Shrew Soft VPN Client V2.1.1 für macOS ab 10.11 - Installationspaket zum Download und zur Installation

1. Installationsvoraussetzungen

Zur korrekten Funktion des ShrewSoft VPN Clients sind zwei Kernel Extension namens TUNTAP zwingend erforderlich. Diese findet man hier: http://tuntaposx.sourceforge.net/download.xhtml bzw. http://downloads.sourceforge.net/tuntaposx/tuntap_20150118.tar.gz .

Diese Erweiterungen haben nichts mit dem Shrew Soft VPN Client selbst zu tun, er nutzt bzw. basiert aber auf deren Möglichkeiten bezüglich der virtuellen Netzwerkdevices zur Erzeugung eines VPN-Tunnels. Im Übrigen ist das unter der WINDOWS-Version des Shrew Soft VPN Client ähnlich, auch dort wird so etwas installiert.

Ich empfehle genau diese Version zu benutzen, da ich den Shrew Soft VPN Client damit getestet habe - diese funktionieren unter 10.11/10.12 und 10.13. Ich stelle hier nur die Links bereit, laden Sie das entsprechende Paket **selbst herunter und installieren es**. Diese TUNTAP Kernel Extensions erzeugen virtuelle Netzwerkinterfaces, die der VPN-Client dann entsprechend nutzen kann. Das gleiche benutzt im Übrigen auch das OpenVPN https://openvpn.net/ bzw. dessen macOS-Ableger Tunnelblick https://tunnelblick.net/ . Ohne diese TUNTAP-Erweiterung kann der Shrew Soft VPN Client nicht funktionieren, bitte beachten Sie das. Der Installer für den Shrew Soft VPN Client prüft das Vorhandensein von TUNTAP und bricht ab, wenn diese nicht installiert sind oder sich nicht an den richtigen Stellen im Dateisystem des Macs befinden - bezogen auf das oben angegebene TUNTAP-Paket. Manuell kann das auch selbst überprüft werden:

```
$ cd /Library/Extensions
$ ls
```

Es müssen eine `tap.kext` und eine `tun.kext` aufgelistet werden. Weiterhin müssen unter `/Library/LaunchDaemons` jeweils eine `net.sf.tuntaposx.tap.plist` und eine `net.sf.tuntaposx.tun.plist` existieren, ob diese auch geladen und aktiv sind, kann man mit

```
$ kextstat
```

überprüfen. Diese müssen dann gelistet sein. Falls jemand den **Cisco Systems Anyconnect VPN Client** bereits benutzt - hier gibts **Konflikte mit TUNTAP** - hier nachzulesen: http://tuntaposx.sourceforge.net/faq.xhtml - hat aber nix mit dem Shrew Soft VPN Client selbst zu tun. Wäre aber u.U. zu beachten.

Desweiteren müssen Sie macOS mindestens in der Version 10.11 (El Capitan) oder höher (Sierra bzw. High Sierra) installiert haben.

Qt4 und **OpenSSL** werden für dieses Installationspaket **nicht mehr extra benötigt** - diese Komponenten sind bereits integriert und Teil der Frameworks, welche automatisch installiert werden (alle bisher veröffentlichten Pakete benötigten das explizit - ich habe das vollständig implementiert).

Ich empfehle dringend, Reste alter ShrewSoft-Installationen vor dem Einsatz meines Installationspakets zu entfernen - um keine Konflikte zwischen altem und neuen Shrew Soft VPN Client herbeizurufen. Da ich natürlich nicht einschätzen kann, was der User vorher schon mit seinem macOS bezüglich Shrew Soft VPN Client ausprobiert bzw. installiert hatte, kann ich hier keine direkte Hilfestellung geben. Anhaltspunkte gäbe es aber - siehe unter Pkt. 4 dieser Anleitung.

2. Download als fertiges Installationspaket

Hier können Sie das fertige Installationspaket, generiert aus den veröffentlichten Original-Sourcen, zur freien Benutzung herunterladen:

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

Checksummen:
SHA-256: fa01613fdb72c54bcd27f000a82f78897457ad7edbab1a9d7323d9fcc5c4b2ae
MD5: 3d073c92398bd3a73c6821041e3ae40d

Bitte überprüfen Sie die *Echtheit und Unversehrtheit des Downloads* zum *Schutz vor möglichen Manipulationen* auf ihrem Mac wie folgt - starten Sie dazu ein **Terminal** unter **Programme > Dienstprogramme**:

```
$ cd /Users/<ihr_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
$
```

Vergleichen Sie die Checksummen auf Übereinstimmung - das entspricht meinem bereitgestellten Paket.

Inhaltlichen Support - bezogen auf die Funktionen der Software - kann ich natürlich keinen geben, schließlich habe ich ja den Programmcode nicht selbst entwickelt. Meine Arbeit bestand lediglich darin, aus dem Quellcode fertige ausführbare Programme zu erzeugen, die unter macOS 10.11/10.12/10.13 lauffähig sind und als installierbares Paket für alle zur Nutzung unter macOS 10.11/10.12/10.13 verfügbar zu machen.

Hinweise zu den Lizenz- und Nutzungsbedingungen:

Shrew Soft Inc. als Entwickler und Eigentümer des Programmcodes stellt den Quellcode selbst öffentlich zur freien Benutzung bereit: "This version is distributed under an OSI approved open source license" - "Diese Version steht unter einer von OSI genehmigten Open-Source-Lizenz". Sehr lobenswert. Sie stellen ihre Arbeit allen zur Verfügung - in diesem Sinne möchte ich das ebenfalls in Anlehnung meiner zusätzlichen Arbeit auch tun - andere Interessierte ebenfalls daran teilhaben lassen.

3. Installation des Shrew Soft VPN Clients

Gehen Sie in den Ordner, wo Sie das Paket gespeichert haben, normalerweise ist das ~/Downloads - also Ihr normaler Downloadordner. Doppelklicken Sie darauf und starten den Installationsprozess, wie Sie das auch von anderen Installationen bereits gewohnt sind.

Die Installation wurde von mir auf 4 verschiedenen Macs getestet, in allen Fällen hat das problemlos funktioniert:

- Macbook Pro unter 10.12.6
- iMac unter 10.11.6 (auf dem habe ich das Paket erzeugt/compiliert)
- Macbook Pro 10.13 mit APFS
- iMac unter 10.12.6

4. Was wird alles genau installiert und wohin ?

Das Installationspaket wurde mit dem Tool "Packages" erzeugt - findet man hier http://s.sudre.free.fr/Software/Packages/about.html , ein feines Helferlein. Compiliert wurde alles auf meinem iMac mit El Capitan 10.11.6, Xcode 8.2.1 als Entwicklungsumgebung - nur so konnte ich auch macOS 10.11 einbeziehen. Mit Xcode9 ab macOS 10.12.6 compiliert liefen die Apps nur noch ab 10.12.6 bzw. 10.13 . Das war mir dann doch zu einschränkend, 10.11 wollte ich schon noch dabei haben.

Das Paket zum Installieren ist ebenfalls um OpenLDAP beim Erzeugen erweitert worden, damit kann man auch XAUTH (dafür ist es erforderlich gewesen) nutzen. Ich habe das erfolgreich testen können. Damit enthält das Installationspaket alle möglichen Funktionen, die der urspüngliche Entwickler vorgesehen hat.

Gern möchte ich darüber Aufschluß geben, was der Installer genau wohin installiert - Transparenz ist immer gut.

```
/usr/local/sbin/iked
/Library/LaunchDaemons/net.shrew.iked.plist
/usr/local/etc/iked.conf
/usr/local/bin/ikec
/usr/local/shrew/lib/libcrypto.1.0.0.dylib
/usr/local/shrew/lib/liblber-2.4.2.10.8.dylib
/usr/local/shrew/lib/libldap-2.4.2.10.8.dylib
/usr/local/shrew/lib/libssl.1.0.0.dylib
/usr/local/share/man/man/man1/ikec.1
/usr/local/share/man/man/man1/qikec.1
/usr/local/share/man/man/man1/qikea.1
/usr/local/share/man/man/man5/iked.conf.5
/usr/local/share/man/man/man8/iked.8
/Library/Frameworks/ShrewSoftIdb.framework
/Library/Frameworks/ShrewSoftIke.framework
/Library/Frameworks/ShrewSoftIp.framework
/Library/Frameworks/ShrewSoftIth.framework
/Library/Frameworks/ShrewSoftLog.framework
```

```
/Library/Frameworks/ShrewSoftPfkey.framework
/Library/Frameworks/ShrewSoftQtCore.framework
/Library/Frameworks/ShrewSoftQtGui.framework
/Applications/Shrew Soft VPN Access Manager.app
/Applications/Shrew Soft VPN Client Connect.app
```

Im letzten Schritt der Installation wird mittels `launchctl load /Library/LaunchDaemons/net.shrew.iked.plist` der `iked`-Daemon gestartet. Damit ist der Shrew Soft VPN Client einsatzbereit.

Wer übrigens eine komfortable Lösung sucht, um den `launchd`, also den "Dienstmanager" unter macOS zu prüfen/steuern/ändern, sollte sich mal die App Lingon X von Peter Borg ansehen, findet man hier: https://www.peterborgapps.com/lingon/ Das nutze ich schon einige Jahre - auch eines meiner must-have-Tools.

Die Frameworks unter `/Library/Frameworks/ShrewSoft*` sind gemeinsame Komponenten, die alle Teilprogramme des Shrew Soft VPN Clients gleichermaßen benutzen. Das gleiche gilt für die dynamischen Bibliotheken (unter WINDOWS wären das quasi .DLLs) unter `/usr/local/shrew/lib` . Meine Entscheidung, das nach `/usr/local/shrew` zu legen, ist darin begründet, um keine Konflikte mit anderer installierter Software zu provozieren die die gleichen Kompontenten verwendet, falls jemand schon OpenSSL oder OpenLDAP installiert haben sollte - z.B. per Homebrew.

Das alles ist konform mit Apples Entwicklerrichtlinien und erzeugt keine Probleme bei unter normalen Bedingungen aktiviertem SIP (System Integration Protection). Bei älteren veröffentlichten Versionen war das eben nicht so - ich habe das diesbezüglich alles korrigiert, wir wollen ja den VPN Client auch unter 10.12 und 10.13 nutzen.

Im normalen Programmordner startet man die App "Shrew Soft VPN Access Manager", die für einen Connect dann die App "Shrew Soft VPN Client Connect" aufruft, welche dann wiederrum den Tunnel-Status anzeigt.

Folgende Zusatzkomponenten wurden für den Shrew Soft VPN Client verwendet, der hier als fertige Package zum Download bereitgestellt wird:

- **OpenSSL 1.0.2l** (long-term Support bis 31.12.2019) - wird dynamisch gelinkt gegen `/usr/local/shrew/lib`
- **Qt4 4.8.7** - wird dynamisch gelinkt als Framework unter `/Library/Frameworks/ShrewQt*`
- **OpenLDAP 2.4.45** - wird dynamisch gelinkt gegen `/usr/local/shrew/lib`
- **cmake 3.9.4** (Hilfsprogramm zum Compilieren des Quellcodes)

Wer immer schon gern wissen wollte, welche dynamischen Bibliotheken einzelne Apps verwenden, hier der Befehl zum anzeigen für den `iked` als Beispiel, geht aber mit jeder App:

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

Die Ausgabe zeigt dann alle vom `iked` benutzen dynamischen Bibliotheken an (dyn libs genannt).

```
iMacAmftIT:~ heiko$ 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/shrew/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
1238.60.2)
    /usr/local/shrew/lib/libldap-2.4.2.10.8.dylib (compatibility version
13.0.0, current version 13.8.0)
    /usr/local/shrew/lib/liblber-2.4.2.10.8.dylib (compatibility version
13.0.0, current version 13.8.0)
    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
307.5.0)
iMacAmftIT:~ heiko$
```

Ein uninstall-Script stelle ich hier in Kürze ebenfalls zur Verfügung.

5. Troubleshooting und mögliche Probleme

►Möglicherweise kann es Schwierigkeiten mit dem Shrew Soft VPN Client geben, wenn man wenigstens 1x das eingebaute VPN des macOS verwendet hat. In diesem Fall scheint wohl macOS den `racoon` mittels `launchd` automatisch beim Systemstart zu starten, was wohl dazu führt, das die erforderlichen Ports **500 und 4500 (UDP) und 10000 (TCP)** dann vom `racoon` benutzt werden und der `iked` des Shrew Soft VPN Clients diese *nicht mehr binden kann* - das ist genauso als ob man versucht, zwei verschiedene Webserver auf dem gleichen Port (z.B. 80 oder 443) zu starten. Das kann natürlich nicht funktionieren. Im Protokoll sieht man dann sowas wie "cannot bind socket" - die Ports werden also schon benutzt bzw. sind belegt. Im Prinzip haben der `racoon` (Teil des macOS) und der `iked` (Teil vom Shrew Soft VPN Client) identische Aufgaben - aber nur einer von beiden kann aktiv sein.

In diesem Fall mal folgendes testen:

```
$ sudo killall racoon
$ sudo launchctl unload /Library/LaunchDaemons/net.shrew.iked.plist
$ sudo launchctl load /Library/LaunchDaemons/net.shrew.iked.plist
```

testen. Alternativ ginge auch mal folgendes:

```
$ sudo killall racoon
$ sudo launchctl unload /Library/LaunchDaemons/net.shrew.iked.plist
$ sudo /usr/local/sbin/iked
```

Das startet den `iked` direkt in der Shell, also interaktiv ohne den `launchd`, prüfen was für Ausgaben

auf der Konsole/Shell angezeigt werden. Meldungen, die sonst nur in die Protokolldateien geschreiben werden, sieht man jetzt direkt auf der Shell. Das hilft Probleme schneller zu erkennen und einzugrenzen.

Hat man das eingebaute VPN des macOS *niemals* benutzt (wie ich beispielsweise) kommt dieses Problem mit dem `racoon` *nicht* zum Tragen.

►Weiterhin kann bei **aktiver Blockierung der Ports 500 (UDP), 4500 (UDP) und 10000 (TCP) durch den Firewall** ebenfalls kein Tunnel aufgebaut werden. Diese Ports werden für VPN/IPSec benötigt, da muss Datenverkehr erlaubt werden. Hier ist also das Setting des Firewalls beim macOS entsprechend anzupassen.

6. Getestete IPSec-Router und Appliances

Mit der intensiven Benutzung des Shrew Soft VPN Client hatte ich nach einigen Tests mit verschiedenen anderen Clients unter WINDOWS - aktuell war da WINDOWS XP  - begonnen, da ich eine günstige Lösung brauchte, um auf die in meinem IT-Kundenumfeld häufig eingesetzten Router der Fa. LANCOM https://www.lancom-systems.de/ per IPSec zugreifen zu können. Das ist nun schon einige Jahre her - damals machte der LANCOM-eigene VPN-Client (der auch nur eine gebrandete Version des NCP-Clients darstellt) unter XP noch hier und da Probleme - das konnte ich nicht gebrauchen. So bin ich dann also die vielen Jahre beim Shrew Soft VPN Client geblieben, meine VPN-Connection-Basis umfasst inzwischen über 200 Connections. Ich nutze den unter sowohl unter WINDOWS 10 Pro und eben auf dem Mac.

Folgende IPSec-Geräte konnte ich bis jetzt erfolgreich testen, mit denen der Shrew Soft VPN Client erfolgreich VPN machen kann - wobei unter Kenntnis der IPSec-VPN-Parameter prinzipiell alles gehen sollte, was IPSec kann :

- LANCOM Router
- AVM's Fritzboxen
- pfSense (Router-Appliance auf Softwarebasis (FreeBSD) meist als VM) samt XAUTH
- diverse CISCOs mit IPSec
- einige NETGEARs mit IPSec (genaue Modellbezeichnungen weiss ich jetzt leider nicht mehr)

Meist setzte ich dabei auf AES bzw. 3DES mit entweder 128bit oder 256bit und optional PFS und/oder NAT-T, je nachdem was der VPN-Router ermöglicht bzw. was die Infrastruktur ermöglicht oder besser erfordert. Häufig kommt dabei PSK (Pre Shared Keys = Passwort basiert) zum Einsatz, kann aber ebenso erfolgreiche Benutzung zertifikatsbasierenden VPN mit dem Shrew Soft VPN Client bestätigen - das konnte ich allerdings nur mit LANCOM-Routern bisher testen, funktioniert auch. Ich habe nur beim LANCOM ein kleines Problem oder besser Limitierung: Meine PKIs haben eigentlich ein RootCA UND ein IntermediateCA, wobei das IntermediateCA die UserCerts signiert. Diese Kette aus RootCA>IntermediateCA>UserCert kann der LANCOM so nicht, Supportrückfrage dort hatte das leider bestätigt. Geht leider nur RootCA>UserCert, d.h. das RootCA muss das UserCert signieren. Das ging aber korrekt und läuft derzeit bei einigen meiner Kunden völlig stabil. Hier sollte aber LANCOM nochmal nacharbeiten - mir gefällt das so nicht.

Ebenfalls funktioniert die dynamische IP-Zuweisung des VPN-Tunnels durch den Router korrekt - an dieser Stelle war die andere VPN-Lösung IPSecuritas http://www.lobotomo.com/products/IPSecuritas/ aus dem Rennen raus - nie ist mir das dort gelungen, die Tunnel-IP dynamisch zu beziehen, auch

Multi-Subnet-Routing ging nicht (entgegen deren Dokumentation). Diese setzt ja auf dem `racoon` auf, dem macOS-builtin-VPN. Das war also keine Option, ausserdem gefällt mir das Connection-Managment von IPSecuritas überhaupt nicht. Die Limits liegen aber wohl eher am `racoon` - denn IPSecuritas ist mehr oder weniger nur eine GUI für den `racoon`. Ich weiss - macOS hat einen VPN-IPSec-Client eingebaut, der aber definitiv nicht bug-frei ist und in gewissen Umgebungen nicht korrekt funktioniert. Der ist also auch keine echte Alternative, für sowas habe ich einfach keine Zeit zum Testen, bis *das mal irgendwann gehen könnte*. Wo ich dann also wieder beim Shrew Soft VPN Client bin.

Auch funktioniert das Multi-Subnet-Routing mit dem Shrew Soft VPN Client korrekt. Das kommt bei einigen meiner Kunden zum Tragen, die wiederrum weitere LAN-LAN-Tunnel an deren Routern haben, wo ich zu remote-Zwecken erst deren Router per VPN anspreche, dann aber auch auf deren weitere Subnetze weiter kommen muss. Insofern habe ich eigentlich keine Wünsche mehr offen was den Shrew Soft VPN Client angeht. Was ich benötige geht wirklich alles und vor allen Dingen absolut zuverlässig und stabil - und jetzt auch wieder unter den aktuellen macOS.

Darum habe ich mir auch die Mühe gemacht und einiges an Zeit investiert, damit das wieder unter macOS nutzbar wird - weil ich den Shrew Soft VPN Client wirklich benötige und zwar fast täglich.

7. Fragen rund um die Sicherheit dieses Pakets

Da ich nun mal keinen (leider) kostenpflichtigen Developer-Account bei Apple besitze und ich diesen auch nicht wirklich benötige - *den Shrew Soft VPN Client lauffähig zu machen ergab sich im Wesentlichen deswegen, weil ich den selbst dringend benötige, um zwischen WIN und macOS VPN-Verbindungen austauschen zu können* - ist diese Package natürlich auch nicht digital signiert. Das geht nur, wenn man so einen Developer-Account besitzt. Ich will ja diese Software weder im Appstore anbieten noch anderweitig irgendwie vermarkten. Darf ich ja auch nicht - ich bin ja nicht der geistige Eigentümer, auch wenn Quellcode veröffentlicht wird, gibts ja immer noch Rechte derer, die das Werk geschaffen haben. Nun bin ich nicht anonym, wer ich bin und was ich mache, ist hier entsprechend verlinkt mit allem Pipapo. Man findet also den Weg zu mir. Das ist ja auch gut so, denn das schafft natürlich auch ein gewisses Grundvertrauen. Weiterhin gehe ich davon aus, das diejenigen, die einen IPSec-VPN-Client wie den Shrew Soft VPN Client benutzen wollen oder müssen bzw. überhaupt mit IPSec-VPN konfrontiert werden, nicht unbedingt der Nutzergruppe "Novice (aka "Anfänger/Einsteiger")" zuzuordnen sind, diejenigen also mit macOS grundsätzlich sicher umgehen können und das OS auch entsprechend kennen. An diese Nutzergruppe richtet sich auch meine Bereitstellung des Shrew Soft VPN Client - da nicht alle, die zwar fit mit macOS sind, auch gleichzeitig tiefere Kenntnisse im Bereich Compiler, C/C++ usw. besitzen. Das ist dann nochmal eine *Nutzergruppe in der Nutzergruppe* für sich. Diese sind - auch das setze ich voraus - selbst in der Lage, Aspekte der Systemsicherheit umzusetzen und diese auch fachlich korrekt zu bewerten und ggf. selbst anzupassen. Im professionellen Bereich geht es aus Prinzip nicht darum, was Apple oder andere wie Microsoft gerne hätten, sondern was der Anwender an Aufgaben zu erfüllen hat. Das kann konform sein oder muss eben notfalls in Teilen oder im Ganzen umgangen werden, weil es aus verschiedenen Gründen dem Ziel entgegenwirkt. Die Profis wissen, was ich genau meine. Das Enterprise-IT-Umfeld hat da andere Regeln - und ich behaupte inzwischen fest, Apple hat kein Interesse mehr am Enterprise-Segment in Allgemeinen. Entwicklungen wie die Hardwareausstattung an den Macbook Pro's hinsichtlich Schnittstellen, der macOS-Server, der Mac Pro - aus meiner Sicht hinsichtlich Einsatz in der Unternehmens-IT - meinem Tätigkeitsumfeld - alles nur noch ein Trauerspiel. Die Nutzung als Produktiv-System rückt in immer weitere Ferne. Zum Glück haben wir ja "untenrum" immer noch eine UNIX-Basis, mit der man auch dank der Linux-Community viel umsetzen

kann, was man nicht gleich unbedingt mit macOS in Verbindung bringen würde. Auch der Shrew Soft VPN Client ist klar diesem Bereich zuzuordnen - macOS ist mehr oder weniger ein Abfallprodukt der Linux-/FreeBSD-Anpassung der Sourcen des VPN-Clients. Deswegen bin ich heute immer noch macOS-User, sonst hätte ich diese Plattform schon längst ad acta gelegt.

Grundsätzlich kann ich hier versichern, **das ich keinerlei Änderungen am eigentlichen Programmcode vorgenommen habe** - um das gleich zu Beginn klarzustellen. Meine Arbeit bestand im Wesentlichen nur darin, dem Compiler und Linker über dessen Steuerdateien, die `CMakeLists.txt`, so anzupassen, dass der Compiler und Linker alles findet und demzufolge sich der Quellcode auch in ein nutzbares Programm verwandeln lässt. Die Sourcen stammen aus dem Jahr 2013, jetzt ist 2017 und da hat sich auch in den Systemumgebungen macOS, Compiler, Linker natürlich einiges geändert - man muss also Anpassungen an aktuelle Befindlichkeiten der Systeme und deren Tools vornehmen. Genau das habe ich gemacht - **ohne Änderungen am Programmcode selbst** - mit dem Ergebnis, eine nutzbare Software generiert zu haben, die vielleicht sogar noch dem einen oder anderen OS-Upgrade standhält. Da ja nicht jeder User in der Lage ist, einen Compiler zu benutzen, geschweige denn mit Quellcode umzugehen, will ich denjenigen praktisch das fertige Ergebnis zur Verfügung stellen - muss nur noch installiert werden und man kann es dann sofort benutzen. Wem das nicht reicht - der kann ja gern alles selber bauen - eine Anleitung unter Benutzung der Original-Sourcen habe ich ja hier bereits ausführlich beschreiben, allerdings in englischer Sprache. Das Ganze nahm urspünglich im Apfeltalk-Forum seinen Anfang, wer Fragen hat, melde sich bitte dort -siehe hier:
https://www.apfeltalk.de/community/threads/vpn-file-importieren.514010/

Ich habe ja zu der ganzen Frage Sicherheit eine klare Meinung als IT-Profi. Wer wirklich glaubt, das Verfahren wie das digitale Signieren von Programmcode jedwegliche Sicherheitsprobleme beseitigt - dem muss ich leider sagen: dem ist nicht so. Beispiele gibts da bereits genug, denn auch damit kann man wohl viel Böses machen, mehr als man vielleicht vermuteten mag. Da das ja alles auf einer PKI (Public Key Infrastuktur) beruht - defacto vergleichbar mit SSL-Verbindungen und deren Zertifikatsmangment dahinter, gibts leider da auch Lücken. Kommt jemand an einen privaten Schlüssel (Key) eines Zertifikats - war's das mit der Sicherheit, zumindest solange bis das bemerkt wird und das entsprechende Zertifikat zurückgezogen wird. Aber dann ist oft bereits der Schaden eingetreten. Auch werde ich das Gefühl nicht ganz los, das Apple das nicht ganz ohne Eigennutz macht - deren Kontrolle über unsere Systeme wird größer. Ob ich das gut finde ? - eher nicht. Sie versuchen es alle - Apple, Microsoft, Google. Es ist immer zu hinterfragen, ob das, was da dem User kommuniziert wird, auch der wirkliche Hintergrund ist - oft trifft das nicht oder nur teilweise zu. Ich mache 20 Jahre IT ausschließlich im Unternehmensbereich - da ist mein Blick schon immer kritisch gewesen. Es gehts immer um's Business - daran sollte man immer denken. Die Unternehmen wollen und müssen Geld verdienen - mit offenen, aber auch verdeckten Karten. Letztlich sind Erscheinungen wie die Ransomware, Cryptotrojaner und die ganze Palette auch immer ein Spiegel - den man den Herstellern vor Augen halten könnte und fragen sollte, warum mein Betriebssystem dagegen so schlecht abgesichert war oder ist. Oder ? Also meine lieben User - immer wachsam bleiben und manches auch mal kritisch hinterfragen, ob hinter einem vermeintlichen "Sicherheitsfeature" mehr steckt als man auf den ersten Blick vermutet. Es gilt immer und überall - zuerst den Kopf einschalten.

Wie sagte Benjamin Franklin schon im Jahr 1775:

"*Those who would give up essential Liberty, to purchase a little temporary Safety, deserve neither Liberty nor Safety.*"

Übersetzung:

"*Diejenigen, die die essentielle Freiheit aufgeben, um eine kleine vorübergehende Sicherheit zu*

erwerben, verdienen weder Freiheit noch Sicherheit."

So - also viel Spass mit dem Shrew, ich hoffe ich konnte dem Einen oder Anderen damit helfen.

zurück zur Startseite

From:
http://kb.amft-it.de/ - **Amateurfunk - Knowledge Base und Wiki by DL1BZ**

Permanent link:
http://kb.amft-it.de/doku.php?id=kb-macos:shrewvpnclient-download

Last update: **09.11.2017 11:12**