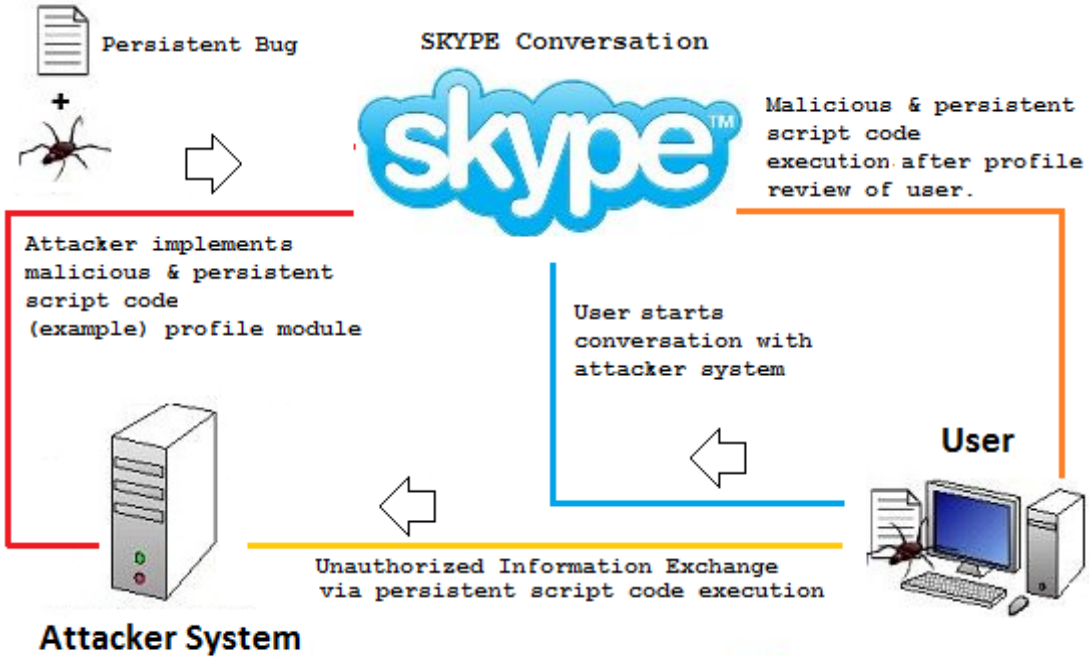
4.2.1

CLIENT SIDE SKYPE EXPLOITATION (Remote & Local)



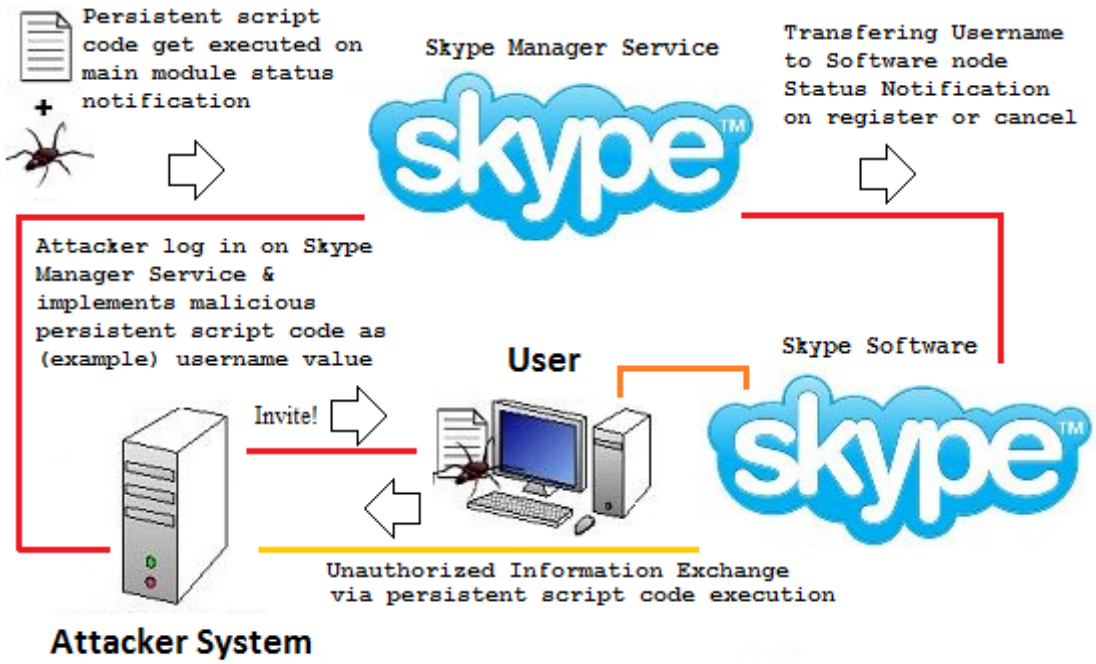
4.2.2

PERSISTENT SCRIPTCODE INJECTION #1 SKYPE EXPLOITATION (Remote & Local)



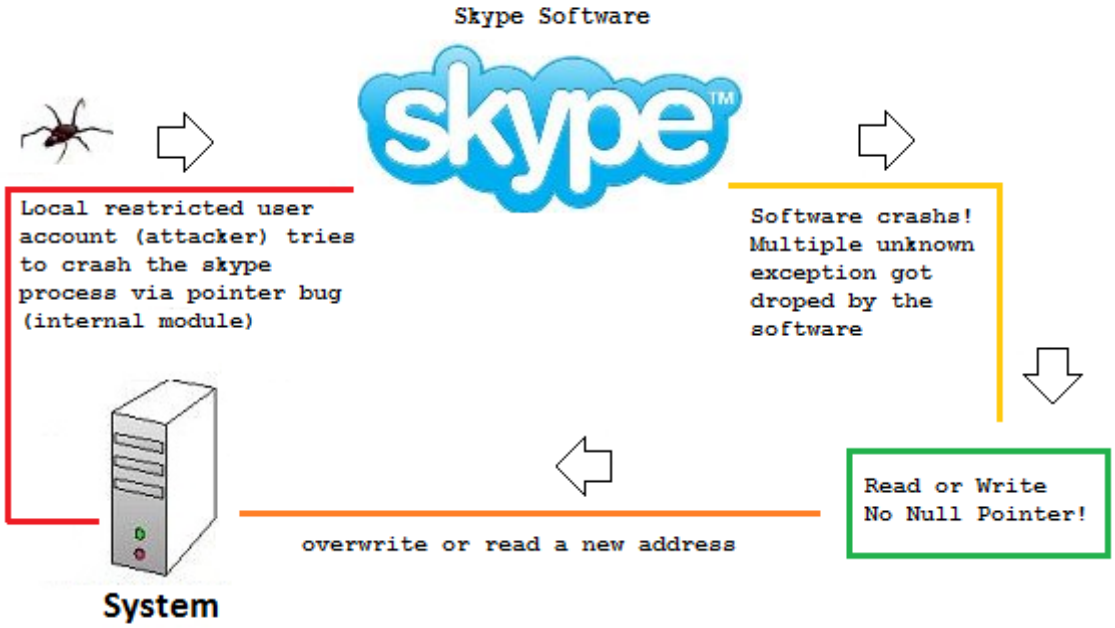
4.2.3

PERSISTENT SCRIPTCODE INJECTION #2 SKYPE EXPLOITATION (Remote & Local)



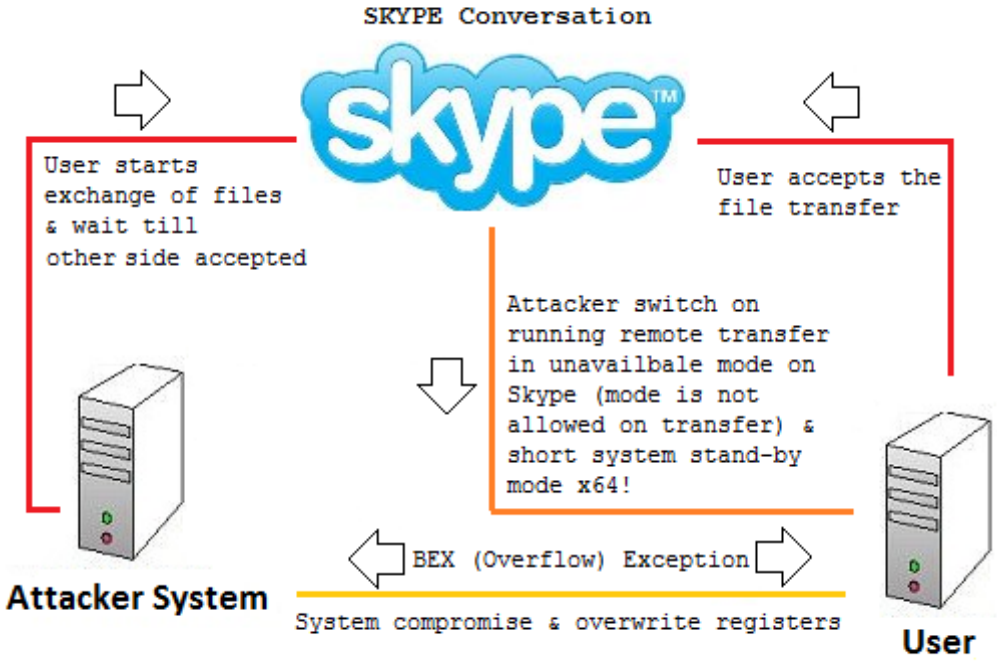
4.2.4

MEMORY CORRUPTION (POINTER) SKYPE EXPLOITATION (Local)

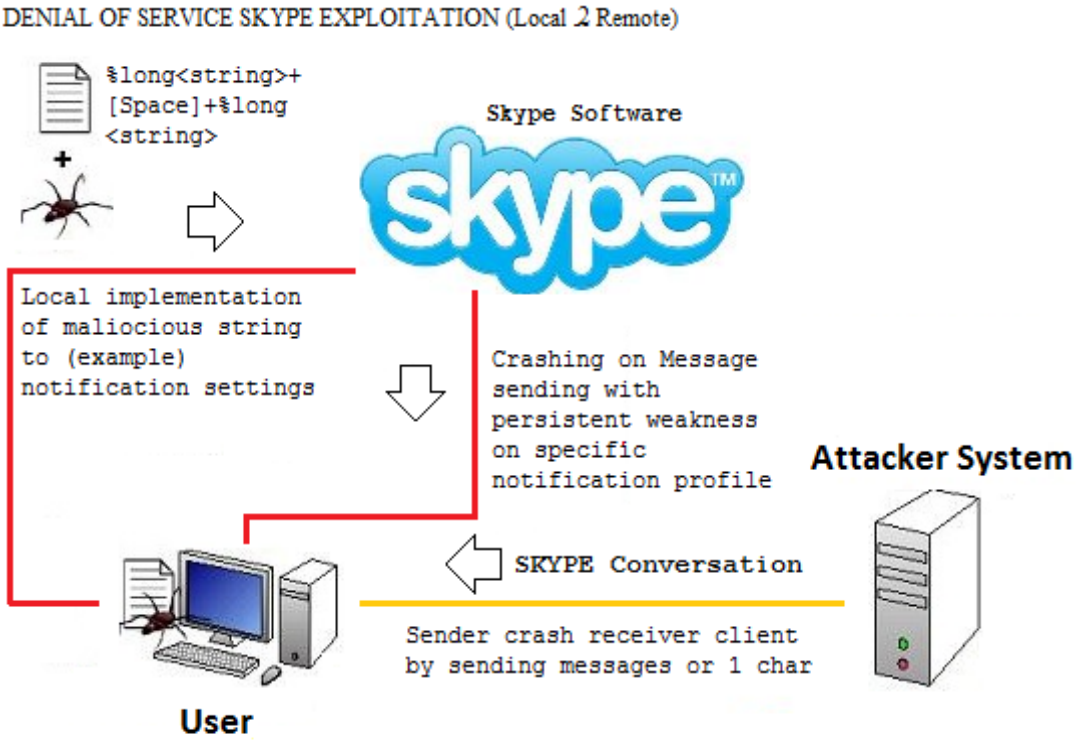


4.2.5

TRANSFER BUFFER OVERFLOW SKYPE EXPLOITATION (Remote)



4.2.7



5. Presentation of own detected 0 day Skype vulnerabilities

Now I will even explain how to exploit Skype and show everything that i found testing this software product. In total **5** vulnerabilities were found:

- 5.6 Skype v5.3.x v2.2.x v5.2.x – Denial of Service Vulnerability (Local to Remote)(+Video)(Linux,Windows & Mac) [Medium[-]
- 5.2 Skype 5.3.x 2.2.x 5.2.x - Persistent Software Vulnerability (Local & Remote)(Linux,Windows & Mac) [High[-]
- 5.5 Skype v5.2.x and v5.3.x – Memory Corruption Vulnerability (Local) (+Video)(Linux,Windows & Mac) [Medium]
- 5.3 Skype v5.3.x - Transfer Standby Buffer Overflow Vulnerability (Remote) (Windows) [High]
- 5.1 Skype 5.3.x 2.2.x 5.2.x - Persistent Profile XSS Vulnerability (Local & Remote)(Linux,Windows & Mac) (+Video) [High]

OS Independent: **4** Vulnerabilities

I will explain the weaknesses individually and what impact they might have on the user. All my tests were recorded in a video session which is currently being processed for release.

Skype v5.3.x v2.2.x v5.2.x - Local Denial of Service Vulnerability (Map: 4.2.7)
First, the stable local & remote denial of service attack in Skype. As with other messengers, it is possible to allow for specific words Notify. Something called Highlight text in irc. For example if a notify on the word "Benjamin" is made, everytime this word is now used in a conversation the user gets notified. The function looks like this ...

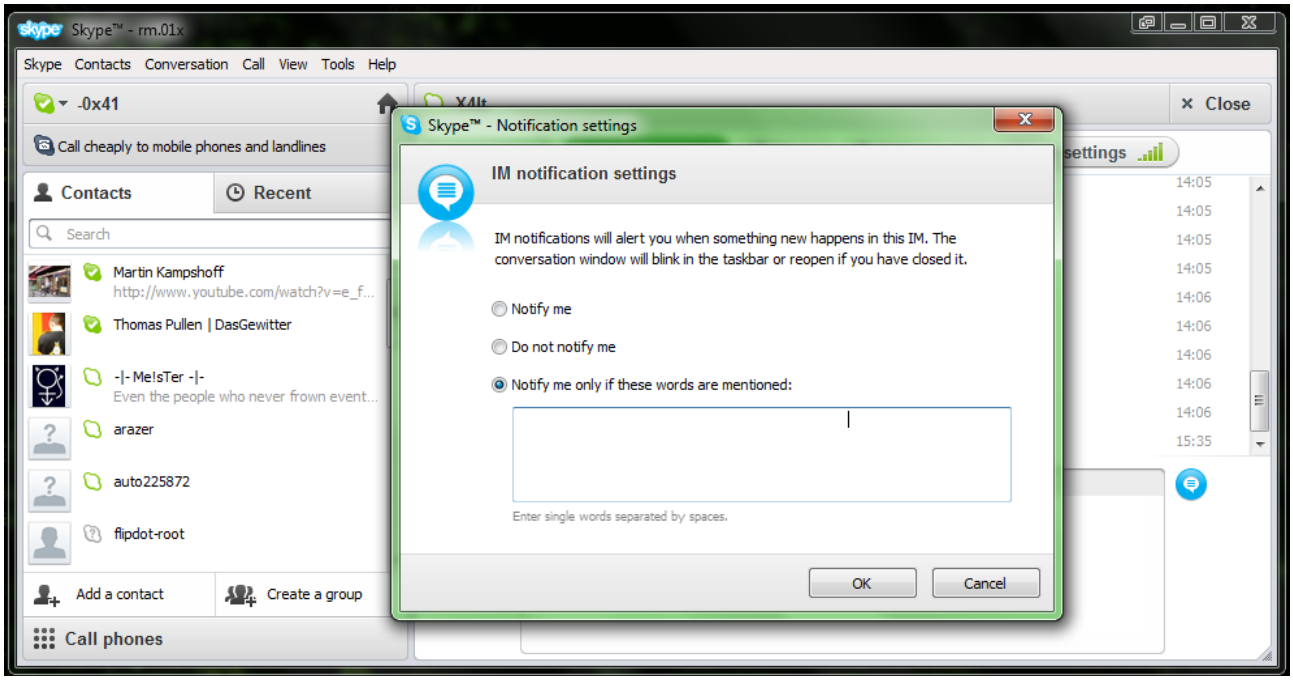


Abbildung 1: Benachrichtigungskonfiguration or Notificationsettings

The steps to do so are the following ...

Notificationsettings => Input Field(Checkbox 3) => Input string => Save the words/strings => Check when a return or a receive is made on the stored words/strings => User Notification if word/string is found in the words/strings stored in the notification list

The stable denial of service vulnerability is detected on the windows & Mac OS version of Skype. The Bug is located in the notification module of the Skype client. The vulnerability allows an local attacker to crash the complete skype process via APPHang(application-hang) & APPCrash. The bug allows an local attacker to block the users message board by including on the savebox ...

%long<string>+[Space]+%long<string>

message. The input field of the notifification-option has no size restriction as maximum. After including the string into the application, it will never verify what is inside the notification option.

The result is in a stable denial of service: apphang & appcrash ... also after a restart because its saved with the account. The bug has a persistent weakness when including on a specific user profile. This can result in a stable denial of service vulnerability after the exchange of v-cards. When the user with the manipulated profile sends a message to me(receiver) the vulnerability crashes the client after receiving 1 character of text.

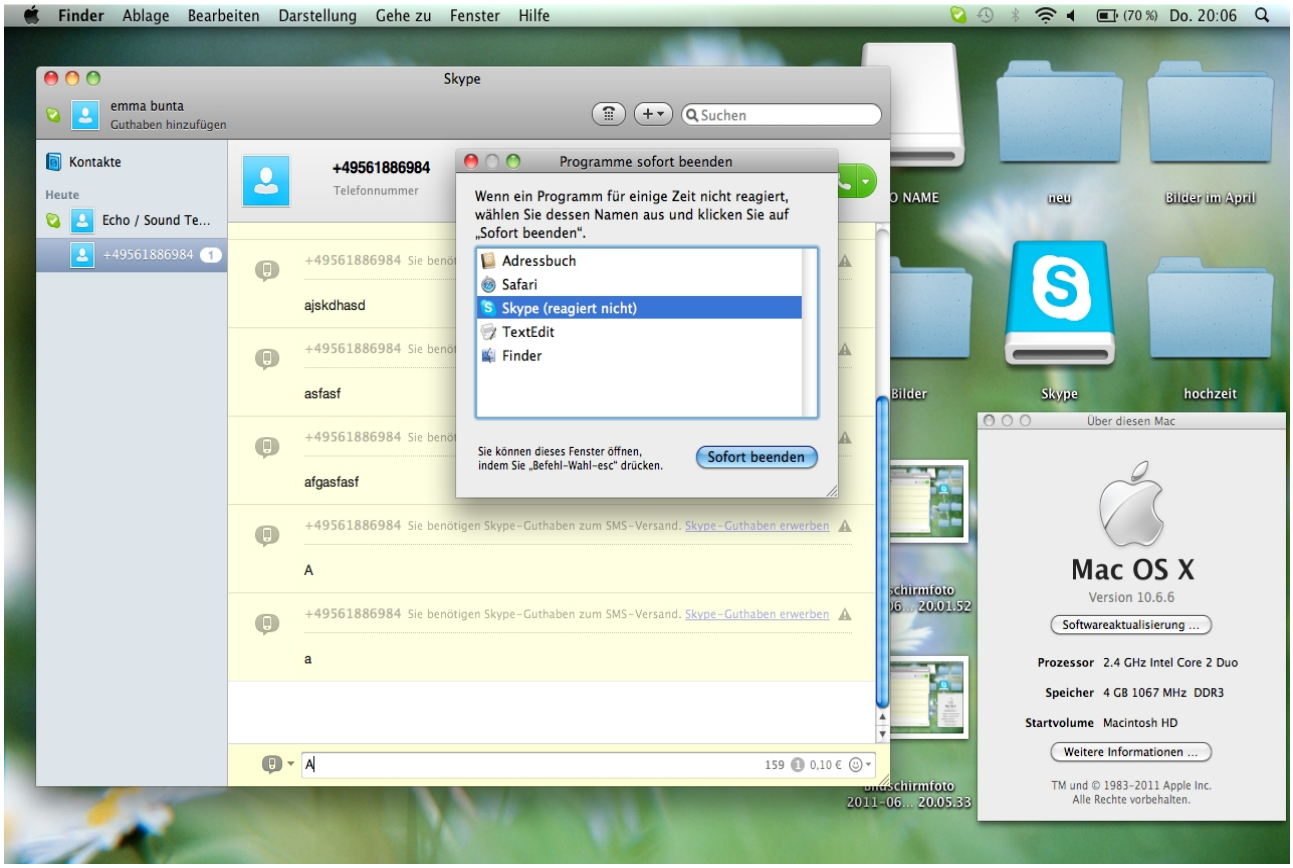
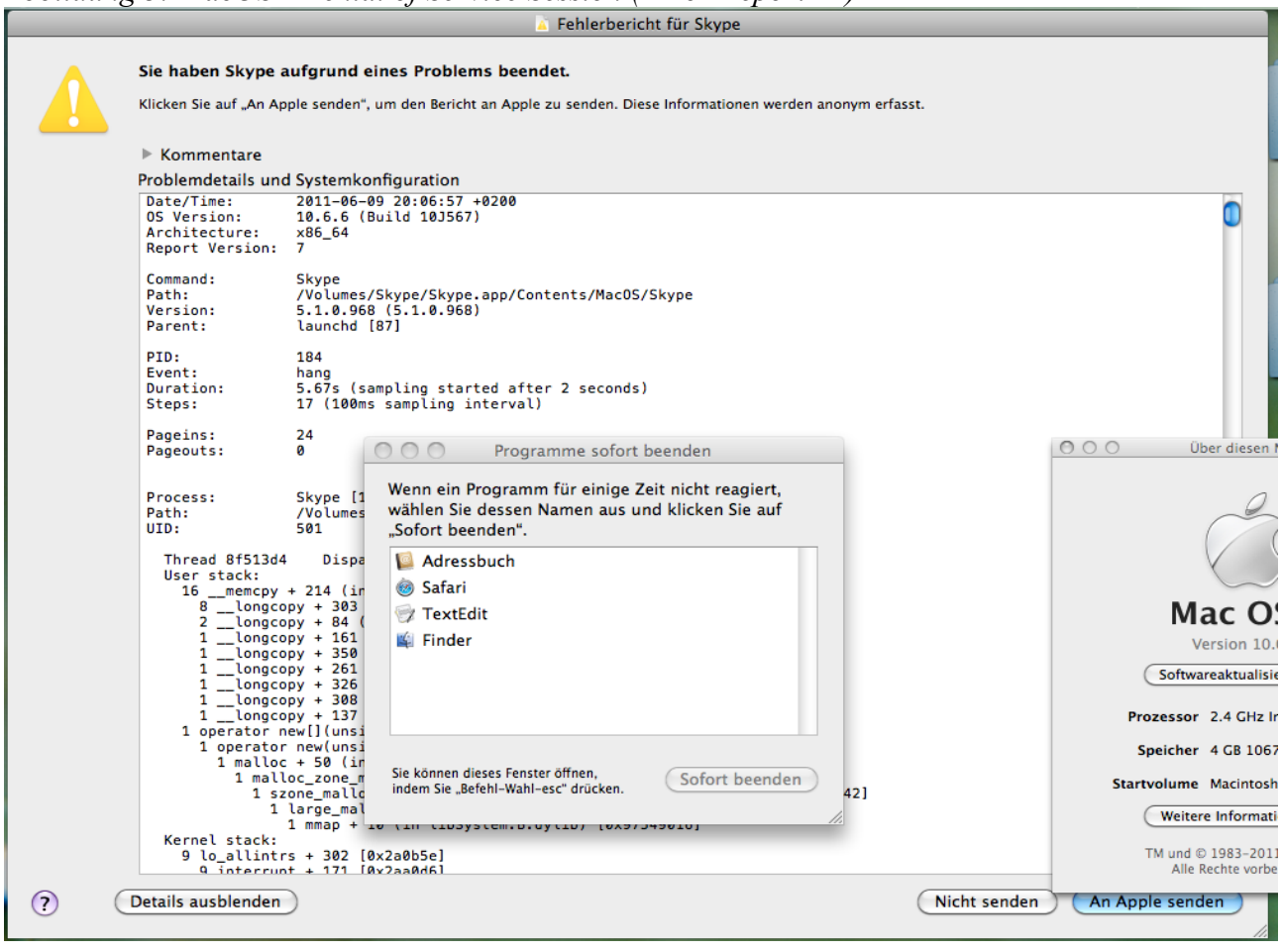


Abbildung 2: MacOS - Denial of Service Session (Software Crash Receiver)

Abbildung 3: MacOS - Denial of Service Session (Error Report #1)



```
OS Version:      10.6.6 (Build 10J567)
Architecture:    x86_64
Report Version:  7

Command:         Skype
Path:            /Volumes/Skype/Skype.app/Contents/MacOS/Skype
Version:         5.1.0.968 (5.1.0.968)
Parent:          launchd [87]

PID:             184
Event:           hang
Duration:        5.67s (sampling started after 2 seconds)
Steps:           17 (100ms sampling interval)

Pageins:         24
Pageouts:        0


Process:         Skype [184]
Path:            /Volumes/Skype/Skype.app/Contents/MacOS/Skype
UID:             501

  Thread 8f513d4    DispatchQueue 100
  User stack:
    16 __memcpy + 214 (in commpage [libSystem.B.dylib]) [0xffff0876]
      8 __longcopy + 303 (in commpage [libSystem.B.dylib]) [0xffff132f]
      2 __longcopy + 84 (in commpage [libSystem.B.dylib]) [0xffff1254]
      1 __longcopy + 161 (in commpage [libSystem.B.dylib]) [0xffff12a1]
      1 __longcopy + 350 (in commpage [libSystem.B.dylib]) [0xffff135e]
      1 __longcopy + 261 (in commpage [libSystem.B.dylib]) [0xffff1305]
      1 __longcopy + 326 (in commpage [libSystem.B.dylib]) [0xffff1346]
      1 __longcopy + 308 (in commpage [libSystem.B.dylib]) [0xffff1334]
      1 __longcopy + 137 (in commpage [libSystem.B.dylib]) [0xffff1289]
    1 operator new[](unsigned long) + 17 (in libstdc++.6.dylib) [0x91f5f703]
      1 operator new(unsigned long) + 36 (in libstdc++.6.dylib) [0x91f5f617]
        1 malloc + 50 (in libSystem.B.dylib) [0x97548278]
```

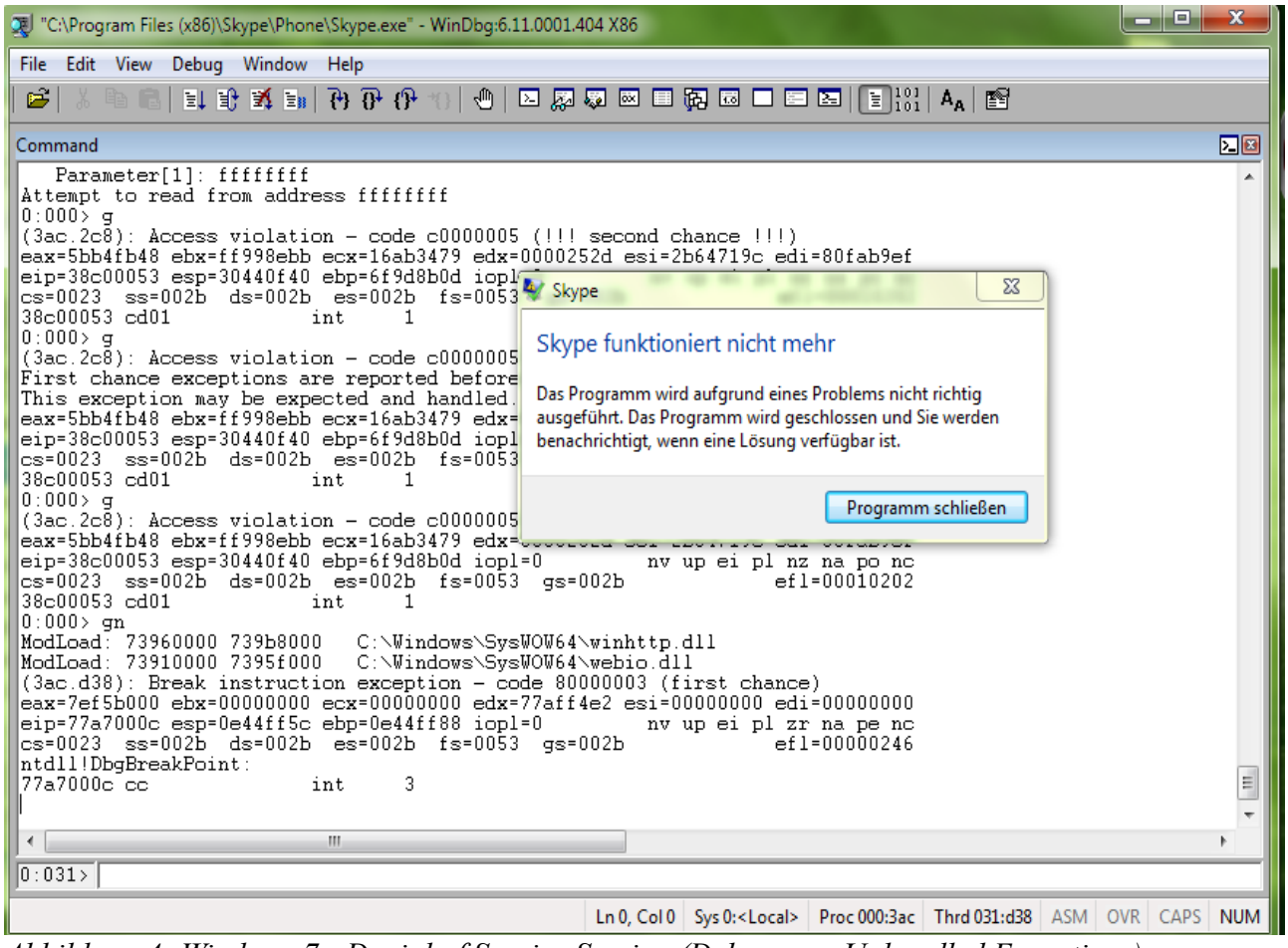


Abbildung 4: Windows 7 - Denial of Service Session (Debugger - Unhandled Exceptions)

When you try to delete the string out of the "Notification Settings" the software also tries to check the input and crashes after execution of the module. The attack model is triggered by the following steps:

Schema: Notificationsettings => Input Field(Checkbox 3) => Input(unrestricted) => %long<string>+[Space] +%long<string> => Save the words/strings => Check when a return or a receive is made on the stored words/strings => Crash on every remote & local message on the profile (Exploitation)

The denial of service bug is vulnerable to Mac OS, Windows & the Linux version (+mobile) of Skype.

Skype 2.8.x, 5.3.x & 2.2.x – Persistent Software Vulnerability (Map: 4.2.3)
Now we have gotten a really nice bug that allows persistent remote script code inclusion directly into the Skype software. Months ago I started looking at the status messages and bars from Skype.

In my tests, I was looking for specific points in the software such as transmitted messages from embedded (inbound) services. When I got notified a few weeks ago about the new service manager for Skype,

I've been looking at this and made small function test. “So far so good I thought, nice piece of software.” But when I logged off my account the status message changed to "The administrator has removed you from Skype Manager called "USER NAME "(HD output).

I had finally found what I'd been looking for. Then I naturally logged back onto Skype Service Manager. This time I tried to log on with different tags as my Skype user name, but that unfortunately didn't work because of security restrictions. “Okay.” I thought to myself. “I can't do that yet.” I logged into Skype Manager with a normal name and went to the profile names. There, I didn't try to log in with the user name again, but rather just tried to run an update. How perfectly it worked. The registration didn't work given that it had already been validated, but the update ran problemfree.

Unfortunately, you couldn't insert more than fifty characters due to the restriction from the update. Then, I began to try to inject different little scripts for tags and afterward just signed out of Skype Service Manager. At the login for the user name, I was left with filtered output.

Status Message: The Administrator has removed you from Skype Manager called ">"(DB Username output)

Okay, I had come as far as finding a form in Skype's software that allowed you to send a direct status update. The problem was that the software would filter a normal tag like, '>"<iframe src=http://www.vulnerability-lab.com>' itself. I then began to throw the scripts through different character encoding calculator applications (HTML, hex-url, base-64). When I was done, I tried all of the different encodings in the user name section. After a quick update, I logged out to see whether it worked, and saw this:

Okay, I had come as far as finding a form in Skype's software that allowed you to send a direct status update. The problem was that the software would filter a normal tag like, '>"<iframe src=http://www.vulnerability-lab.com>' itself. I then began to throw the scripts through different character encoding calculator applications (HTML, hex-url, base-64). When I was done, I tried all of the different encodings in the user name section. After a quick update, I logged out to see whether it worked, and saw this:

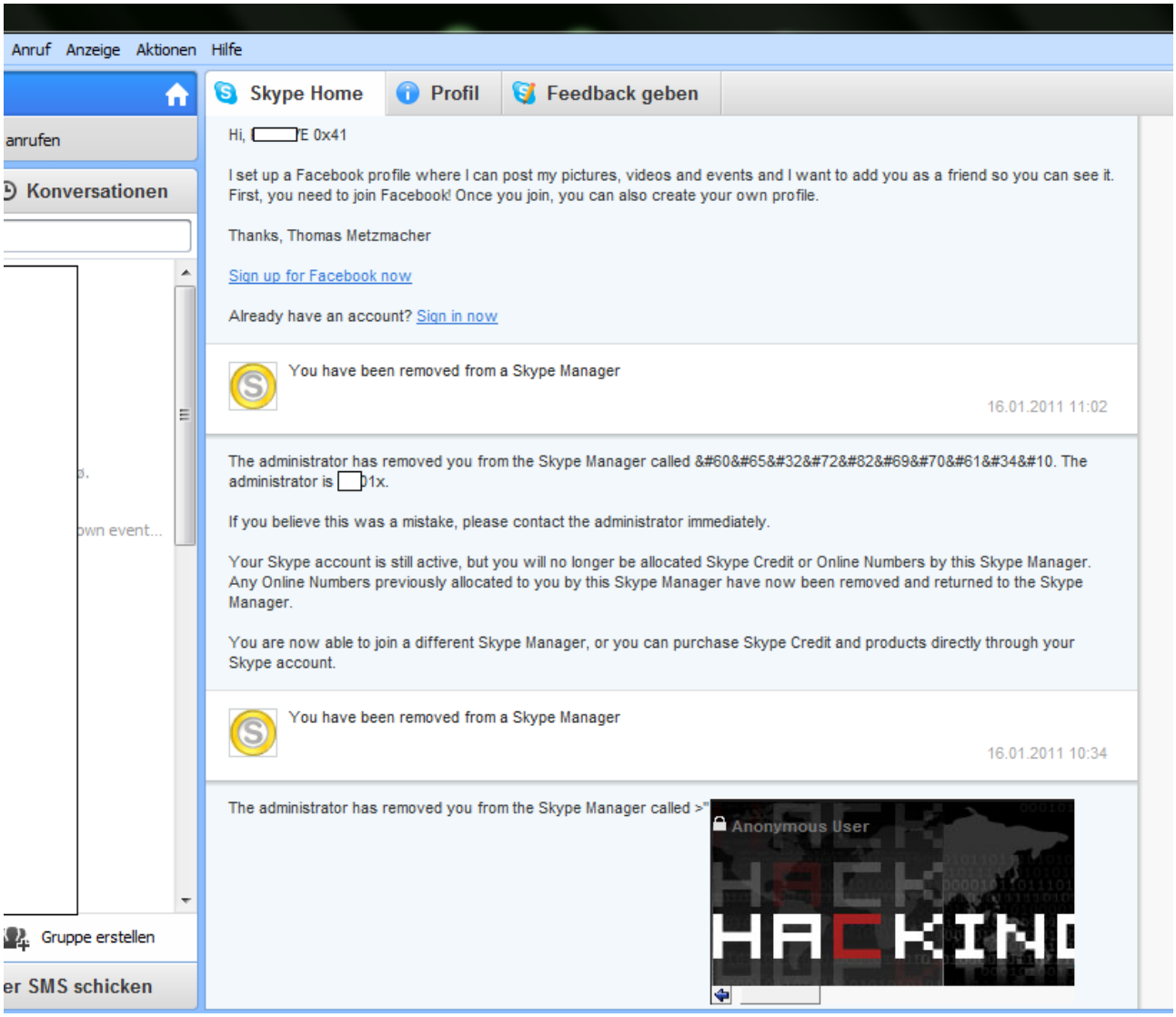


Abbildung 5: Execution of persistent script code on skypes statusbar message main module

Now it was possible to run scripts through updating the status in Skype with those underlying vulnerabilities. You simply cannot just ignore the status as it is permanently set in the menu. The security hole is also not just hinged upon something within the operating system. The weakness lies in the validation of the output of the status change, but also in the user name validation of the Skype Manager service. At the end of my tests it was clear, it didn't matter which service you let update your status with, you just had to find some type of correlating inbound service (usern- output) and you can run your favorite executables in the software.

Schema: Install => register => Login1 => register 2 => Check1 => Login2 => Change username input to Tag (Script Code HTML/JS) => Cancel Service => Redisplay in status message bar index (Exploitation)

PoC:
%3E%22%3C%69%66%72%61%6D%65%20%73%72%63%3D%68%74%74%70%3A%2F%2F%76%75%6C%6E%2D%64%62%2E%63%6F%6D%3E

Skype 2.8.x & 5.3.x - Persistent Profile XSS Vulnerability (Map: 4.2.2)
At least a second input validation vulnerability has been identified on the profile input/output of the Skype client software. The bug is located on the "myphone" profile input fields. The affected vulnerable output area is the active user preview were the script code get executed. The lab researcher Levent Kayan (noptrix) has discovered the vulnerability 2011-07-15 on vulnerability-lab.com. The security risk of the vulnerability is estimated as high(-) because exploitation requires to be listed on the active users preview section of the software.

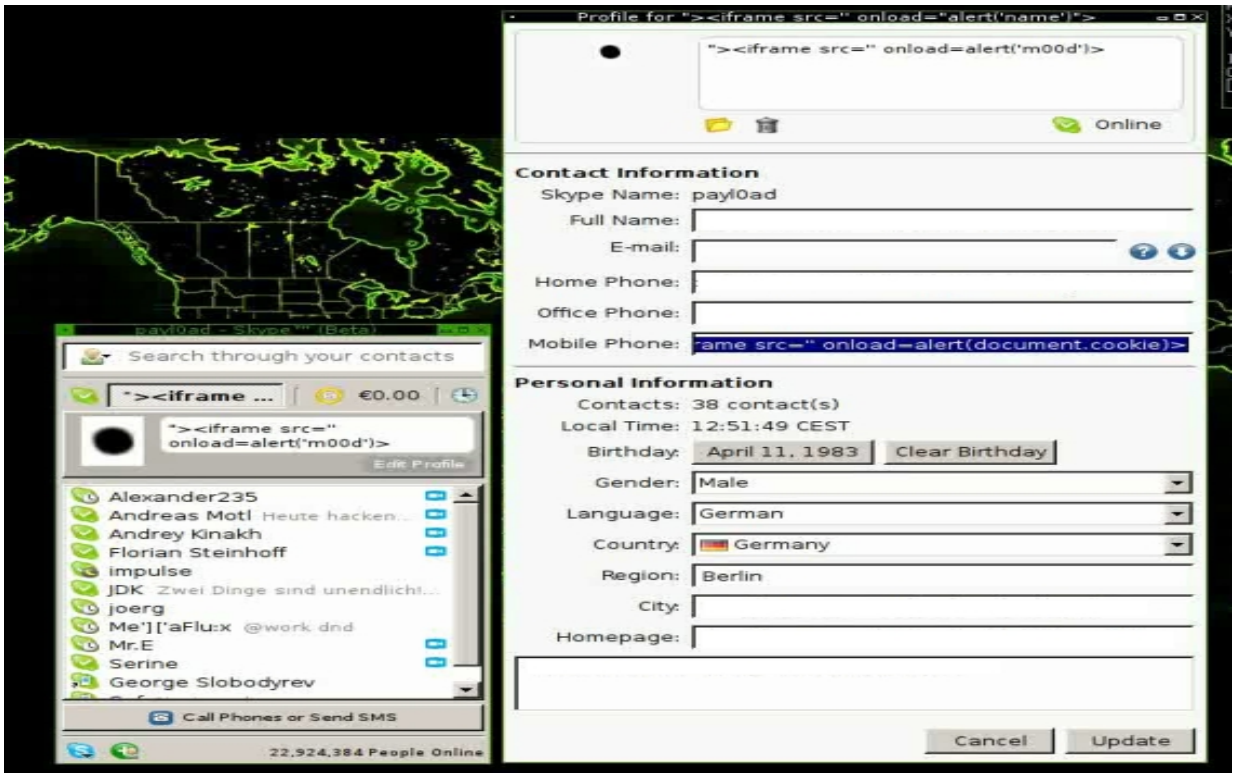


Abbildung 6: Vulnerable Input Field of the Skype Profile Section

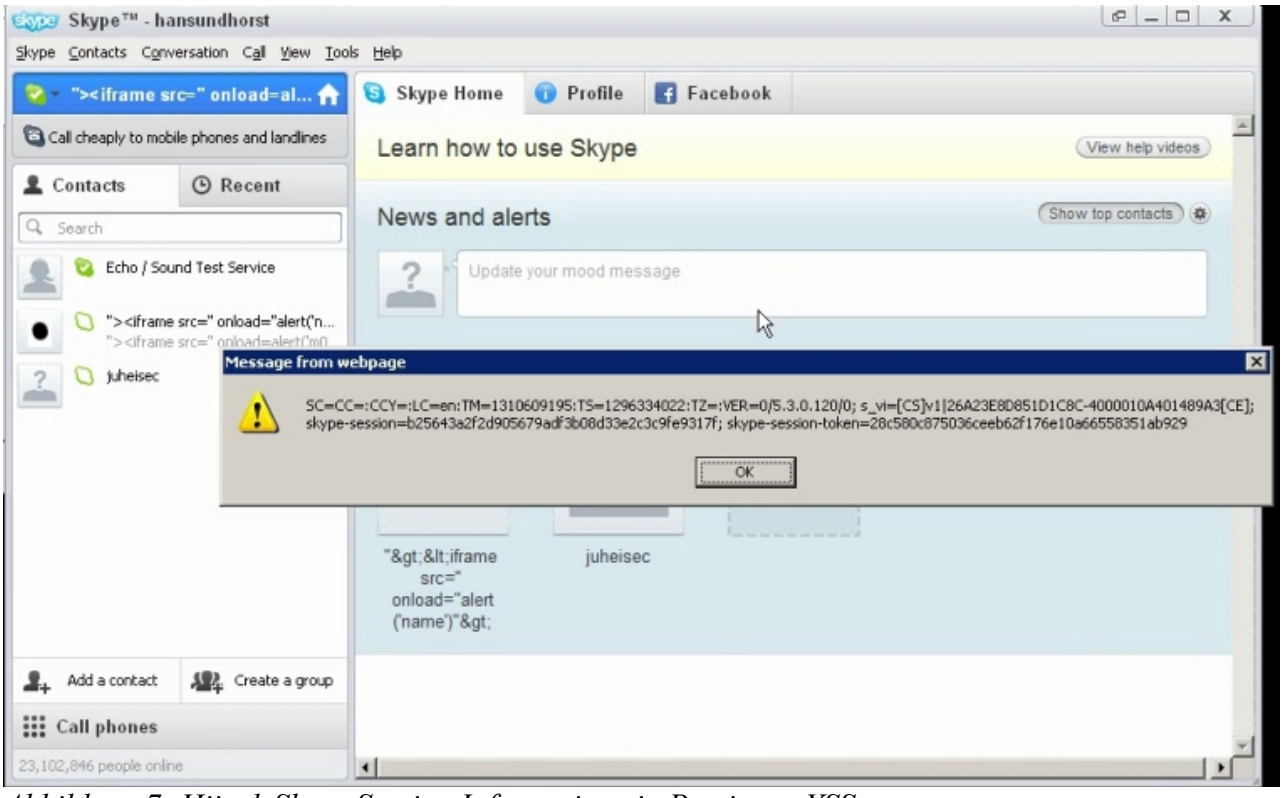


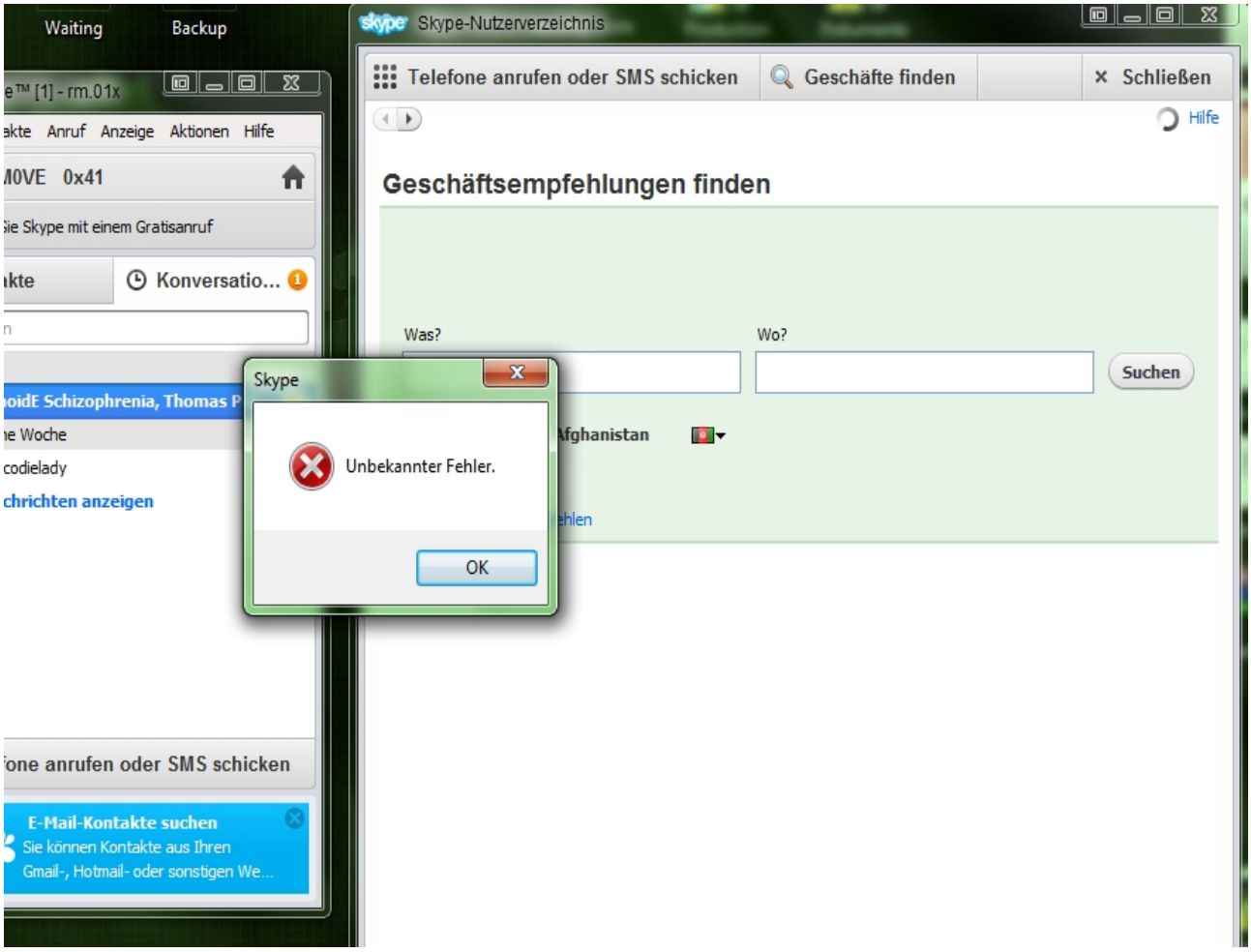
Abbildung 7: Hijack Skype Session Information via Persistent XSS

The vulnerability allows an remote attacker to implement malicious persistent script code over an input field (myphone) on the user profile settings. The successfully exploitation of the vulnerability allows an attacker to hijack customer sessions or can lead to malicious persistent script code execution over the review display listing of active users.

Schema: Install => Login => Change profile my-phone input (Script Code HTML/JS) => Save Input => Redisplay in active user preview (Exploitation)

PoC: "><iframe src=" onload=alert('mphone')>

Skype 5.2.x & 5.3.x – Pointer Vulnerability (Memory Corruption) (Map: 4.2.4)
After we found a non restricted input field on the directory request we have done some input tests. Benjamin K.M. Detected critical pointer vulnerability which is vulnerable to the Mac OSx & Windows version of Skype. The bug/vulnerability is located in 2 input forms of a unicode http search request to the Skype search directory server. The vulnerability allows an local attacker to crash the complete Skype process via an unknown unhandled software exception(memory-corruption).



After this unknown unhandled exception we tried to include different large inputs with chars like %, 0, A+

A short while later we got the following results on Mac OS version of the Skype client ...

Skype wurde unerwartet beendet.

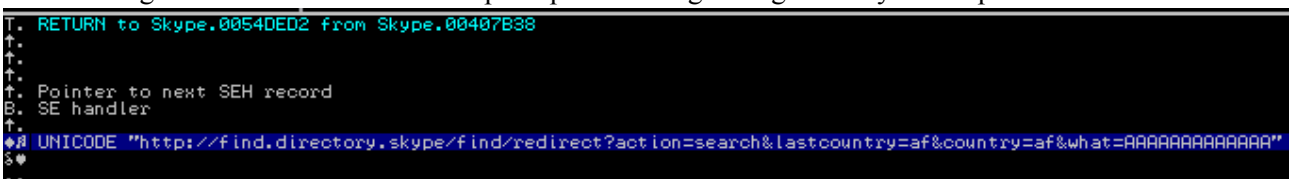
Klicken Sie auf „An Apple senden“, um den Bericht an Apple zu senden. Diese Informationen werden ano

▶ Kommentare

Problemdetails und Systemkonfiguration

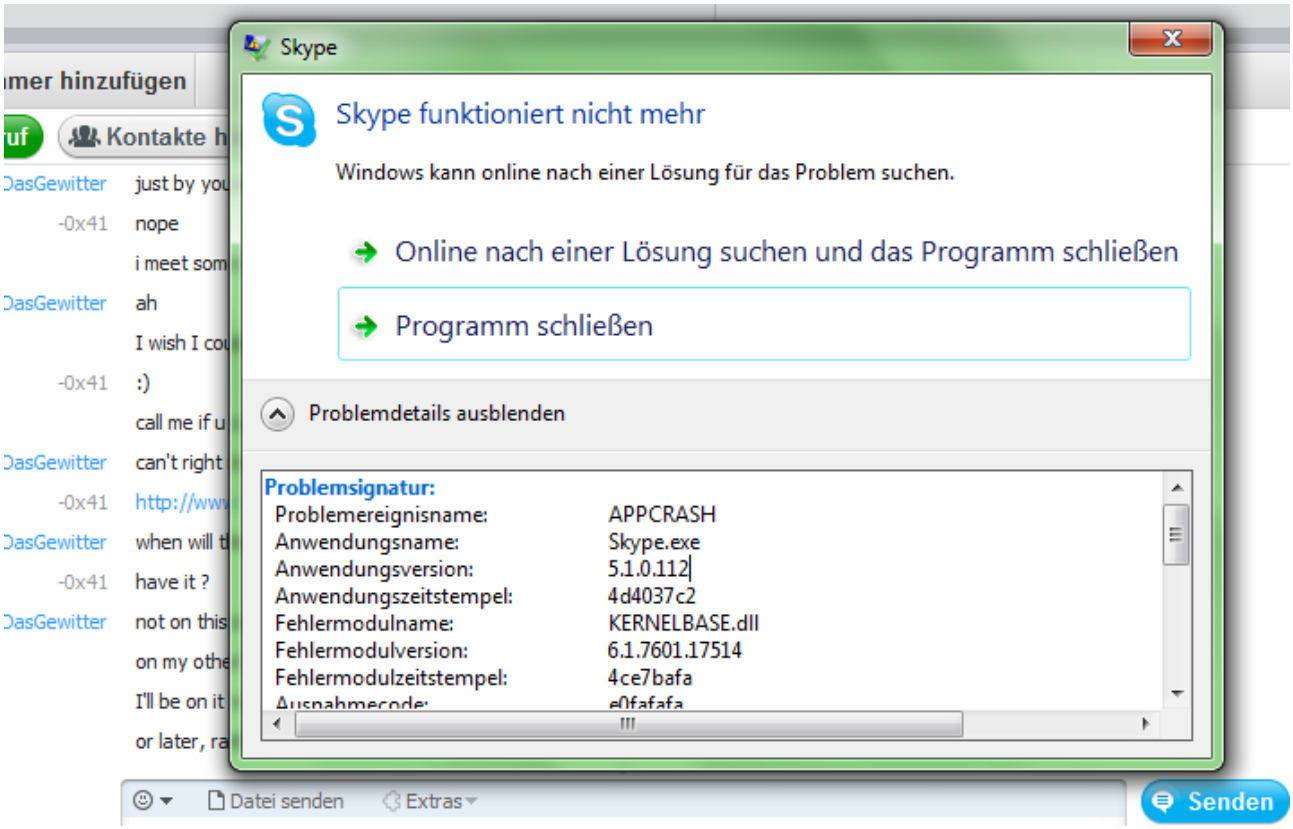
```
Process:          Skype [61833]
Path:             /Applications/Skype.app/Contents/MacOS/Skype
Identifier:       com.skype.skype
Version:          2.8.0.851 (2.8.0.851)
Code Type:        X86 (Native)
Parent Process:   launchd [166]

Date/Time:        2010-10-04 21:31:44.063 +0200
OS Version:       Mac OS X 10.6.4 (10F569)
Report Version:   6

Interval Since Last Report:          2972437 sec
Crashes Since Last Report:           1117
Per-App Interval Since Last Report:  2630855 sec
Per-App Crashes Since Last Report:   2
Anonymous UUID:                      2DCE5869-9F5B-46AD-950A-2295F0CBFC3A

Exception Type:  EXC_CRASH (SIGABRT)
Exception Codes: 0x0000000000000000, 0x0000000000000000
Crashed Thread:  6

Application Specific Information:
abort() called

Thread 0:  Dispatch queue: com.apple.main-thread
0   libSystem.B.dylib               0x95747ef6 __kill + 10
1   libSystem.B.dylib               0x95747ee8 kill$UNIX2003 + 32
2   libSystem.B.dylib               0x957da62d raise + 26
3   libSystem.B.dylib               0x957f06e4 abort + 93
4   com.skype.skype                 0x0023237f 0x1000 + 2298751
5   ???                             0xffffffff 0 + 4294967295
6   com.apple.CoreFoundation        0x93249faf __CFRunLoopRun + 2079
7   com.apple.CoreFoundation        0x93249094 CFRunLoopRunSpecific + 452
8   com.apple.CoreFoundation        0x93248ec1 CFRunLoopRunInMode + 97
9   com.apple.HIToolbox             0x97db6f9c RunCurrentEventLoopInMode + 392
10  com.apple.HIToolbox             0x97db6d51 ReceiveNextEventCommon + 354
11  com.apple.HIToolbox             0x97db6bd6 BlockUntilNextEventMatchingListI
12  com.apple.AppKit                0x9477ea89 _DPSNextEvent + 847
13  com.apple.AppKit                0x9477e2ca -[NSApplication nextEventMatchin
14  com.apple.AppKit                0x9474055b -[NSApplication run] + 821
15  com.skype.skype                 0x0021c5fb 0x1000 + 2209275
16  com.skype.skype                 0x0021c81b 0x1000 + 2209819
17  com.skype.skype                 0x00002cab 0x1000 + 7339
18  com.skype.skype                 0x00002bd9 0x1000 + 7129

Thread 1:  Dispatch queue: com.apple.libdispatch-manager
0   libSystem.B.dylib               0x9570d942 kevent + 10
1   libSystem.B.dylib               0x9570e05c _dispatch_mgr_invoke + 215
2   libSystem.B.dylib               0x9570d519 _dispatch_queue_invoke + 163
3   libSystem.B.dylib               0x9570d2be _dispatch_worker_thread2 + 240
4   libSystem.B.dylib               0x9570cd41 _pthread_wqthread + 390
5   libSystem.B.dylib               0x9570cb86 start_wqthread + 30

Thread 2:
0   libSystem.B.dylib               0x95715066 __semwait_signal + 10
1   libSystem.B.dylib               0x95740c64 nanosleep$UNIX2003 + 188
```

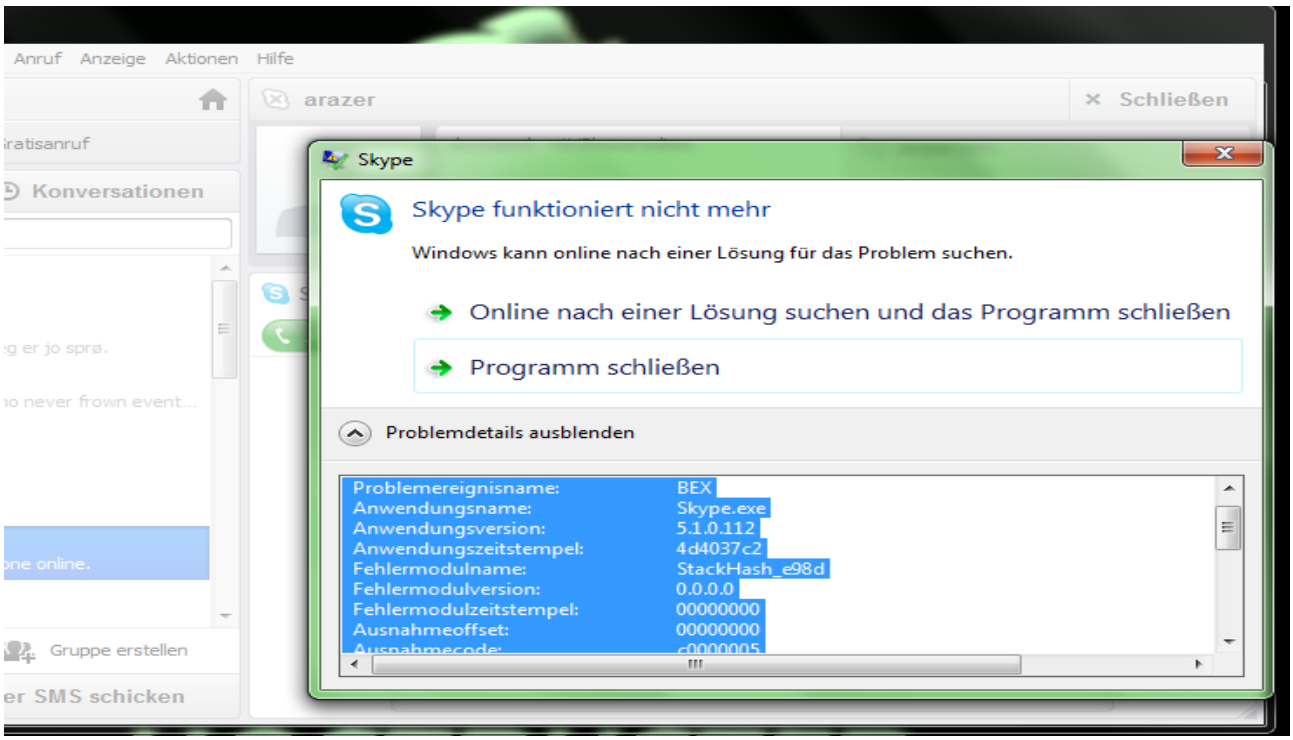
After a short look at the debugger we saw that the software was requesting the search directory with our large string inside. The content was too big to process & the client crashes after the input of the large unicode search string. When the user tries to dump the process the growing memory on the process freeze.



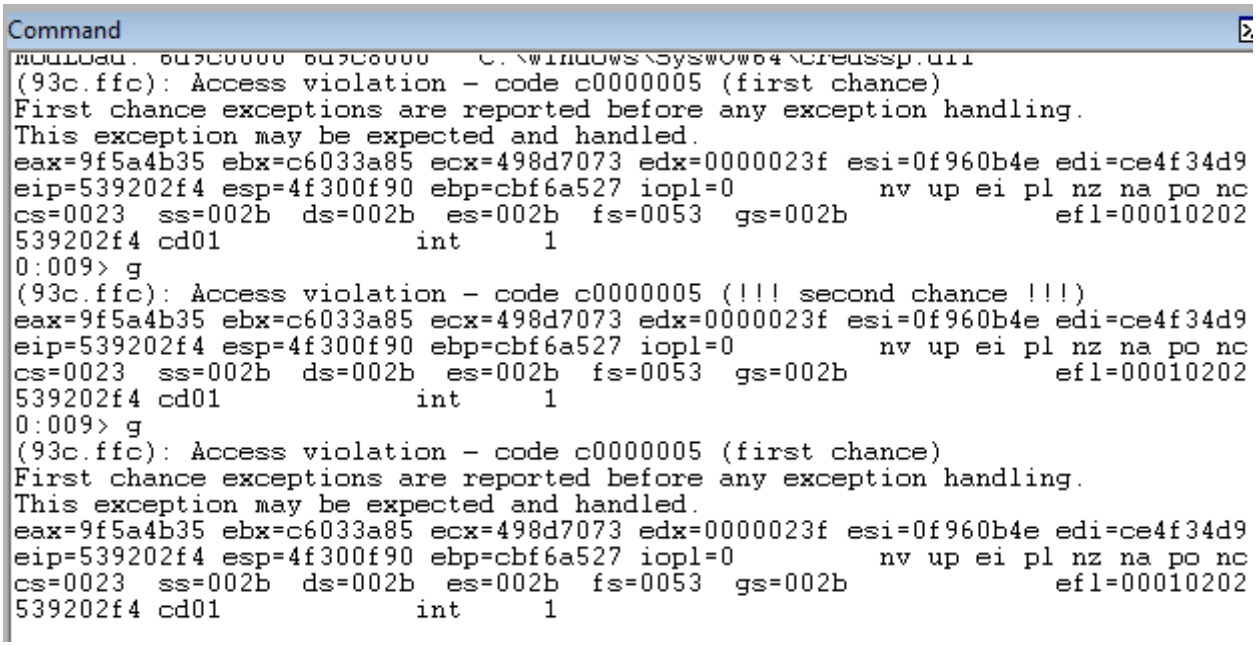
Schema: Install => Login => Open "Geschäftsempfehlungen" => Insert large uni-code string => Requesting Find Directory Server => AppCrash (Memory Corruption)

Skype 5.3.x – Transfer Buffer Overflow Vulnerability (Map:4.2.5)
Next, I got busy testing transfers of data in different system modes. This means, for instance, someone sending a file to someone else while your status is set to Away. Then the files can't be sent or received, and are stuck in the query. Then the other user sets their status to Away. The data query is stopped between the messengers. When the user has sent a file and it goes into stand-by, upon awakening, the result is a remote buffer overflow on 32 and 64 bit versions of Windows 7.





We also recorded some pictures of the access violations around the local buffer overflow & BEX exceptions.



Schema: Install => Login => Startup Data Transfer => Unavailable Mode => Stand-by Mode => Re-activate => Buffer Overflow (BEX)

Skype does not allow to send transfer of files when the status is unavailable or busy. When the conversation is started & a user transfers a file & the receiver acknowledges the process ... He can switch when processing in another mode. By switching to the stand-by mode with your box the running transfer got held on the query with no reaction. The successful exploitation may result in a remote buffer overflow on both client sides.

6. Skype Security, Repsonse/Patch & Cooperation

First we want to talk about the respone/patch time-line and feedback that we got from the Skype security team. After our third submission we got forwarded to a security developer of the Skype. He was responsible for testing and verifying the bugs that we found. The process is described below to get a general idea of how things went.

Information Disclosure > Feedback/Response > Verification > Reproduce > Fix/Patch

To get an idea on how fast the whole proces was the general time line of the persistant software bug with a rating of high is shown below.

2011-07-10: Vendor Notification
2011-07-11: Vendor Response/Feedback
2011-07-12: Vendor Fix/Patch
2011-07-15: Public or Non-Public Disclosure

This secon time-line shows the time-frame in which the persistent software vulnerability (OS idependent) with a rating of high was fixed.

2011-07-22: Vendor Notification
2011-07-23: Vendor Response/Feedback
2011-08-01: Vendor Fix/Patch
2011-09-06: Public or Non-Public Disclosure

As you can see from the time-lines Skype has a very dedicated and fast working security team. They are most of the time able to reproduce bugs in a short timespan and often they can confirm the security issues within hours. When we submitted the first bug we got a fast response by the security

team which had been able to recreate the bug within minutes. As pointed out earlier after third bug submission we got some help by a security developer of the skype team. He helped us on verification & confirmed 4 of our found (6) (detected) vulnerabilities with a short time-span.

After our submissions arrived skype security responded:

Skype v5.3.x v2.2.x v5.2.x – Local 2 Remote Denial of Service Vulnerability
This will be fixed in future releases.

Skype 2.8.x, 5.3.x & 2.2.x – Persistent Software Vulnerability
We will ensure the trusted admin attack vector is fixed.

Skype 2.8.x & 5.3.x - Persistent Profile XSS Vulnerability
Reported and fixed.

Skype 5.2.x & 5.3.x – Pointer Vulnerability (Memory Corruption)
This will also be fixed in a future version.

Skype 5.3.x – Transfer Buffer Overflow Vulnerability
Skype has repeatedly failed to reproduce this issue based on the steps provided.

7. (Review) Security Session Videos

- 7.1 Skype (VoIP) - Denial of Service Vulnerability.wmv **[HD]**
- 7.2 Skype (VoIP) - Persistent Cross Site Scripting Vulnerability.wmv **[HD]**
- 7.3 Skype (VoIP) - [Pointer Bug] Memory Corruption.wmv **[HD]**

8. Credits & Laboratory

The new Vulnerability-Lab is now live! We are certainly excited about this project and have much to work toward. The Vulnerability-Lab works very hard in bringing Europe and the world a great amount of information regarding vulnerabilities and urgent security advisories. If you are a vendor, Vulnerability-Labs can be an extremely valuable resource for information in detail about the current state of security for your software. Vulnerability-Lab is a research team that finds vulnerabilities, security holes, and bad security practices in software and applications, bringing this information to one site where vendors may be notified in a professional and timely manner.

The Vulnerability-Lab is comprised currently of eleven members who range from experts in the field of Information Security to managers of information and site content. All of these members, however, are greatly interested in security and is their primary concern. The research team releases, on average, 25-40 vulnerabilities a month, ranging from important to critical. The process of releasing vulnerabilities and advisories is always generally followed in a professional manner. Sensitive Information is censored and any contribution from third parties that may include or seem to encourage malicious or stolen content, or personal/group agendas is strictly forbidden. More information about this can be found in the FAQ. Not only does the Vulnerability-Lab provide advisories for software, but it also allows the customisation of these advisories down to particular vendors, types of vulnerabilities, dates, and even informative videos. If your goal is to only be notified of security holes in your software and to work with the researchers to have it patched, then this option is naturally available. However, collaboration amongst our team and with other teams and vendors is a priority, as education and knowledge always lie in the forefront.

Read our blog or join our forum if you would simply like to read more and keep up with the fast-paced world of information security and what is going on in our labs! Vulnerability-Lab is committed to bringing vulnerabilities to light and collaborating with researchers for the betterment of software and application security. If you are a member of a research team and would like to work with Vulnerability-Lab, send us an E-Mail including who you are and what you are interested in contributing. We also need sponsors! If you are a vendor or research team that would like to employ our services, we would be more than happy to oblige. Donations are, of course, also always welcome. This is a very dedicated and talented team of researchers and workers. Investing in the Vulnerability-Lab will help nurture both the security of your software as a vendor and also status of application and software security world-wide. Please contact us if you are interested sponsoring, benefits and internet prevention system projects for customers.

Domains: www.vulnerability-lab.com , www.vuln-lab.com or www.vuln-db.com

Contact: admin@vulnerability-lab.com
Support: support@vulnerability-lab.com
Research: research@vulnerability-lab.com
Submit Advisories: submit@vulnerability-lab.com