Hochschule Offenburg
University of Applied Sciences
Unternehmens- und IT-Sicherheit

UNITS (WS15/16)

Fakultät für Medien und Informationswesen

Userspace IPsec-Processing unter Windows

Author:
Noel Kuntze

Supervisor:
Prof. Dr. D. Westhoff

Bearbeitungszeitraum: 28. April 2016 bis 28. September 2016

Druckdatum: 28. September 2016

Eidesstattliche Erklärung

Hiermit versichere ich eidesstattlich, dass die vorliegende Bachelor-Thesis von mir selbstständig und ohne unerlaubte fremde Hilfe angefertigt worden ist, insbesondere, dass ich alle Stellen, die wörtlich oder annähernd wörtlich oder dem Gedanken nach aus Veröffentlichungen, unveröffentlichten Unterlagen und Gesprächen entnommen worden sind, als solche an den entsprechenden Stellen innerhalb der Arbeit durch Zitate kenntlich gemacht habe, wobei in den Zitaten jeweils der Umfang der entnommenen Originalzitate kenntlich gemacht wurde. Ich bin mir bewusst, dass eine falsche Versicherung rechtliche Folgen haben wird.

Haslach, den ....

Noel Kuntze

1 Einleitung

1.1 Abstract

Der Mangel an Unterstützung für TUN/TAP-Geräte unter Windows von strongSwan, sowie Fehler und fehlende Unterstützung für Mehrfaktorauthentifizierung im nativen IPsec-Stack von modernen Windows-Versionen zusammen mit dem Mangel an frei verfügbaren Alternativen zu den mangelhaften existierenden Lösungen lässt den Bedarf an einem Client für IPsec-Roadwarrior-VPNs ungedeckt.

Diese Arbeit zielt darauf ab, Unterstützung für den OpenVPN TAP-Treiber für Windows in strongSwan zu implementieren, sodass die Grundlage für die Implementierung eines offenen Clients auf Basis von strongSwan gelegt ist.

1.2 Motivation

Die Motivation für die Arbeit ist der Mangel eines modernen, flexiblen, offenen und gepflegten IPsec-VPN-Clients für die Windows-Platform. Aufgrund der Erfahrungen aus der Unterstützung des strongSwan-Projekts wurde der Bedarf für so eine Software entdeckt, der durch die Ergänzung von fehlender Funktionalität abgedeckt wird. Der Mangel an starker Kryptografie und existierende Implementierungsfehler im nativen Client von neueren Windowsversionen wird oft bemängelt und ist im Jahr 2016 nicht mehr adequat zur Bedrohungslage durch staatliche Angreifer.

Inhaltsverzeichnis

Abbildungsverzeichnis

Codeverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ABI Application Binary Engine

AEAD Authenticated Encryption with Associated Data

AH Authentication Header

API Application Programming Interface

ARP Address Resolution Protocol

BA Bachelorarbeit

BEET Bound-End-to-End-Tunnel

BGP Border Gateway Protocol

CA Certificate Authority

CBC Cipher Block Chaining

CP Configuration Payload

CPU Central Processing Unit

CRL Certificate Revocation List

DH Diffie Hellman

EAP Extensible Authentication Protocol

ESP Encapsulated Security Payload

FD File Descriptor

GUI Graphical User Interface

HMAC Hashed Message Authentication Code

HSO Hochschule Offenburg

ICMP Internet Control Message Protocol

IETF Internet Engineering Task Force

IKE Internet Key Exchange

ISAKMP Internet Security Association and Key Management Protocol

ISIS Intermediate System To Intermediate System

IP Internet Protocol

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

IPH IP Helper

IPsec Internet Protocol Security

ISP Internet Service Provider

L2TP Layer 2 Tunneling Protocol

MAC Message Authentication Code

MTU Maximum Transmission Unit

MSS Maximum Segment Size

NAT Network Address Translation

OCSP Online Certificate Status Protocol

OSPF Open Shortest Path First

PPTP Point to Point Tunneling Protocol

PBR Policy Based Routing

PFS Perfect Forward Secrecy

PSK PreShared Key

RFC Request For Comment

RFC Request For Comment

RW Roadwarrior

SA Security Association

SAD Security Association Database

SP Security Policy

SPD Security Policy Database

SPI Security Parameter Index

SSTP Secure Socket Tunneling Protocol

TCP Transmission Control Protocol

TOS Type Of Service

TS Traffic Selector

TTL Time To Live

UDP User Datagram Protocol

WFP Windows Filtering Platform

VICI Versatile IKE Configuration Interface

VM Virtual Machine

VPN Virtual Private Network

VTI Virtual Tunnel Interface

2 Grundlage

Um IPsec zu verstehen, müssen zuerst die Netzwerkgrundlagen verstanden werden.

2.1 Netzwerke

Moderne Computernetzwerke basieren auf Ethernet und Internet Protocol version 4 (IPv4) oder Internet Protocol version 6 (IPv6), wie standardisiert von der Internet Engineering Task Force (IETF) in diversen Request For Comments (RFCs). Der Empfang und das Senden von Daten geschieht mit Hilfe von Netzwerkschnittstellen, die die Computer mit dem Netzwerk verbinden. Die Adressierung der Computer im Netzwerk geschieht mithilfe der Internet Protocol (IP)-Adresse des Empfängers. Um bidirektionale Kommunikation zu ermöglichen, enthält ein IP-Paket als Quelle die IP-Adresse des Senders. Die Übertragung von IP-Paketen zwischen verschiedenen Computern findet durch Nutzung der vorhandenen Übertragungswege statt, sei es Ethernet über entsprechende Kabel oder über andere Träger, wie Glasfaser oder Funk. Die Adressierung der Netzwerkschnittstellen erfolgt über deren MAC-Adressen, was für Internet Protocol Security (IPsec) jedoch nicht relevant ist, da nur IP-Pakete oder die Payload davon, also das Protokoll auf der Transportschicht, geschützt werden.

2.2 Routing

Beim Routen eines Pakets wird nach dem Empfang oder Senden eines IP-Pakets durch Nutzung des sogenannten "Routing Table" nachgeschlagen, wohin ein Paket weitergeleitet werden muss, um den Empfänger zu erreichen. Hierbei wird nach dem längsten passenden Präfix im Routing Table gesucht, um die korrekte Route zu finden. Der Time To Live (TTL)-Wert des Pakets wird überprüft und dekrementiert. Wenn er 0 erreicht, wird das Paket verworfen und eine Internet Control Message Protocol (ICMP)-Fehlermeldung an den Absender verschickt.

2.2.1 Policy Based Routing

Policy Based Routing ist eine Sonderform des Routings. Hierbei wird nicht die Zieladresse zum Finden der Route genutzt, sondern eine Regel.

Linux implementiert Policy Based Routing (PBR) durch die die Nutzung von Regeln, die Pakete auf Basis von entweder Firewall-Markierungen, der Ziel- oder Quelladresse, dem Wert des Type Of Service (TOS)-Felds oder der benutzten eingehenden oder ausgehenden Netzwerkschnittstelle in bestimmte Routing-Tabellen umleitet. Die Firewallmarkierung kann ist 32 Bit lang und kann in der Firewall (iptables, nftables, ebtables) auf eigens gewählte Werte und mit eigens gewählten Regeln gesetzt werden. Linux erlaubt Anwendungen die Firewallmarkierung direkt im Netzwerksockel zu setzen.

Windows implementiert kein PBR.

2.3 IPsec

IPsec ist ein Konzept zum Absichern von beliebigen IP-Paketen oder deren Nutzlasten. Es stellt Vertraulichkeit, Authentizität und Schutz vor Replay-Attacken bereit. Die Implementierung der Schutzmechanismen beruht auf sogenannten Security Associations (SAs) und Security Policy

(SP), die genutzt werden um die Daten zu schützen. Die Kombination aus zugehörigen SAs und SPs wird allgemein als CHILD_SA bezeichnet.

IPsec an sich lebt nur im Kernel, jedoch wird für die Aushandlung von Sitzungsschlüsseln, Algorithmen und dem Erneuern derselben eine Komponente im Userland benötigt.

Im Userland existiert in der Regel ein Systemdienst (Daemon), der eine oder mehrere Versionen von Internet Key Exchange (IKE) spricht und so ermöglicht IPsec SAs mit anderen Computern auszuhandeln um IP-Pakete zu schützen. Es steht jedem Systemadministrator jedoch frei die IPsec SAs und SPs manuell zu konfigurieren.[1]

Der Daemon kommuniziert mit dem Kernel und übergibt ihm die ausgehandelten IPsec SAs und SPs, die in der Security Association Database (SAD) und Security Policy Database (SPD) verwaltet werden.

Die Kommunikation zwischen den Diensten wird über Authentifizierungsverfahren wie PreShared Key (PSK), Zertifikatsauthentifizierung, RSA- oder ECDSA-Schlüssel oder Extensible Authentication Protocol (EAP) unter Benutzung des Oakley-Protokols[2] abgesichert, welches mithilfe des Diffie Hellman (DH)-Schlüsselaustauschprotokolls geheime Schlüssel zwischen den Teilnehmern aushandelt.[3][4]

2.3.1 IKE

IKEv1/ISAKMP IKEv1 wird oft als äquivalent zu Internet Security Association and Key Management Protocol (ISAKMP) verwendet, wobei es auch keinen für diese Arbeit bedeutsamen Unterschied gibt. Daher werden die Begriffe hier auch äquivalent genutzt. IKEv1 ist die erste, älteste Version des IKE-Protokolls, welches zum Aushandeln von CHILD_SAs genutzt wird.

IKEv2 IKEv2 ist die logische Weiterentwicklung von IKEv1 und beinhaltet Verbesserungen im Bezug auf den Verbindungsaufbau, Robustheit, Sicherheit und Flexibilität[5]. Mit der neuen Version wurde XAUTH durch EAP ersetzt, was die Delegation der Authentifizierung zu einem oder mehreren RADIUS-Servern ermöglicht und weitere Authentifizierungsmodi ermöglicht. Des weiteren wurde der unsichere "Aggressive Mode entfernt" und der Standard klarer formuliert.

IKE_SA Eine IKE_SA bezeichnet eine Verbindung, die zwei Teilnehmer mittels IKE ausgehandelt haben. Eine IKE_SA wird durch die IP-Adressen, sowie durch Security Parameter Indexes (SPIs) identifiziert. Beiden Teilnehmern ist ein mittels IKE ausgehandeltes geteiltes Geheimnis bekannt, welches genutzt wird um Nachrichten zwischen den Teilnehmern zu verschlüsseln und zu authentisieren.

IKE_SAs haben zu CHILD_SAs eine 1:N-Beziehung. Eine einzelne IKE_SA kann keine, eine oder mehrere CHILD_SAs verwalten.

CHILD_SA Eine CHILD_SA ist die Sammlung aller zugehörigen IPsec SAs und SPs, die zu einem Tunnel gehören. Wenn der Tunnel mit Komprimierung ausgehandelt wurde, so gehören

[1] SK05, S. 18.
[2] Hil98.
[3] Cha+14, S. 50.
[4] Dou+98, S. 8.
[5] Cha+14, S. 136, 137.

die Komprimierungs-SAs, die dabei erstellt werden, auch dazu[6][7].

Das Aushandeln einer CHILD_SA wird unter IKEv1 im sogenannten Quick Mode durchgeführt, der dem Aggressive Mode oder Main Mode folgt. Eine CHILD_SA wird als ganzes verwaltet. Wenn eine einzelne SA gelöscht wird, so wird auch die dazugehörige SA in der Gegenrichtung und die SPs gelöscht. Um die kryptografischen Schlüssel von CHILD_SAs zu erneuern muss sie komplett neu ausgehandelt werden. In verschiedenen Implementierungen wird das Rekeyen von CHILD_SAs mit IKEv1 unterschiedlich gehandhabt. Für IKEv2 ist das Verhalten standardisiert[8].

In der Regel werden unterschiedliche Schlüssel für jede einzelne SA einer CHILD_SA und jeden einzelnen eingesetzten kryptografischen Algorithmus gesetzt.

Das Zuordnen von IPsec-Paketen zu IPsec-SAs wird bewerkstelligt, indem nach dem SPI des Pakets in der SAD gesucht wird.

Unterschiede in Schreibweisen in der Bachelorarbeit und den RFCs Diese spezielle Begrifflichkeit stammt aus dem "strongSwan"-Quellcode, in dem die Datenstrukturen für eine IKE SA und eine CHILD SA Unterstriche enthalten. Um die Begriffe einheitlich zu verwenden wird der Unterstrich immer mitgeschrieben. Dies verdeutlicht auch, dass hier speziell von "strongSwan" gesprochen wird. In den RFCs über IPsec wird ebenfalls von IKE_SAs und CHILD_SAs gesprochen, dort jedoch ohne Unterstrich. Was "strongSwan" ist wird in Unterabschnitt 3.2 erläutert.

PFS Ohne Perfect Forward Secrecy (PFS) werden die Schlüssel für die CHILD_SAs vom Schlüssel der IKE_SA abgeleitet. Wenn PFS eingesetzt wird, so wird stattdessen ein DH-Schlüsselaustausch eingesetzt, um die Schlüssel zu generieren.[9]. Wenn ein kryptografischer Schlüssel für den Verschlüsselungsalgorithmus einer IPsec SA kompromittiert wird, so kann ein Angreifer nur den Verkehr entschlüsseln, der mit dieser SA verschlüsselt wurde.

Rekeying und Reauthentication Beide IKE-Versionen unterstützen das Reauthentifizieren von IKE SAs und das Rekeyen von CHILD_SAs[10]. Der Zweck der Reauthentifizierung einer IKE_SA ist zu überprüfen, ob die Authentifizierungsinformation eines Teilnehmers noch gültig ist, zum Beispiel ob ein Zertifikat ausgelaufen ist oder gesperrt wurde über eine Certificate Revocation List (CRL) oder mittels Online Certificate Status Protocol (OCSP). Der Zweck von Rekeying ist die kryptografischen Schlüssel in gewissen Abständen zu wechseln, zum Beispiel alle 6 Stunden, oder wenn 4 GB übertragen wurden oder wenn eine gewisse Anzahl Pakete übertragen wurde. Dies wird getan, damit die Menge an Daten, die bei einer Kompromittierung eines Schlüssels offengelegt wird, begrenzt ist.

2.3.2 Modi

IPsec unterstützt verschiedene Modi wie Daten über das VPN versendet werden. Die Modi unterscheiden sich hinsichtlich der Tatsache wie sie die IP-Pakete übertragen

[6] Abr+01, S. 7.
[7] Cha+14, S. 61.
[8] Cha+14, S. 16.
[9] Cha+14, S. 13.
[10] Cha+14, S. 616.

Transport-Modus Im Transport-Modus wird der IP-Header des zu schützenden Pakets nicht übertragen. Der IP-Header wird auf Basis der eingetragenen IP-Adressen der genutzten IPsec-SAs bestimmt und nach der Überprüfung des empfangenen Pakets ergänzt. Der Sinn dahinter ist, dass der Overhead der SAs verringert wird, wenn die Quell- oder Zieladressen der getunnelten Pakete sich nicht von denen der SAs unterscheiden.

Abbildung 1: Transport-Modus

IP-Kopf	*IPsec-Paketkopf*	*Nutzlast (Schicht 4 und höher)*	*IPsec-Anhänger*

Tunnel-Modus Im Tunnel-Modus werden die kompletten IP-Pakete übertragen und mittels des ausgehandelten Protokolls geschützt.

Abbildung 2: Tunnel-Modus

Äußerer IP-Kopf	*IPsec-Paketkopf*	*Innerer IP-Paketkopf mit Nutzlast*	*IPsec-Anhänger*

BEET Bound-End-to-End-Tunnel (BEET) ist ein spezieller Modus, bei dem "virtuelle" IP-Adressen für die Endpunkte ausgehandelt werden. Die IP-Header der übertragenen Pakete werden jedoch nicht mitgesendet, sodass eine Struktur wie im Transport-Modus entsteht. Die Quell- und Zieladressen werden nach dem Empfang des IPsec-Pakets durch Suchen der SPI in der SAD ergänzt, sodass das Paket an eine lokale Anwendung ausgeliefert werden kann.[11]

2.3.3 Protokolle

ESP ESP ist das Standardprotokoll zum Absichern von IPsec-basierten Virtual Private Networks (VPNs), da es Vertraulichkeit, Integrität und Replay-Schutz für die übertragenen Pakete unterstützt. Dabei wird optional ein Verschlüsselungsalgorithmus, sowie ein Hashed Message Authentication Code (HMAC) eingesetzt oder ein Authenticated Encryption with Associated Data (AEAD)-Algorithmus.[12]

UDP Encapsulation UDP Encapsulation ist ESP in einer UDP-Hülle. Es wird standardmäßig eingesetzt, wenn erkannt wurde, dass zwischen den Teilnehmern Network Address Translation (NAT) eingesetzt wird.[13] Es wird auch eingesetzt, wenn einer der Teilnehmer keine reinen ESP-Pakete versenden kann, wie zum Beispiel wenn IPsec im Userspace implementiert ist und aus einem beliebigen Grund kein Sockel implementiert wurde, der es ermöglicht reine ESP-Pakete zu versenden.

AH AH ist ein Protokoll, welches den Header des darunterliegenden IP-Pakets absichert, sowie die darin eingebetteten Daten. Im Header des darunterliegenden Pakets werden nur die statischen

[11] Appendix B PRJ15.
[12] Ste05b.
[13] Mar+05.

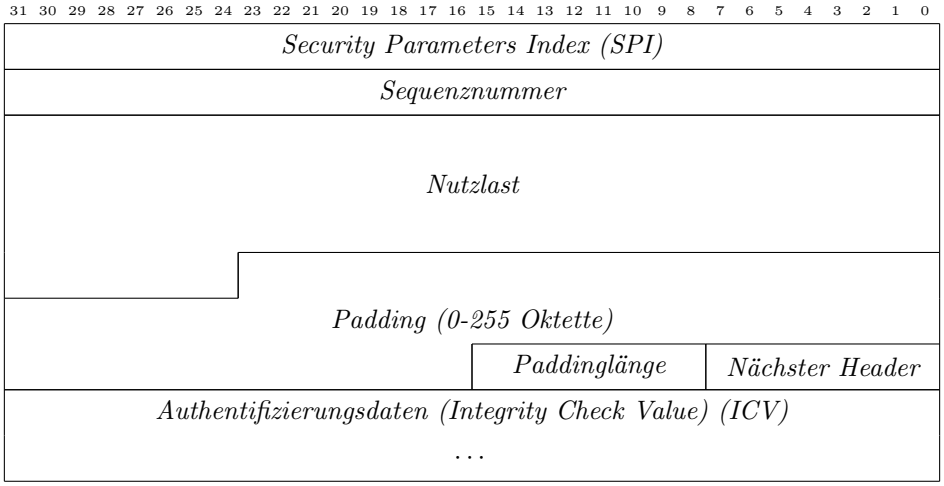


Abbildung 3: ESP

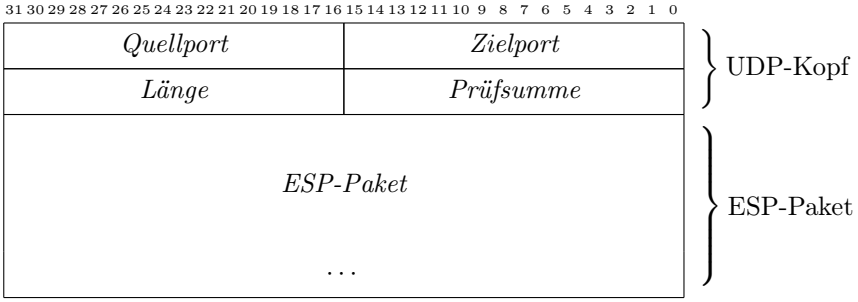


Abbildung 4: UDP Encapsulation

Felder abgesichert. AH bietet Integrität und Replay-Schutz. Es unterstützt keine Vertraulichkeit, da der Verkehr nur mittels eines HMAC abgesichert wird.[14]

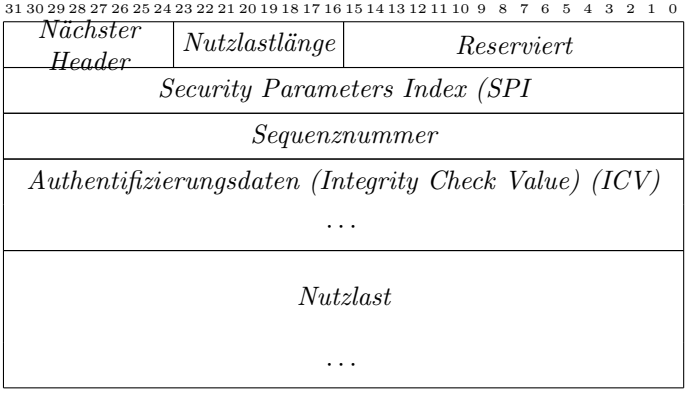


Abbildung 5: AH

2.3.4 MTU und MSS

Beim Tunneln von Paketen muss beachtet werden, dass die Maximum Transmission Unit (MTU) im Tunnel geringer ist als die MTU des physikalischen Trägers. Bei der Berechnung der MTU des

[14] Ste05a.

Tunnels müssen die Header der beteiligten Protokolle (IP, (User Datagram Protocol (UDP),) ESP/AH), sowie die Blockgrößen der eingesetzten Verschlüsselungsalgorithmus und der eingesetzte Message Authentication Code (MAC) berücksichtigt werden. Gewöhnliche Computer nehmen in der Regel eine MTU von 1500 Byte an und berechnen aus dieser MTU die TCP-Maximum Segment Size (MSS) und geben diese beim Verbindungsaufbau mit Transmission Control Protocol (TCP) an. Da die daraus resultierenden Pakete jedoch dann zu groß sind, müssen diese fragmentiert werden. Dies wird mit einer ICMP-Nachricht bewerkstelligt, die an den Sender geschickt wird. Es gibt jedoch leider viele TCP-Implementierungen, die dann nicht fragmentieren und so keine Verbindung aufbauen können. Des weiteren gibt es viele Internet Service Provider (ISP), die ICMP-Nachrichten verwerfen. Dieses Problem kann umgangen werden, indem auf dem IPsec-Router der MSS-Wert von TCP-Nachrichten heruntergesetzt wird, sodass eine niedrigere MSS als die MTU des Tunnels zuzüglich des TCP-Overheads entsteht. Die Fragmentierung von IP-Paketen auf dem IPsec-Router sollte das eigentlich umgehen, aber es funktioniert in der Regel nicht. Dies kann auch aus Sicherheitsgründen so sein, da für die Überprüfung des Packets bezüglich der ausgehandelten SPs zwischen den Teilnehmern erst alle Fragmente gesammelt werden müssten, was bezüglich des Speicherbedarfs problematisch sein kann.[15]

2.3.5 Standardszenarien für IPsec

IPsec wird alltäglich für die Absicherung von Verkehr zwischen einem Router oder VPN-Server und Roadwarrior (RW) genutzt. Ein weiteres Szenario ist die Absicherung von Verkehr zwischen zwei Routern über einen Site-to-Site-Tunnel.

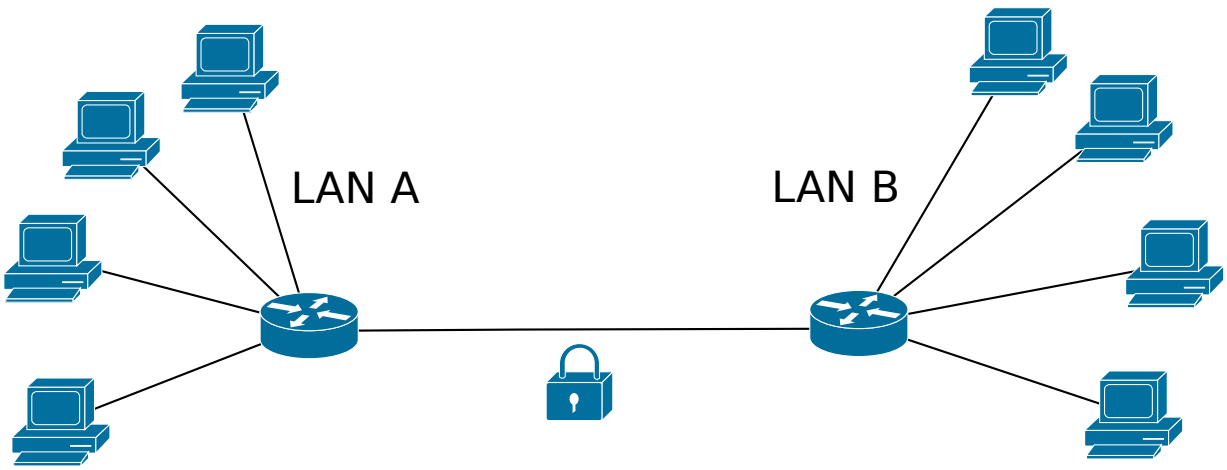


Abbildung 6: Site-to-Site-Szenario

2.3.6 IPsec unter Linux

Unter Linux gibt es zwei verschiedene IPsec-Stacks. Es existiert einerseits der KLIPS-Stack, der vom FreeS/WAN-Projekt entwickelt wurde und auf Route based IPsec basiert.

Des weiteren gibt es den XFRM-Stack, welcher auf Policy based IPsec basiert, jedoch mithilfe eines Virtual Tunnel Interface (VTI) auch für Route based IPsec genutzt werden kann.

[15] Ste05b, Kapitel 3.4 Inbound Packet Processing.

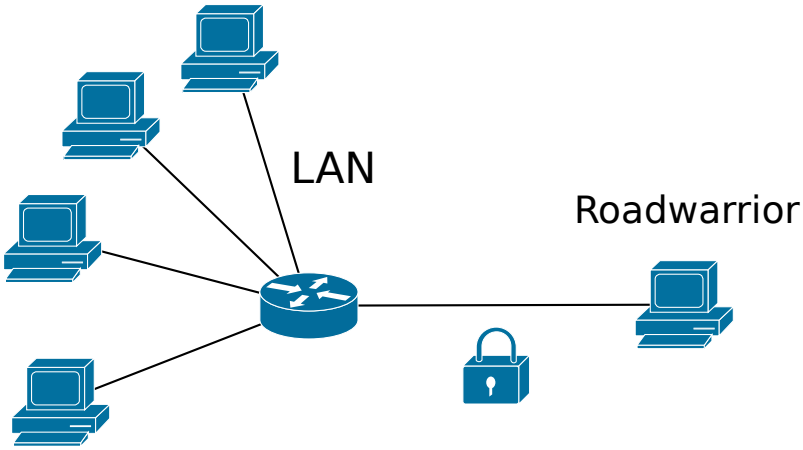


Abbildung 7: Roadwarrior-Szenario

Beide Stacks sind konform zu RFC 2367[16].

Die Kommunikation mit dem Kernel bezüglich Netzwerkfunktionen geschieht über den Netlink-Sockel. Das ”strongSwan”-Projekt implementiert mit dem Dienst ”charon” einen ISAKMP/IKEv1 und IKEv2-Keying-Daemon auf Linux. Die notwendigen Funktionen für die Kommunikation mit dem Kernel sind in ”charon” im Plugin ”kernel-netlink” implementiert. Über Netlink werden auch die SAD und SPD verwaltet.

2.3.7 IPsec unter Windows

Microsoft hat im Laufe der Entwicklungsgeschichte von Windows drei verschiedene IPsec-Implementierungen implementiert, die teilweise parallel existierten.

In der Firewall von Windows existiert eine Implementierung, die Policy-based ist und daher flexibler ist und es erlaubt Firewall-Regeln zu schreiben, die abhängig davon sind, ob ein Paket mit IPsec geschützt war.

”strongSwan” implementiert mit ”charon-svc” einen ISAKMP/IKEv1 und IKEv2-Keying-Daemon auf Windows. Die Kommunikation mit dem Kernel ist dort in den Plugins ”kernel-wfp” und ”kernel-iph” implementiert.

Windows Agile VPN Client Seit Windows 7 existiert in Windows eine Implementierung von IPsec im Netzwerkmnanager, der den Config-Modus implementiert, sodass Windows als RW operieren kann. Diese Implementierung nutzt eine Route-based-Implementierung, wie in Unterunterabschnitt 2.3.8 erklärt. Diese Implementierung heißt offiziell ”Windows Agile VPN Client” und implementiert Point to Point Tunneling Protocol (PPTP), IPsec+Layer 2 Tunneling Protocol (L2TP), Secure Socket Tunneling Protocol (SSTP) und IKEv2. Sie existiert seit Windows 7.

Die Fähigkeiten dieser Implementierung lässt sich aus den Tabellen im Appendix unter Unterabschnitt 6.1 ablesen.

IKEEXT Windows wird mit einem Dienst namens ”IKEEXT” ausgeliefert, welcher genutzt werden kann, um IKE_SAs und CHILD_SAs auszuhandeln. Standardmäßig ist dieser Dienst

[16] DB98.

aktiv und lauscht auf UDP Port 500 und 4500. Diese Implementierung unterstütz den Config-Modus nicht und ist daher nicht für die Benutzung als RW geeignet.[17]

WFP Windows Filtering Platform (WFP) ist die grundlegende Technologie im Netzwerkstack von Windows, mit der IPsec im Kernel, sowie die Firewall dort implementiert wurde. "charon" kann mit WFP mithilfe des "kernel-wfp"-Plugins kommunizieren und SAs und SPs in die jeweiligen Datenbanken einfügen.

WFP hat mehrere Probleme, unter anderem mit virtuellen IP-Adressen, UDP-Encapsulation, mit mehreren Traffic Selectors im Transport-Modus und mit Protokoll oder Port-Beschränkungen in den SPs von Net-To-Net-SPs[18]

Martin Willi hat die Problematik mit virtuellen IPs in einem Git-Commit beschrieben:

> WIP: Windows virtual IP notes
>
> When not using skipAsSource with the installed virtual IP, the IP gets promoted as source address for the already available routes over that interface. When setting the flag, all of our manually installed IPsec routes still use the interfaces main address as source address, as the route uses that interface.
>
> To fix this issue, we probably need a dedicated interface for virtual IPs that allows us to install our separated routes over that interface.
>
> Using the MS Loopback adapter kinda works; when disabling skipAsSource, an address installed to that adapter gets used and outgoing traffic flows as expected. Inbound traffic, though, fails with STATUS_IPSEC_CLEAR_TEXT_DROP, probably related to:
>
> https://wiki.strongswan.org/projects/strongswan/wiki/Kernel-wfp#Accessing-Gateway-internal-address-in-a-net-to-net-tunnel [Mar14]

Aufgrund der Problematik bezüglich virtueller IPs ist es nicht sinnvoll einen RW-Client auf Basis von WFP zu erstellen.[19][20]

Die Problematik mit UDP-Encapsulation wird so beschrieben, dass unaufgeforderte UDP-Encapsulation-Pakete mit dem Ereignis "STATUS_INTERNAL_ERROR" verworfen werden. Laut dem Text beeinflusst das nur Pakete, die über eine SA mit UDP-Encapsulation empfangen werden und die der Kernel nicht zu einer bereits bekannten Verbindung zuordnen kann. Eine TCP-Verbindung, die der Windows-Host über eine solche SA aufgebaut hat, ist zum Beispiel von diesem Problem unbeeinflusst.[21]

Der Wiki-Artikel über kernel-wfp beschreibt, dass die Datenstrukturen, die in Windows für die Definition einer Transport-Policy genutzt werden, nur die Spezifizierung eines einzelnen Selektors zulassen.[22] Das verhindert die Nutzung mehrer Selektoren für eine CHILD_SA im Transport-Modus. Dies könnte zum Beispiel für L2TP/IPsec genutzt werden, um das L2TP-Pseudowire und ICMP-Pakete zwischen den teilnehmenden Hosts abzusichern.

[17] 16f.
[18] Mar14.
[19] TM14.
[20] Mar14.
[21] Mar14.
[22] Mar14.

Der Kernel überprüft die Header der Transportschicht nicht, wenn Pakete weitergeleitet werden. Daher werden Protokoll- und Portselektoren für Forwarding-SPs nicht unterstützt. strongSwan ignoriert solche Selektoren daher, wenn kernel-wfp genutzt wird und Forwarding-Policies installiert werden. Für Input- und Output-Policies werden sie jedoch erzwungen.[23]

Des weiteren stellt WFP keine Statistiken über die einzelnen SAs bereit, was verhindert, dass volumenbasiertes Rekeying und das Auslesen von Nutzungsstatistiken der SAs implementiert werden kann.

IPH Windows bietet die IP Helper (IPH)-Funktionen an, die das Verwalten von IP-Adressen, Routen und Geräten erleichtert. "strongSwan" nutzt diese Funktionen im "kernel-iph"-Plugin, um ein Kernel-Interface für Windows mit diesen Funktionen zu implementieren.

2.3.8 Route Based

Route based heißt, dass Pakete, die über das VPN gesendet werden sollen, in eine virtuelle Netzwerkschnittstelle geroutet werden. Da gewöhnliches Routing auf Basis der Zieladresse geschieht, unterstützt solche eine solche Implementierung in der Regel keine SPs, die nur bestimmte Protokolle oder Ports schützen oder wenn das der Fall ist, so ist die Kommunikation mit den anderen Ports oder Protokollen nicht mehr möglich, da jegliche Pakete, die in die Schnittstelle gelangen und von den SPs nicht zugelassen sind, verworfen werden.

Implementierungen von IPsec auf Endgeräten sind in der Regel Route based, gleichzeitig werden die ausgehandelten SPs jedoch weiterhin erzwungen. Route based VPNs werden für gewöhnlich dann genutzt, wenn dynamisches Routen genutzt wird, wie zum Beispiel Border Gateway Protocol (BGP), Intermediate System To Intermediate System (ISIS) oder Open Shortest Path First (OSPF), da beim Verändern der Routen die SPs nicht verändert und damit keine neuen CHILD_SAs ausgehandelt werden müssen. Ein weiterer Vorteil ist, dass Neulinge es leichter haben den Paketfluss zu verfolgen.

Um zu verhindern, dass der IKE-Verkehr über das VPN geleitet wird, wird eine Route für die IP-Adresse des Peers in den Routing Table installiert, die den Verkehr am VPN vorbei leitet. Aus diesem Grund ist es auch nicht möglich mit Route based IPsec mit der IP des Peers unter der Nutzung von IPsec zu kommunizieren.

Eine Möglichkeit Pakete, die nicht von den Policies geschützt werden, nicht ins VPN zu leiten und damit die Kommunikation mit ihnen zu ermöglichen ist PBR anzuwenden und ebendiese Pakete speziell zu markieren und eine andere Routing-Entscheidung zu treffen.

2.3.9 Policy Based

Policy based bedeutet, dass die IPsec-SPs, die ausgehandelt wurden, direkt genutzt werden welche Pakete geschützt werden. Eine solche Implementierung ermöglicht es, IPsec zur Absicherung von sehr genau spezifiziertem Verkehr zu nutzen, wie zum Beispiel nur ICMP-Pakete mit Typ 0, Code 0 (Echo Reply), sowie Typ 8 und Code 0 (Echo Request). Policy based IPsec ermöglicht es IPsec direkt in den Stack zu integrieren, ohne notwendigerweise die Routen zu verändern. Um sicherzustellen, dass der IKE-Verkehr des IKE-Daemons nicht mittels einer IPsec-SA geschützt

[23] Mar14.

wird, was unter Umständen die Kommunikation zwischen den IKE-Daemons verschiedener Systeme verhindern kann, werden bestimmte Optionen auf dem jeweiligen Netzwerksockel gesetzt, die Pakete, die von diesem Sockel kommen, von den SPs und SAs nicht bearbeitet werden.

2.3.10 TUN und TAP-Geräte

TUN- und TAP-Geräte sind virtuelle Netzwerkadapter, die von lokalen Anwendungen genutzt werden können um vom Kernel IP-Pakete oder Ethernet-Rahmen zu empfangen. Pakete und Rahmen, die der Kernel über ein TUN- oder TAP-Gerät erhält werden gleich behandelt als wären sie über eine physikalische Schnittstelle empfangen worden. Sie werden auch gleich verwaltet, so können ihnen zum Beispiel IP-Adressen zugewiesen und Routen über so eine virtuelle Schnittstelle angelegt werden.

TUN-Geräte TUN-Geräte sind virtuelle Netzwerkschnittstellen, die auf der Netzwerkschicht arbeiten. Das heißt, dass sie IP-Pakete senden und empfangen ohne einen darunterliegenden Ethernet-Rahmen. Das heißt im Umkehrschluss das über ein TUN-Gerät keine Kollisionsdomäne erreichbar ist, ARP nicht benutzbar ist und das ”next hop”-Feld einer Route in der Routing-Tabelle keine Bedeutung hat, wenn die Route über ein TUN-Gerät führt.

TAP-Geräte TAP-Geräte verhalten sich wie physikalische Ethernet-Schnittstellen. Das heißt, dass prinzipiell über sie ARP und Ethernet gesprochen werden können und eine Kollisionsdomäne erreichbar ist. TAP-Geräte können genutzt werden um eine Brücke zwischen einem lokalen Netzwerk und einem entfernten Netzwerk über einen VPN-Dienst mithilfe eines virtuellen Brückengeräts zu bauen.

3 Basis

Als Basis der Arbeit wurde der quelloffene VPN-Dienst strongSwan gewählt, welcher von der Hochschule für Technik Rapperswill entwickelt wird. Um die Pakete vom Kernelspace in Empfang nehmen zu können, wird der quelloffene TAP-Gerätetreiber von OpenVPN Technologies, Inc. genutzt. Dieser stellt funktionierende und nutzbare TUN- und TAP-Geräte bereit, mit denen gearbeitet werden kann. Für die Verarbeitung der Pakete wird die in strongSwan integrierte Bibliothek libipsec, das Plugin kernel-libipsec, sowie bestehende Teile von libstrongswan aus strongSwan genutzt.

3.1 Bestehende Implementierungen für Windows

In Abschnitt 6 ist eine Auflistung aller allgemein bekannten IPsec-Clients für Windows, inklusive Matrizen der jeweils unterstützten Algorithmen und Verfahren in den ISAKMP- und IKE-Standards.

3.2 strongSwan

strongSwan implementiert einen beträchtlichen Teil der Funktionalität der RFCs für IKEv2 und ISAKMP/IKEv1[24]. Die Software ist multithreaded und in objektorientiertem C geschrieben. Die

[24] 16g.

Grundlage wurde von Martin Willi und Jan Hutter in ihrer Diplomarbeit im Jahr 2005 gelegt[25]. ”strongSwan” wird unter anderem von der secunet AG in den SINA-Boxen für Hochsichere Anwendungen, Alcatel-Lucent, Astaro und der US-Regierung genutzt.[26] Der strongSwan-Quellcode wird unter der GPLv2 veröffentlicht. Kunden des strongSwan-Projekts können den Quellcode unter einer anderen Lizenz beziehen.

Um Code zu den Kernbibliotheken oder den verschiedenen Programmen des Projekts beizutragen, muss dieser unter der MIT-X11-Lizenz beigetragen werden. Dies ist erforderlich, damit der Quellcode in das Projekt integriert werden kann, das Projekt gegebenenfalls zur GPLv3 wechseln kann und um es den Entwicklern überhaupt zu ermöglichen, den Quellcode unter einer anderen Lizenz weiterzugeben[27].

3.2.1 charon

`charon` ist der Daemon, der in der Diplomarbeit[28] entwickelt wurde. Er arbeitet parallel und wurde in objektorientiertem C geschrieben, so wie die Software Xine. Intern werden verschiedene Bibliotheken genutzt, angefangen von ”libstrongswan” bis zu `libcharon`. Es existieren verschiedene Versionen von `charon-svc` für verschiedene Zwecke, zum Beispiel `charon-svc`. `charon-svc` ist eine Version von `charon`, die als Dienst auf Windows genutzt werden kann.

Jegliche Funktionalität von strongSwan ist in Form einer Bibliothek untergebracht und in objektorientiertem C geschrieben. Das ermöglicht es die Namensräume in strongSwan relativ frei zu halten und Objekte gleichen Typs gleich zu behandeln, sowie die Duplikation von kurzem Code zu vermeiden.

Ein Beispiel dafür ist libstrongswan, welche unter anderem allgemeine Datenstrukturen wie Linked Lists, Arrays und Hashtables, sowie Code für TUN-Geräte und Multithreading implementiert.

Kernel-Interfaces `charon` benötigt für die Verwaltung von IP-Adressen, Routen, TUN-Schnittstellen, Firewallregeln und der SPD und der SAD Zugriff auf den Kernel. Die dafür nötigen Funktionen werden in Plugins für `libcharon` implementiert. Die Plugins implementieren ein standardisiertes Interface für die Methoden, die sie implementieren. Die momentan existierenden Plugins, die Kernel-Schnittstellen implementieren, sind ”kernel-pfroute”, ”kernel-netlink”, ”kernel-wfp” und ”kernel-iph”.[29] Um die Schnittstellen zum Kernel nutzen zu können, muss `charon` mit entsprechenden Rechten ausgeführt werden. Die gängigste Methode hierfür ist, die .exe-Datei mit Administratorrechten auszuführen. Alternativ ist es auch möglich, die Datei mit einem Dienstkonto, welches die entsprechenden Rechte hat, auszuführen.

3.2.2 Userspace Processing

Userspace Processing wurde in `charon` von Guiliano Grassi und Ralf Sager für Android implementiert, um die Entwicklung der strongSwan-App für Android zu ermöglichen. Auf Android ist Kernelspace Processing von IPsec-Paketen nicht möglich, da die Privilegien der Anwendung das nicht erlauben.

[25] JM05.
[26] And11, Folie 8.
[27] 14, Contributing.
[28] JM05.
[29] 16h.

Um Userspace Processing von IPsec-Paketen zu implementieren benötigt die Anwendung, die die Pakete verarbeiten soll, Zugriff auf die Pakete und muss, damit die darin enthaltenen Pakete an den Zielort weitergeleitet werden können, eine Möglichkeit haben sie an den Kernel zum Routing zu übergeben. In der Regel wird dies mit TUN- oder TAP-Geräten bewerkstelligt, die einen virtuellen Netzwerkadapter darstellen. Am einen Ende hängt der Kernel mit seiner Routing-Engine und den Firewallregeln, am anderen Ende hängt die Anwendung, die Pakete auf einen File Descriptor (FD) oder ein Handle schreibt und davon liest.

Pakete, die in den Adapter gesendet werden werden von der Anwendung empfangen und Pakete, die in das Handle geschrieben werden, tauchen auf dem Adapter auf Seiten des Betriebssystems auf.

OpenVPN TAP-Treiber Der OpenVPN-Tap-Treiber ist unter den Namen "TAP-Windows" bekannt und wurde von OpenVPN Technologies, Inc. für den Einsatz mit OpenVPN entwickelt. Er steht unter der GPLv2.

Der Treiber stellt unter Windows ein Ethernet-Gerät dar, inklusive Kollisionsdomäne. Dies ist für dein Einsatz mit IPsec etwas unbeholfen, da IPsec IP-Pakete überträgt und keine Ethernet-Rahmen. Der Treiber unterstützt, wie OpenVPN, das Tunneln von Ethernet-Frames jedoch (TAP-Modus). Er bewerkstelligt das, indem er den Ethernet-Rahmen eines eingehenden Pakets vom Host verwirft und nur das IP-Paket weiterleitet. Vice Versa ergänzt er IP-Pakete, die er von der Software erhält um einen Ethernet-Rahmen. Um über den Adapter kommunizieren zu können stellt der Treiber einen virtuellen Router bereit, dessen IP-Adresse, wie die lokale und die entfernte IP-Adresse(n) konfigurierbar ist.

Der Treiber reagiert nicht auf jede ARP-Anfrage, sondern nur auf solche, bei denen die Quelladresse im ARP-Request der lokalen konfigurierten gleicht, sowie die erfragte Adresse der konfigurierten entfernten Adresse (oder einer der Entfernten, falls ein gesamtes Netzwerk als entfernte IP konfiguriert wurde) gleicht.

Die MAC-Adresse des virtuellen Routers hängt von der GUID des Adapters ab, welche zufällig ist und bei der Erstellung des Adapters initialisiert wird. Die MAC-Adresse kann auch über die Registry verändert werden.

Um mit dem virtuellen Adapter kommunizieren zu können, wird ein Handle mit dem Gerät geöffnet, über das dann asynchron gelesen und geschrieben werden kann. Des weiteren kann das Gerät über das Handle mittels IoCtl konfiguriert werden.

Im September 2016 wurde eine Anfrage an OpenVPN Technologies, Inc. gestellt, ob die Headerdatei für die IoCtl-Werte und die verschiedenen Pfade, um das Handle für das Gerät öffnen können, unter die MIT-X11-Lizenz gestellt werden könnte. Der Grund dafür ist, dass geistiges Eigentum, das unter der GPL-Lizenz ist, nicht in ein Projekt unter der MIT-X11-Lizenz integriert werden kann, da die MIT-X11-Lizenz freier ist als die GPL-Lizenz. Die Anfrage wurde am 18. September 2016 auf dem Hackathon von OpenVPN diskutiert und von James Yonan, dem Mitbegründer von OpenVPN Technologies, Inc. und Verantwortlichen für den Quellcode des TAP-Windows6-Treibers ist, hat ihr entsprochen. Bei Abgabe der Bachelorarbeit (BA) war eine entsprechend lizensierte Version der Header-Datei noch nicht verfügbar.

Betriebsmodi Der TAP-Treiber beherrscht drei verschiedene Betriebsmodi für verschiedene Zwecke. Laut dem Quellcode des Treibers ist der Point-To-Point-Modus veraltet.

TUN-Modus Im TUN-Modus schneidet der Treiber den Ethernet-Rahmen um das IP-Paket ab und reicht das Paket über das Handle an die Software weiter. Pakete, die an den Treiber gegeben werden, werden um einen Ethernet-Rahmen ergänzt, bei dem die Zieladresse die MAC-Adresse des virtuellen Adapters ist und die Quelladresse die MAC-Adresse des virtuellen Routers. In diesem Modus existiert eine lokale IP-Adresse, ein virtueller Router und ein entferntes Netzwerk.

TAP-Modus Im TAP-Modus kopiert der Treiber jeden Ethernet-Paket zwischen Anwendung und Gerät.

Point-To-Point-Modus Im Point-To-Point-Modus existiert eine lokale IP und eine entfernte IP. Empfangene Ethernet-Frames werden entfernt und nur das enthaltene IP-Paket weitergereicht.

libipsec libipsec ist eine Bibliothek, die IPsec im Tunnelmodus implementiert. Sie ist Teil des strongSwan-Quellcodes. Sie wird für die Verarbeitung von IP-Paketen in der Regel auf Systemen genutzt, die keine (funktionierenden) IPsec-Implementierung enthalten.

libipsec besteht aus zwei Komponenten: Einerseits die Bibliothek, die die SAD und die SPD implementiert, sowie die Verarbeitung (libipsec) und ein Plugin, welches ein Kernel-Interface für das Verwalten der SAD und SPD emuliert (kernel-libipsec).

libipsec empfängt und sendet Pakete über die gleichen Sockets, den der restliche Teil des Daemons benutzt, um mit anderen IKE-Peers zu kommunizieren. Das sind Sockets, die auf UDP mit Port 500 und 4500 gebunden sind; die IKE-Standardports. libipsec unterstützt die Verarbeitung von UDP-Encapsulation-Paketen, sowie von normalen ESP-Paketen, da sie sich strukturell nur gering unterscheiden. Jedoch wird aufgrund der Tatsache, dass mit den gegebenen Sockets kommuniziert wird, UDP-Encapsulation forciert. Um ESP-Pakete verschicken zu können, müsste ein Socket geöffnet werden der das verschicken von solchen Paketen erlauben würde. Das ist jedoch auf der Plattform, für die die Bibliothek ursprünglich geschaffen wurde (Android) nicht erlaubt, da die Privilegien der Anwendung dort sehr beschränkt sind.

Prinzipiell wäre es jedoch möglich dies zu implementieren. Dafür müsste ein raw-Socket geöffnet werden, über den rein IP gesprochen wird.

libstrongswan libstrongswan ist eine interne Bibliothek von strongSwan, die von den verschiedenen Versionen von `charon` geladen wird, um diverse Funktionalität zu erhalten, wie Linked-Lists, Hashtables, Arrays, sowie die Funktionen um TUN-Geräte zu erstellen und zu öffnen.

Der Code für das lesen und schreiben von Paketen existiert bereits für Linux, Mac OSX und FreeBSD. Die Operationen basieren dabei auf File Descriptors, welche mit poll() multiplext nach Daten abgefragt werden.

4 Vorgehensweise

Zuerst wurden alle nötigen Änderungen ausfindig gemacht. Dies beinhaltete die Implementierung von Code von für IO-Multiplexiung unter Windows.

Des weiteren mussten fehlende Features gefunden werden. Einen Hinweis darauf ließ sich bereits auf den Wiki-Seiten über die Plugins für Windows finden. Dieser ergab, dass die Verwaltung von virtuellen IPs unter Windows momentan nicht unterstützt wurde.

Weitere Punkte wurden nach dem ersten Kompilierungsversuch, sowie der Beobachtung des Verhaltens während dem Tunnelaufbau herausgefunden.

Die Unterstützung für virtuelle IP-Adressen wurden aus dem Zweig `win-vip` vom strongSwan Git-Repository bezogen und angepasst. Der Großteil des Quellcodes stammt also von Martin Brunner, der diesen Code geschrieben hat. Der Code musste noch um die Beachtung eines Konfigurationswertes ergänzt werden.

Die Installation der Routen in kernel-libipsec wurde angepasst, sodass es mit dem TAP-Treiber funktioniert.

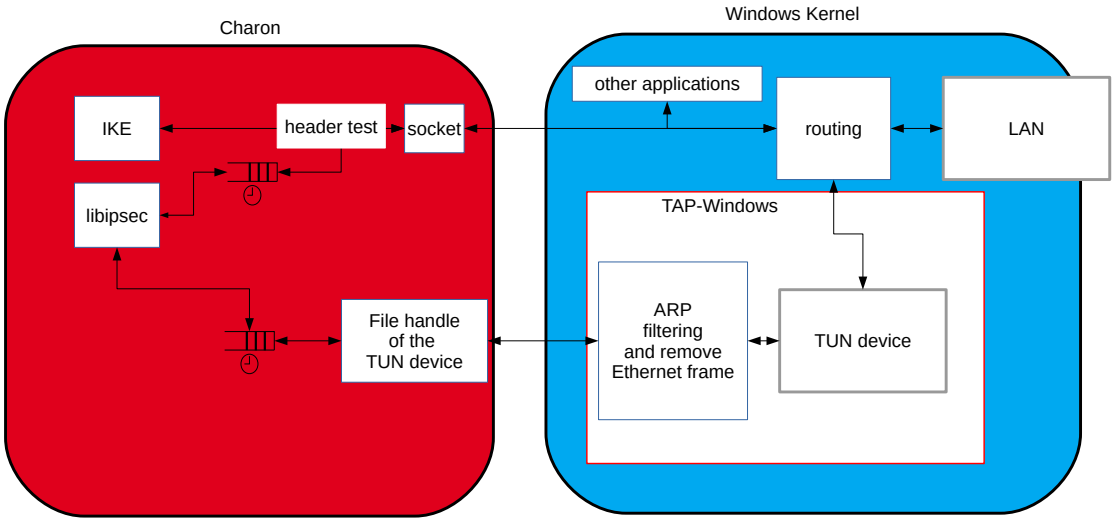


Abbildung 8: Datenflussdiagramm

4.1 Portierung von libipsec auf Windows

Die Portierung von libipsec auf Windows, sodass sie dort lauffähig ist, ist das Ziel der Bachelorarbeit. Die zu portierenden Teile des Programmcodes betreffen nur die Codesegmente, die plattformspezifische Application Programming Interfaces (APIs) oder Application Binary Engines (ABIs) verwenden, also primär alles um Dateiein- und Ausgabe, sowie das Verwalten von TUN-Geräten. Unter Unix und Linux werden hier FDs verwendet. Unter Windows werden stattdessen Handles genutzt, welche einen anderen Dateityp darstellen. Des weiteren unterscheidet sich die Methode zum Multiplexen von Lese-Aufrufen auf einer Liste von Handles oder FDs stark. Unter Unix und Linux wird hierfür poll() genutzt, unter Windows geschieht das jedoch Event-basiert mit WaitForMultipleObjects(). Explizite Beispiele hierfür sind im Abschnitt über die Portierung von libipsec sichtbar. Bei der Portierung wurde für das Verwalten von Speicherabschnitten primär alloca() genutzt, statt malloc() und seine Unterarten. Der Unterschied hierbei ist die Speicherdauer. Mit alloca() allokierter Speicher ist nur bis zum Verlassen der Funktion gültig und wird automatisch freigegeben. Mit malloc() allokierter Speicher muss manuell freigegeben werden. alloca() ist ein Feature, welches nicht standardisiert ist. Daher lässt sich strongSwan nicht mit allen C-Bibliotheken übersetzen. Auf der Wiki-Seite über den Windows-Port wird

erwähnt, dass nur die Kompilierung mittels MinGW-W64 unterstützt wird.[30]

4.2 Bestehende Implementierung

Die bestehende Implementierung umfasst die eigentliche Bibliothek, eine Implementierung eines Kernel-Interface zwischen libipsec und `charon`, sowie Code in libstrongswan um TUN-Geräte zu öffnen. Martin Willi hat 2014 bereits die gesamte Portierungsarbeit für version 5.2.0 von strongSwan gemacht. Der Port unterstützt Windows 7 / server 2008 R2 und neuere Versionen von Microsoft Windows.

4.3 Unterstützte Kryptographie

Die unterstützte Kryptographie ist notwendigerweise von den deklarierten Identifikatoren für IKE beschränkt. strongSwan unterstützt mehr Verschlüsselungsalgorithmen als in den RFCs deklariert wurde. Aus diesem Grund nutzt strongSwan Identifikatoren teilweise aus dem privaten Bereich, wenn die Identifikatoren für den eingesetzten Algorithmus nicht standardisiert sind.

strongSwan unterstützt eine große Anzahl von Algorithmen im Vergleich zu anderen Implementierungen, wie in den Tabellen in Abschnitt 6 dagelegt wird. Wenn libipsec genutzt wird, so können alle Algorithmen, die im Userspace implementiert sind, für die Absicherung von Verkehr genutzt werden.

4.4 Portierung

4.4.1 libipsec

libipsec implementiert die Verarbeitung von Paketen, das Erzwingen der SPs, das Verwalten der SPs, SAs, Routen und der TUN-Geräte. Die hierbei relevanten Dateien sind unter "/src/libipsec/" zu finden.

Standardmäßig installiert strongSwan die IP-Adressen die per Config-Mode oder Configuration Payload (CP) empfangen wird auf dem ausgehenden Interface (Linux, Kernelspace processing), was für die Kommunikation mit dem Peer genutzt wird, oder auf dem Loopback-Adapter (Windows). Um die IP-Adresse für TUN-Geräte korrekt zu setzen, nutzt libipsec den Parameter `charon.install_virtual_ip_on` von strongswan.conf, der während der Initialisierung des Plugins gesetzt wird.

Header Die Bibliotheken und Plugins um libipsec nutzen diverse Datenstrukturen und Konstanten, die in C-Header-Dateien von Linux definiert sind. Diese sind unter Windows nicht verfügbar. Aus diesem Grund wurde eine Kopie der relevanten Definitionen in den Quellcode kopiert und fehlende Teile ergänz. Spezifisch wurde die Definition eines IPv4-Headers und eines IPv6-Headers aus der glibc kopiert und die fehlenden Protokollkonstanten per Hand ergänzt.

Ursprünglich wurden diese in "src/libipsec/win32.h" abgelegt und in "src/libpsec/ip_packet.h" und "src/libipsec/ip_packet.c" inkludiert. Dabei wurden die Header aus den Headern des Kernels oder den Headenr der GNU LIBC kopiert. Als Name für das Ifdef-Guard wurde `WINDOWS_32_PROTOCOL_HEADERS` gewählt. Ein Ifdef-Guard wird benötigt um das Reimportieren von Header-Dateien in C zu unterbinden. Wenn eine Header-Datei erneut importiert wird, dann

[30] 15b.

führt das dazu, dass die Kompilierung abbricht, da eine Variable redefiniert wird. Die Redefinierung einer Variable produziert in C eine Warnung oder einen Fehler. C-Code wird in der Regel mit dem Argument `-Wall` kompiliert, welches alle Warnungen zu Fehlern umwandelt. Das wird getan, um Warnungen nicht zu übersehen. Der Compiler generiert Warnungen, wenn er Programmierfehler erkennt, die berichtenswert sind, aber kein größeres Hinderniss für die Kompilierung sind.

Die Header wurden am Ende der BA vereinfacht und stark verändert, sodass sie nicht mehr unter der GPL stehen, sondern unter der MIT-X11-Lizenz. Die Header wurden von Tobias Brunner bereitgestellt und inkludiert. Das wurde getan, um die Änderungen ins Projekt eintragen zu können. Es stellt sich die Frage, ob es ausreicht, dass die nun benutzten Header so verschieden von denen aus dem Original sind, dass es lizenzrechtlich in Ordnung ist diese so in ein Projekt unter der MIT-X11-Lizenz zu integrieren. Das ist jedoch außerhalb des Aufgabengebiets der BA.

Code 1: win32.h-Headerdatei für den TAP-Windows6-Treiber

```
1  #ifndef WIN32_H
2  #define WIN32_H
3
4  #define TAP_WIN_COMPONENT_ID "tap0901"
5
6  #define ADAPTER_KEY
       "SYSTEM\\CurrentControlSet\\Control\\Class\\{4D36E972-E325-11CE-BFC1-08002BE10318}"
7
8  /*
9   * ======================
10  * Filesystem prefixes
11  * ======================
12  */
13
14 #define USERMODEDEVICEDIR "\\\\.\\Global\\"
15 #define TAP_WIN_SUFFIX     ".tap"
16
17 /*
18  * TAP IOCTL constants and macros.
19  *
20  */
21 #define TAP_WIN_CONTROL_CODE(request,method) \
22   CTL_CODE (FILE_DEVICE_UNKNOWN, request, method, FILE_ANY_ACCESS)
23
24 /* Present in 8.1 */
25
26 #define TAP_WIN_IOCTL_GET_MAC                 TAP_WIN_CONTROL_CODE (1,
       METHOD_BUFFERED)
27 #define TAP_WIN_IOCTL_GET_VERSION             TAP_WIN_CONTROL_CODE (2,
       METHOD_BUFFERED)
28 #define TAP_WIN_IOCTL_GET_MTU                 TAP_WIN_CONTROL_CODE (3,
       METHOD_BUFFERED)
29 #define TAP_WIN_IOCTL_GET_INFO                TAP_WIN_CONTROL_CODE (4,
       METHOD_BUFFERED)
30 #define TAP_WIN_IOCTL_CONFIG_POINT_TO_POINT TAP_WIN_CONTROL_CODE (5,
       METHOD_BUFFERED)
31 #define TAP_WIN_IOCTL_SET_MEDIA_STATUS       TAP_WIN_CONTROL_CODE (6,
       METHOD_BUFFERED)
```

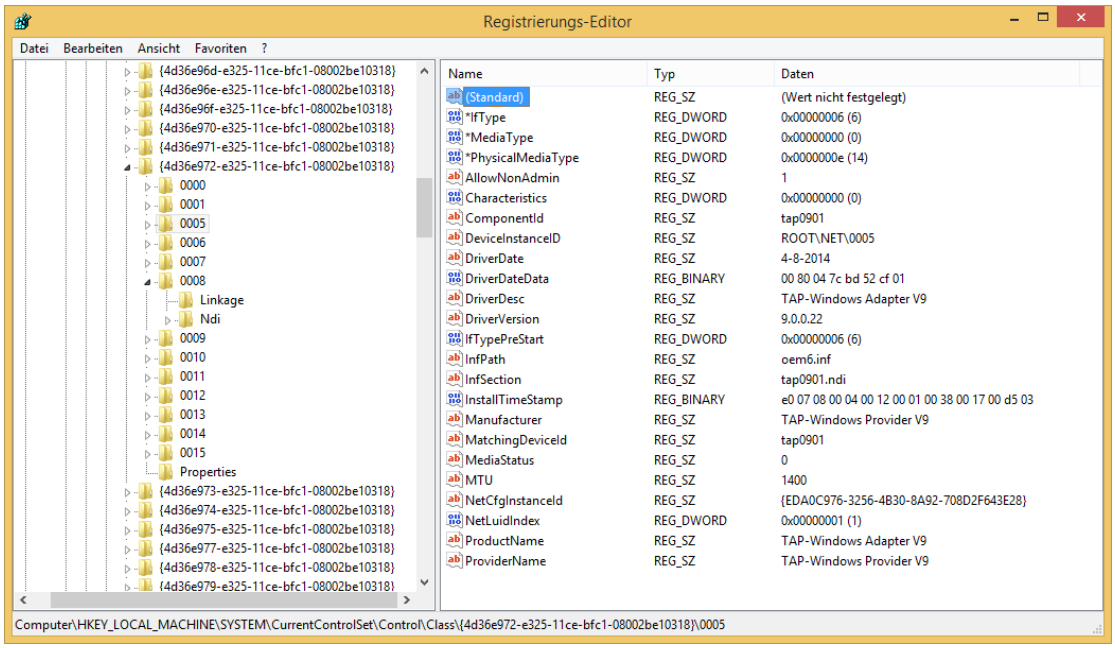
```
32 #define TAP_WIN_IOCTL_CONFIG_DHCP_MASQ       TAP_WIN_CONTROL_CODE (7,
       METHOD_BUFFERED)
33 #define TAP_WIN_IOCTL_GET_LOG_LINE           TAP_WIN_CONTROL_CODE (8,
       METHOD_BUFFERED)
34 #define TAP_WIN_IOCTL_CONFIG_DHCP_SET_OPT    TAP_WIN_CONTROL_CODE (9,
       METHOD_BUFFERED)
35
36 /* Added in 8.2 */
37
38 /* obsoletes TAP_WIN_IOCTL_CONFIG_POINT_TO_POINT */
39 #define TAP_WIN_IOCTL_CONFIG_TUN             TAP_WIN_CONTROL_CODE (10,
       METHOD_BUFFERED)
40 #define TAP_WIN_IOCTL_CONFIG_SET_SRC_CHECK   TAP_WIN_CONTROL_CODE (11,
       METHOD_BUFFERED)
41
42
43 #endif /* WIN32_H */
```

Die Headerdatei ”win32.h” definiert verschiedene Makros für die Kommunikation mit und Konfiguration des TAP-Windows6-Treibers.

Das Makro `TAP_WIN_COMPONENT_ID` mit Wert ’tap0901” wird genutzt, um in der Registry valide TAP-Geräte unter den Netzwerkverbindungen zu finden. Dabei wird der String-Wert `ComponentId` überprüft. Wenn sein Inhalt ”tap0901” ist, dann ist das Gerät ein valides TAP-Gerät.

Die existierenden Netzwerkadapter können unter HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Class\{4D36E972-E325-11CE-BFC1-08002BE10318} gefunden werden.

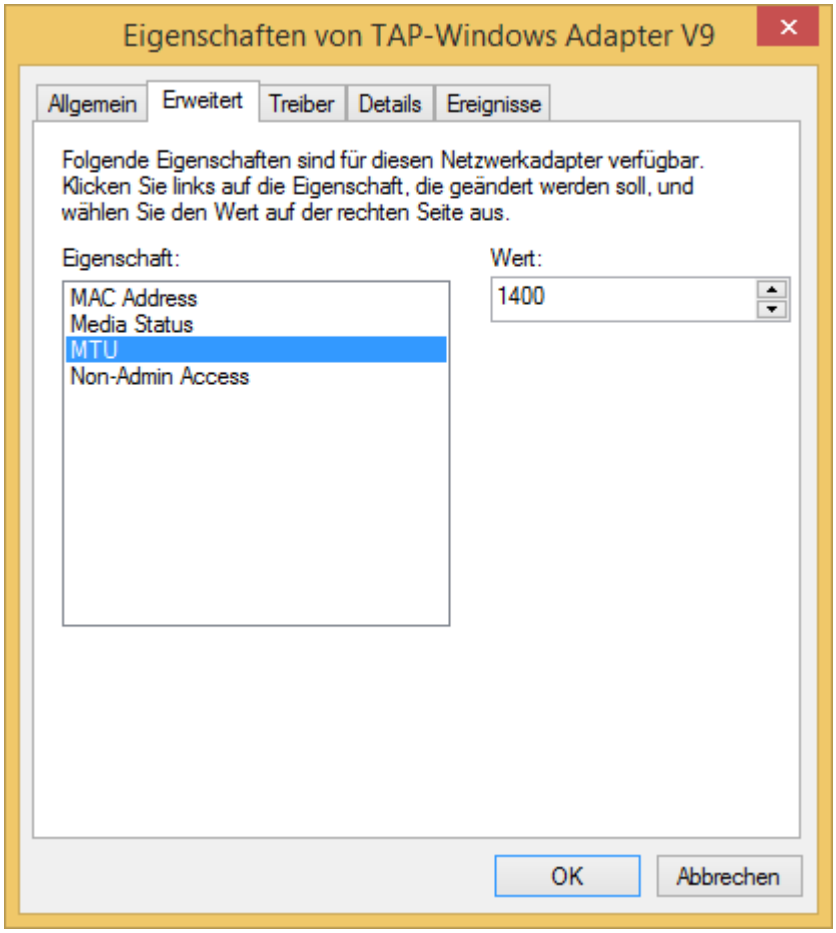
Abbildung 9: Registry-Eintrag über ein TAP-Gerät



In Abbildung 9 sind die verschiedenen Einträge eines Geräts unter dem oben genannten Pfad zu sehen beispielhaft an einem TAP-Gerät zu sehen. Das TAP-Gerät wurde mit dem TAP-

Windows6-Treiber erstellt.

Abbildung 10: Netzwerkverwaltung eines TAP-Geräts



In Abbildung 10 sind die vom Benutzer konfigurierbaren Optionen zu sehen. Sie werden als Werte in der Registry gespeichert und können auch dort verändert werden.

Die Makros USERMODEDEVICEDIR und TAP_WIN_SUFFIX werden zusammen mit der GUID des Geräts genutzt, um ein Handle im Geräteverzeichnis zu erstellen. Der sich daraus ergebende Pfad wird genutzt, um ein Handle zu erstellen, welches der TAP-Windows6-Treiber nutzt, um mit dem Programm im Userspace zu kommunizieren und umgekehrt.

Die Definitionen, die mit TAP_WIN_IOCTL beginnen definieren die Werte, die mit `DeviceIoControl` benutzt werden können, um das TAP-Gerät zu konfigurieren. Details über die verschiedenen Werte sind in Tabelle 1 zu finden.

4.4.2 kernel-libipsec

Die primäre Aufgabe hier war die Installation der Routen in `kernel_libipsec_ipsec.c` für Windows anzupassen, da hier ein Gateway verwendet wird auf einem TAP-Gerät statt ein echtes TUN-Gerät, sowie die Anpassung des Codes für die Ein- und Ausgabe und einige Methoden in `kernel_libipsec_router.c` da das Multiplexen der Handles, sowie die Benachrichtigung von anderen Threads auf Windows anders abläuft als auf Linux und UNIX.

Multiplexing Für das Multiplexen der Eingabe stehen unter Windows zwei Verfahren zur Verfügung:

- WaitForMultipleObjects
- IOCompletionPorts

WaitForMultipleObjects[31] funktioniert mit einem Array aus Handles. In der Struktur des Handles ist das event-Attribut auf ein einzgartiges Event gesetzt, welches genutzt wird um das Handle zu finden, dessen Lesevorgang abgeschlossen oder abgebrochen wurde. Nach dem Kopieren des Handles in das Array und dem Setzen des Events wird ein asynchroner Lesevorgang gestartet. Wenn er sofort beendet wurde, wird das entsprechende Event gesetzt. Dadurch beendet der Aufruf von WaitForMultipleObjects() nach dem Aufruf direkt, falls ein Lesevorgang schon zuvor erfolgreich war und der Programmcode wird etwas kürzer.

IOCompletionPorts[32] funktionieren, indem man einen IOCompletionPort mit CreateIoCompletionPort() anlegt und Handles mit ihm asoziiert. Bei der Asoziierung wird ein einzigartiger Schlüssel übergeben, der bei der Signalisierung wieder zurückgegeben wird um das Handle identifizieren zu können. Der CompletionPort kann erst nach dem Schließen der asoziierten Handles geschlossen werden. Mit der Funktion GetQueuedCompletionStatus() kann der ausführende Thread dann auf abgeschlossene I/O-Operationen warten. Die Nachteile dieser Methode sind, dass jegliche Operationen auf den Handles eine Benachrichtigung an GetQueuedCompletionStatus() generieren, obwohl Schreib-Vorgänge nicht von Interesse sind.

Des weiteren ähnelt die Nutzung von WaitForMultipleObjects() deren von poll() insoweit, dass die Handles/FDs auch nach der Benutzung mit der Funktion weiterhin einzeln genutzt werden können.

Da es relativ einfach ist WaitForMultipleObjects() zu nutzen, wurde diese Methode für die Implementierung von handle_plain() genutzt.

Die Zustände der Originalimplementierung sind in Abbildung 11 dargestellt.

Wie aus dem Diagramm hervorgeht, wird im Prinzip nur ein Array mit Strukturen des Typs `pollfd` erstellt, dann mit einem Notification-FD und den FDs der TUN-Geräte befüllt und als gewünschte Events `POLLIN` gesetzt. Das heißt, dass der Aufruf von poll() ohne Fehler beendet wird, wenn Daten auf dem FDs zur Verfügung stehen. Danach wird poll() auf die `pollfd`-Datenstruktur ausgeführt und analysiert, auf welchen FDs Daten zur Verfügung stehen. Dieser Code ist relativ kurz im Vergleich zum Analogon mit WaitFor*Objects() Funktionen unter Windows. Dies geht aus den vergleichbaren Implementierungsmöglichkeiten, die sich aus den `WaitForSingleObject()`- und `WaitForMultipleObjects`-Funktionen ergeben. Dies wird im Diagramm Abbildung 13 und Abbildung 14 hervor. Diese Diagramme haben weitaus mehr Zustände als Abbildung 11.

Der Code für die Implementierung von `handle_plain()` auf Windows ist 239 Zeilen lang. Daher wurde darauf verzichtet, sie hier in den Text zu kopieren. Stattdessen ist sie im Appendix auf Seite 64 zu finden. Abbildung 14 stellt die Zustände in der Implementierung von `handle_plain()` unter Windows dar.

Die Funktionen zum Starten von asynchronen Leseoperationen auf Handles wurden in `start_read()` ausgelagert und das Übersetzen von Fehlercodes in menschenlesbare Texte in `format_error()`.

[31] 16k.
[32] 16b.

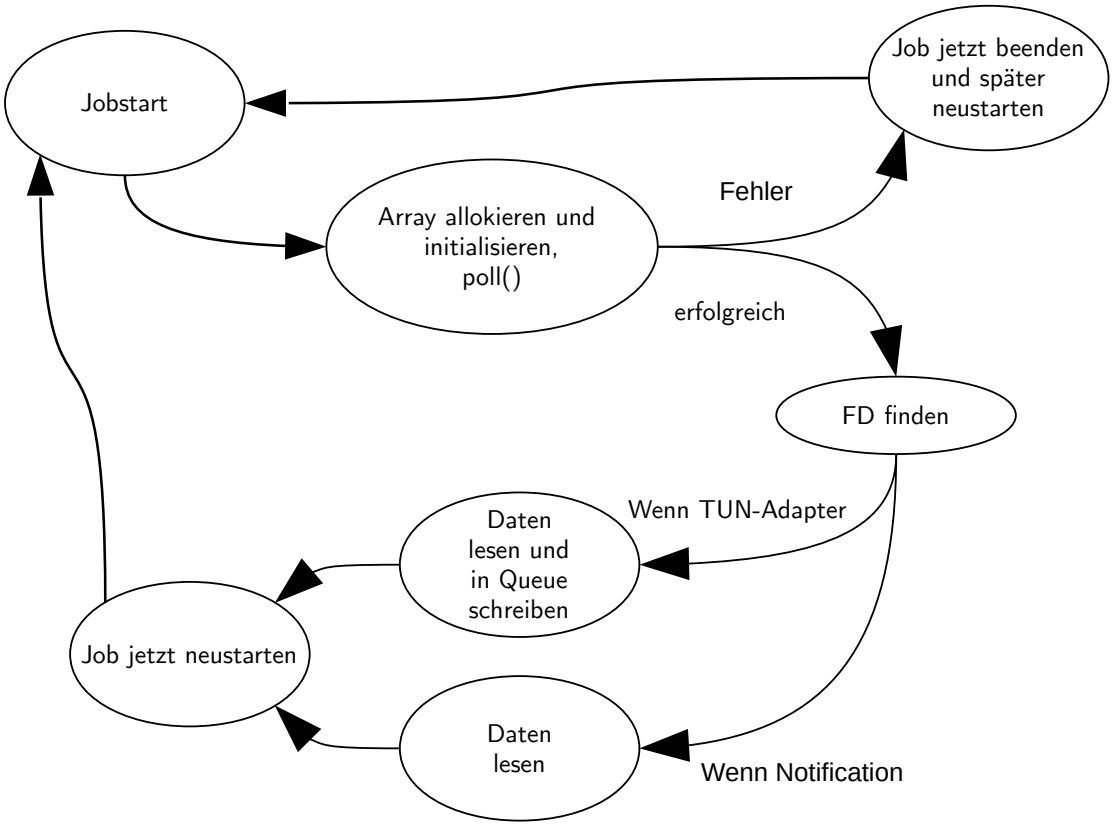


Abbildung 11: Zustände in handle_plain() mittels poll()

Bei Fehlern beim Lesen startet sich der Job mit JOB_REQUEUE_FAIR neu, sodass der Job als letzter vom Job-Manager in `charon` neugestartet wird, wenn hoffentlich der Fehler nicht erneut auftritt.

Die Funktion `start_read()` ist in Code 2 abgebildet. Sie startet eine asynchrone Leseoperationen mit dem übergebenen `handle_overlapped_buffer_t`-Objekt und überprüft den Rückgabewert von `ReadFile()`. Sie gibt einen booleschen Wert zurück, bei dem True einen Erfolg signalisiert und False einen Fehlschlag.

Code 2: Code von start_read()

```
1  /*
2   * Enqueue a Read operation on the given handle with the given struct
3   * @return: bool
4   */
5  static BOOL start_read(handle_overlapped_buffer_t *structure, HANDLE event)
6  {
7          DWORD error;
8          BOOL status;
9          /* Initialise read with the allocate overwrite structure */
10         status = ReadFile(structure->fileHandle, structure->buffer.ptr,
11                 structure->buffer.len, NULL, structure->overlapped);
12         error = GetLastError();
13         if (status)
14         {
15             /* Read returned immediately */
16             /* We need to signal the event ourselves */
```

```
17              SetEvent(event);
18              return TRUE;
19          }
20          else
21          {
22
23              switch(error)
24              {
25                  case ERROR_SUCCESS:
26                  case ERROR_IO_PENDING:
27                  {
28                      /* all fine */
29                      return TRUE;
30                      break;
31                  }
32                  default:
33                  {
34                      char *error_message = format_error(error);
35                      DBG2(DBG_ESP, "Error %d.", format_error);
36                      free(error_message);
37                      return FALSE;
38                      break;
39                  }
40              }
41
42          }
43  }
```

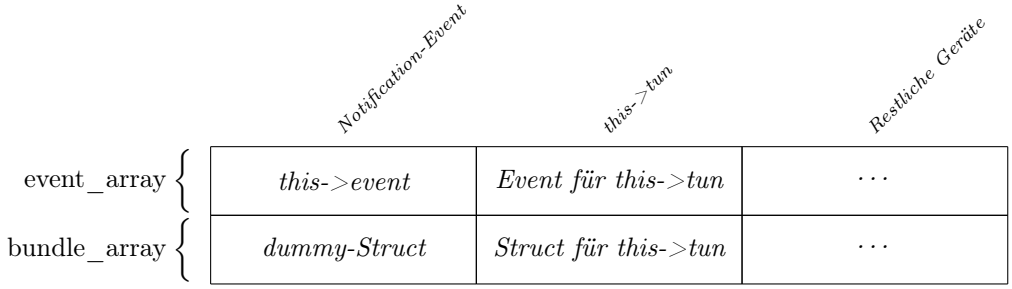
Die Funktion `format_error()` ist in Code 3 abgebildet und formatiert nur einen Fehler, der mit `GetLastError()` abgefragt wurde und ruft die entsprechende Fehlermeldung aus der Fehlerliste in Windows ab. Dabei wird die Fehlermeldung auf dem Heap allokiert und der Pointer auf sie an die aufrufende Anwendung zurückgegeben.

Code 3: Code für format_error()

```
1  /*
2   * Formats an error message based on error. Takes info from system.
3   * @return formatted error message
4   */
5  static char* format_error(DWORD error)
6  {
7              char *lpMsgBuf = NULL;
8              FormatMessage(
9                  FORMAT_MESSAGE_FROM_SYSTEM |
10                 FORMAT_MESSAGE_IGNORE_INSERTS,
11                 NULL,
12                 error,
13                 MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
14                 (LPTSTR) &lpMsgBuf,
15                 0,
16                 NULL);
17             return lpMsgBuf;
18 }
```

Der Unterschied zwischen der Implementierung von Abbildung 13 und Abbildung 14 ist,

Abbildung 12: Arrays aus handle_plain()



dass Abbildung 14 effektiver und schneller ist, da die Puffer nach jedem Lesevorgang beibehalten werden und IO-Operationen weiterlaufen können. In Abbildung 13 werden die Puffer und Operationen immer freigegeben und gestoppt. Das ist unnötig, bringt die Implementierung jedoch näher an das Verhalten von Abbildung 11. Es wurde das Verfahren aus Abbildung 14 implementiert, da es schneller und effektiver ist.

In `handle_plain()` werden zwei verschiedene Arrays benutzt, deren Strukturen in Abbildung 12 dargestellt sind.

bundle_array Ein Array aus Strukturen vom Typ `handle_overlapped_buffer_t`.

event_array Ein Array aus Events (HANDLE).

Alle Strukturen vom Typ `handle_overlapped_buffer_t` beinhalten jeweils ein Objekt vom Typ `HANDLE`, ein Objekt vom Typ `*OVERLAPPED` und ein Objekt vom Typ chunk_t. Das Handle gehört zu einem TUN-Gerät, das `OVERLAPPED`-Objekt wird für asynchrone IO-Operationen benötigt und das Objekt vom Typ chunk_t beinhaltet den Pufferspeicher und dessen Länge für die asynchronen Lesevorgänge. Die Puffer und die `OVERLAPPED`-Objekte müssen während der Lesevorgänge in zugänglichen Speicherbereichen sein.

Das `event_array` Array zum Multiplexen mittels WaitForMultiplEvents() genutzt und beinhaltet die gleichen Events, wie die Strukturen des Typs OVERLAPPED. Bei der Ausführung von `WaitForMultipleObjects()` auf das Array wird darauf gewartet, dass eine vorher gestartete Leseoperation auf einem Handle fertig gestellt wird. Das signalisiert ein Event im Array und dessen Position im Array verrät welche Position im bundle_array einen nun gefüllten Puffer hat. Die Position signalisiert hierbei nur die niedrigste Position im Array, deren Event signalisiert wurde. Ein im Array höher gelegenes Event könnte auch signalisiert sein. Der Puffer kann nun gelesen werden und an die Queue zu libipsec gehängt werden. Nach dem Lesen wird die OVERLAPPED-Struktur und das Event zurückgesetzt, sowie der Puffer mit Nullen initialisiert und die Länge zurückgesetzt. Danach wird ein neuer Lesevorgang auf dem oder den Handles gestartet, deren Leseoperationen fertig sind.

Wenn das Array nicht mittels `WaitForSingleObject()` nach signalisierten Events durchsucht wird, so ist es theoretisch möglich, dass beim Verlassen der Funktion `WaitForMultipleObjects()` mehrere Events signalisiert sind und nur das Event mit der niedrigsten Position im Array zurückgesetzt und das Paket gelesen wird. Das bewirkt, dass nur Pakete vom jeweils niedrigsten Handle gelesen werden.

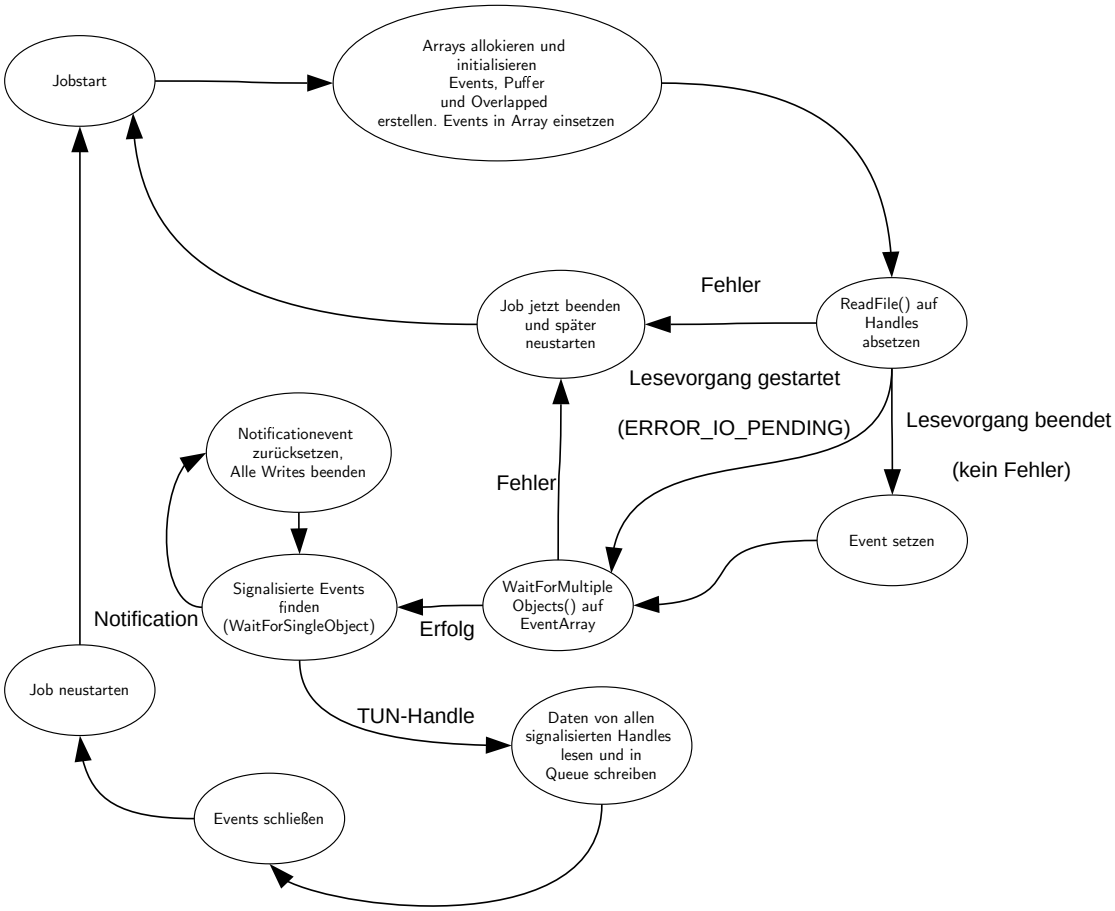


Abbildung 13: Zustände in handle_plain() mittels WaitForMultipleObjects(), Variante 1

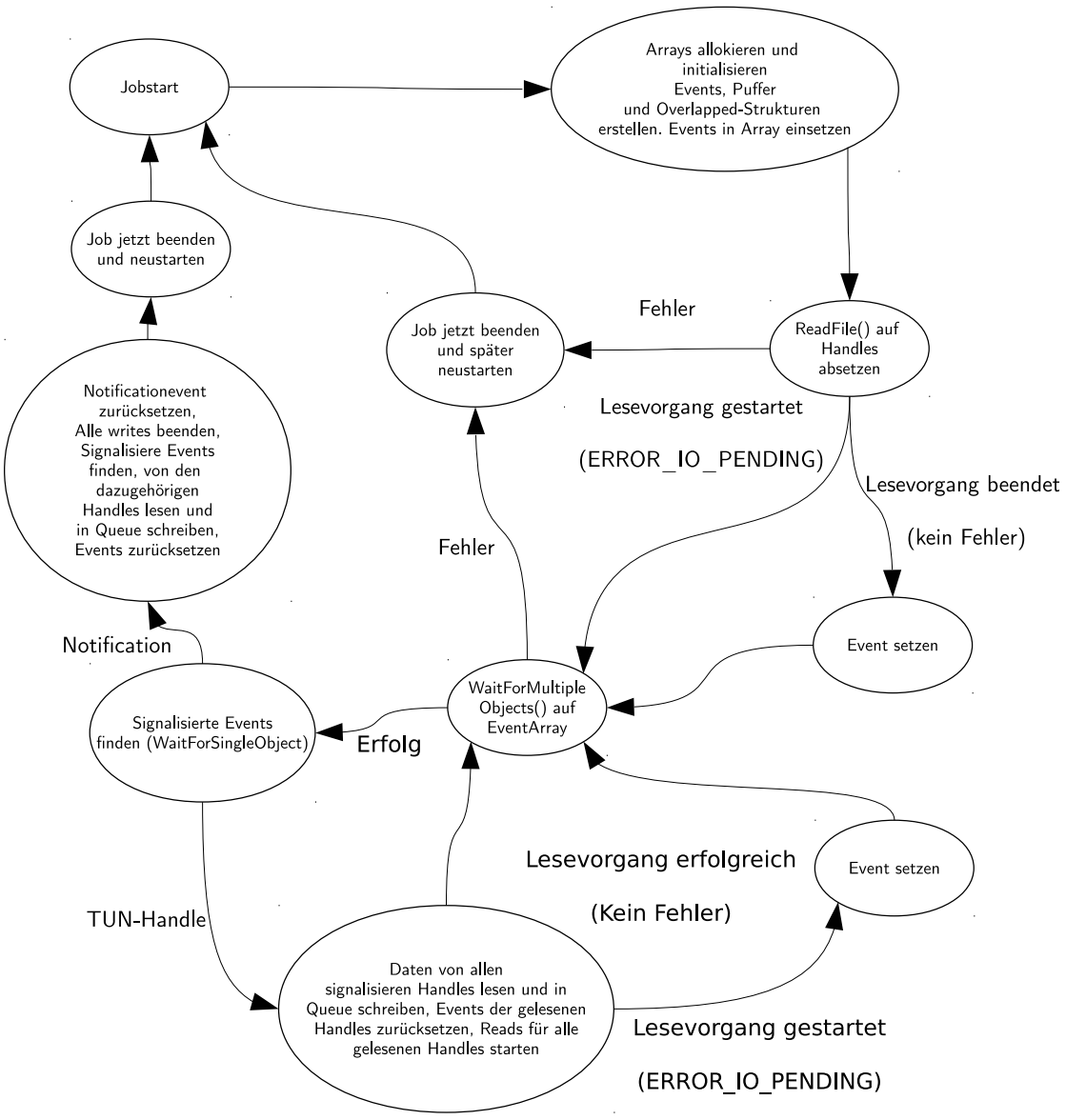


Abbildung 14: Zustände in handle_plain() mittels WaitForMultipleObjects(), Variante 2

Routing libipsec setzt für die von ihr installierten Routen standardmäßig das ”Gateway” oder ”Next Hop”Feld nicht. Der Grund dafür ist, dass libipsec bisher nur auf Betriebssytemen genutzt wurde, die echte Layer-3-Geräte implementieren und daher keine Kollisionsdomäne über das Gerät erreichbar ist. Daher ergibt es keinen Sinn das ”next hop”-Feld zu setzen. Auf Windows ist dies jedoch, wie zuvor erläutert, nicht der Fall, da der TAP-Windows-Treiber TUN-Geräte als Ethernet-Gerät emuliert. Der Struct `route` is vom Typ `route_entry_t` und wird in der Funktion `install_route` genutzt, um die zu installierende Route zu bestimmen. Das Gateway-Feld wird genutzt, um das Next-Hop-Feld der Routen im Routing-Table zu setzen. Der TAP-Windows6-Treiber nutzt für IPv6 einen statische IP addresse im link-local-Bereich der IPv6-Spezifikation. Für IPv4 lässt sich die IP-Adresse des Adapters implizit konfigurieren, wie zuvor erklärt. Das Object, welches mittels `host_create_from_string` erstellt wird, wird von der Funktion auf dem Heap abgelegt. Damit ist es auch nach dem Verlassen des Funktionsblocks noch valide. Es wird automatisch beim Zerstören des `route`-Objekts freigegeben.

Code 4: Patch für die Routen-Installation von libipsec

```
#ifdef __linux__
#elif defined(WIN32)
        /* Set out special gateway */
        family = route->src_ip->get_family(route->src_ip);
        switch(family)
        {
            case AF_INET:
            {
                /* For IPv4, the nxt hop is 169.254.128.128 (Configured next
                    hop) */
                host_t *gw = host_create_from_string("169.254.128.128", 0);
                route->gateway = gw;
                break;
            }
            case AF_INET6:
            {
                /* For IPv6, the next hop is fe80::8 (TAP-Windows6 magic router
                    gw) */
                host_t *gw = host_create_from_string("fe80::8", 0);
                route->gateway = gw;
                break;
            }
            default:
                DBG2(DBG_ESP, "Unknown Protocol family %d encountered. Not
                    setting a next hop.", family);
                break;
        }
#else
        /* on Linux we cant't install a gateway */
        route->gateway = charon->kernel->get_nexthop(charon->kernel, dst, -1,
            src);
#endif
```

Das Plugin ”kernel-iph” installiert Routen mit einer Metrik von 10. Andere Software von Windows, die Routen installiert, benutzen eine höhere Metrik.

Routen mit einer Prefixlänge von 0 werden auf zwei Routen mit der Prefixlänge 1 aufgeteilt.

Dies wird gemacht um die Standardroute, welche ebenfalls eine Metrik von 10 besitzt, nicht zu überschreiben.

Die Implementierung von `handle_plain()` auf Windows unterstütz, wie die Implementierung auf Linux und UNIX das Multiplexen von IO von mehreren TUN-Adaptern. Der Grund dafür ist, dass verschiedene TUN-Implementierungen gegebenenfalls verschiedene Adapter für mehrere Tunnel benötigen. Mit dem TAP-Windows-Treiber ist dies zwar nicht der Fall, es wurde der Vollständigkeit halber jedoch mitemplementiert.

4.4.3 libstrongswan

Hier galt es das Öffnen, Konfigurieren und Schließen von TUN-Geräten mit dem TAP-Windows-Treiber zu implementieren.

Der Treiber ist kompiliert auf der OpenVPN-Seite verfügbar[33] und der Quellcode auf Github[34]. Die Basis für die Implementierung war hier der existierende Programmcode in OpenVPN, spezifisch in der Datei "src/openvpn/tun.c.".

TAP-Geräte erstellen Das Erstellen von TAP-Geräten ist im Moment nicht unterstützt. Der Grund dafür ist, dass die Funktionalität von `devcon.exe` dafür nachgebaut werden müsste. Des weiteren ist der Quellcode von `devcon.exe` zwar verfügbar[35], aber nur unter einer proprietären Lizenz von Microsoft, der MS-PL[36], verfügbar. Eine Reimplementierung müsste Code aus dem Quellcode von `devcon.exe` nutzen, welcher dann auch unter der MS-PL stünde. Die Verbreitung des Codes wäre dann nicht möglich, da die MS-PL mit der GPL inkompatibel ist[37]. Da der Code unter der MIT-Lizenz beim strongSwan-Projekt eingereicht werden muss und dann dual-lizensiert wird. Wenn nur der Zugriff auf die benötigte API benutzt werden würde, so wäre es möglich, die Unterstützung zu implementieren. Aus Zeitgründen wurde das nicht getan.

TAP-Geräte suchen Um das TAP-Gerät zu nutzen versucht der Code zuerst nach Netzwerkgeräten mit der ID "tap0901" zu suchen, welche die ID für TAP-Geräte ist. Dies wird über die Registry getan. Hierbei wird im Pfad

```
HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Control\\Class\\{4D36E972-E325-11CE-BFC1-08002BE10318
```

nach Verbindungen gesucht, deren "ComponentId"-Feld den Wert "tap0901" besitzen. Bei passenden Schlüsseln wird der Wert "NetCfgInstanceId" an eine linked list angehängt. Jeder Netzwerkadapter unter Windows wird mittels seiner NetCfgInstanceId identifiziert. Vor dem Beenden der Funktion wird die linked list zurückgegeben. Diese Funktionalität ist in der Funktion `get_tap_reg()` in der Datei "src/libstrongswan/networking/tun_device.c" gekapselt.

Der Code hierfür ist im Appendix unter Code 31 aufzufinden.

TAP-Geräte öffnen TAP-Geräte sind im Usermode über den Globalen Addressraum für Userspaceadapter unter "\\.\\Global\\$NetCfgInstanceId.tap" erreichbar. `$NetCfgInstanceId` steht hierbei für die vorher mit `get_tap_req()` gefundenen `NetCfgInstanceId`-Werte der gefundenen TAP-Adapter.

[33] `https://openvpn.net/index.php/open-source/downloads.html`
[34] `https://github.com/OpenVPN`
[35] `https://github.com/Microsoft/Windows-driver-samples/tree/master/setup/devcon`
[36] `https://github.com/Microsoft/Windows-driver-samples/blob/master/LICENSE`
[37] 16c.

```
CreateFile(device_path, GENERIC_READ \| GENERIC_WRITE, 0,
                    0, OPEN_EXISTING, FILE_ATTRIBUTE_SYSTEM |
                        FILE_FLAG_OVERLAPPED, 0);
```

Das Handle kann nun genutzt werden um Pakete vom TAP-Gerät zu lesen und zu schreiben. Mittels DeviceIoControl kann die Konfiguration des Geräts geändert werden, wie die IP des virtuellen Routers, den Status des Geräts sowie der Modus des Geräts.

Für die Portierung wurden die FDs für Handles ausgetauscht und für das Benachrichtigen über Änderungen in der Liste der TUN-Geräte wurde ein Event verwendet.

Für das Lesen und Schreiben vom und auf dem Handle wird ein Event benötigt, welches in die OVERLAPPED-Struktur eingefügt wird. Das Event wird dynamisch mittels `CreateEvent()` erstellt und nach der Nutzung mittels `CloseHandle()` geschlossen.

Ändern der MTU eines TAP-Geräts Der Treiber ermöglicht es, die MTU über die Windows-Registry zu ändern. Dort wird der Schlüssel "MTU" dafür genutzt. Er steht standardmäßig auf "1500", welches die Standard-MTU von Ethernet ist. Alternativ ist es vermutlich möglich, die Funktionen `SetIpInterfaceEntry`[38] dafür zu nutzen, welche Teil der IP-Helper-API ist. Es ist zu beachten, dass der MTU-Eintrag in der Registry nur beim Erstellen des Geräts ausgelesen werden. Daher bewirkt das Ändern dieses Eintrags nur eine Änderung der MTU bei einem Neustart des Computers.

Konfiguration eines TAP-Geräts Die Konfiguration eines TAP-Geräts geschieht mit den IOCTL-Werten aus /autoreffig:tap-ioctls.

Im Funktionskopf von `init_tun` wird das Argument `name_tmpl` übergeben, welches auf Linux/UNIX den Anfang des Gerätenamens angibt, welcher dort `ipsec` ist. Da das Erstellen von TAP-Geräten auf Windows in charon (noch) nicht implementiert ist, wird es ignoriert.

Um ein TAP-Gerät erfolgreich zu konfigurieren muss zuerst eines gefunden werden. Das wird mit der Funktion `find_tap_devices` getan. Sie sucht in der Registry, wie im Paragraph "TAP-Geräte suchen" beschrieben. Nachdem alle TAP-Geräte gefunden wurden, wird versucht im globalen Geräteverzeichnis ein Handle auf dem Gerät zu öffnen. Der Pfad zum Gerät wird hierbei aus dem globalen Geräteverzeichnis, der GUID des Geräts, sowie der Endung ".tap" zusammengebaut. Danach werden die folgenden Konfigurationsschritte ausgeführt.

- Das Gerät wird in den TUN-Modus setzen. Dabei werden als Parameter die lokale IP (169.254.128.127), die IP des virtuellen Routers (169.254.128.128) und die Netzmaske des entfernten Netzwerks (255.255.255.255) gesetzt. `(TAP_WIN_IOCTL_CONFIG_TUN)`
- Die Überprüfung des "Sender protocol address"-Feldes muss deaktiviert werden. `(TAP_WIN_IOCTL_CONFIG_SET_SRC_CHECK)`
- Der Adapter muss in den "up"-Zustand gesetzt werden (einschalten). Nach dem Setzen des Zustands muss kurz gewartet werden um sicher zu gehen, dass das Gerät auch aktiv ist. `(TAP_WIN_IOCTL_SET_MEDIA_STATUS)`

[38] 16i.

Code 5: Konfiguration eines TAP-Geräts

```
1  /**
2   * Initialize the tun device
3   */
4  static bool init_tun(private_tun_device_t *this, const char *name_tmpl)
5          [...]
6  #elif defined(WIN32)
7          enumerator_t *enumerator;
8          char *guid;
9          BOOL success = FALSE;
10         /* Get all existing TAP devices */
11         linked_list_t *possible_devices = find_tap_devices();
12         memset(this->if_name, 0, sizeof(this->if_name));
13
14         /* Iterate over list */
15         enumerator = possible_devices->create_enumerator(possible_devices);
16         /* Try to open that device */
17         while(enumerator->enumerate(enumerator, &guid))
18         {
19             if (!success){
20                 /* Set mode */
21                 char device_path[256];
22                 /* Translate dev name to guid */
23                 /* TODO: Fix. device_guid should be */
24                 snprintf (device_path, sizeof(device_path), "%s%s%s",
                       USERMODEDEVICEDIR, guid, TAP_WIN_SUFFIX);
25
26                 this->tunhandle = CreateFile(device_path, GENERIC_READ |
                       GENERIC_WRITE, 0,
27                     0, OPEN_EXISTING, FILE_ATTRIBUTE_SYSTEM |
                           FILE_FLAG_OVERLAPPED, 0);
28                 if (this->tunhandle == INVALID_HANDLE_VALUE)
29                 {
30                     DBG1(DBG_LIB, "could␣not␣create␣TUN␣device␣%s", device_path);
31                 }
32                 else
33                 {
34                     memcpy(this->if_name, guid, strlen(guid));
35                     success = TRUE;
36                 }
37             }
38             else
39             {
40                 break;
41             }
42             /* device has been examined or used, free it */
43             free(guid);
44         }
45
46         /* possible_devices has been freed while going over the enumerator.
47          * Therefore it is not necessary to free the elements in the list now.
48          */
49         enumerator->destroy(enumerator);
50         possible_devices->destroy(possible_devices);
51         /* If we didn't find one or could open one, we need to bail out.
```

```
52          * We currently can not create new devices.
53          */
54         if (!success)
55         {
56             return FALSE;
57         }
58         /* set correct mode */
59         /* We set a fake gateway of 169.254.254.128 that we route packets over
60          The TAP driver strips the Ethernet header and trailer of the Ethernet
                frames
61          before sending them back to the application that listens on the handle
                */
62         struct in_addr ep[3];
63         ULONG status = TRUE;
64         DWORD len;
65         /* Local address (just fake one): 169.254.128.127 */
66         ep[0].S_un.S_un_b.s_b1 = 169;
67         ep[0].S_un.S_un_b.s_b2 = 254;
68         ep[0].S_un.S_un_b.s_b3 = 128;
69         ep[0].S_un.S_un_b.s_b4 = 127;
70         /*
71          * Remote network. The tap driver validates it by masking it with the
                remote_netmask
72          * and then comparing hte result against the remote network (this value
                here).
73          * If it does not match, an error is logged and initialization fails.
74          * (local & remote_netmask ? local)
75          * The driver does proxy arp for this network and the local address.
76          */
77         /* We need to integrate support for IPv6, too. */
78         /* Just fake a link local address for now (169.254.128.128) */
79         ep[1].S_un.S_un_b.s_b1 = 169;
80         ep[1].S_un.S_un_b.s_b2 = 254;
81         ep[1].S_un.S_un_b.s_b3 = 128;
82         ep[1].S_un.S_un_b.s_b4 = 128;
83         /* Remote netmask (255.255.255.255) */
84         ep[2].S_un.S_un_b.s_b1 = 255;
85         ep[2].S_un.S_un_b.s_b2 = 255;
86         ep[2].S_un.S_un_b.s_b3 = 255;
87         ep[2].S_un.S_un_b.s_b4 = 255;
88
89         if(!DeviceIoControl (this->tunhandle, TAP_WIN_IOCTL_CONFIG_TUN,
90             ep, sizeof (ep),
91             ep, sizeof (ep), &len, NULL))
92         {
93             DBG1 (DBG_LIB, "WARNING: The TAP-Windows driver rejected a
                    TAP_WIN_IOCTL_CONFIG_TUN DeviceIoControl call.");
94         }
95
96         ULONG disable_src_check = FALSE;
97         if(!DeviceIoControl(this->tunhandle, TAP_WIN_IOCTL_CONFIG_SET_SRC_CHECK,
98                     &disable_src_check, sizeof(disable_src_check),
99                     &disable_src_check, sizeof(disable_src_check), &len, NULL))
100        {
101            DBG1 (DBG_LIB, "WARNING: The TAP-Windows driver rejected a
```

```
                TAP_WIN_IOCTL_CONFIG_SET_SRC_CHECK␣DeviceIoControl␣call.");
102         }
103         ULONG driverVersion[3] = {0 , 0, 0};
104         if(!DeviceIoControl(this->tunhandle, TAP_WIN_IOCTL_GET_VERSION,
105                 &driverVersion, sizeof(driverVersion),
106                 &driverVersion, sizeof(driverVersion), &len, NULL))
107         {
108             DBG1(DBG_LIB, "WARNING:␣The␣TAP-Windows␣driver␣rejected␣a␣
                    TAP_WIN_IOCTL_GET_VERSION␣DeviceIoControl␣call.");
109         }
110         else
111         {
112             DBG1(DBG_LIB, "TAP-Windows␣driver␣version␣%d.%d␣available.",
                    driverVersion[0], driverVersion[1]);
113         }
114         /* Set device to up */
115
116         if (!DeviceIoControl (this->tunhandle, TAP_WIN_IOCTL_SET_MEDIA_STATUS,
117                 &status, sizeof (status),
118                         &status, sizeof (status), &len, NULL))
119         {
120             DBG1 (DBG_LIB, "WARNING:␣The␣TAP-Windows␣driver␣rejected␣a␣
                    TAP_WIN_IOCTL_SET_MEDIA_STATUS␣DeviceIoControl␣call.");
121         }
122
123             /* Give the adapter 2 seconds to come up */
124
125         sleep(2);
126         return TRUE;
127         [...]
128 }
```

Lesen und Schreiben vom Adapter Beim Öffnen des Adapters wurde ein Handle erstellt. Durch asynchrone Lese- und Schreiboperationen mittels `ReadFile()` und `WriteFile()` unter Benutzung von `OVERLAPPED`-Strukturen und Events lassen sich Pakete lesen und schreiben.

Die Methoden hierfür wurden in `tun_device.c` implementiert und nutzen einfache Event-basierte asynchrone IO-Operationen mittels `ReadFile()` und `WriteFile()`. Die Implementierung für `read_packet` ist in Code 6 zu sehen.

Die benötigten Variablen werden auf dem Stack allokiert, da sie nach dem Lesen des Pakets nicht mehr benötigt werden. Dabei wird ein Putter für das Paket in Größe der MTU des Geräts allokiert. Ein einzelnes Event wird für die asynchrone Leseoperation benötigt, welches ins Feld `hevent` der Struktur eingesetzt wird. Es wird ohne Name erstellt, da es nicht zur Benachrichtigung oder Synchronisation von mehreren Threads genutzt wird. Nach dem Starten des Lesevorgangs mittels `ReadFile` wird auf die Benachrichtigung des Events gewartet. Wenn ein Paket gelesen wurde, wird es mittels `chunk_clone` auf den Heap kopiert und in das im Funktionskopf übergebene Objekt vom Typ `chunk_t` geschrieben. Da das Warten auf das Event blockierend ist, wird dort ein Thread-Cancelation-Punkt gesetzt, damit der Thread dort gegebenenfalls sicher beendet werden kann.

Code 6: Code für read_packet

```
1  METHOD(tun_device_t, read_packet, bool,
2          private_tun_device_t *this, chunk_t *packet)
3  {
4          bool old, status;
5          DWORD error;
6          OVERLAPPED overlapped;
7              chunk_t data;
8          HANDLE read_event = CreateEvent(NULL, FALSE, FALSE, FALSE);
9
10         ResetEvent(read_event);
11
12         error = GetLastError();
13         switch(error)
14         {
15             case ERROR_SUCCESS:
16                 break;
17             default:
18                 /* Just fine. Don't do anything. */
19                 break;
20         }
21
22         memset(&overlapped, 0, sizeof(OVERLAPPED));
23
24         overlapped.hEvent = read_event;
25
26             data = chunk_alloca(get_mtu(this));
27
28         /* Read chunk from handle */
29         status = ReadFile(this->tunhandle,
30                     &data.ptr,
31                     data.len,
32                     NULL,
33                     &overlapped);
34         error = GetLastError();
35
36         if (status)
37         {
38             /* Read returned immediately. */
39             SetEvent(read_event);
40         }
41         else
42         {
43             switch(error)
44             {
45                 case ERROR_SUCCESS:
46                 case ERROR_IO_PENDING:
47                     /* all fine */
48                     break;
49                 default:
50                     DBG1(DBG_LIB, "reading from TUN device %s failed: %u",
                            this->if_name,
51                          error);
52                     CloseHandle(read_event);
53                     return FALSE;
```

```
54                          break;
55                  }
56          }
57          old = thread_cancelability(TRUE);
58
59          WaitForSingleObject(read_event, INFINITE);
60
61          thread_cancelability(old);
62
63          *packet = chunk_clone(data);
64
65          CloseHandle(read_event);
66          return TRUE;
67  }
```

Beim Schreiben von Paketen ist ähnlich implementiert wie das Lesen von Paketen. Auch hier werden die Variablen auf dem Stack allokiert, ein Event erstellt und es in die `OVERLAPPED`-Struktur eingesetzt. Danach wird Der Schreibvorgang gestartet und auf seine Beendigung gewartet. Fehler werden abgefangen und angemessen behandelt. Da das Warten auf das Event blockierend ist, wird dort ein Thread-Cancelation-Punkt gesetzt, damit der Thread dort gegebenenfalls sicher beendet werden kann.

Code 7: Code für write_packet

```
1  METHOD(tun_device_t, write_packet, bool,
2          private_tun_device_t *this, chunk_t packet)
3  {
4          bool status;
5          DWORD error;
6          OVERLAPPED overlapped;
7          HANDLE write_event;
8
9          write_event = CreateEvent(NULL, FALSE, FALSE, NULL);
10         error = GetLastError();
11         if (error != ERROR_SUCCESS)
12         {
13            DBG1(DBG_LIB, "creating an event to write to the TUN device %s
                   failed: %d",
14                      this->if_name, error);
15            return FALSE;
16         }
17
18
19         memset(&overlapped, 0, sizeof(OVERLAPPED));
20
21         overlapped.hEvent = write_event;
22
23         status = WriteFile(
24                  this->tunhandle,
25                  packet.ptr,
26                  packet.len,
27                  NULL,
28                  &overlapped
29                  );
30         error = GetLastError();
```

```
31
32          if (status) {
33              /* Read returned immediately. */
34              SetEvent(write_event);
35          }
36          else
37          {
38              switch(error)
39              {
40                  case ERROR_SUCCESS:
41                  case ERROR_IO_PENDING:
42                      /* all fine */
43                      break;
44                  default:
45                      DBG1(DBG_LIB, "writing to TUN device %s failed: %u",
46                              this->if_name, error);
47                      CloseHandle(write_event);
48                      return FALSE;
49                      break;
50              }
51           }
52          WaitForSingleObject(write_event, INFINITE);
53
54          CloseHandle(write_event);
55
56          return TRUE;
57  }
```

tun_device_t Die Klasse für TUN-Geräte musste für Windows angepasst werden. Spezifisch mussten neue Funktionen zum Lesen und Schreiben implementiert werden, sowie Attribute geändert werden. Auf Windows wird ein Objekt vom Typ `HANDLE` genutzt, statt eines FD. Das `HANDLE` wird dabei von der Funktoon `init_tun()` gesetzt, welche beim Instanzieren der Klasse im Konstruktur aufgerufen wird.

Code 8: Code für das Erstellen von tun_device_t

```
1  tun_device_t *tun_device_create(const char *name_tmpl)
2  {
3          private_tun_device_t *this;
4
5          INIT(this,
6                  .public = {
7                          .read_packet = _read_packet,
8                          .write_packet = _write_packet,
9                          .get_mtu = _get_mtu,
10                         .set_mtu = _set_mtu,
11                         .get_name = _get_name,
12             /* For WIN32, that's a handle. */
13 #ifdef WIN32
14             .get_handle = _get_handle,
15 #else
16                         .get_fd = _get_fd,
17 #endif /* WIN32 */
18                         .set_address = _set_address,
```

```
19                          .get_address = _get_address,
20                          .up = _up,
21                          .destroy = _destroy,
22                  },
23 #ifdef WIN32
24          .tunhandle = NULL,
25 #else
26
27                  .tunfd = -1,
28                  .sock = -1,
29 #endif /* WIN32 */
30          );
31
32          if (!init_tun(this, name_tmpl))
33          {
34                  free(this);
35                  return NULL;
36          }
37 #ifdef WIN32
38          DBG1(DBG_LIB, "opened TUN device: %s", this->if_name);
39 #else
40          DBG1(DBG_LIB, "created TUN device: %s", this->if_name);
41
42          this->sock = socket(AF_INET, SOCK_DGRAM, 0);
43          if (this->sock < 0)
44          {
45                  DBG1(DBG_LIB, "failed to open socket to configure TUN device");
46                  destroy(this);
47                  return NULL;
48          }
49 #endif /* WIN32 */
50          return &this->public;
51 }
```

Code 9: Relevanter code für tun_device_t->destroy()

```
1 METHOD(tun_device_t, destroy, void,
2          private_tun_device_t *this)
3 {
4 #ifdef WIN32
5          /* close file handle, destroy interface */
6          CloseHandle(this->tunhandle);
7 #else
8          [...]
9 #ifdef __FreeBSD__
10         [...]
11 #endif
12         [...]
13 #endif
14         DESTROY_IF(this->address);
15         free(this);
16 }
```

Beim Destruktur für die Klasse muss nur statt dem Zerstören eines TUN-Geräts nur das Handle des TAP-Geräts geschlossen werden.

4.4.4 kernel-iph

kernel-iph ist ein Plugin für `charon`, das das Kernel-Interface für die Verwaltung von IP-Adressen und Routen implementiert.

Ursprünglich war das Verwalten von IP-Adressen im Master-Zweig nicht implementiert. Eine vollständige Implementierung existiert im Zweig `win-vip`, die in den Zweig 'windows-libipsec' gemerged wurde, um eine vollständige Implementierung dort zu erhalten.

Nachgerüstet werden musste Code zum Beachten des `charon.install_virtual_ip_on` Parameters der in der Datei `strongswan.conf` definiert werden kann und den libipsec nutzt um die Installation der empfangenen IP-Adresse(n) auf den richtigen Adapter zu erzwingen. Der Code kopiert den Adapternamen während der Initialisierung des Plugins, daher ändert ein Ändern des Parameters während der Laufzeit nicht die Schnittstelle, auf die die IP-Adresse installiert werden würde.

Die bekannten IP-Adressen werden in kernel-iph in einer Datenstruktur gespeichert, um sie schnell nachschlagen zu können und Berechnungen zu tätigen. Die Struktur wird einerseits durch manuelles Einfügen und Entfernen von selbstinstallierten Adressen bewerkstelligt, als auch durch Events, die die Anwendung über ein Handle vom Kernel empfängt.

Code 10: Ergänzung zu private_kernel_iph_net_t

```
1  /**
2   * Private data of kernel_iph_net implementation.
3   */
4  struct private_kernel_iph_net_t {
5          [...]
6          /**
7           * Whether to install virtual IPs
8           */
9          bool install_virtual_ip;
10
11         /**
12          * Where to install virtual IPs
13          */
14
15         char *install_virtual_ip_on;
16 };
```

Die Klasse für das Plugin `kernel-iph` wurde um zwei Attribute für die Installation von IP-Adressen erweitert. Einerseits ein boolescher Wert, der die Installation aktiviert und deaktiviert und andererseits einen `char *`, der genutzt wird um das Gerät zu konfigurieren, auf welchem alle IP-Adressen installiert werden sollen.

Code 11: Code für add_ip

```
1 METHOD(kernel_net_t, add_ip, status_t,
2     private_kernel_iph_net_t *this, host_t *vip, int prefix, char *name)
3 {
4          if (!this->install_virtual_ip)
5          {     /* disabled by config */
6              return SUCCESS;
7          }
8          MIB_UNICASTIPADDRESS_ROW row;
```

```
9           u_long status;
10          iface_t *iface = NULL, *entry = NULL;
11
12          /* Print out all known interfaces */
13          enumerator_t *enumerator = this->ifaces->create_enumerator(this->ifaces);
14          while(enumerator->enumerate(enumerator, &entry))
15          {
16              DBG1(DBG_KNL, "interface %s\n index: %d\n description %s\n type %d",
                    entry->ifname, entry->ifindex, entry->ifdesc, entry->iftype);
17          }
18          enumerator->destroy(enumerator);
19          /* name of the MS Loopback adapter */
20          if (!this->install_virtual_ip_on ||
                this->ifaces->find_first(this->ifaces, (void*)iface_by_name,
21                          (void**)&iface, this->install_virtual_ip_on) != SUCCESS)
22          {
23              name = "{DB2C49B1-7C90-4253-9E61-8C6A881194ED}";
24          }
25          else
26          {
27              name = this->install_virtual_ip_on;
28          }
29          host2unicast(vip, prefix, &row);
30
31          row.InterfaceIndex = add_addr(this, name, vip, TRUE);
32          if (!row.InterfaceIndex)
33          {
34              DBG1(DBG_KNL, "interface '%s' not found", name);
35              return NOT_FOUND;
36          }
37
38          status = CreateUnicastIpAddressEntry(&row);
39          if (status != NO_ERROR)
40          {
41              DBG1(DBG_KNL, "creating IPH address entry failed: %lu", status);
42              remove_addr(this, vip);
43              return FAILED;
44          }
45          return SUCCESS;
46  }
```

Die Methode `add_ip` in Code 11 des `kernel-iph`-Plugins musste angepasst werden. Ursprünglich war sie nur ein Stub, der `return NOT_SUPPORTED;` enthielt. Nach dem der Quellcode aus dem Zweig `win-vip` gemerged wurde, wurde der Code erweitert um die Attribute `install_virtual_ip_on` und `install_virtual_ip` zu beachten. Dabei wird zuerst überprüft, ob IP-Adressen überhaupt installiert werden sollen und danach, ob eine Schnittstelle konfiguriert ist. Wenn eine Schnittstelle konfiguriert ist, dann wird sie in der internen Schnittstellenliste von `kernel-iph` mittels `this->ifaces->find_first()` nachgeschlagen um zu überprüfen ob sie existiert. Wenn sie existiert, wird die IP-Adresse mittels `host2unicast` und `CreateUnicastAddressEntry` darauf installiert. Wenn sie nicht existiert, wird die IP-Adresse auf dem Loopback-Adapter installiert.

Die Methode `del_ip` in Code 12 wurde komplett aus dem Zweig `win-vip` übernommen und

bedarf keiner Veränderung. Sie übersetzt die übergebene IP in ein Objekt vom Typ `MIB_UNICASTIPADDRESS_ROW`, welches dann den Index der "virtuellen" IP-Adresse ermittelt und mithilfe von `DeleteUnicastIpAddressEntry()` gelöscht.

Code 12: Code für del_ip

```
METHOD(kernel_net_t, del_ip, status_t,
    private_kernel_iph_net_t *this, host_t *vip, int prefix, bool wait)
{
        if (!this->install_virtual_ip)
        {    /* disabled by config */
            return SUCCESS;
        }

        MIB_UNICASTIPADDRESS_ROW row;
        u_long status;

        host2unicast(vip, prefix, &row);

        row.InterfaceIndex = remove_addr(this, vip);
        if (!row.InterfaceIndex)
        {
            DBG1(DBG_KNL, "virtual IP %H not found", vip);
            return NOT_FOUND;
        }

        status = DeleteUnicastIpAddressEntry(&row);
        if (status != NO_ERROR)
        {
            DBG1(DBG_KNL, "deleting IPH address entry failed: %lu", status);
            return FAILED;
        }

        return SUCCESS;
}
```

Um die Variablen, der Klasse zu initialisieren, musste die Funktion `kernel_iph_net_create()` angepasst werden. Aufgrund der Unterstützung für die Installation von IP-Adressen müssen die neuen Attribute `install_virtual_ip_on` und `install_virtual_ip` initialisiert werden. Da beide Attribute ihre Werte aus dem Konfigurationssystem beziehen, werden diese mithilfe von `lib->settings->get_bool` und `lib->settings->get_str` initialisiert.

Code 13: Ergänzung zu kernel_iph_net_create()

```
*
 * Described in header.
 */
kernel_iph_net_t *kernel_iph_net_create()
{
        [...}
        .install_virtual_ip = lib->settings->get_bool(lib->settings,
                        "%s.install_virtual_ip", TRUE, lib->ns),
        .install_virtual_ip_on = lib->settings->get_str(lib->settings,
                        "%s.install_virtual_ip_on", NULL, lib->ns),
       [...]
```

```
12 }
```

Für das Implementieren des Codes für den TAP-Treiber werden Magic Numbers benötigt, welche sich im Quellcode von OpenVPN finden lassen. Diese wurden übernommen und in einer Header-Datei angelegt. Teilweise wurden auch die Konstanten für die Erzeugung der Namen der Events für das IO-Multiplexen dort abgelegt. Die Datei ist im Appendix unter Code 30 zu finden.

Der gesamte gepatchte Quellcode von strongSwan ist auf der CD unter strongSwan/ zu finden Der Patch selbst ist unter patches/strongSwan/tap-handling.patch zu finden.

4.5 TAP-Treiber

Im Zuge der BA wurde ein Patch für den TAP-Windows-Treiber entwicklelt, um die Prüfung der Quell-IP der ARP-Requests zu deaktivieren. Das ist erforderlich, um mit der virtuellen IP des Clients mit dem entfernten IP-Netzwerk kommunizieren zu können.

4.5.1 Bauen und Installieren des Treibers

Um den Treiber zu bauen, wird zuallererst der Quellcode benötigt, sowie die Vorraussetzungen, welche auf der Github-Seite dort aufgelistet sind. Die aktuellen Minimalvorraussetzungen sind Python 2.7, das Microsoft Windows 7 WDK (Windows Driver Kit) und ein Windows Code-Signing-Certificate. Das Zertifikat muss nicht valid sein, wenn der Treiber nicht produktiv ausgerollt werden soll. Der Grund ist, dass unter Windows das Überprüfen der Treibersignatur deaktiviert werden kann, aber der Treiber muss trotzdem signiert sein. Der Trust-path muss aber nicht zu einem öffentlichen Trust Anchor (eine CA) führen. Um den Treiber signieren zu können, wird die gesamte Trust Chain benötigt, sowie der private Schlüssel des Zertifikats. Um ein Zertifikat mit dem dazugehörigen privaten Schlüssel importieren zu können, müssen die Dateien in einen PKCS#12-Container gepackt werden. Dieser kann mit OpenSSL erstellt werden, zum Beispiel mit diesem Befehl:

Code 14: OpenSSL PKCS#12

```
1 openssl pkcs12 -export -in key key.key -in certificate.pem -out pkcs12.p12
```

Die mit diesem Befehl erstellte Datei pkcs12.p12 kann dann durch einen Doppelklick auf sie in den Windows importiert werden. Folgend ist ein beispielhafter Befehl zum Bau des Treibers.

Code 15: TAP-Windows bauen

```
1 python buildtap.py -c -b -d --cert="csc" --sign
    --crosscert="C:\Users\Noel\certs\rootca.pem"
```

Die Ausgabe des Befehls listet die Kompilierung, sowie das Signieren der Dateien auf. Wie zu sehen ist, muss das Root-CA-Zertifikat als Datei vorhanden sein. Das Code Signing certificate wird nur mit seinem Common Name angegeben. Damit das WDK es findet, muss es in "Eigene Zertifikate" zu finden sein.

Mit Windows im Testmodus kann der Treiber geladen werden und TAP-Geräte konnen mittels tapinstall.exe (Was eigentlich `devcon.exe` ist). Die Datei ist im Quellcode des Treibers nicht mitenthalten. Sie kann jedoch durch die Installation des Treiber-Pakets von der OpenVPN-Webseite erhalten werden. Nach der Installation des Pakets ist die Datei unter *C:\Program*

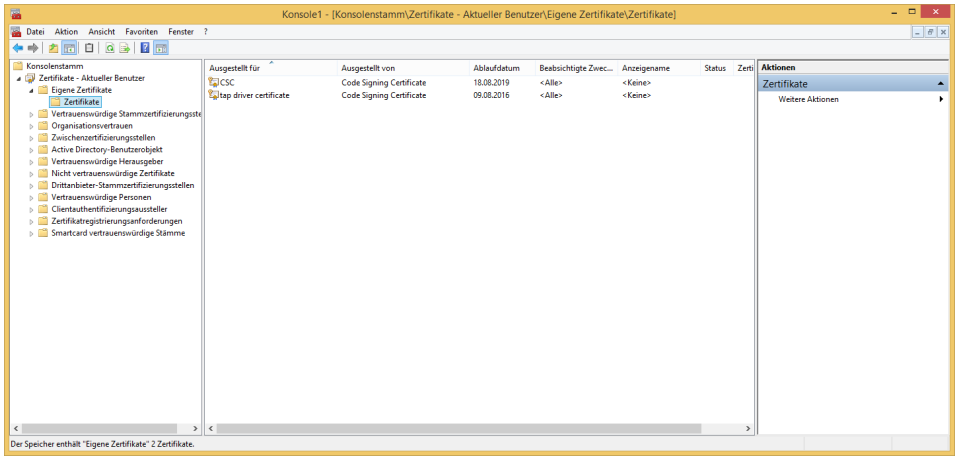


Abbildung 15: Eigene-Zertifikate-Menü

Files\TAP-Windows\bin\tapinstall.exe zu finden und kann zur Erstellung und zum Löschen von TAP-Geräten genutzt werden.

Der erste Parameter ist ”install” oder ”remove”. Der Pfad zur .inf-Datei über den Treiber muss beim Erstellen eines Geräts angegeben werden. Hier im Beispiel ist es die Datei, die mittels des Python-Skripts erstellt wurde. Bei der Installation dieses Treibers wird der kompilierte Treiber installiert, welcher sich unter `C:\Users\Noel\tap-driver\dist\amd64\OemVista.inf` befindet.

Code 16: Erstellung eines TAP-Geraets

```
1 "C:\Program Files\TAP-Windows\bin\tapinstall.exe" install
      "C:\Users\Noel\tap-driver\dist\amd64\OemVista.inf" tap0901
```

Code 17: Löschung aller TAP-Geraete

```
1 "C:\Program Files\TAP-Windows\bin\tapinstall.exe" remove tap0901
```

ABI Die ABI des Treibers ist undokumentiert. Im Folgenden ist eine Auflistung der Funktionen der ABI. Sie wurden aus der Datei ”src/device.c” extrahiert und aufgeschlüsselt. Für mehr Informationen bezüglich des Verhaltens und der genauen Eingabeparameter sollte die Datei konsultiert werden.

Patch Bei der Entwicklung des Patchs wurde viel Zeit darauf verwendet die funktionalen Komponenten zu finden, die bei der Verarbeitung von Address Resolution Protocol (ARP)-Requests ausgeführt werden, sowie die Codepfade zu finden, die bei der Verarbeitung von Daten abgelaufen werden.

Der entwickelte Patch deaktiviert die Überprüfung der Quell-IP der ARP-Requests, die vom Treiber verarbeitet werden. Wenn die Überprüfung erfolgreich war, so wird der ARP-Request beantwortet und dem Kernel wird damit mitgeteilt, wie die MAC-Adresse des virtuellen Routers lautet.

Für die Implementierung der Routen wurde die Nutzung eines virtuellen Routers gewählt, da das Routen aller IP-Adressen über das Gerät die ARP-Tabelle des Clients mehr gefüllt hätte und eine ARP-Adressabfrage bei jeder Kommunikation mit einer neuen IP-Adresse zu Folge hätte.

IOCtl	Makro	Eingabe	Ausgabe	Zweck	Kommentar
1	TAP_WIN_IOCTL_GET_MAC	NULL	char[6]	Gibt die MAC-Adresse des Geräts zurück	
2	TAP_WIN_IOCTL_GET_VERSION	NULL	ULONG[3]	Gibt die Version des Treibers zurück	
3	TAP_WIN_IOCTL_GET_MTU	NULL	ULONG	Gibt die MTU des Geräts zurück	
4	TAP_WIN_IOCTL_GET_INFO	NULL	N/A	Gibt Informationen über den Adapter zurück	Ist laut Code für NDIS6 nicht implementiert
5	TAP_WIN_IOCTL_CONFIG_POINT_TO_POINT	IPADDR[2] (2*4 CHAR)	NULL	Setzt das Gerät in den Point-To-Point-Modus	
6	TAP_WIN_IOCTL_SET_MEDIA_STATUS	ULONG	NULL	Setzt das Gerät in den "Up" oder "Down"-Zustand	
7	TAP_WIN_IOCTL_CONFIG_DHCP_MASQ	IPADDR[4] (4*4 CHAR)	NULL	Aktiviert den internen DHCP-Server, DHCP-IP-Adresse, DHCP-Netzmaske, DHCP-Server-IP, Leasetime	
8	TAP_WIN_IOCTL_GET_LOG_LINE	allokierter String (char *)	NULL	Gibt eine Debug-Log-Zeile zurück	
9	TAP_WIN_IOCTL_CONFIG_DHCP_SET_OPT	char[256]	NULL	Setzt die DHCP-Optionen	
10	TAP_WIN_IOCTL_CONFIG_TUN	IPADDR[3] (3*4 char)	NULL	Setzt das Gerät in den TUN-Modus, lokale IP, entferntes Netzwerk, entfernte Netzmaske	
11	TAP_WIN_IOCTL_CONFIG_SET_SRC_CHECK	ULONG	NULL	Deaktiviert oder Aktiviert den ARP-Source-Check. "0" deaktiviert ihn. "1" aktiviert ihn (Standard)	

Tabelle 1: TAP-Windows-Treiber IOCtls

Der Patch wurde entwickelt, indem nach der Behandlung von ARP-Paketen im Quellcode des Treibers gesucht wurde. Dafür wurde nach "ARP" und "arp" mittels "grep" im Quelltext gesucht. Dabei wurde die Funktion "ProcessARP" in "src/txpath.c" gefunden, in der ARP-Requests verarbeitet werden. In dieser Funktion werden verschiedene Felder des Pakets überprüft, unter anderem das "Sender protocol address"-Feld.

Die Überprüfung des "Sender protocol address"-Felds wurde optional gemacht, wobei es standardmäßig aktiviert ist. Die Kontrolle dieses Feldes kann über ein IOCTL-Wert gesteuert werden. Der momentane Zustand der Option wird im Adapter-Kontext des TAP-Adapters gespeichert.

Er wurde 2016-08-15 in der OpenVPN Community auf freenode im Kanale #openvpn-meeting besprochen und bekam ein feature-ack. Ein Code-ack steht noch aus. Das Ticket ist unter `https://community.openvpn.net/openvpn/ticket/721` zu finden.

Der komplette Patch ist auf der CD unter "patches/tap-windows/0001-implement-source-ip-check.patch" zu finden.

4.6 Test des Codes

Zum Testen des Codes wurde er mit der implementierten TAP-Treiber-Unterstützung kompiliert und im Netzwerk getestet, sowie mit TravisCI in einer standardisierten Umgebung gebaut. Dabei werden verschiedene Standardkonfiguriatonen auf Linux und auf Mac OSX kompiliert, sowie von Linux auf Windows crosscompiled und überprüft, ob es Fehler während des Build-Vorgangs auftragen. TravisCI überprüft alle Builds auf Fehler und sendet eine Benachrichtigung aus, wenn der Build-Vorgang fehlgeschlagen ist.

Im Testszenario für den Netzwerktest wurde der Quellcode auf Linux für Windows 64-Bit cross kompiliert und in einer Virtual Machine (VM) mit Windows 8.1 ausgeführt.

Die gebauten Binärdateien wurden über ein geteiltes Verzeichnis der VM zugänglich gemacht. Danach wurden sie in "C:\\Users\\Thermi\\bin kopiert und in 'cmd' mit Administratorrechten ausgeführt. Die nötige Dateistruktur sollte wie in Abbildung 17 auf Seite 43 aussehen.

Im Unterordner "swanctl" befindet sich die Dateien "swanctl.conf", sowie die Ordner "bliss", "ecdsa", "pkcs8", "pkcs12", "pubkey", "rsa", "x509", "x509aa", "x509ac", "x509ca", "x509crl" und"x509ocsp". Im Ordner "rsa" befindet sich der private Schlüssel für den Client. Im Ordner "x509" befindet sich das Zertifikat des Clients und im Ordner "x509ca" alle Certificate Authority (CA)-Zertifikate vom Zertifikat des Clients bis zum selbstsignierten Wurzelzertifikat, sowie das Zertifikat der Server-CA.

Die Datei `this_log` wurde von der Logger-Konfiguration des Dienstes erstellt. Hierbei wurde zuerst `charon-svc.exe` in einem ersten Fenster ausgeführt. In einem zweiten Fenster wurde die Konfiguration geladen und der Tunnelaufbau mittels `swanctl.exe` gestartet. Hierbei wurden `swanctl.exe -load-all` und `swanctl -i -child bar`. Nach dem Testen wurde der Tunnel, sollte `charon-svc.exe` nicht gecrasht sein, mit `swanctl -t -ike foo` beendet, sodass keine virtuellen IP-Adressen oder Routen für den nächsten Test zurückbleiben.

Wenn `charon-svc.exe` getötet wird, während ein Tunnel mit Routen und IP-Adressen aufgebaut ist, so verbleiben IP-Adressen und Routen auf dem Adapter, was verhindern kann dass dieselben Routen und IP-Adressen wieder installiert werden. Das führt dazu, dass der Aufbau der CHILD_SA fehlschlägt und ein neuer Tunnel nicht aufgebaut werden kann. In diesem Fall müssen sie manuell über die Nutzung von `route delete` oder `netsh` gelöscht werden oder Windows

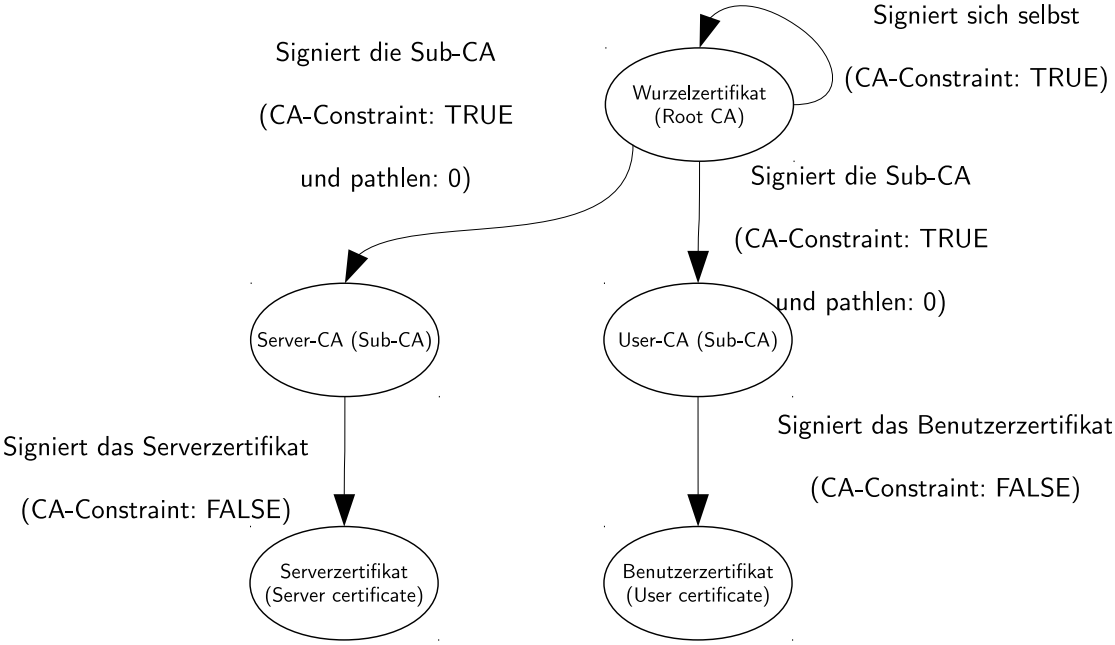


Abbildung 16: Eine exemplarische CA-Struktur

neu gestartet werden.

OpenSSL auf Windows wird für die Kryptographie benötigt und liegt nicht in den Standardsuchpfaden für Bibliotheken und Header. Aus diesem Grund werden die Header und Bibliotheken über LDFLAGs in Include-Pfade auf der Kommandozeile übergeben.

Die Bibliotheken von OpenSSL wurden durch die Installation von OpenVPN beschafft. Bei der Installation von OpenVPN werden direkt die Bibliotheken von OpenSSL mitinstalliert.

Die unter Code 29 und Code 28 bereitgestellten Konfigurationen wurden zur Konfiguration des Dienstes genutzt.

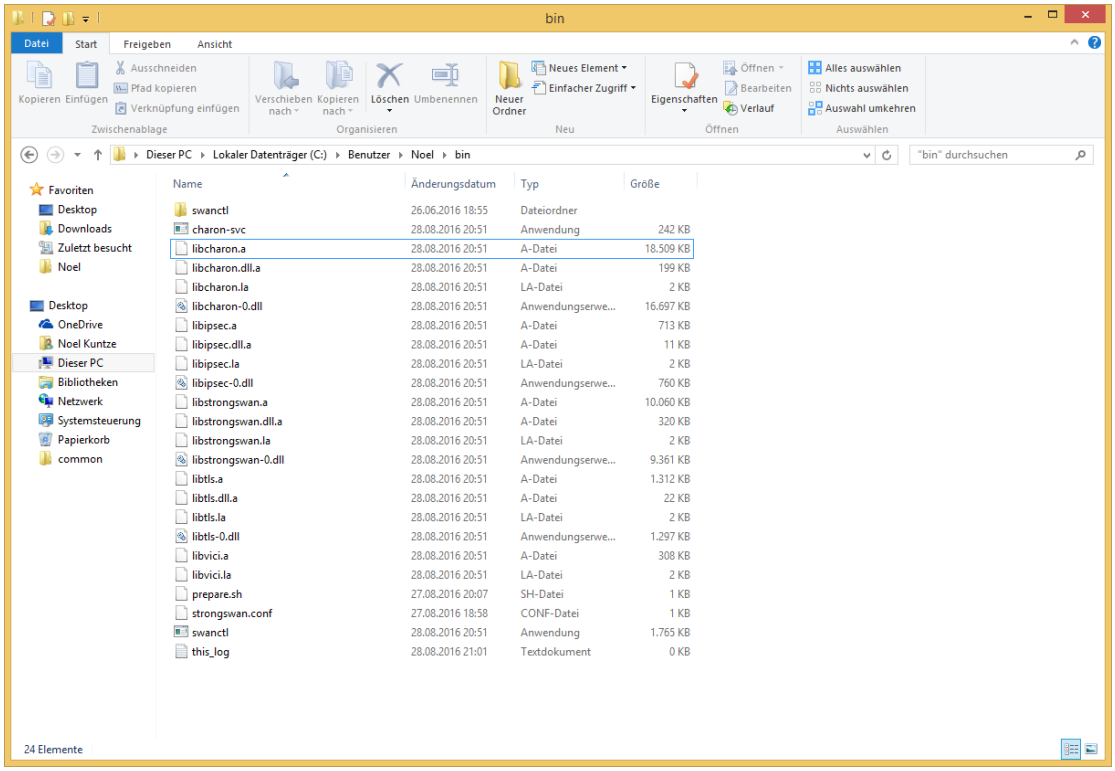


Abbildung 17: Ordnerstruktur nach dem Kopieren der Dateien

Code 18: ./configure und make

```
1 ./configure --host=x86_64-w64-mingw32 --prefix=/ --libdir=/bin --bindir=/bin
      --sbindir=/bin --disable-defaults --enable-monolithic --enable-static
      --enable-svc --enable-ikev2
2 --enable-ikev1 --enable-nonce --enable-pem --enable-pkcs1 --enable-x509
      --enable-openssl --enable-socket-win --enable-kernel-wfp --enable-kernel-iph
      --enable-pubkey --enable-swanctl
3 --with-swanctldir=swanctl --with-strongswan-conf=strongswan.conf
      --enable-libipsec --enable-kernel-libipsec --enable-eap-tls
      --enable-mschapv2 --enable-eap-peap --enable-eap-gtc
4 --enable-eap-dynamic --enable-eap-identity --enable-md4 --enable-ipseckey
      --enable-dnscert --enable-files --enable-sha3 host_alias=x86_64-w64-mingw32
      CC=x86_64-w64-mingw32-gcc
5 CFLAGS=-g -O2 -Wall -Werror -Wno-pointer-sign -Wno-format-security -Wno-format
      -mno-ms-bitfields
      -I/home/thermi/FH-Stuff/UNITS-8/Bachelorarbeit/win32-headers/files/include/
6 LDFLAGS=-L/home/thermi/FH-Stuff/UNITS-8/Bachelorarbeit/win32-headers/files/lib/
      -L/home/thermi/FH-Stuff/UNITS-8/Bachelorarbeit/win32-headers/files/bin/
7 make clean && make
```

Der Befehl konfiguriert den strongSwan Quellcode zum Bauen der benötigten Plugins, sowie einiger Extras für die Authentifizierung und dazu den 64-Bit Compiler des MINGW32-Projekts zu nutzen.

Der Code wurde getestet, indem versucht wurde eine VPN-Verbindung zu einem IKE-Peer unter meiner Kontrolle aufzubauen. Der ausgehandelte Traffic Selector (TS) wurde so gewählt, dass die Beschränkung von Route based VPNs keine Rolle spielen. Die genutzte Konfiguration

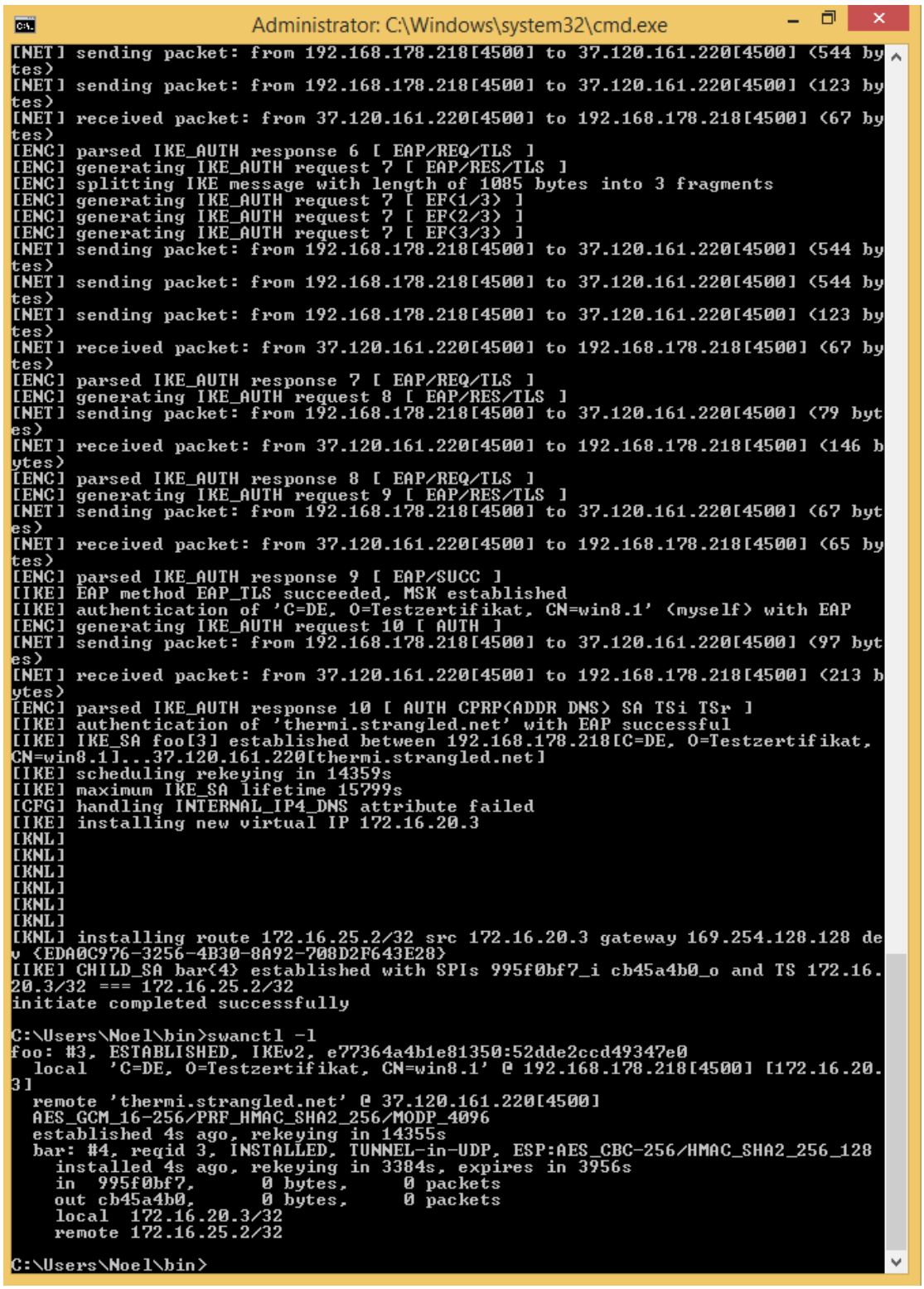


Abbildung 18: Ausgabe des Tunnelaufbaus und swanctl -l

ist im Appendix unter Code 28 und Code 29 zu finden.

4.7 Test der Verbindung

Das Testen der Verbindung wurde durch durch einfaches Pingen über die Verbindung bewerkstelligt. Dabei wird durch Dumpen der Pakete mit tcpdump und Wireshark überprüft, ob die ARP-Requests auf dem TUN-Adapter beantwortet werden und ob das Paket auf dem anderen Endpunkt auftaucht. Des weiteren werden die Traffic-Counter der SAs überprüft, um zu überprüfen, ob die Pakete jemals verarbeitet wurden.

4.8 Leistung

Die Leistung der Implementierung wurde mittels einer Verbindung vom Gast zum Host der Testplattform ermittelt. Als Testumgebung wurde eine Virtualbox-VM genutzt, die auf einem Computer mit einem Intel(R) Core(TM) i7-3820 CPU ausgerüstet ist. Diese Central Processing Unit (CPU) hat vier Kerne. Die verschiedenen Prozesse wurden nicht auf bestimmte Kerne gepinnt. Der CPU-Governor wurde in den Performance-Modus gestellt. Die Messpunkte wurden mittels einer simplen Software in `Python 2.7.12` ermittelt, die im Abstand von einer Sekunde die CPU-Auslastung der Kerne ermittelt. Für die Konfiguration des Initiators und des Responders wurden die Konfigurationen aus Code 19 und Code 20 genutzt. Die Ciphersuite `null-sha1` wurde gewählt, um den Overhead bei der Verarbeitung der Pakete in `charon` zu minimieren.

Code 19: Initiator-Konfiguration für den Geschwindigkeitstest

```
1  connections {
2          speedtest {
3                  version = 2
4                  fragmentation = yes
5                  remote_addrs = 192.168.178.43
6                  mobike = yes
7                  proposals = aes256gcm16-prfsha256-modp4096
8                  vips = 0.0.0.0
9                  local {
10                         auth = psk
11                         id = speedtest-initiator
12                 }
13                 remote {
14                         auth = psk
15                         id = speedtest-responder
16                 }
17                 children {
18                         speedtest-child {
19                                 esp_proposals = null-sha1
20                                 remote_ts = 172.17.0.1
21                         }
22                 }
23
24         }
25 }
26
27 secrets {
28         ike-speedtest {
```

```
29                  secret = speedtest
30                  id = speedtest-responder
31                  id = speedtest-initiator
32          }
33  }
```

Code 20: Responder-Konfiguration für den Geschwindigkeitstest

```
1   connections {
2           speedtest {
3                   version = 2
4                   fragmentation = yes
5                   mobike = yes
6                   proposals = aes256gcm16-prfsha256-modp4096
7                   pools = alpha
8                   local {
9                           auth = psk
10                          id = speedtest-responder
11                  }
12                  remote {
13                          auth = psk
14                          id = speedtest-initiator
15                  }
16                  children {
17                          speedtest-child {
18                                  esp_proposals = null-sha1
19                                  local_ts = 172.17.0.1
20                          }
21                  }
22
23          }
24  }
25
26  pools {
27          alpha {
28                  addrs = 172.16.16.1-172.16.16.10
29          }
30  }
31  secrets {
32          ike-speedtest {
33                  secret = speedtest
34                  id = speedtest-initiator
35                  id = speedtest-responder
36          }
37  }
```

Die CHILD_SA wird vom Initiator mithilfe von `swanctl` aufgebaut. Das Kommando ist in Code 21 sichtbar.

Code 21: Kommando für den Tunnelaufbau des Geschwindigkeitstests

```
1   swanctl -i --child speedtest-child
```

Code 22: Kommando für den Start des iperf-Servers

```
1   iperf3 -f m -s
```

Code 23: Kommando für den Start des iperf-Clients

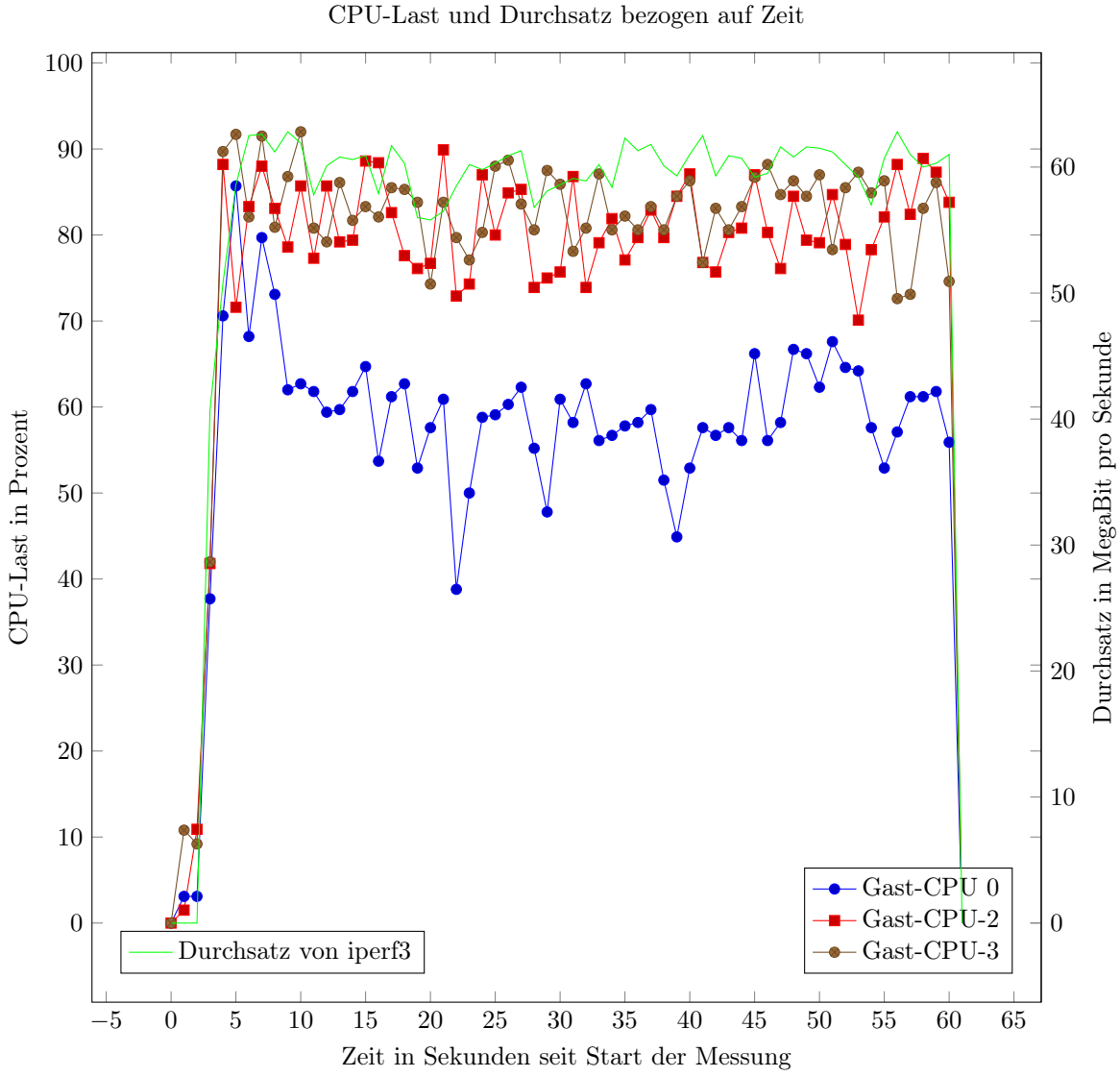
```
1 iperf3 -t 60 -c 172.17.0.1
```

Der `iperf3`-Server wurde mit dem Kommando aus Code 22 gestartet und der Client mit dem Kommando aus Code 23.

Damit das Routing auf dem Initiator korrekt funktioniert, wurde ihm mittels eines pools und `Config Mode` eine ”virtuelle” IP-Adresse zugewiesen. Dies war und ist nötig, damit die SPs auf dem Initiator installiert werden können. Die IP `172.17.0.1` ist eine lokale IP auf dem resolver. Sie wurde als einer der Teilnehmer beim Geschwindigkeitstest genutzt. Als Software wurde `iperf3` in Version `3.1.3` genutzt. Die Binärdatei für den Windows-Gast wurde von `https://iperf.fr/iperf-download.php` bezogen. Auf dem Linux-Host wurde `iperf3` in Version `3.1.3` von einem Archlinux-Mirrorserver mittels `pacman` bezogen.

Der `iperf3`-Server wurde auf dem Host in einer Bash-Shell mittels dem Befehl in Code 22 gestartet.

Abbildung 19: Statistik über iperf3-Leistung durch den Tunnel



In Abbildung 19 wurde der Durchsatz einer `iperf3`-Messung durch den konfigurierten Tunnel dargestellt.

Die Leistung des TAP-Adapters scheint auf etwa 60 MegaBit pro Sekunde beschränkt zu sein. Dies ist hier jedoch nicht einer möglichen schlechte Implementierung der IO-Operationen in charon geschuldet, sondern dem TAP-Windows6-Treiber selbst. Es ist seit langem bekannt, dass die Leistung des Treibers sehr schlecht ist[39]. Es fällt auf, dass die CPU-Last nie auf 100%, sondern nur auf 90% steigt und die erste CPU des Gasts in der Mitte des Tests eine relativ geringe Auslastung von 70% bis 40% erfährt.

4.9 Notwendige Features für Nutzung als RW auf Windows

DNS-Resolver Um DNS-Server installieren zu können, muss die Unterstützung dafür noch implementiert werden. DNS-Einträge werden in IKE mittels `Config Mode` oder CP an den Initiator übertragen und dort konfiguriert.

Windows 10 Windows verwaltet die DNS-Resolvereinträge für jede Schnittstelle einzeln, statt global in einer Datei. Seit Windows 10 wird jeder konfigurierte Resolver nach einem Namen gefragt und die schnellste Antwort wird genutzt. Das ist gefährlich, da in VPNs zu Unternehmensnetzen für gewöhnlich private DNS-Zonen für die Adressierung von Computern im Unternehmensnetzwerken genutzt werden. Ein Angreifer kann den Verkehr zwischen dem Host und dem DNS-Resolver in einem unprivilegierten Netzwerk abhören und so in Erfahrung bringen mit welchen internen Hosts kommuniziert wird. Des weiteren kann er diese Anfragen beantworten und die Race-Condition zwischen den Resolvern versuchen zu gewinnen, um den Verkehr zum Rechner umzuleiten und den Host so anzugreifen. Dies kann unter Windows mittels Firewallregeln verhindert werden, die mittels WFP verwaltet werden.

Authentifizierung für VICI In die Versatile IKE Configuration Interface (VICI)-API von charon ist keine Authentifizierung integriert. Auf Windows wird die VICI-API über einen TCP-Socket genutzt, der auf Port 4502 horcht. charon bindet den Dienst dabei an die IP-Adresse `127.0.0.1`. Damit ist der Dienst für alle lokalen Anwendungen erreichbar und jedes Programm kann charon konfigurieren und kontrollieren. Um dem vorzubeugen müsste eine Authentifizierungsschicht in VICI implementiert werden. Unter Linux/UNIX macht charon die VICI-API nur über einen Unix-Socket zugänglich, welcher mittels den konfigurierten Zugriffsrechten standardmäßig nur für den ”root”-Benutzer zugänglich ist. Die Anwendungen, die als ”root”-Benutzer ausgeführt werden, sind in der Regel ausgewählt und vertrauenswürdig. Wenn eine Anwendung als ”root”-Benutzer ausgeführt wird, hat sie in der Regel, wenn keine weiteren Sicherheitsmaßnahmen getroffen wurden, bereits Zugriff auf alle Systemfunktionen und kann beliebige Aktionen im System ausführen. Der Wiki-Artikel über VICI zeigt die Unsicherheit des VICI-Protokolls auf.

> The VICI protocol runs over a reliable transport protocol. As the protocol itself currently does not provide any security or authentication properties, it is recommended to run it over a UNIX socket with appropriate permissions.

[16j]

[39] 15a.

Dynamische Authentifizierungsaufforderungen Charon beherrscht noch keine Weitergabe von Authentifizierungsaufforderungen, die über XAUTH oder EAP empfangen werden und nach mehreren Strings fragen. Ein solches Szenario ist die Authentifizierung mittels einer PIN und eines Passworts. charon kann im Moment nur einen einzelnen String über XAUTH oder EAP übertragen. Ein weiteres Problem ist, dass eine PIN, die zum Konfigurationszeitpunkt eingegeben wird, nicht zwingend zum Authentifizierungszeitpunkt gültig ist.

4.10 Probleme

Während der BA wurden mehrere Probleme identifiziert, die mit unbekanntem Verhalten von Funktionen und Standards zusammenhängen.

C-Standard Eines dieser Probleme ist, dass der C-Standard die Deklaration einer Variable direkt nach einem ”case” eines switch-cases nur erlaubt, wenn die Deklaration in einem Codeblock stattfindet. Der folgende Code produziert eine Fehlermeldung:

Code 24: Problematik mit C-Standard

```
1  void main() {
2      int foo = 0;
3
4      switch(foo)
5      {
6          case 0:
7              char *bar;
8              break;
9          default:
10             break;
11     }
12
13 }
```

Die Problematik kann umgangen werden, wenn der Code im case mit geschweiften Klammern umfasst wird, wie hier:

Code 25: Abhilfe für Problematik mit C-Standard

```
1  void main() {
2      int foo = 0;
3
4      switch(foo)
5      {
6          case 0:
7          {
8              char *bar;
9              break;
10         }
11         default:
12             break;
13     }
14 }
```

Code 26: Fehlermeldung von gcc

```
test.c: In Funktion »main«:
test.c:7:13: Fehler: eine Marke kann nur Teil einer Anweisung sein, und eine
    Deklaration ist keine Anweisung
             char *bar;
             ^~~~
```

Interessanterweise ist eine Deklaration in einem "case" erlaubt, jedoch nur nicht direkt nach der Deklaration des "case". Daher funktioniert dieser Code auch.

Code 27: Abhilfe mittels NOOP

```
void main() {
    int foo = 0;

    switch(foo)
    {
        case 0:
            ;
            char *bar;
            break;
        default:
            break;
    }
}
```

WaitForSingleObject() Wie erläutert wird WaitForMultipleObjects() genutzt, um die IO-Operationen zu multiplexen. Diese Funktion beendet im Programmcode mit dem Wert "1", was bedeuted dass das zweite Objekt (In C werden Arrays von 0 auf indexiert.) im Array signalisiert wurde. Bei der Überprüfung mit WaitForSingleObject() ist das jedoch nicht der Fall. Wenn das Event signalisiert wäre, dann würde der folgende Aufruf den Wert "WAIT_OBJECT_0" (0x0) zurückgeben:

```
WaitForSingleObject(event_array[offset], 0)
```

] Das ist jedoch nicht der Fall. Der Aufruf beendet mit "WAIT_TIMEOUT" (0x102, 258), was bedeuted, dass das Event nicht signalisiert ist. Dieser Rückgabewert ist jedoch falsch, da das Event tatsächlich signalisiert wird.

Das Problem ist besonders evident bei der Betrachtung des Debug-Logs, welches sich auf Seite 69 finden lässt. Es wird vom Code in auf Seite 64 ausgegeben.

TAP-Treiber schneidet DHCP-Pakete ab Beim Testen des Lesens ist aufgefallen, dass DHCP-Pakete abgeschnitten an die Anwendung weitergegeben werden. Dies hatte mit dem Problem mit `WaitForSingleObject()` die Auswirkung, dass es sehr schwierig war das eigentliche Problem herauszufinden, da der Treiber scheinbar nur ein einziges Paket übergab und dieses beschädigt war.

GetLastError() Des weiteren gibt die Funktion "GetLastError()" den Fehlerwert 0 zurück, wenn sie nicht direkt nach dem Funktionsaufruf, der den Fehlerstatus setzte, aufgerufen wird. Es ist irrelevant, ob zwischen dem Aufruf von "GetLastError()" nur eine einfache Zuweisung oder

eine If-Abfrage steht. Dies ergibt keinen Sinn, da sich der Fehlerstatus scheinbar ändert, ohne dass eine Funktion aufgerufen wird, die das tun könnte.

5 Fazit

In der Arbeit wurden die nötigen Änderungen für strongSwan präsentiert, um libipsec auf Windows lauffähig zu machen. Dies inkludiert die Implementierung der Routinen für das Öffnen und Konfigurieren von TAP-Geräten, die vom TAP-Windows-Treiber bereitgestellt werden. Des weiteren wurde die Installation von IP-Adressen getestet und erweitert, sowie Code zum Lesen und Schreiben von Paketen auf Handles auf Windows.

Im Verlauf der Arbeit wurden mehrere Probleme mit dem C-Standard, sowie mit verschiedenen Windowsfunktionen gefunden. Sie wurden in der Arbeit dokumentiert und um sie herumgearbeitet. Es war etwas überraschend, dass so simple Arbeiten, wie die Implementierung von Multiplexing mittels Events Probleme bereiten.
Es ist ungewiss, ob die nötigen Features für eine Implementierung eines vollständigen Roadwarrior-Clients unter Windows noch entwickelt werden.

Die gesamte Arbeit ist gemeinfrei und unterliegt der GPLv3. Somit kann sie, sowie der Quellcode, frei vertrieben werden, in der Hoffnung dass sie jemandem nutzt. Der Quellcode der Arbeit, sowie komplette Git-Bäume und diffs vom jeweiligen Root-Commit der jeweiligen Softwareprojekte sind auf der CD mitenthalten.

Literatur

[13] *Shrew Soft VPN Client Administrators Guide*. 2013. URL: https://www.shrew.net/static/help-2.1.x/vpnhelp.htm (besucht am 13. 08. 2016).

[14] *Contributions - strongSwan*. 15. Juli 2014. URL: https://wiki.strongswan.org/projects/strongswan/wiki/Contributions (besucht am 14. 08. 2016).

[15a] *OpenVPN / Mailing Lists*. 17. Jan. 2015. URL: https://sourceforge.net/p/openvpn/mailman/message/33244187/ (besucht am 26. 09. 2016).

[15b] *Windows - strongSwan*. 3. Okt. 2015. URL: https://wiki.strongswan.org/projects/strongswan/wiki/Windows (besucht am 13. 08. 2016).

[16a] *bintec Secure IPSec Client*. 14. Apr. 2016. URL: http://pim.bintec-elmeg.com/common/ajax.php?modul_id=101&klasse=produkte_pdf&bereich=admin&com=detail&kanal=xml&system_id=871&sprache=en.

[16b] *CreateIoCompletionPort function (Windows)*. 2016. URL: https://msdn.microsoft.com/en-us/library/windows/desktop/aa363862(v=vs.85).aspx (besucht am 09. 08. 2016).

[16c] *gnu.org*. 8. Juli 2016. URL: https://www.gnu.org/licenses/license-list.en.html#GPLIncompatibleLicenses (besucht am 21. 09. 2016).

[16d] *IKEv1CipherSuites - strongSwan*. 2016. URL: https://wiki.strongswan.org/projects/strongswan/wiki/IKEv1CipherSuites (besucht am 13. 08. 2016).

[16e] *IKEv2CipherSuites - strongSwan*. 2016. URL: https://wiki.strongswan.org/projects/strongswan/wiki/IKEv2CipherSuites (besucht am 13. 08. 2016).

[16f] *IPsec Key Exchange*. 2016. URL: https://technet.microsoft.com/en-us/library/cc731752(v=ws.11).aspx (besucht am 25. 08. 2016).

[16g] *IpsecStandards - strongSwan*. 30. März 2016. URL: https://wiki.strongswan.org/projects/strongswan/wiki/IpsecStandards (besucht am 27. 07. 2016).

[16h] *PluginList - strongSwan*. 2016. URL: https://wiki.strongswan.org/projects/strongswan/wiki/PluginList (besucht am 29. 08. 2016).

[16i] *SetIpInterfaceEntry function (Windows)*. 2016. URL: https://msdn.microsoft.com/en-us/library/windows/desktop/aa814465(v=vs.85).aspx (besucht am 26. 09. 2016).

[16j] *Vici - strongSwan*. 2016. URL: https://wiki.strongswan.org/projects/strongswan/wiki/Vici (besucht am 29. 08. 2016).

[16k] *WaitForMultipleObjects function (Windows)*. 2016. URL: https://msdn.microsoft.com/en-us/library/windows/desktop/ms687025(v=vs.85).aspx (besucht am 10. 08. 2016).

[16l] *Windows7 - strongSwan*. 12. Juli 2016. URL: https://wiki.strongswan.org/projects/strongswan/wiki/Windows7 (besucht am 31. 07. 2016).

[Abr+01] Abraham Shacham u. a. *RFC 3173 - IP Payload Compression Protocol (IPComp)*. Sep. 2001. URL: https://tools.ietf.org/html/rfc3173 (besucht am 01. 08. 2016).

[And11] Andreas Steffen. *The strongSwan IPsec Solution*. 15. Juni 2011. URL: https://www.strongswan.org/tcg/tcg_munich_2011.pdf.

[Cha+14] Charlie Kaufman u. a. *RFC 7296 - Internet Key Exchange Protocol Version 2 (IKEv2)*. Okt. 2014. URL: https://tools.ietf.org/html/rfc7296 (besucht am 27. 07. 2016).

[DB98] Daniel L. McDonald und Bao G. Phan. *RFC 2367 - PF_KEY Key Management API, Version 2*. Juli 1998. URL: https://tools.ietf.org/html/rfc2367 (besucht am 25. 07. 2016).

[Dou+98] Douglas Maughan u. a. *RFC 2408 - Internet Security Association and Key Management Protocol (ISAKMP)*. Nov. 1998. URL: https://tools.ietf.org/html/rfc2408 (besucht am 27. 07. 2016).

[Hil98] Hilarie K. Orman. *RFC 2412 - The OAKLEY Key Determination Protocol*. Nov. 1998. URL: https://tools.ietf.org/html/rfc2412 (besucht am 30. 07. 2016).

[JM05] Jan Hutter und Martin Willi. »strongSwan II Eine IKEv2-Implementierung für Linux«. Diplomarbeit. Rapperswil: Hochschule für Technik Rapperswil, 16. Dez. 2005. 186 S. URL: http://security.hsr.ch/theses/DA_2005_IKEv2.pdf (besucht am 28. 07. 2016).

[Jür16] Jürgen Honig. *Datenblatt NCP Secure Entry Client Windows*. 26. Jan. 2016. URL: https://www.ncp-e.com/fileadmin/pdf/library/datasheets/NCP_DB_Entry_Client_Win32_64.pdf (besucht am 14. 08. 2016).

[Mar+05] Markus Stenberg u. a. *RFC 3948 - UDP Encapsulation of IPsec ESP Packets*. Jan. 2005. URL: https://tools.ietf.org/html/rfc3948 (besucht am 27. 07. 2016).

[Mar14] Martin WIlli. *git.strongswan.org Git - WIP: Windows virtual IP notes win-vip*. 16. Sep. 2014. URL: https://git.strongswan.org/?p=strongswan.git;a=commit;h=d50fd962a9bfea277de9131375e793deb80874a5 (besucht am 25. 08. 2016).

[PRJ15] Petri Jokela, Robert Moskowitz und Jan Melen. *RFC 7402 - Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)*. Apr. 2015. URL: https://tools.ietf.org/html/rfc7402 (besucht am 27. 07. 2016).

[SK05] Stephen Kent und Karen Seo. *RFC 4301 - Security Architecture for the Internet Protocol*. Dez. 2005. URL: https://tools.ietf.org/html/rfc4301 (besucht am 27. 07. 2016).

[Ste05a] Stephen Kent. *RFC 4302 - IP Authentication Header*. Dez. 2005. URL: https://tools.ietf.org/html/rfc4302 (besucht am 27. 07. 2016).

[Ste05b] Stephen Kent. *RFC 4303 - IP Encapsulating Security Payload (ESP)*. Dez. 2005. URL: https://tools.ietf.org/html/rfc4303 (besucht am 27. 07. 2016).

[TM14] Tobias Brunner und Martin Willi. *Kernel-wfp - strongSwan*. 7. Okt. 2014. URL: https://wiki.strongswan.org/projects/strongswan/wiki/Kernel-wfp (besucht am 25. 08. 2016).

6 Appendix

6.1 Featurematrix

Die Daten aus diesen Tabellen wurden unter Nutzung der öffentlich zugänglichen Dokumente auf den entsprechenden Herstellerwebseiten erstellt.

Wenn bei einem symmetrischen Verschlüsselungsalgorithmus kein Modus angegeben war, wurde Cipher Block Chaining (CBC) angenommen.

Wenn bei einem Authentifizierungsmodus nicht alle unterstützen Permutationen angegeben waren, wurden alle standardisierten Permutationen als unterstützt angenommen.

Wenn ein Feature nicht explizit als unterstützt angegeben wurde, so wurde angenommen dass es nicht unterstützt wird.

Offenbar ist der "bintec Secure IPsec Client" nur ein Rebranding des "NCP Secure Entry Client", wenn man vom Graphical User Interface (GUI) ausgehen kann. Ein weiteres Indiz ist, dass im Installer des "bintec Secure IPsec Client" der String "NCP engineering GmbH" auftaucht. Daher wäre es zu verstehen, wenn aus den Dokumentationen der beiden Produkte Featureparität hervorkäme. Dem ist aber nicht so, wie aus den Tabellen hervorgeht.

Diese Tabellen beziehen sich nur auf die Fähigkeiten, die ab Windows 7 unterstützt sind. Sie machen keine Aussage über die Unterstützung der Software auf anderen Platformen und hat keinen Anspruch auf Vollständigkeit.

Symbolik

x Unterstützt

o Nicht unterstützt

? unbekannt

Referenzen zur Bibliographie

- strongSwan[40][41]
- Windows Agile VPN Client[42]
- Shrewsoft VPN Client[43]
- NCP Secure Entry Client[44]
- bintec Secure IPsec Client[45]

[40] 16d.
[41] 16e.
[42] 16l.
[43] 13.
[44] Jür16.
[45] 16a.

Software	IKE-Versionen
strongSwan	IKEv1, IKEv2
Windows Agile VPN Client	IKEv1+L2TP, IKEv2
Shrewsoft VPN Client	IKEv1
NCP Secure Entry Client	IKEv1, IKEv2
bintec Secure IPsec Client	IKEv1, IKEv2

Tabelle 2: Unterstützte IKE-Versionen der IPsec-Implementierungen

Software	Lizenz
strongSwan	MIT/GPLv2
Windows Agile VPN Client	Proprietär
Shrewsoft VPN Client	Shareware
NCP Secure Entry Client	Proprietär
bintec Secure IPsec Client	Proprietär

Tabelle 3: Lizenzen der IPsec-Implementierungen

6.2 Testkonfiguration

Code 28: Testkonfiguration - swanctl.conf

```
connections {
        foo {
            version = 2
            dpd_delay = 10
            dpd_timeout = 60
            fragmentation = yes
            send_cert = always
            remote_addrs = 37.120.161.220
            proposals = aes256gcm16-prfsha256-modp4096
            vips = 0.0.0.0
                        mobike = no
                        encap = yes
            local {
                auth = eap-tls
                certs = certificate.pem
            }
            remote {
                auth = pubkey
                id = thermi.strangled.net
                cacerts = serverca.pem
            }
            children {
                    bar {
                        dpd_action = restart
                        start_action = none
                        esp_proposals = aes256-sha256-ecp521
                        remote_ts = 172.16.25.2/32
                    }
            }
        }
}
```

Software / Modus	strongSwan	Windows	Shrewsoft	NCP	bintec
AES-128-CBC	x	x	x	x	x
AES-192-CBC	x	x	x	x	x
AES-256-CBC	x	x	x	x	x
AES-128-GCM-8	x	o	o	o	o
AES-128-GCM-12	x	o	o	o	o
AES-128-GCM-16	x	o	o	o	o
AES-192-GCM-8	x	o	o	o	o
AES-192-GCM-12	x	o	o	o	o
AES-192-GCM-16	x	o	o	o	o
AES-256-GCM-8	x	o	o	o	o
AES-256-GCM-12	x	o	o	o	o
AES-256-GCM-16	x	o	o	o	o
AES-128-CTR	x	o	o	o	o
AES-192-CTR	x	o	o	o	o
AES-256-CTR	x	o	o	o	o
AES-128-CCM-8	x	o	o	o	o
AES-128-CCM-12	x	o	o	o	o
AES-128-CCM-16	x	o	o	o	o
AES-192-CCM-8	x	o	o	o	o
AES-192-CCM-12	x	o	o	o	o
AES-192-CCM-16	x	o	o	o	o
AES-256-CCM-8	x	o	o	o	o
AES-256-CCM-12	x	o	o	o	o
AES-256-CCM-16	x	o	o	o	o
DES-CBC	o	o	x	o	o
3DES-CBC	x	x	x	x	x
BLOWFISH-128-CBC	x	o	x	x	x
BLOWFISH-192-CBC	x	o	x	x	x
BLOWFISH-256-CBC	x	o	x	x	x
CAMELLIA-128-CBC	x	o	o	o	o
CAMELLIA-192-CBC	x	o	o	o	o
CAMELLIA-256-CBC	x	o	o	o	o
CAMELLIA-128-CCM-8	x	o	o	o	o
CAMELLIA-128-CCM-12	x	o	o	o	o
CAMELLIA-128-CCM-16	x	o	o	o	o
CAMELLIA-192-CCM-8	x	o	o	o	o
CAMELLIA-192-CCM-12	x	o	o	o	o
CAMELLIA-192-CCM-16	x	o	o	o	o
CAMELLIA-256-CCM-8	x	o	o	o	o
CAMELLIA-256-CCM-12	x	o	o	o	o
CAMELLIA-256-CCM-16	x	o	o	o	o
SERPENT-128-CBC	x	o	o	o	o
SERPENT-192-CBC	x	o	o	o	o
SERPENT-256-CBC	x	o	o	o	o
TWOFISH-128-CBC	x	o	o	o	o
TWOFISH-192-CBC	x	o	o	o	o
TWOFISH-256-CBC	x	o	o	o	o
CAST-128-CBC	x	o	x	o	o
chacha20poly1305	x	o	o	o	o

Tabelle 4: Unterstützte Algorithmen für Vertraulichkeit der IPsec-Implementierungen

Modus \ Software	strongSwan	Windows	Shrewsoft	NCP	bintec
MD5	x	o	x	x	x
SHA-1	x	x	x	o	x
SHA-256	x	x	o	x	x
SHA-384	x	x	o	x	x
SHA-512	x	o	o	x	x
SHA-256-96	x	x	o	o	o
AES-XCBC	x	o	o	o	o
AES-128-GMAC	x	o	o	o	o
AES-192-GMAC	x	o	o	o	o
AES-256-GMAC	x	o	o	o	o

Tabelle 5: Unterstützte Algorithmen für Authentizität der IPsec-Implementierungen

Modus \ Software	strongSwan	Windows	Shrewsoft	NCP	bintec
MODP-768	x	o	x	x	x
MODP-1024	x	x	x	x	x
MODP-1536	x	o	x	x	x
MODP-2048	x	x	x	x	x
MODP-3072	x	o	x	x	x
MODP-4096	x	o	x	x	x
MODP-6144	x	o	x	x	x
MODP-8192	x	o	x	x	x
MODP-1024s160	x	o	o	o	o
MODP-2048s224	x	o	o	o	o
MODP-2048s256	x	o	o	o	o
ECP-192	x	o	o	x	o
ECP-224	x	o	o	x	o
ECP-256	x	o	o	x	o
ECP-384	x	o	o	x	o
ECP-521	x	o	o	x	o
ECP-224BP	x	o	o	o	o
ECP-256BP	x	o	o	o	o
ECP-384BP	x	o	o	o	o
ECP-512BP	x	o	o	o	o
NTRU-112	x	o	o	o	o
NTRU-128	x	o	o	o	o
NTRU-192	x	o	o	o	o
NTRU-256	x	o	o	o	o
NEWHOPE-128	x	o	o	o	o

Tabelle 6: Unterstützte Schlüsselaustauschprotokolle der IPsec-Implementierungen

Code 29: Testkonfiguration - strongswan.conf

```
1 charon-svc {
2         filelog {
3                 this_log.txt{
4                         default=2
5                         job = 1
6                         mgr = 0
7                         enc = 0
8                         asn = 0
```

Software / Modus	strongSwan	Windows	Shrewsoft	NCP	bintec
Hybrid	x	x	x	x	x
PSK	x	o	x	x	o
PSK + XAUTH	x	o	x	x	o
X.509	x	x	x	x	x
EAP-MD5	x	o	o	x	o
EAP-PAP	x	o	o	x	o
EAP-CHAP	x	o	o	x	o
EAP-MSCHAPv2	x	x	o	x	o
EAP-GTC	x	o	o	o	o
EAP-TLS	x	x	o	x	o
EAP-TTLS	x	o	o	o	o
EAP-AKA	x	o	o	o	o
EAP-TNC	x	o	o	o	o
TNC-IMC	x	o	o	o	o
TNC-IMV	x	o	o	o	o

Tabelle 7: Unterstützte Authentifizierungsmethoden der IPsec-Implementierungen

Software / Modus	strongSwan	Windows	Shrewsoft	NCP	bintec
CRL	x	x	o	x	x
OCSP	x	?	o	x	x

Tabelle 8: Unterstützte Mechanismen zum Zurückziehen von Zertifikaten der IPsec-Implementierungen

Software / Modus	strongSwan	Windows	Shrewsoft	NCP	bintec
Tunnel-Modus	x	x	x	x	x
Transport-Modus	x	x	o	o	o
BEET-Modus	x	o	o	o	o

Tabelle 9: Unterstützte Tunnel-Modi der IPsec-Implementierungen

Software / Modus	strongSwan	Windows	Shrewsoft	NCP	bintec
Main Mode	x	x	x	x	?
Aggressive Mode	x	x	x	x	?
Quick Mode	x	o	x	x	?
Config Mode	x	x	x	x	x

Tabelle 10: Unterstützte IKE-Modi der IPsec-Implementierungen

```
9                       flush_line = yes
10                      ike_name = yes
11                      append = no
12                  }
13          }
14  }
```

Code 30: Code von win32.h

```
1  /*
```

Feature \ Software	strongSwan	Windows	Shrewsoft	NCP	bintec
NAT-T	x	x	x	x	x
DPD	x	x	x	x	x
MOBIKE	x	x	o	o	o
GUI	o	x	x	x	x
IPsec über TCP	o	o	o	x	x
IPv6	x	x	x	x	x
Attribut-Zertifikate	x	?	o	o	?
PFS	x	o	x	x	x
IKE-Fragmentierung	x	Nur IKEv1	x	o	o
Komprimierung	x	o	o	x	o

Tabelle 11: Unterstützte Features der IPsec-Implementierungen

```
2  * Copyright (C) 2016 Noel Kuntze
3  *
4  * Permission is hereby granted, free of charge, to any person obtaining a copy
5  * of this software and associated documentation files (the "Software"), to deal
6  * in the Software without restriction, including without limitation the rights
7  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
8  * copies of the Software, and to permit persons to whom the Software is
9  * furnished to do so, subject to the following conditions:
10 *
11 * The above copyright notice and this permission notice shall be included in
12 * all copies or substantial portions of the Software.
13 *
14 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
15 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
17 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
18 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
19 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
20 * THE SOFTWARE.
21 */
22
23 #ifndef WIN32_H
24 #define WIN32_H
25
26 #define WIN32_TUN_READ_EVENT_TEMPLATE "WIN32-libipsec-read-device-%d"
27 #define WIN32_TUN_WRITE_EVENT_TEMPLATE "WIN32-libipsec-write-device-%d"
28 #define WIN32_TUN_EVENT_LENGTH 80
29 #define TAP_WIN_COMPONENT_ID "tap0901"
30
31 #define ADAPTER_KEY
       "SYSTEM\\CurrentControlSet\\Control\\Class\\{4D36E972-E325-11CE-BFC1-08002BE10318}"
32 #define NETWORK_CONNECTIONS_KEY
       "SYSTEM\\CurrentControlSet\\Control\\Network\\{4D36E972-E325-11CE-BFC1-08002BE10318}"
33
34 /*
35  * ==========================
36  * Filesystem prefixes
37  * ==========================
38  */
```

```
39
40 #define USERMODEDEVICEDIR "\\\\.\\Global\\"
41 #define SYSDEVICEDIR      "\\Device\\"
42 #define USERDEVICEDIR     "\\DosDevices\\Global\\"
43 #define TAP_WIN_SUFFIX    ".tap"
44
45 /*
46  * TAP IOCTL constants and macros.
47  *
48  */
49 #define TAP_WIN_CONTROL_CODE(request,method) \
50   CTL_CODE (FILE_DEVICE_UNKNOWN, request, method, FILE_ANY_ACCESS)
51
52 /* Present in 8.1 */
53
54 #define TAP_WIN_IOCTL_GET_MAC               TAP_WIN_CONTROL_CODE (1,
      METHOD_BUFFERED)
55 #define TAP_WIN_IOCTL_GET_VERSION           TAP_WIN_CONTROL_CODE (2,
      METHOD_BUFFERED)
56 #define TAP_WIN_IOCTL_GET_MTU               TAP_WIN_CONTROL_CODE (3,
      METHOD_BUFFERED)
57 #define TAP_WIN_IOCTL_GET_INFO              TAP_WIN_CONTROL_CODE (4,
      METHOD_BUFFERED)
58 #define TAP_WIN_IOCTL_CONFIG_POINT_TO_POINT TAP_WIN_CONTROL_CODE (5,
      METHOD_BUFFERED)
59 #define TAP_WIN_IOCTL_SET_MEDIA_STATUS      TAP_WIN_CONTROL_CODE (6,
      METHOD_BUFFERED)
60 #define TAP_WIN_IOCTL_CONFIG_DHCP_MASQ      TAP_WIN_CONTROL_CODE (7,
      METHOD_BUFFERED)
61 #define TAP_WIN_IOCTL_GET_LOG_LINE          TAP_WIN_CONTROL_CODE (8,
      METHOD_BUFFERED)
62 #define TAP_WIN_IOCTL_CONFIG_DHCP_SET_OPT   TAP_WIN_CONTROL_CODE (9,
      METHOD_BUFFERED)
63
64 /* Added in 8.2 */
65
66 /* obsoletes TAP_WIN_IOCTL_CONFIG_POINT_TO_POINT */
67 #define TAP_WIN_IOCTL_CONFIG_TUN            TAP_WIN_CONTROL_CODE (10,
      METHOD_BUFFERED)
68 #define TAP_WIN_IOCTL_CONFIG_SET_SRC_CHECK  TAP_WIN_CONTROL_CODE (11,
      METHOD_BUFFERED)
69
70
71 #endif /* WIN32_H */
```

Code 31: Code für das Suchen eines TAP-Geräts

```
1 /*
2  * Searches through the registry for suitable TAP driver interfaces
3  * On Windows, the TAP interface metadata is stored and described in the
       registry.
4  * It returns a linked list that contains all found guids. The guids describe
       the interfaces.
5  */
6
```

```
7  linked_list_t *find_tap_devices()
8  {
9      char enum_name[256], unit_string[256],
10     instance_id[256], component_id[256],
11     component_id_string[] = "ComponentId",
12     instance_id_string[] = "NetCfgInstanceId";
13     LONG status;
14     uint32_t i = 0;
15     DWORD len, type;
16     HKEY adapter_key, unit_key;
17     linked_list_t *list = linked_list_create();
18
19     /*
20      * Open parent key. It contains all other keys that
21      * describe any possible interfaces.
22      */
23     status = RegOpenKeyEx(
24             HKEY_LOCAL_MACHINE,
25             ADAPTER_KEY,
26             0,
27             KEY_READ,
28             &adapter_key);
29
30     if (status == ERROR_SUCCESS)
31     {
32         while (TRUE)
33         {
34             len = sizeof (enum_name);
35             status = RegEnumKeyEx(
36                     adapter_key,
37                     i,
38                     enum_name,
39                     &len,
40                     NULL,
41                     NULL,
42                     NULL,
43                     NULL);
44             if (status == ERROR_SUCCESS)
45             {
46                 snprintf(unit_string, sizeof (unit_string), "%s\\%s",
47                         ADAPTER_KEY, enum_name);
48
49                 status = RegOpenKeyEx(
50                         HKEY_LOCAL_MACHINE,
51                         unit_string,
52                         0,
53                         KEY_READ,
54                         &unit_key);
55
56                 if (status == ERROR_SUCCESS)
57                 {
58                     len = sizeof (component_id);
59                     status = RegQueryValueEx(
60                             unit_key,
61                             component_id_string,
```

```
62                              NULL,
63                              &type,
64                              component_id,
65                              &len);
66
67                      if (status == ERROR_SUCCESS && type == REG_SZ)
68                      {
69                          len = sizeof (instance_id);
70                          status = RegQueryValueEx(
71                                  unit_key,
72                                  instance_id_string,
73                                  NULL,
74                                  &type,
75                                  instance_id,
76                                  &len);
77
78                          if (status == ERROR_SUCCESS && type == REG_SZ)
79                          {
80                              if (!strcmp(component_id, TAP_WIN_COMPONENT_ID))
81                              {
82                                  /* That thing is a valid interface key */
83                                  /* link into return list */
84                                  char *guid = malloc(sizeof(instance_id));
85                                  memcpy(guid, instance_id, sizeof(instance_id));
86                                  list->insert_last(list, guid);
87                              }
88                          }
89                      }
90                      else
91                      {
92                          DBG2(DBG_LIB, "Error opening registry key: %s\\%s",
93                                  unit_string, component_id_string);
94                      }
95                      RegCloseKey(unit_key);
96                  }
97                  else if (status != ERROR_SUCCESS)
98                  {
99                      DBG2(DBG_LIB, "Error opening registry key: %s", unit_string);
100                 }
101                 i++;
102             }
103             else if (status == ERROR_NO_MORE_ITEMS)
104             {
105                 break;
106             }
107             else
108             {
109                 DBG2(DBG_LIB, "Error enumerating registry subkeys of key: %s",
110                         ADAPTER_KEY);
111             }
112         }
113     }
114     else
115     {
116         DBG2(DBG_LIB, "Error opening registry key: %s", ADAPTER_KEY);
```

```
117     }
118
119     RegCloseKey(adapter_key);
120     return list;
121 }
```

Code 32: Code für handle_plain auf Windows

```
1  /**
2   * Job handling outbound plaintext packets
3   */
4  static job_requeue_t handle_plain(private_kernel_libipsec_router_t *this)
5  {
6  #ifdef WIN32
7          void **key = NULL;
8          bool oldstate;
9          uint32_t length, event_status = 0, i = 0, j = 0, offset;
10         handle_overlapped_buffer_t *bundle_array = NULL, dummy,
               tun_device_handle_overlapped_buffer;
11         OVERLAPPED *overlapped = NULL;
12         HANDLE *event_array = NULL, tun_device_event;
13         tun_device_t *tun_device = this->tun.tun;
14         enumerator_t *tuns_enumerator;
15
16         memset(&tun_device_handle_overlapped_buffer, 0,
               sizeof(handle_overlapped_buffer_t));
17         /* Reset synchronisation event */
18         ResetEvent(this->event);
19
20         length = this->tuns->get_count(this->tuns);
21
22         this->lock->read_lock(this->lock);
23         /* Read event for this->tun */
24
25         /* allocate arrays for all the structs we need */
26         /* events, overlapped structures and bundles. */
27         /* event_array holds all the HANDLE structures for the events that are
28          * used for notifying the thread of finished reads and writes.
29          */
30
31         overlapped = alloca((length+2)*sizeof(OVERLAPPED));
32         event_array = alloca((length+2)*sizeof(HANDLE));
33         bundle_array = alloca((length+2)*sizeof(handle_overlapped_buffer_t));
34
35         memset(overlapped, 0, (length+2)*sizeof(OVERLAPPED));
36         memset(bundle_array, 0, (length+2)*sizeof(handle_overlapped_buffer_t));
37
38         /* These are the arrays we're going to work with */
39
40         /* first position is the event we use for synchronisation  */
41         /* Insert notification event */
42         event_array[i] = this->event;
43         /* Insert dummy structure */
44         bundle_array[i] = dummy;
45         i++;
```

```
46
47          /* second position is this->tun */
48          /* insert event object for this->tun device */
49          tun_device_event = CreateEvent(NULL, FALSE, FALSE, FALSE);
50          if (!tun_device_event)
51          {
52              char *error_message = format_error(GetLastError());
53              free(error_message);
54              return JOB_REQUEUE_FAIR;
55          }
56          event_array[i] = tun_device_event;
57          ResetEvent(event_array[i]);
58          /* bundle for the read on this->tun */
59          /* Reserve memory for the buffer*/
60          tun_device_handle_overlapped_buffer.buffer =
                chunk_alloca(tun_device->get_mtu(tun_device));
61          /* Initialise the buffer */
62          memset(tun_device_handle_overlapped_buffer.buffer.ptr, 0,
                tun_device_handle_overlapped_buffer.buffer.len);
63
64          tun_device_handle_overlapped_buffer.fileHandle =
                tun_device->get_handle(tun_device);
65          tun_device_handle_overlapped_buffer.overlapped = overlapped;
66
67          tun_device_handle_overlapped_buffer.overlapped->hEvent= tun_device_event;
68
69          bundle_array[i] = tun_device_handle_overlapped_buffer;
70
71          i++;
72
73          /* Start ReadFile for this->tun.handle */
74          if (!start_read(&tun_device_handle_overlapped_buffer,
                tun_device_handle_overlapped_buffer.overlapped->hEvent))
75          {
76                  // TODO: Cleanup heap
77                  this->lock->unlock(this->lock);
78                  return JOB_REQUEUE_FAIR;
79          }
80          /* pad bundle_array with two empty structures */
81          /* iterate over all our tun devices, create event handles, reset them,
                queue read operations on all handles */
82
83
84          tuns_enumerator = this->tuns->create_enumerator(this->tuns);
85          while(tuns_enumerator->enumerate(tuns_enumerator, key, &tun_device))
86          {
87              /* Allocate structure and buffer */
88
89              bundle_array[i].buffer =
                    chunk_alloca(tun_device->get_mtu(tun_device));
90              memset(bundle_array[i].buffer.ptr, 0, bundle_array[i].buffer.len);
91              bundle_array[i].fileHandle = tun_device->get_handle(tun_device);
92              /* Allocate and initialise OVERLAPPED structure */
93              bundle_array[i].overlapped = alloca(sizeof(OVERLAPPED));
94              (*bundle_array[i].overlapped) = overlapped[i];
```

```
95              memset(&bundle_array[i].overlapped, 0, sizeof(OVERLAPPED));
96              /* Create unique name for that event. */
97              /* Create unique event for read accesses on that device
98               * No security attributes, no manual reset, initial state is
                     unsignaled,
99               * name is the special name we created
100              */
101             bundle_array[i].overlapped->hEvent = CreateEvent(NULL, FALSE, FALSE,
                    FALSE);
102             // event_array[i] = OpenEvent(EVENT_ALL_ACCESS, FALSE,
                    tun_device->get_read_event_name(tun_device));
103             event_array[i] = bundle_array[i].overlapped->hEvent;
104
105             if (event_array[i] == NULL)
106             {
107                 char *error_message = format_error(GetLastError());
108                 free(error_message);
109                 return JOB_REQUEUE_FAIR;
110             }
111             i++;
112
113             /* Initialise read with the allocate overwrite structure */
114             DBG2(DBG_ESP, "Reading on %s", tun_device->get_name(tun_device));
115             if (!start_read(&bundle_array[i],
                    bundle_array[i].overlapped->hEvent))
116             {
117                     // TODO: Cleanup heap
118                     this->lock->unlock(this->lock);
119                     return JOB_REQUEUE_FAIR;
120             }
121             i++;
122         }
123         tuns_enumerator->destroy(tuns_enumerator);
124
125         while (TRUE)
126         {
127             /* Wait for a handle to be signaled */
128             /* In the mingw64 sources, MAXIMUM_WAIT_OBJECTS is defined as 64.
                    That means we can wait for a maximum of 64 event handles.
129              * This translates to 63 tun devices. I think this is sufficiently
                    high to not have to implement a mechanism for waiting for more
130              * events /support more TUN devices */
131             oldstate = thread_cancelability(FALSE);
132             event_status = WaitForMultipleObjects(i, event_array, FALSE,
                    INFINITE);
133             thread_cancelability(oldstate);
134             offset = event_status - WAIT_OBJECT_0;
135
136             /* A handle was signaled. Find the tun handle whose read was
                    successful */
137
138             /* We can only use the event_status of indication for the first
                    completed IO operation.
139              * After the event was signaled, we need to test the OVERLAPPED
                    structure in the other array
```

```
140                 * to find out what event was signaled.
141                 */
142                /*
143                 * Probably broken?
144                 */
145                /* Check if an event in the array was signaled. (That is the case if
146                 * the event_status is between WAIT_OBJECT_0 and WAIT_OBJECT_0 +
                        nCount -1)
147                 */
148                if ((WAIT_OBJECT_0 < event_status) && event_status < ((WAIT_OBJECT_0
                       + length - 1)))
149                {
150                    /* the event at event_array[event_status - WAIT_OBJECT_0] has
                           been signaled */
151                    /* It is possible that more than one event was signalled. In
                           that case, (event_status - WAIT_OBJECT_0)
152                     * is the index with the lowest event that was signalled. More
                           signalled events can be found higher
153                     *
154                     * According to the documentation, WAIT_OBJECT_0 is defined as 0
155                     */
156                    if (offset == 0)
157                    {
158                        /* Notification about changes regarding the tun devices.
159                         * Or the object is destroyed.
160                         * We need to rebuild the array. So exit and rebuild. */
161                        /* Cleanup
162                         *  Starts with 1 to skip over the dummy
163                         */
164                        for(j=1;j<i;j++)
165                        {
166                            /* stop all asynchronous IO */
167                            CancelIo(bundle_array[j].fileHandle);
168                            CloseHandle(bundle_array[j].overlapped->hEvent);
169                            memset(bundle_array[j].buffer.ptr, 0,
                                   bundle_array[j].buffer.len);
170                            free(bundle_array[j].buffer.ptr);
171                            ResetEvent(event_array[j]);
172                            CloseHandle(event_array[j]);
173                        }
174                        /* exit */
175                        return JOB_REQUEUE_DIRECT;
176                    }
177                    /* The arrays have the same length and the same positioning of
                           the elements.
178                     * Therefore, if event_array[j] is signaled, the read on
                           bundle_array[i].fileHandle has succeeded
179                     * and bundle_array[j].buffer has our data now.
180                     */
181
182                    char foo[(bundle_array[offset].buffer.len *4)/3 + 1];
183                    memset(foo, 0, (bundle_array[offset].buffer.len *4)/3 + 1);
184                    chunk_to_base64(bundle_array[offset].buffer, foo);
185
186                    ip_packet_t *packet;
```

```
187                 /* clone the buffer */
188                 chunk_t buffer_clone = chunk_clone (bundle_array[offset].buffer);
189                 packet = ip_packet_create(buffer_clone);
190                 if (packet)
191                 {
192                         ipsec->processor->queue_outbound(ipsec->processor,
                            packet);
193                 }
194                 else
195                 {
196                         DBG2(DBG_ESP, "invalid IP packet read from TUN device");
197                 }
198                 /* Reset the overlapped structure, event and buffer */
199                 /* Print out the package for debugging */
200                 /* Don't leak packets */
201                 memset(bundle_array[offset].buffer.ptr, 0,
                        bundle_array[offset].buffer.len);
202                 memset(bundle_array[offset].overlapped, 0, sizeof(OVERLAPPED));
203
204                 if (!start_read(&bundle_array[offset],
                        bundle_array[offset].overlapped->hEvent))
205                 {
206                    /* Cleanup
207                     *  Starts with 1 to skip over the dummy
208                     */
209                     for(j=1;j<i;j++)
210                     {
211                         /* stop all asynchronous IO */
212                         CancelIo(bundle_array[j].fileHandle);
213                         CloseHandle(bundle_array[j].overlapped->hEvent);
214                         memset(bundle_array[j].buffer.ptr, 0,
                                bundle_array[j].buffer.len);
215                         free(bundle_array[j].buffer.ptr);
216                     }
217                     this->lock->unlock(this->lock);
218                     return JOB_REQUEUE_FAIR;
219                 }
220             }
221             /* Function failed */
222             else
223             {
224                 DBG2(DBG_ESP, "waiting for events on the tun device reads
                        failed.");
225
226                 /* Cleanup
227                  *  Starts with 1 to skip over the dummy
228                  */
229                 for(j=1;j<i;j++)
230                 {
231                     /* stop all asynchronous IO */
232                     CancelIo(bundle_array[j].fileHandle);
233                     CloseHandle(bundle_array[j].overlapped->hEvent);
234                     memset(bundle_array[j].buffer.ptr, 0,
                            bundle_array[j].buffer.len);
235                     free(bundle_array[j].buffer.ptr);
```

```
236                         ResetEvent(event_array[j]);
237                         CloseHandle(event_array[j]);
238                     }
239                     this->lock->unlock(this->lock);
240                     return JOB_REQUEUE_FAIR;
241
242                 }
243             }
244             this->lock->unlock(this->lock);
245             return JOB_REQUEUE_DIRECT;
246 #else
247             [...]
248 #endif /* WIN32 */
249 }
```

Code 33: Debug-Log; Zeigt Problematik mit WaitForSingleObject()

```
1  00[DMN] Starting IKE service charon-svc (strongSwan 5.4.1dr1, Windows Client
       6.2.9200 (SP 0.0)
2  00[LIB] plugin 'sha3': loaded successfully
3  00[LIB] plugin 'md4': loaded successfully
4  00[LIB] plugin 'nonce': loaded successfully
5  00[LIB] plugin 'x509': loaded successfully
6  00[LIB] plugin 'pubkey': loaded successfully
7  00[LIB] plugin 'pkcs1': loaded successfully
8  00[LIB] plugin 'dnscert': loaded successfully
9  00[LIB] plugin 'ipseckey': loaded successfully
10 00[LIB] plugin 'pem': loaded successfully
11 00[LIB] plugin 'openssl': loaded successfully
12 00[LIB] plugin 'files': loaded successfully
13 00[LIB] Error opening registry key:
       SYSTEM\CurrentControlSet\Control\Class\{4D36E972-E325-11CE-BFC1-08002BE10318}\Properties
14 00[LIB] TAP-Windows driver version 9.22 available.
15 00[LIB] created TUN device: {EDA0C976-3256-4B30-8A92-708D2F643E28}
16 00[LIB] plugin 'kernel-libipsec': loaded successfully
17 00[LIB] plugin 'kernel-wfp': loaded successfully
18 00[LIB] plugin 'kernel-iph': loaded successfully
19 00[LIB] plugin 'socket-win': loaded successfully
20 00[LIB] plugin 'vici': loaded successfully
21 00[LIB] plugin 'eap-identity': loaded successfully
22 00[LIB] plugin 'eap-gtc': loaded successfully
23 00[LIB] plugin 'eap-dynamic': loaded successfully
24 00[LIB] plugin 'eap-tls': loaded successfully
25 00[LIB] plugin 'eap-peap': loaded successfully
26 00[LIB] feature CUSTOM:kernel-ipsec in plugin 'kernel-wfp' failed to load
27 00[LIB] feature PUBKEY:DSA in plugin 'pem' has unmet dependency: PUBKEY:DSA
28 00[LIB] feature CUSTOM:dnscert in plugin 'dnscert' has unmet dependency: RESOLVER
29 00[LIB] feature CUSTOM:ipseckey in plugin 'ipseckey' has unmet dependency:
       RESOLVER
30 00[LIB] feature PRIVKEY:DSA in plugin 'pem' has unmet dependency: PRIVKEY:DSA
31 00[LIB] feature PRIVKEY:BLISS in plugin 'pem' has unmet dependency: PRIVKEY:BLISS
32 00[LIB] feature CERT_DECODE:PGP in plugin 'pem' has unmet dependency:
       CERT_DECODE:PGP
33 00[LIB] feature CERT_DECODE:OCSP_REQUEST in plugin 'pem' has unmet dependency:
       CERT_DECODE:OCSP_REQUEST
```

```
00[LIB] unloading plugin 'dnscert' without loaded features
00[LIB] unloading plugin 'ipseckey' without loaded features
00[LIB] unloading plugin 'kernel-wfp' without loaded features
00[LIB] loaded plugins: charon-svc sha3 md4 nonce x509 pubkey pkcs1 pem openssl
    files kernel-libipsec kernel-iph socket-win vici eap-identity eap-gtc
    eap-dynamic eap-tls eap-peap
00[LIB] unable to load 8 plugin features (7 due to unmet dependencies)
00[JOB] spawning 16 worker threads
00[LIB] created thread 4016
00[LIB] created thread 3032
00[LIB] created thread 3216
00[LIB] created thread 3876
00[LIB] created thread 3132
00[LIB] created thread 1580
00[LIB] created thread 1244
00[LIB] created thread 4212
00[LIB] created thread 3152
00[LIB] created thread 4208
00[LIB] created thread 2560
00[LIB] created thread 2764
00[LIB] created thread 1604
00[LIB] created thread 4220
00[LIB] created thread 1304
00[LIB] created thread 972
08[ESP] entered handle_plain.
08[ESP] Allocated arrays, opened events
08[ESP] put notification event into index 0
08[ESP] Put TUN {EDA0C976-3256-4B30-8A92-708D2F643E28} event in index 1
08[ESP] Allocated buffer.
08[ESP] Allocated file handle.
08[ESP] Allocated overlapped..
08[ESP] Created event
08[ESP] ReadFile() returned 0
08[ESP] Error 997
08[ESP] Read on tun device is pending.
08[ESP] Enumerating tun devices ...
08[ESP] Waiting for events...
08[ESP] Event triggered with event_status 1
08[ESP] offset == 1
08[ESP] position 1 in array
08[ESP] checking if event is signaled.
08[ESP] WaitForSingleObject returned 258
08[ESP] Event is not signaled.
08[ESP] Waiting for events...
[...]
```

Code 34: Ausgabe von ipconfig und route -4 print

```
C:\Users\Noel\bin>ipconfig

Windows-IP-Konfiguration


Ethernet-Adapter THIS IS A TAP DEVICE:

```

```
   Verbindungsspezifisches DNS-Suffix:
   Verbindungslokale IPv6-Adresse  . : fe80::596b:bf92:963f:63a3%8
   IPv4-Adresse  . . . . . . . . . . : 172.16.20.2
   Subnetzmaske  . . . . . . . . . . : 255.255.255.255
   Standardgateway . . . . . . . . . :

Ethernet-Adapter Ethernet:

   Verbindungsspezifisches DNS-Suffix: thermicorp.lan
   IPv6-Adresse. . . . . . . . . . . : 2a02:8071:9282:e600:3031:9c6b:6485:3cc9
   Temporäre IPv6-Adresse. . . . . . : 2a02:8071:9282:e600:5181:db51:f4d7:2afa
   Temporäre IPv6-Adresse. . . . . . : 2a02:8071:9282:e600:7154:ac74:30c4:cbee
   Verbindungslokale IPv6-Adresse  . : fe80::3031:9c6b:6485:3cc9%3
   IPv4-Adresse  . . . . . . . . . . : 192.168.178.218
   Subnetzmaske  . . . . . . . . . . : 255.255.255.0
   Standardgateway . . . . . . . . . : fe80::a96:d7ff:fe85:e002%3
                                       192.168.178.1

Tunneladapter isatap.{EDA0C976-3256-4B30-8A92-708D2F643E28}:

   Medienstatus. . . . . . . . . . . : Medium getrennt
   Verbindungsspezifisches DNS-Suffix:

Tunneladapter Teredo Tunneling Pseudo-Interface:

   Medienstatus. . . . . . . . . . . : Medium getrennt
   Verbindungsspezifisches DNS-Suffix:

Tunneladapter isatap.thermicorp.lan:

   Medienstatus. . . . . . . . . . . : Medium getrennt
   Verbindungsspezifisches DNS-Suffix: thermicorp.lan


C:\Users\Noel\bin>route -4 print
===========================================================================
Schnittstellenliste
  8...00 ff ed a0 c9 76 ......TAP-Windows Adapter V9
  3...08 00 27 ef 0b 0e ......Intel(R) PRO/1000 MT-Desktopadapter
  1...........................Software Loopback Interface 1
  4...00 00 00 00 00 00 00 e0 Microsoft-ISATAP-Adapter
  5...00 00 00 00 00 00 00 e0 Teredo Tunneling Pseudo-Interface
  6...00 00 00 00 00 00 00 e0 Microsoft-ISATAP-Adapter #2
===========================================================================

IPv4-Routentabelle
===========================================================================
Aktive Routen:
     Netzwerkziel    Netzwerkmaske          Gateway     Schnittstelle Metrik
          0.0.0.0          0.0.0.0    192.168.178.1  192.168.178.218     10
        127.0.0.0        255.0.0.0    Auf Verbindung         127.0.0.1    306
        127.0.0.1  255.255.255.255    Auf Verbindung         127.0.0.1    306
  127.255.255.255  255.255.255.255    Auf Verbindung         127.0.0.1    306
      172.16.20.2  255.255.255.255    Auf Verbindung      172.16.20.2     276
      172.16.25.2  255.255.255.255  169.254.128.128      172.16.20.2      30
```

```
63     192.168.178.0     255.255.255.0    Auf Verbindung    192.168.178.218     266
64   192.168.178.218   255.255.255.255    Auf Verbindung    192.168.178.218     266
65   192.168.178.255   255.255.255.255    Auf Verbindung    192.168.178.218     266
66         224.0.0.0         240.0.0.0    Auf Verbindung          127.0.0.1     306
67         224.0.0.0         240.0.0.0    Auf Verbindung        172.16.20.2     276
68         224.0.0.0         240.0.0.0    Auf Verbindung    192.168.178.218     266
69   255.255.255.255   255.255.255.255    Auf Verbindung          127.0.0.1     306
70   255.255.255.255   255.255.255.255    Auf Verbindung        172.16.20.2     276
71   255.255.255.255   255.255.255.255    Auf Verbindung    192.168.178.218     266
72 ===========================================================================
73 Ständige Routen:
74   Keine
```

6.3 Lizensierung der Arbeit

Diese BA steht unter der GNU General Public License Version 3 (GPLv3) und darf unter Berücksichtigung der Lizensvereinbarung der GPLv3 verfielfältigt und verteilt werden. Der Lizenztext ist in Unterabschnitt 6.4 oder auf der offiziellen Webseite[46] zu finden.

Das Logo der Hochschule Offenburg (HSO) steht unter seiner eigenen Lizenz. Es steht nicht unter der GPLv3. Der gesamte Code steht unter der URL `https://github.com/Thermi/Bachelorarbeit` zur Verfügung.
Diese Arbeit wurde mittels LaTeX erstellt

6.4 Lizenz

[46] `https://www.gnu.org/licenses/gpl.html`

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   (a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

   (b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

   (c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

   (d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

   You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

   (a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

   (b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

   (c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

   (d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

   (e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

   A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

(a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

(b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

(c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

(d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

(e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

(f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

    You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

    A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

    If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

    Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

End of Terms and Conditions

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <textyear>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.
```

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an “about box”.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see `http://www.gnu.org/licenses/`.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read `http://www.gnu.org/philosophy/why-not-lgpl.html`.