Part 1: Building ShrewSoft VPN Client with public source package from developer in macOS 10.11/10.12/10.13 (en)

written 5.10.2017 against macOS versioning 10.12.6 and 10.13.[0]

Breaking news October, 8th 2017: ready-to-run Installer-package of ShrewSoft VPN Client for macOS is ready - without depencies of Qt4 and OpenSSL, all of these now transfered into a private framework. I made a rework with the source code - not the programm code itself - only the files for control the compiling process with cmake (aka CMakeLists.txt) for correct compiling and linking, I've found that the work of the original developer was not bug-free for preperation with cmake and compiling in macOS - so I made corrections with sucess. In a few days you will be found here a download-link with instructions. Be patience please, I need to do all ready for publishing. I will publish the changes in the sources too with comments, what was wrong and why.

Breaking news No.2 October, 12th 2017: I detect some strange things. I compile all with Xcode9 with result, that the two GUI-Apps only work with macOS 10.12.6 as minimum or 10.13 . That means, the usability is more limited as I wanted. So I reactivate my old iMac build in 2007, it sleeps a long time and now I took him out of sleep, because the last macOS it were working on it was 10.11 El Capitain. There I used Xcode8 and repeat the whole process with compiling of ShrewSoft's VPN Client with the result - now I have a distribution package of VPN Client it work from 10.11 up to 10.13. That's better, not all user will be using the newest macOS because different reasons, especially in business IT environments. For the second thing I had success with OpenLDAP, so I can compile a VPN Client with full full functionality, because the XAUTH-feature requires OpenLDAP linked with ShrewSoft VPN Client. So, now I'm ready make a ready-to-install package public. I have arrived at the finish line.

For a long time it was possible to install the ShrewSoft VPN Client with homebrew itself:

```
$ brew tap homebrew/boneyard
$ brew install shrewsoft-vpn-client
```

For some time **this is current no longer possible**, because homebrew/boneyard is discontinued/deprecated and no longer available. Look here https://github.com/Homebrew/homebrew-boneyard for more information. So we don't get the shrewsoft-vpn-client from homebrew anymore. It's over now (at this moment). But we do not need homebrew for it - the Shrew Soft VPN client itself - anymore, **but** we need homebrew for *additional things* like OpenSSL, Qt4 and OpenLDAP around compiling the Shrew Soft VPN client. It makes the whole process much simpler and faster.
I was also never really satisfied with the homebrew version of shrewsoft's vpn client. I had problems with the screen display of the two GUI applications, it looked like problems with fonts. In addition, the compiling process with homebrew had never really gone flawless.

What do we do now ? Looking for something different ? Give up? Never. We build it from the original sources - as simple as that. Like to good old Linux times we have the source code and we make something with it.

In this KB-article I will describe, how it's possible to **compile yourself a working Shrew Soft VPN Client with the macOS** Sierra (10.12) and macOS High Sierra (10.13) based on official source package.

~~At the moment nowhere is a ready-to-run package (.dmg/.pkg/.mpkg) available, which works with macOS 10.12 or 10.13~~.

Meanwhile, there is a "ready-to-use" alternative (compiled with the steps in this documentation and provided from me) instead of build the Shrew Soft VPN client by yourself from source code:

Here you can download the compiled ready-to-run package, based of original sources in addition of this documentation, for free-to-use:

https://www.amft-it.de/dist/macos/ShrewSoftVPNClient-2.1.1_macOS_Oct2017.pkg

checksums:
SHA-256: fa01613fdb72c54bcd27f000a82f78897457ad7edbab1a9d7323d9fcc5c4b2ae
MD5: 3d073c92398bd3a73c6821041e3ae40d

Please check the download against manipulation in a terminal/shell:

```
$ cd /Users/<your_login_name>/Downloads
$ shasum -a 256 ShrewSoftVPNClient-2.1.1_macOS_Oct2017.pkg
fa01613fdb72c54bcd27f000a82f78897457ad7edbab1a9d7323d9fcc5c4b2ae
ShrewSoftVPNClient-2.1.1_macOS_Oct2017.pkg
$ md5 ShrewSoftVPNClient-2.1.1_macOS_Oct2017.pkg
MD5 (ShrewSoftVPNClient-2.1.1_macOS_Oct2017.pkg) =
3d073c92398bd3a73c6821041e3ae40d
$
```

Compare the checksums for safety. This package has additional XAUTH-support, because I compiled the package sucessfully with OpenLDAP. The package need macOS 10.11 as minimum, but run with 10.12 and 10.13 too. More information you will be found on my German download page . I'm sorry, but this download page is in German language only - at the moment I haven't enough time for translation.

This documentation is to understand as "as-it-is", no additional support. I repeat this 3 times with one MBP 10.12, one iMac 10.12 and one MBP 10.13, so I can confirm that the following things are working.

Attention ! For this way you need basic knowledge of UNIX shell things, working with terminal/shell, using shell commands and you know how edit files from shell. It's not for the GUI of macOS. It's for advanced users - not for novices or people they do this never before. If your knowledge only on level "novice", please stop here and get additional help from people they have experiences for such things.

I can not test this with older macOS operating systems - I have only 10.11.6 (iMac late 2007 Core 2 Duo), 10.12.6 (iMac Mid 2013 i5 + MBP 15" Retina Mid 2015 i7) and the fresh 10.13 (MBP Mid 2012 i7) for testing - correct working at all systems I listed here.

0. About me

I have an own IT company in Germany called Amft IT https://www.amft-it.de ,which offers different services in the field of enterprise networks in multi-OS-environments. I'm from the UNIX environment at my education, but today I work with different operating systems, depending on what my customer need. Since the beginning of the 90s I have many experiences especially in the Linux environment in dealing with software, which is only available as source code. Previously on Linux I had to compile everything myself, there were only a few ready-made packages. Today I use primarily the macOS for it. The shrewsoft vpn client is one of my the most important tool for me because I need to exchange the vpn-connection-files between WIN and macOS. I need this strongly for my work - all my customers have IPSec-based business routers for remote access. I checked many other vpn clients for macOS - free and commercial - but none of those fulfilled my expectations. In the meantime, the software is no longer available directly via homebrew, so I had to find another solution. So I did not spend very little time to get this software running on the current macOS 10.12/10.13. I now want to make this knowledge available to other users - the idea behind open source.

Talked enough - so let's begin.

1. Prepare your Mac (optional)

#1 SIP (System Integrity Protection) - for information only

I check all against possible problems with SIP (aka "rootless") activated - which is the default setting on Mac since 10.11, for compiling and running Shrew Soft VPN Client we don't need deactivate SIP. Good news for people which like more system security !

I had compiling and installing all with the SIP option enabled on my iMac with 10.12 - all worked correctly like a charm.

More information about SIP and what that will be do and what not you can read here: https://en.wikipedia.org/wiki/System_Integrity_Protection and here: https://support.apple.com/en-us/HT204899 .

⇒ IMPORTANT: Changes against SIP should be done only by advanced users of macOS, they really know what and why they want to do with this function enable or disable.

#2 Disable Gatekeeper (optional if required)

Start a terminal and type

```
$ sudo spctl –master-disable
```

hit enter and confirm with your password.This *reactivate* the option in **System Preferences > Security & Privacy > Allow apps downloaded from: Anywhere**. This option was no more shown since 10.12, but now you can see this option and select if needed. The background is to load two kernel extensions named TUNTAP, a virtual network interface which we need for shrewsoft vpn client as a third party software. If those couldn't be loaded, you may try to change the option. **I will show only the way how it works if needed.** You will get information about the two kernel extensions here: http://tuntaposx.sourceforge.net . If they are correct signed, macOS with loaded those extensions with the option **App Store and identified developers** without problems**.**

You need to check this by yourself - I don't know the status of your Mac security settings.

2. Prerequisites for compiling shrew-sources on macOS

#1 Install Xcode from MacAppStore

The easiest way is to install Xcode from MacAppStore. After download please start Xcode, accept EULA, it will be do additional things.

An other way ist to install only the command line tools :

```
$ xcode-select –install
```

I use the complete package Xcode, version 9, from the MacAppStore for this documentation. In addition I use an older version of Xcode - 8.2.1 - with my old iMac from 2007 with macOS 10.11.6 successful too. If you use Xcode9, the binary compatibility is only from starting macOS 10.12.6 or higher, macOS lower than 10.12.6 don't work with the compiled Xcode9-binaries of Shrew Soft VPN client. If you want to use it only with your current running macOS on Mac you compile all, ignore this - it's only important for transport the binaries to other, older systems.

#2 Using Homebrew for additional things we need

For fast and simple installing requirements for comiling we will use the help of Homebrew https://brew.sh/ , a package manager for macOS which installs ready-to-run open source packages, mostly well-known from the Linux open source world, for using with macOS.

For compiling we need the following environment: OpenSSL, TunTap, Qt4 (NOT Qt5 !). All of this we will get from and with Homebrew. The best way is to start from a clean environment. At first I recommed a full cleanup of all homebrew-packages:

```
brew remove $(brew list)
```

If it not all will be cleaning, try

```
brew remove $(brew list) --force
```

Next, we uninstall homebrew itself:

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/uninstall)"
```

Maybe there things the uninstaller cannot be cleanup, read what the script wrote. This you need to be clean up by hand.

Now reinstall Homebrew

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

#3 Required homebrew packages

TUNTAP:

The tuntap kernel extensions http://tuntaposx.sourceforge.net/ for macOS allow to create virtual network interfaces. The shrewsoft vpn client needs that for working. Other VPN technology like OpenVPN https://openvpn.net/ (SSL-based VPN) use these too with macOS.

```
brew install Caskroom/cask/tuntap
```

As second option you can use the install package from http://downloads.sourceforge.net/tuntaposx/tuntap_20150118.tar.gz too, homebrew uses exact the same.

As next, we need OpenSSL https://www.openssl.org/ which is not part of macOS (or better - no more part, earlier version of macOS had this implemented). At moment, homebrew installed the newest version of openssl-1.0.2, which has longterm support until 31st December 2019.

My tip: Install simply the Midnight Commander, one of my must-have tool for work with console/shell https://en.wikipedia.org/wiki/Midnight_Commander and https://midnight-commander.org/ as *first option*, **which installs OpenSSL too** :

```
brew install mc
```

- mc (Midnight Commander ist a very useful tool and great file manager for the shell with best greetings to the good old Norton Commander earlier if I was using DOS)

The second option is **install only OpenSSL**, if you don't want install the mc (OpenSSL is required for the ShrewSoft VPN Client) :

```
brew install openssl
```

Qt4 (NOT Qt5!) :

```
$ brew tap cartr/qt4
$ brew tap-pin cartr/qt4
$ brew install qt@4
```

At moment we **need to use only the Qt4**-version for compiling - later I will try to migrate the Shrewsoft VPN Client running with the newest version Qt5 because Qt4 is old and has no offical support for newer macOS - Qt5 has it - since 10.11, but it works up to 10.13. We only need the open-source version of Qt, not the commercial Qt.

Now the Qt4 package is called qt@4 (homebrew's `brew install qt` is installing Qt5 now, they change this !). Sierra get a binary package, High Sierra will be compile Qt4 - needs about 30 min for this. More information about Qt4 and homebrew you will get from here:

https://github.com/cartr/homebrew-qt4

cmake https://cmake.org/ we need too, the build process based on cmake :

```
brew install cmake
```

I'm not sure, but I think if you have installed Qt4 and Qt5 together, you may get problems with compiling. I recommend highly only Qt4 installed. Uninstall Qt5 if exists:

```
$ brew uninstall qt
```

#4 Get source packages for shrewsoft-vpn-client

Download for trust and safety the sources from offical developer here: https://www.shrew.net/download/ike/ike-2.2.1-release.tgz , with browsers it will be saved in ~/Downloads . **If you use Safari for this**: Safari automaticly unzipped the source package. So you will get only a ike-2.2.1-release.tar in your download folder. Now we prepare the build process:

```
$ cd
$ mkdir src
$ cd src
$ cp /Users/<your_username>/Downloads/ike-2.2.1-release.tgz .
$ tar xzvf ike-2.2.1-release.tgz
$ cd ike
```

If you have only a tar-file (for Safari-users):

```
$ cd
$ mkdir src
$ cd src
$ cp /Users/<your_username>/Downloads/ike-2.2.1-release.tar .
$ tar xvf ike-2.2.1-release.tar
$ cd ike
```

Now we will have extracted shrewsoft-vpn-client sources in ~/src/ike. ~/ is a placeholder in UNIX-environments and mean <your_home_directory> of active user. In macOS , if the username "heiko" - the homedir will be /Users/heiko . Do a cd ~/ will be the same as cd /Users/heiko.Let's start now the build process…

3. Build and compiling the shrewsoft-vpn-client

First cleanup old things:

```
$ sudo rm -rf /Applications/Shrew*
$ sudo rm -rf /usr/local/opt/shrew*
$ sudo rm -rf /usr/local/bin/ikec
$ sudo rm -rf /usr/local/sbin/iked
```

If you have the beta-Package installed, which ShrewSoft itself published earlier (a few years ago), look in */usr/bin* and */usr/sbin* too. If yes, with the newer macOS you have a Problem. Without disabling SIP you cannot delete anything in */usr/bin* or */usr/sbin* - *sudo* don't work too ! Only without SIP you have write permissions to these paths/folders ! Think about, change SIP need to boot in recovery-mode (cmd + R) and can be changed with csrutil-tool in terminal / shell, important for cleanup old things after upgrades of macOS they located in */usr* - only */usr/local* is usable with SIP enabled !

check in */Library/LaunchDaemons* , if there a .plist file with *shrew*, do the following:

```
$ sudo launchctl unload <filename_with_*shrew*>.plist
$ sudo rm <filename_with_*shrew*>.plist
```

check /usr/local/etc , if there exists a iked.conf , if yes delete it

```
$ sudo rm /usr/local/etc/iked.conf
```

We will use as target for installing the binaries basicly `/usr/local/[bin,sbin]` for `iked` and `ikec`, additional framework components will be located in `/Library/Frameworks`, the daemon `net.shrew.iked.plist` in `/Library/LaunchDaemon` for automatic start with boot.

That's compliant with the actual developer rules from Apple, so I modified the targets from `/usr` to `/usr/local` in the sources, that's also compliant with the Limitation of SIP if it's active an Mac as default.

In the root of the source package you will have a file named **CMakeLists.txt** - we need **add** a few lines because directories for using, compiling and linking OpenSSL, compare it with your **CMakeLists.txt**:

```
project( IKE )

cmake_minimum_required( VERSION 2.4 )

#add next line - for set prefix path to /usr/local
set(CMAKE_INSTALL_PREFIX "/usr/local")

#add next 2 lines - additional dirs for compiler and linker - it depends
where your OpenSSL is installed - check this
include_directories(/usr/local/opt/openssl/include)
link_directories(/usr/local/opt/openssl/lib)

if( COMMAND cmake_policy )

cmake_policy( SET CMP0003 NEW )

endif( COMMAND cmake_policy )
```

a little bit further down:

```
set(
SEARCH_INC
#add next line - it depends where your OpenSSL is installed - check this
/usr/local/opt/openssl/include
/usr/local/include
/usr/include )

set(
SEARCH_LIB
#add next line - it depends where your OpenSSL is installed - check this
/usr/local/opt/openssl/lib
/usr/local/lib
/usr/lib )

set(
SEARCH_BIN
#add next line - it depends where your OpenSSL is installed - check this
/usr/local/opt/openssl/bin
/usr/local/bin
/usr/pkg/bin
/usr/bin )

set(
SEARCH_SYS
#add next line - it depends where your OpenSSL is installed - check this
/usr/local/opt/openssl/share
/usr/local
/usr/share
/usr )
```

we stay in the root of the sources, thats important. Now we can start with cmake:

```
$ cmake -DQTGUI=YES -DNATT=YES -DCMAKE_INSTALL_PREFIX=/usr/local -
DQT_QMAKE_EXECUTABLE=/usr/local/Cellar/qt@4/4.8.7_2/bin/qmake .
```

I decided to use the absolut path to the qmake-binary of Qt4, so we will have no conflicts with possible other things from Qt. You can check and test *as an other option against /usr/local/bin* this too:

```
$ cmake -DQTGUI=YES -DNATT=YES -DCMAKE_INSTALL_PREFIX=/usr/local -
DQT_QMAKE_EXECUTABLE=/usr/local/bin/qmake .
```

Both ways should be worked. Decide by yourself which option you want to use.

Build-Flags are:

-DQTGUI=YES - build the GUI binaries *recommended*

-DNATT=YES - build the NAT-T Support *recommended*

-DQT_QMAKE_EXECUTABLE - set the path to the **qmake** from Qt4 *required* if -DTGUI=YES is set

More information about the build-flags you can get from the README in the root of source. **With LDAP-support (possible with OpenLDAP package installed required) compiling was not successful at moment. I need to check this deeper about reasons. Now - *without LDAP-***

***support* for ShrewSoft VPN client.**

If all ok, it should be completed.

Now we compile and install all:

```
$ make
$ sudo make install
```

The last command requires sudo - because the install process will be write something in /Library/Frameworks , this a system-relevant directory and protected. It is very important, that the make process is starting from the root of the source. Otherwise, the compiling process will be break.

Now we need to do some additional things:

```
$ cd script/macosx/
```

modify the file **net.shrew.iked.plist** from :

```
<array>
 <string>/usr/sbin/iked</string>
 <string>-F</string>
</array>
```

to

```
<array>
 <string>/usr/local/sbin/iked</string>
 <string>-F</string>
</array>
```

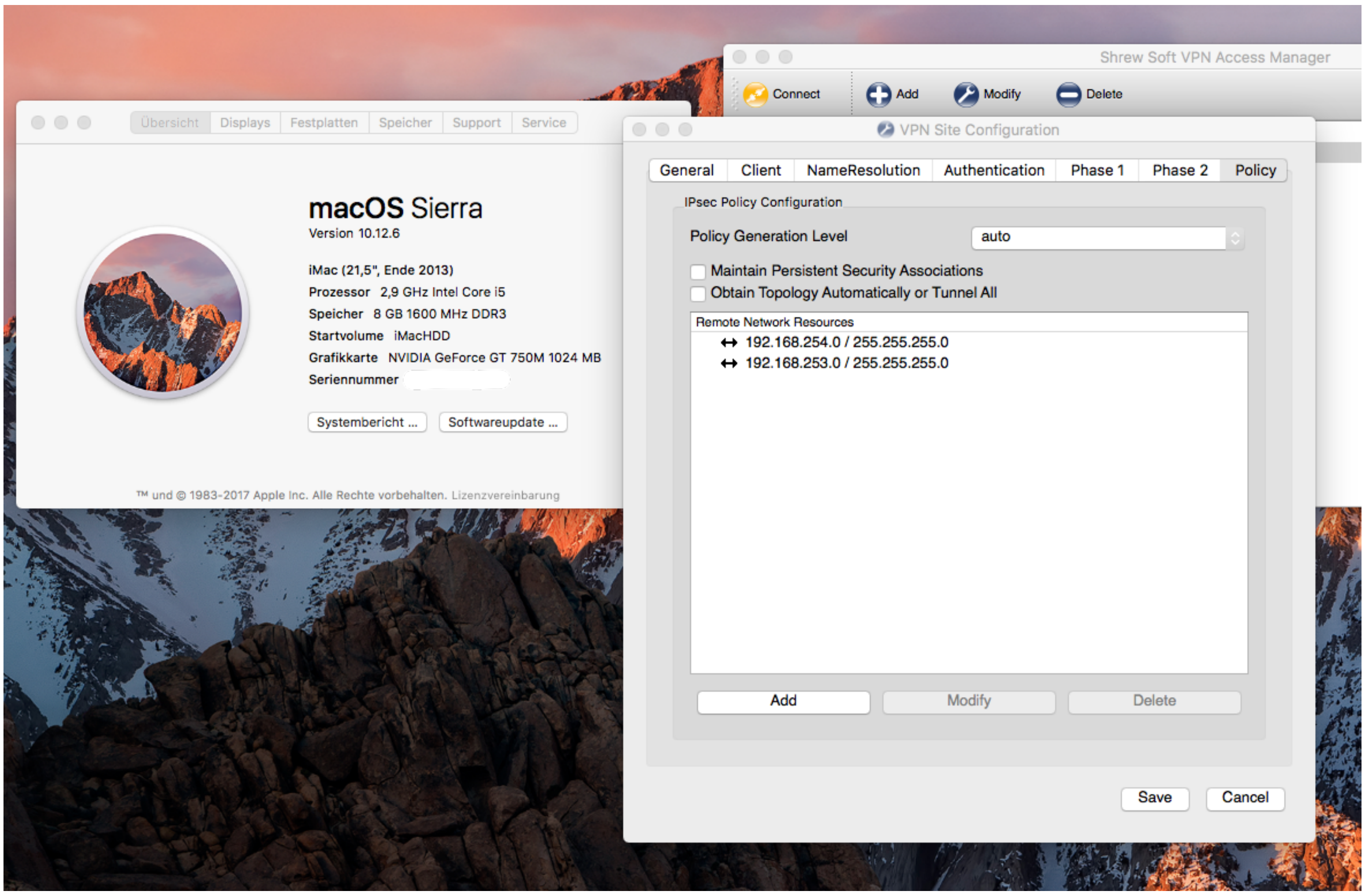
next step:

```
$ sudo cp net.shrew.iked.plist /Library/LaunchDaemons
$ sudo cp /usr/local/etc/iked.conf.sample /usr/local/etc/iked.conf
$ cd /Library/LaunchDaemons
$ sudo launchctl load net.shrew.iked.plist
```

check if iked is running:

```
$ sudo ps x | grep iked
5507 ?? Ss 0:00.02 /usr/local/sbin/iked -F
```

We are ready. You will find the two GUI-Applications in your */Applications* folder, if all is ok it need to be working. You will have now a working ShrewSoft VPN Client.



macOS Sierra
Version 10.12.6

iMac (21,5", Ende 2013)
Prozessor 2,9 GHz Intel Core i5
Speicher 8 GB 1600 MHz DDR3
Startvolume iMacHDD
Grafikkarte NVIDIA GeForce GT 750M 1024 MB
Seriennummer

™ und © 1983-2017 Apple Inc. Alle Rechte vorbehalten. Lizenzvereinbarung

IPsec Policy Configuration

Policy Generation Level: auto

- [ ] Maintain Persistent Security Associations
- [ ] Obtain Topology Automatically or Tunnel All

Remote Network Resources
↔ 192.168.254.0 / 255.255.255.0
↔ 192.168.253.0 / 255.255.255.0

That's all, folks. Happy VPNing…

4. Using ShrewSoft VPN Client

You can start the "VPN Access Manager" from your /Applications folder. It generated a directory *~/.ike* with the subdirs *sites* and *certs* in your home folder. If you have a *.vpn* file, copy this to the *sites* location:

```
$ cp <filename>.vpn .ike/sites
```

It's important, that this file has an **extension with .vpn** - **otherwise the client don't detetct this file**. In the WIN-version you can do this without extension, here with macOS not.

There also a command line tool, ikec . You can use this too for start a VPN connection (maybe for scripting):

```
$ ikec -r "<filename>.vpn" -a
```

In all cases, the *iked* **must be running**. Check this. If not, look at my troubleshooting section.

I can confirm and verfify working ShrewSofts VPN client with the following IPSec-VPN-based router/software:

- Fritzbox from AVM (here in Germany one of most used router for home and small business)
- LANCOM
- Cisco with IPSec-VPN

- pfsense as a software router appliance
- some NETGEAR router with IPSec

If you know your IPSec parameters exactly - every IPSec-based VPN router should be worked with the ShrewSoft VPN client. In most cases you will be worked with PSK (pre-shared-keys aka "password-based", XAUTH as extension works too) - I can confirm (checked with LANCOM) a working IPSec-VPN with PKI-certs too. I used this too with the ShrewSoft VPN client - but you need a correct setup of your PKI infrastructure with certificates. That's a very wide field - and will not be a part of this documentation.

You can use Shrew's .vpn-files get from WINDOWS too. Simply import these files, after import you will find these files saved in your home-folder ~/.ike/sites . Export to WINDOWS is possible too.

5. Troubleshooting and things good to know

You need first to check your firewall settings. It's possible that the firewall is blocking useful ports. For IPSec we need to **open UDP ports 500 and 4500, TCP port 10000 and the ip protocol ESP**. If there's some of these ports/protocol blocking with firewall , IPSec/VPN cannot working. So check this first ! Behind a NAT-router you need (maybe) to activate NAT-T (aka "NAT-transversal") in your VPN-connection-settings. Check this too - it depends strongly from your environment for internet access. Only in some cases the ShrewSoft VPN Client will be the source of connection problems - but I can confirm in public networks (hotel, public hotspots) they are blocking some of the required ports - in this case you haven't a chance for connect with IPSec-VPN - it's not your problem with Mac and/or ShrewSoft VPN Client. Other VPN-technologies like OpenVPN (SSL-based and usable over proxy server) may work in this scenario. Behind a proxy server normally you can't work with IPSec-VPN too. Think and remember this if some of these cases exist !

This documentation is based and translated from my description/thread (only in German language) of this process in a German forum : https://www.apfeltalk.de/community/threads/vpn-file-importieren.514010/ I'm the user "dg2dra" there.

#1 iked is not running

iked needs to be running for using ShrewSoft VPN Client. Test it "by hand" on shell, if it starts not with the launchd:

```
$ sudo /usr/local/sbin/iked
```

It will only start, if a *//usr/local/etc/iked.conf* really exists ! If not, read my docu here, I described this how.

In some cases the is an other small problem. Do this, first close all Shrew-GUI-Apps (Click the App-Window active and close with **cmd + Q**):

```
$ cd /Library/LaunchDaemons
$ sudo launchctl unload net.shrew.iked.plist
$ sudo killall iked
$ sudo rm -fr /var/run/ikedi
$ sudo rm -fr /var/run/vpncontrol.sock
$ sudo launchctl load net.shrew.iked.plist
$ sudo ps x | grep iked
```

Check this, if it could be the problem.

The second problem was identified by macOS own build-in VPN daemon *racoon*. If you use sometime macOSes build-in VPN, the OS starts up to now his own vpn-daemon *racoon* automaticly. Both running at the same time don't work - Shrew's iked says then: "cannot bind socket". I think both daemons use the same ports - which don't work. Only one (`racoon` OR `iked`) daemon can use the ports 500,4500 (both UDP) and 10000 (TCP) at the same time. So try to kill the racoon:

```
$ sudo killall racoon
```

Try re-register the iked as a daemon:

```
$ cd /Library/LaunchDaemons
$ sudo launchctl unload net.shrew.iked.plist
$ sudo launchctl load net.shrew.iked.plist
$ sudo ps x | grep iked
```

check if it now works. Also check if the 2 kernel-extension TUNTAP are loaded:

```
$ kextstat | grep tuntap
```

#2 Compiling against Qt4 will absolutly not work

Strange. In this case you can first test, if the 2 command line tools can be build. Use the *cmake* without-*DQTGUI=YES* and the *-DQT_QMAKE_EXECUTABLE=…* . This will be compile only *iked* and *ikec*, Qt4 is not required in this case.

#3 was working - after system things changed or play around with homebrew not

Strange too. Most likely problems with loading the dynamic link libraries. Mainly changing paths to the libraries. Here I linked some of the shrew-things direct against absolut path like Qt4 and OpenSSL. For instance, if the OpenSSL version updated with homebrew, maybe the old version will be deleted and/or paths changed. **Check the paths, mainly OpenSSL,** and edit or correct the *CMakeLists.txt* with the new paths. Repeat *cmake*, *make* and *sudo make install* after change. Here I need to do some tests with the compiler to stabilize this case. At the moment I had only success with put the absolut paths for the linker, especially the OpenSSL .h header files and dynamic libs. If you want to know, which dynamic libraries a binary/programm is using, try this with the help of `otool` :

```
$ otool -L /usr/local/sbin/iked

/usr/local/sbin/iked:

ShrewSoftIke.framework/Versions/2.2.1/ShrewSoftIke (compatibility version
2.2.1, current version 2.2.1)

ShrewSoftIp.framework/Versions/2.2.1/ShrewSoftIp (compatibility version
2.2.1, current version 2.2.1)
```

```
ShrewSoftPfkey.framework/Versions/2.2.1/ShrewSoftPfkey (compatibility
version 2.2.1, current version 2.2.1)

/usr/local/opt/openssl/lib/libcrypto.1.0.0.dylib (compatibility version
1.0.0, current version 1.0.0)

/usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version
1252.0.0)

ShrewSoftIdb.framework/Versions/2.2.1/ShrewSoftIdb (compatibility version
2.2.1, current version 2.2.1)

ShrewSoftLog.framework/Versions/2.2.1/ShrewSoftLog (compatibility version
2.2.1, current version 2.2.1)

ShrewSoftIth.framework/Versions/2.2.1/ShrewSoftIth (compatibility version
2.2.1, current version 2.2.1)

/usr/lib/libc++.1.dylib (compatibility version 1.0.0, current version
400.9.0)
```

You get as result which dynamic libraries were used from `/usr/local/bin/iked` . Check if all exists in correct locations. Btw, it works with every binary, not only the ShrewSoft VPN Clients.

For the Qt-GUI apps:

```
$ otool -L /Applications/Shrew Soft VPN Client
Connect.app/Contents/MacOS/Shrew Soft VPN Client Connect

/Applications/Shrew Soft VPN Client Connect.app/Contents/MacOS/Shrew Soft
VPN Client Connect:

ShrewSoftIke.framework/Versions/2.2.1/ShrewSoftIke (compatibility version
2.2.1, current version 2.2.1)

ShrewSoftIdb.framework/Versions/2.2.1/ShrewSoftIdb (compatibility version
2.2.1, current version 2.2.1)

ShrewSoftLog.framework/Versions/2.2.1/ShrewSoftLog (compatibility version
2.2.1, current version 2.2.1)

/usr/local/opt/openssl/lib/libcrypto.1.0.0.dylib (compatibility version
1.0.0, current version 1.0.0)

/usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version
1252.0.0)

/usr/local/opt/qt@4/lib/QtGui.framework/Versions/4/QtGui (compatibility
version 4.8.0, current version 4.8.7)

/usr/local/opt/qt@4/lib/QtCore.framework/Versions/4/QtCore (compatibility
version 4.8.0, current version 4.8.7)

ShrewSoftIth.framework/Versions/2.2.1/ShrewSoftIth (compatibility version
2.2.1, current version 2.2.1)

/usr/lib/libc++.1.dylib (compatibility version 1.0.0, current version
400.9.0)
```

Here you will see the additional Qt-libs using the GUI-Apps from Shrew.

6. Future plans

My first idea will be to migrate the source code for compiling against Qt5. I think it's more safer for next OS changes. Which time I will be start - I don't know at the moment. I need to check many things to understand the work of compiler and linker. You will get an update here about this later.

back to Start