Gravierende Fehler in VPN-Protokollen und deren Lösung

Schutz (be)dürftig

Ein VPN strahlt Sicherheit aus – aber wie es um sie tatsächlich bestellt ist, bleibt meist im Dunkeln. Jenseits der drei wichtigen Protokolle SSL, SSH und IPsec finden sich fast nur Applikationen mit gravierenden Mängeln. Das Linux-Magazin erklärt die Hintergründe und beschreibt Auswege. Peter Gutmann

Protokolle und Programme für VPNs (Virtuelle Private Netze) gibt es viele, sowohl als Open Source als auch proprietär. Unklar ist aber meist, wie es um ihre Sicherheit bestellt ist. Der naive Ansatz mancher Entwickler, etwas Blowfish-Verschlüsselung und RSA-Schlüsselaustausch auf einen Netzwerksocket zu legen, reicht bei weitem nicht aus. Dieser Beitrag erklärt die typischen Fallen beim Entwickeln einer VPN-Applikation und beschreibt, wie Krypto-Experten diese Schwächen vermeiden.

Wer seine eigene VPN-Software entwickeln oder eine vorhandene Applikation um VPN-Funktionalität ergänzen will, sollte diese Grundlagen kennen und berücksichtigen. Leider finden sich für fast alle Fehler auch Beispiele aus der Praxis, die zu teils schwerwiegenden Sicherheitslücken führen. Bereits die in dem Kasten „Sicherheitsanforderungen“ genannten Punkte korrekt zu implementieren ist recht verzwickt. Abbildung 1 zeigt, wie eine sichere VPN-Implementierung aussehen sollte. Links ist das Handshake-Protokoll angedeutet, rechts die Datenübertragung.

Für den Handshake eignen sich Protokolle, wie sie auch TLS [11], SSH [12] oder IPsec-IKE [13] verwenden. Während dieser Phase handeln beide Seiten das Schlüsselmaterial aus, das sie dann bei der Datenübertragung einsetzen. Jedes Datenpaket erhält mehrere Schutzschichten: Die innerste Hülle sorgt für Vertraulichkeit (etwa mit 3DES), die nächste stellt die Integrität sicher (per HMAC-SHA1) und die äußere Schutzhülle kümmert sich um die Integrität des Datenstroms (hier mit dem Sliding-Window-Algorithmus von IPsec).

Viele VPN-Implementierungen halten sich aber nicht an diese Vorlage, sei es aus Unwissenheit oder um die vermeintlich überflüssige Komplexität zu vermeiden. Dabei stolpern die Entwickler immer wieder in die gleichen Fallen.

Einige VPN-Protokolle verschlüsseln mit RC4. Seit Windows 3.1 hat Microsoft diesen Algorithmus bis in die späten 1990er an allen möglichen Stellen eingesetzt. Sie haben es fast jedes Mal geschafft, RC4 falsch zu verwenden – daher meiden sie ihn inzwischen. Microsoft-freie Software setzt ihn aber heute noch ein, beispielsweise Eclipt [http://freshmeat.net/projects/ecliptsecuretunnel/] und Mirrordir [http://mirrordir.sf.net].

RC4: Eine Stromchiffre ...

RC4 ist eine einfach zu verwendende und daher populäre Stromchiffre, die mit einer Pseudozufallszahlenfolge arbeitet. Zum Verschlüsseln verknüpft der Absender den Schlüsselstrom per XOR mit den Klartextdaten (Abbildung 2, oben). Der Empfänger erzeugt denselben Schlüsselstrom und entschlüsselt den Chiffretext wieder mit XOR.

Dummerweise ist XOR aber kommutativ. Oft sind Chiffre und Klartext schon bekannt, etwa weil jemand eine E-Mail an einen VPN-Anwender sendet oder weil er die Reaktion einer E-Commerce-Applikation beobachtet, bei der er etwas eingekauft hat – dann deckt XOR direkt den Schlüsselstrom auf (Abbildung 2, mitte). Mit diesen Daten sind dann auch VPN-Pakete zu entschlüsseln, deren Inhalt vorher noch verborgen war.

Selbst ohne eigene Daten in das VPN einzuschleusen, kann ein neugieriger Zeitgenosse immerhin die XOR-Verknüpfung zweier Klartextpakete ermitteln: Er muss nur die verschlüsselten Pakete per XOR verknüpfen, die Schlüsseldaten beider Pakete heben sich dabei gegenseitig auf. Microsoft versuchte einmal, die eigenen RC4-Programme durch den Wech-

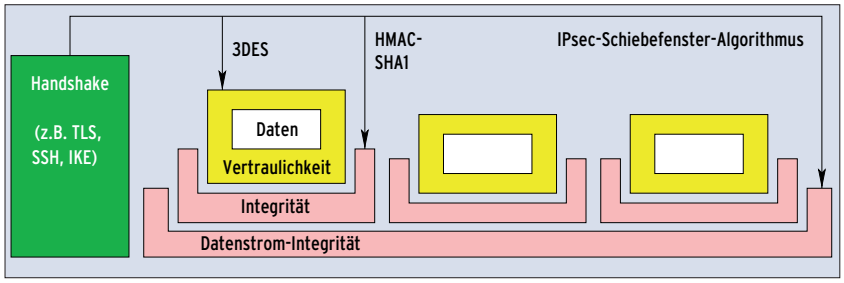

Abbildung 1: Ein sicheres VPN-Protokoll besteht aus einem Handshake-Protokoll (gute Vorlagen sind TLS, SSH oder das IKE von IPsec), mit dessen Schlüsselmaterial es dann die Daten chiffriert (etwa mit 3DES) und sie Integritäts-gesichert (etwa per HMAC-SHA1) überträgt.

sel von 32- auf 128-Bit-Schlüssel besser zu sichern. Das Kunststück konnte nicht gelingen, da die Schlüssellänge für diese Sicherheitslücken irrelevant ist.

Ein besonders einfacher Angriff gelingt bei RC4-basierten Challenge-Response-Protokollen: Die Challenge und die (ver- oder entschlüsselte) Response genügen, um den Schlüsselstrom per XOR zu erfahren. Übrigens haben 802.11-Hersteller (WLAN) diesen Teil von WEP klammheimlich aufgegeben, als ihnen jemand erklärte, was sie da standardisierten.

... mit vielen Problemen

RC4 hat noch weitere kryptographische Schwächen, die teilweise schon seit Jahren bekannt sind. Sie sind zwar weniger gravierend, aber es bleibt fragwürdig, wenn ein neu entwickeltes Produkt heute noch RC4 ohne besondere Vorkehrungen einsetzt. Zudem erlaubt es RC4 dem Angreifer, die Nachricht nach Belieben zu ändern. Dazu später mehr.

Die Alternative zu Stromchiffren sind Blockchiffren. Die bekanntesten Vertreter heißen DES und 3DES, das recht aktuelle AES, IDEA (aus PGP 2), Blowfish (aus Bruce Schneiers Buch „Angewandte Kryptografie“) und CAST (PGP 5). Es gibt keine grundlegenden Unterschiede zwischen diesen Algorithmen. 3DES ist die konservativste Wahl, es wurde am genauesten analysiert, ist seit einer Ewigkeit bekannt und wird von praktisch allem unterstützt. Der Nachfolger AES ist zwar schneller, aber auch jünger und hatte daher weniger Zeit, sich im harten Krypto-Alltag zu behaupten. Der Vorteil ist zudem relativ: Selbst eine ältere 1-GHz-CPU verschlüsselt mit 3DES immerhin etwa 50 MBit/Sekunde, mit AES schafft sie rund 100 MBit/Sekunde. Auf der Freeswan-Homepage sind weitere Performance-Daten zu finden [1].

Blockchiffren

Wie der Name andeutet, verschlüsselt eine Blockchiffre die Daten blockweise, meist 64 oder 128 Bit am Stück. Sie ist schwerer zu benutzen als RC4, die Byteweise arbeitet: Nachrichten sind in der

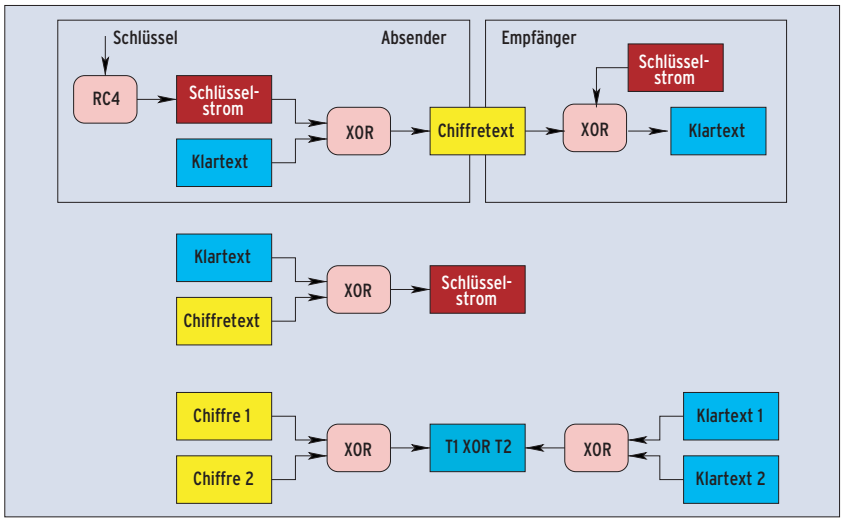

Abbildung 2: Bei RC4 generiert der Absender einen Schlüsselstrom, den er per XOR mit dem Klartext verknüpft (oben). Wer Klartext in das System injiziert und das Chiffrat abhört, kann den Schlüsselstrom ermitteln (mitte). Aus zwei Chiffretexten lässt sich die XOR-Verknüpfung der Klartexte entschlüsseln (unten).

Sicherheitsanforderungen

Ein VPN muss einige Anforderungen erfüllen, um tatsächlich sicherer zu sein als das Internetprotokoll. Dazu gehören:

Vertraulichkeit: Ein Angreifer darf den Inhalt einer Nachricht nicht erfahren. Die Nachricht einfach nur verschlüsseln ist nicht ausreichend. Ein Angreifer kann unter Umständen schon Schaden anrichten, wenn er nur die Länge der Nachricht weiß, bestimmte Bitmuster in den verschlüsselten Daten bemerkt oder ein einzelnes Datenbit aufdeckt (etwa ein Flag).

Authentizität und Zugangskontrolle: Ein Angreifer sollte es nicht schaffen, unbemerkt Daten unter falschem Namen ins VPN einzuschleusen. Die Abwehr kann sehr knifflig sein – der Saboteur könnte eine authentische Nachricht aufzeichnen und später ins VPN einfügen. Der Empfänger muss das bemerken.

Integritätssicherung: Ein Angreifer soll den Inhalt einer Nachricht nicht unbemerkt manipulieren können. Diese Anforderung lässt sich noch erweitern, um auch die Integrität des Datenflusses sicherzustellen. Dem Angreifer soll es nicht gelingen, Nachrichten einzufügen oder zu wiederholen (»Zahle 10000 Euro auf mein Konto«), zu löschen (»Achtung, jemand manipuliert deine Nachrichten«) oder die Reihenfolge zu ändern (»rm backup«, »mv valuable_data backup«). Für viele Angriffe auf den Datenfluss muss der Angreifer weder die Vertraulichkeit und Authentizität noch die Integrität der Nachrichten brechen.

Verfügbarkeit: Eine weitere, etwas schwächere Anforderung ist der Schutz vor Denial-of-Service-Attacken. Auf die Verfügbarkeit hat eine Sicherheitsschicht häufig keinen Einfluss, beispielsweise muss sich der IP-Stack selbst vor SYN-Flooding-Angriffen schützen. Dennoch ist DoS-Schutz eine wichtige Aufgabe für ein VPN, da es viele aufwändige Krypto-Algorithmen berechnet (RSA oder Diffie-Hellman-Schlüsseltausch). Ein Angreifer könnte den Server mit vielen Client-Connect-Nachrichten überfluten. Den Client belastet das kaum, es genügt, wenn er Zufallswerte schreibt. Der Server muss jedoch kostspielige Krypto-Berechnung durchführen, nur um zu bemerken, dass die Nachricht Datenmüll enthält.

Regel keine Vielfachen von 64 oder 128 Bit lang. Im DES-Standard wurden daher einige Betriebsmodi mit unterschiedlichen Eigenschaften definiert. Beispielsweise verbergen sie Datenmuster und arbeiten Byte-weise.

Gute und schlechte Modi

Der unsicherste Modus ist ECB (Electronic Codebook), er verschlüsselt die 64-Bit-Blöcke unabhängig voneinander. Obwohl jedes Kryptographie-Buch vor diesem Modus ausdrücklich warnt, kommt er in VTun zum Einsatz [http://vtun.sourceforge.net/].

Da ECB jeden Datenblock einzeln verschlüsselt, führt ein gegebener Klartext immer zum selben Chiffretext. Wenn ein VPN-User »ABC« in den Tunnel sendet, das Netzwerk abhört und sieht, dass seine Daten verschlüsselt »XYZ« lauten, weiß er in Zukunft, dass der Chiffretext »XYZ« dem Klartext »ABC« entspricht. Durch einfaches Ausprobieren kann er an Teile von fremden Nachrichten kommen, ohne den Schlüssel zu kennen. Vergleichen lässt sich das mit einem Unix-Passwort-Verschlüsselungsalgorithmus, der keinen Salt benutzt.

Die Situation ist sogar noch schlimmer: Da alle Blöcke unabhängig voneinander sind, kann ein Angreifer bequem Copy & Paste-Angriffe durchführen, doch dazu später mehr. Bei jedem Krypto-Experten schrillen die Alarmglocken, wenn er Verschlüsselungssoftware mit ECB-Modus sieht – ähnlich wie bei einem Entwickler, der „Programmieren: Visual Basic und HTML“ im Lebenslauf eines Job-Bewerbers entdeckt. Andere Modi haben diese Probleme nicht, siehe Abbildung 3. Am weitesten verbreitetet ist CBC (Cipher Block Chaining). Bei CBC und CFB (Cipher Feedback) hängt Block »*n*« vom Inhalt des Blocks »*n*-1« ab. Ein zufällig gewählter Initialisierungsvektor (IV, der Block »-1«) schützt auch den ersten Nachrichtenblock.

Selbst wenn derselbe Klartext mehrmals mit dem gleichen Schlüssel verschlüsselt wird, unterscheiden sich die Chiffretexte. Die oben beschriebenen Trivialangriffe auf ECB sind so viel schwieriger, wenn auch nicht unmöglich. Das so genannte Geburtstagsparadoxon (siehe Kasten „Geburtstagsparadoxon“) lässt erwarten, dass sich bei einer 64-Bit-Blockchiffre nach nur 2^{32} Blöcken einer wiederholt. Aus diesem Grund verwendet AES 128-Bit-Blöcke.

In der Kryptographie sind diese Schwächen als Certificational Weakness bekannt. Es ist zwar nicht sehr wahr-

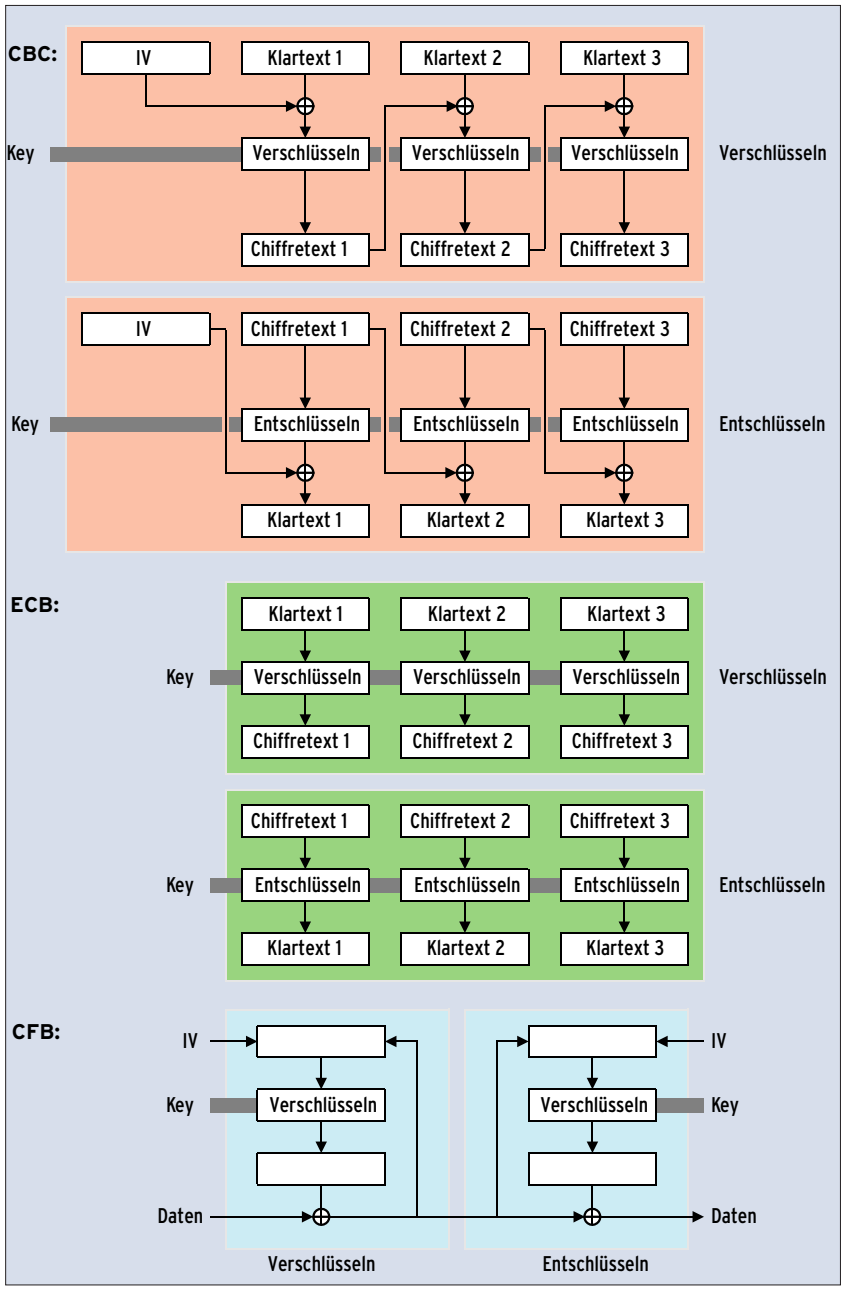

Abbildung 3: Für Blockverschlüsselungs-Algorithmen sind mehrere Betriebsmodi definiert, um auch größere Nachrichten zu verarbeiten. Bei CBC (oben) und CFB (unten) hängt jeder Chiffreblock vom Klartext und dem vorherigen Chiffretext ab, während ECB (mitte) jeden Block einzeln verschlüsselt.

Geburtstagsparadoxon

Wie groß muss eine Gruppe sein, dass mit 50 Prozent Wahrscheinlichkeit zwei am selben Tag des Jahres Geburtstag haben? Die intuitive Antwort liegt meist weit neben der richtigen. Dass auf einer Party jemand am selben Tag Geburtstag hat wie der Gastgeber, ist mit 0,3 Prozent (1/365) recht unwahrscheinlich. Erst bei einer Party mit 253 Gästen ist eine Wahrscheinlichkeit von 50 Prozent gegeben. Sollen dagegen nur zwei beliebige Gäste am selben Tag Geburtstag haben, dann genügt bereits eine Gruppe von 23 Personen.

Dieses Paradoxon hat große Bedeutung für viele Sicherheitsprobleme in der Kryptographie: Dieselbe Fehleinschätzung führt zum Beispiel dazu, dass Blockgrößen und MAC-Codes häufig zu kurz gewählt werden.

scheinlich, dass sie jemand tatsächlich ausnutzen kann. Wer aber die Wahl zwischen einem anfälligen und einem immunen Algorithmus oder Modus hat, sollte sich immer für die sichere Variante entscheiden. Vielleicht findet doch jemand einen Weg, diese theoretische Schwäche tatsächlich auszunutzen.

Theorie und Praxis

Eine ganze Reihe realer Angriffe auf Sicherheitsprotokolle bewiesen in den letzten Jahren, wie schnell sich das Blatt wenden kann: Noch kurz bevor die Details so manch erfolgreicher Attacke veröffentlicht wurden, hielt man die jeweiligen Schwächen für rein hypothetisch und praktisch nicht ausnutzbar. Gute Protokollentwickler sind bei ihren Designs daher überaus vorsichtig, um auch gegen Lücken gewappnet zu sein, die manche Kryptologen nur müde belächeln. So vermeiden sie es, ein oder zwei Jahre später alles neu aufsetzen zu müssen, nur weil doch jemand einen Angriffsweg gefunden hat.

Auch IVs haben potenzielle Probleme. Vor allem muss ein Protokoll sie auch einsetzen – Tunnelvision [http://open.nit.ca/wiki/?page=TunnelVision] und Zebedee [http://www.winton.org.uk/zebedee/] verzichten darauf. Der IV muss zufällig und unvorhersehbar sein [2], [3]. SSL und SSH benutzen als IV des Pakets »*n*« aber den letzten Block von Paket »*n*-1«. Hört ein Angreifer »*n*-1« ab, kennt er auch den IV von Paket »*n*«. TLS 1.1, der Nachfolger von SSL 3, behebt diese Schwäche. Die IPsec-Entwickler kannten das Problem seit Jahren und vermieden den Fehler [4]. Die Schwäche gilt derzeit zwar als Certificational Weakness, niemand hat einen Weg gefunden, damit echten Schaden anzurichten. Doch das bedeuten nicht, dass man sie beruhigt ignorieren darf.

Mache Programmierer denken, dass verschlüsselte Nachrichten auch vor Manipulation geschützt wären. Nichts könnte falscher sein. Oft ist es sehr einfach, den verschlüsselten Text beliebig zu ändern, ohne den Schlüssel zu kennen. Ein Beispiel ist das oben beschriebene RC4, zu finden in Eclipt, Mirrordir und einer Menge von Microsoft-Programmen.

Integrität der Daten

Abbildung 4 zeigt, wie leicht ein einziges geändertes Bit die Bedeutung einer verschlüsselten Nachricht verdreht. Bei RC4 kann ein Saboteur seine Manipulation Byte-genau ansetzen, da der Algorithmus jedes Klartext-Byte auf genau ein Chiffretext-Byte abbildet. Bei anderen Stromchiffren wie CFB ist das Bit-Umdrehen ähnlich einfach. Dennoch benutzen mehrere VPN-Lösungen RC4 oder CFB ohne Integritätsschutz.

Bei Blockchiffren gelingt die Operation nicht so zielgenau, da ein geändertes Chiffretext-Bit etwa die Hälfte aller Klartext-Bits umdreht. Arbeitet die Chiffre im ECB-Modus, kann der Angreifer mit den

bereits angedeuteten Copy & Paste-Techniken arbeiten. Damit durchbricht er spielend die Sicherheit von VTun, trotz 3DES-Verschlüsselung und ohne den Key zu kennen.

Kopier-Angriff

Wesentlich ist, dass ECB jeden Block unabhängig von allen anderen verschlüsselt. Abbildung 5 zeigt, wie sich das ausnutzen lässt: Der Angreifer sendet über einen Banking-Server eine reguläre Überweisung an sich selbst. Die Bank schickt die Transaktion über ihr VPN, das der Angreifer belauscht und so die verschlüsselte Fassung seines Auftrags erfährt. Aus dieser verschlüsselten Nachricht schneidet er den Block heraus, der seine (verschlüsselte) Kontonummer enthält, und fügt ihn in eine fremde Überweisung ein.

Ohne den Schlüssel zu kennen und ohne die fremde Überweisung zu entschlüsseln, gelingt es so, gezielt Daten zu manipulieren. Der Angreifer weiß zwar nicht, wie viel seine Aktion einbringt oder von wem das Geld stammt. Ist er mit dem Ergebnis auf seinem nächsten Kontoauszug unzufrieden, wiederholt er den virtuellen Raubzug einfach.

Beim CBC-Modus ist dieser Angriff nicht so einfach möglich, da jeder Block vom vorherigen abhängt. Einige VPNs verlassen sich daher auf CBC oder CFB, um die Schwäche von ECB zu vermeiden. Tatsächlich schützen CBC und CFB besser vor Manipulationen, ihr Vorsprung ist aber nicht sehr groß, denn sie haben die – in manchen Situationen nützliche – Eigenschaft, sich nach einem Fehler wieder aufzusynchronisieren.

Ein fehlerhafter Block zerstört zwar auch seinen Nachfolger, danach lässt sich die Nachricht aber wieder korrekt dechiffrieren. Der Copy & Paste-Angriff ist daher weiterhin möglich, wenn der Angreifer in Kauf nimmt, dass er die Daten am Beginn des eingefügten Blocks sowie den Folgeblock unvorhersehbar ändert. Bei CFB und RC4-ähnlichen Stromchiffren sind Manipulationen am letzten Block nicht erkennbar.

Echter Integritätsschutz

Ein funktionierender Integritätsschutz ist nur durch spezielle Maßnahmen möglich. Keiner der Standardmodi leistet dies – erst in jüngster Zeit versuchen einige Neuentwicklungen, eine Integritätssicherung einzubauen. Diese Modi sind aber noch kaum erprobt und außerdem meist mit Patenten belastet. Der erste Versuch, einen solchen Schutz herzustellen, ist häufig eine einfache CRC-Prüfsumme. Manche VPNs nutzen nicht einmal diesen Minimalschutz. So verwenden Cipe [http://sites.inka.de/sites/bigred/devel/cipe.html] und die überholte SSH-Version 1 zwar CRC, nur ist diese Prüfsummme unsicher.

CRCs entdecken versehentliche Änderungen sehr zuverlässig, gegen absichtliche Manipulationen sind sie aber völlig machtlos. Es ist sehr einfach, eine Nachricht mit einem zuvor bestimmten CRC zu generieren – und damit in einer Fälschung den CRC-Code der Originalnachricht nachzubilden. Selbst ein verschlüsselter CRC hilft hier nicht.

Der genaue Ablauf der CRC-Angriffe ist recht komplex, aber en détail in [5] nachzulesen. Interessanterweise galt der Einsatz von CRC-32 bereits ein Jahrzehnt lang als hypothetische Schwäche von Kerberos, bevor ein SSHv1-Exploit diesen Fehler praktisch ausnutzte. Die Schwäche sorgte dafür, dass SSHv1 heute beinahe bedeutungslos ist.

Angemessenen Integritätsschutz leisten spezielle kryptographische Prüfsummen, so genannte MACs (Message Authentication Code). Aber auch hier lauern Fallen. Wer sein eigenes Protokoll nach der IPsec-Vorlage entwickelt, ohne deren Interna verstanden zu haben, neigt dazu, den MAC ebenfalls auf 96 Bit zu beschränken. Dabei ist dieser Wert bei IPsec rein pragmatisch gewählt: So passt der gesamte AH-Header mit seinen 128 Bit in vier 32-Bit-Wörter. Immerhin sind 96 Bit wesentlich besser als die auch gern genommenen 32 Bit.

SSL/TLS und SSHv2 belassen den MAC korrekt bei vollen 160 Bit (eine übliche Größe guter MAC-Algorithmen). VPNd [http://sunsite.dk/vpnd/] setzt zwar auch einen MAC ein, beschneidet ihn aber auf mickrige 16 Bit. Auf einer T1-Leitung dauert es gerade mal eine halbe Sekunde, bis zwei Pakete denselben MAC tragen – das Geburtstagsparadoxon lässt grüßen. VPNd kann statt dieses MAC auch mit einer selbst gestrickten 16-Bit-Prüfsumme arbeiten (»checksum ()« in der Datei »crypto.c«):

```
while(length--)
  sum=((sum<<1)|((sum>>15)&1))^*data++;
```

Dieser Algorithmus hat recht ärmliche Eigenschaften. Die Prüfsumme ist zyklisch: Nach 16 Bytes mit identischen Daten stehen in den niederwertigen 16 Bits nur Einsen, nach weiteren 16 Datenbytes enthalten alle Bits Nullen. Danach wiederholt sich der Zyklus.

Integrität des Datenflusses

Spätestens wenn es um den Datenfluss geht, ist Sicherheit knifflig. Selbst wenn ein VPN die Nachrichten korrekt verschlüsselt und deren Integrität sicherstellt, kann ein Angreifer Chaos stiften:

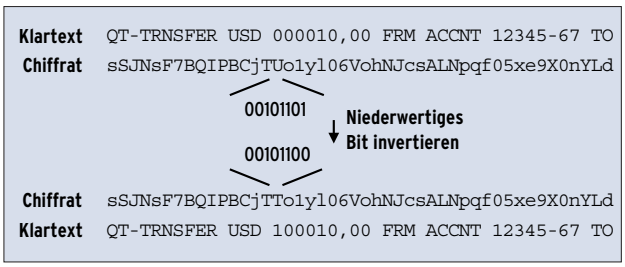

Abbildung 4: Ein einziges geändertes Bit kann bei RC4-verschlüsselten Daten schwerwiegende Folgen haben: Statt 10 Dollar spült dieser EDI-Auftrag ganze 100 010 Dollar auf das Konto des Empfängers.

Deposit	$10,000	in acct.	number	12-3456-	789012-3
H2nx/GHE	KgvldSbq	GQHbrUt5	tYf6K7ug	S4CrMTvH	7eMPZcE2
H2nx/GHE	5guZEHVr	GQHbrUt5	tYf6K7ug	Pts21LGb	a8oaNWpj
H2nx/GHE	5guZEHVr	GQHbrUt5	tYf6K7ug	S4CrMTvH	7eMPZcE2

Abbildung 5: Aus der verschlüsselten Form einer reguläre Überweisung (oben) kopiert der Angreifer zwei Blöcke in einen fremden Bankauftrag und leitet ihn so auf sein eigenes Konto (unten). Schuld ist der ECB-Modus der Blockchiffre.

Er wiederholt Nachrichten, löscht sie, fügt neue ein oder ändert ihre Reihenfolge. Kaum ein VPN schützt davor. Denselben Fehler begingen auch frühe IPsec-Versionen, die kaum Integritätssicherung für den Datenfluss enthielten. Der Grund: IPsec tunnelt nur IP-Pakete und IP geht selbst von unzuverlässigen Leitungen aus. In dieser Welt kümmert sich TCP um verlorene, vertauschte oder duplizierte Pakete.

Ähnlich wie CRC, das zwar zufällige Fehler zuverlässig erkennt, bei vorsätzlichen Änderungen aber kläglich versagt, sind auch mit beabsichtigten Netzwerkfehlern allerhand hässliche Dinge möglich. TCP und verwandte Protokolle waren nie für den Schutz vor vorsätzlichen Manipulationen ausgelegt. Steve Bellovin – einer der Väter von IPsec – veröffentlichte einen ganzen Katalog von Problemen **[6]**, die daraus entstehen. In IPsec sind diese Schwächen inzwischen beseitigt (siehe Abbildung 6), in allen anderen bisher erwähnten VPN-Applikationen aber nicht.

Falsche Erwartungen

VPN-Benutzer gehen oft automatisch davon aus, dass ihnen ein VPN mit seinen Sicherheitsfunktionen auch einen verlässlicheren Dienst (QoS, Qualit of Service) bietet als ein ungeschütztes Netz. Diese Annahme ist sogar recht nahe liegend, da ein VPN ja mehr bieten soll als einfaches IP. Es kann auch sein, dass die Anwender etwas anderes als IP durch den Tunnel schicken; Protokolle, die erst gar nicht versuchen, die Datenfluss-Integrität zu sichern. So überträgt etwa VPE [http://savannah.nongnu.org/projects/vpe] Ethernet-Frames.

In diesen Fällen erwarten VPN-Entwickler und -Anwender vom jeweils anderen, dass er sich um die Datenflussintegrität kümmert. Dabei ist die Lösung so einfach: Eine Paketnummer innerhalb des kryptographisch geschützten Umschlags genügt. Der Empfänger kann die Pakete dann in die richtige Reihenfolge bringen, Dubletten entfernen und sich bei Bedarf über vermisste Pakete beklagen.

Bei einer TCP-Verbindung hat es der Empfänger einfach, das Protokoll sorgt für einen verlässlichen Datenstrom. Jede Dublette, Auslassung oder verdrehte Reihenfolge deutet auf einen Angriff hin. Bei UDP ist es deutlich schwieriger, da das Protokoll selbst keinen zuverlässigen Dienst garantiert. IPsec **[7]** löst das Problem durch einen Schiebefenster-Algorithmus (Sliding Window). Er sorgt für einen abschnittsweise verlässlichen Dienst auf IPsec-Ebene.

Stattdessen könnte ein VPN auch eine Schutzschicht auf UDP aufsetzen, das Ganze wie eine TCP-Verbindung betrachten und mit obiger Sequenznummer arbeiten. In beiden Fällen muss der Entwickler aber sehr umsichtig sein, um alle Detailfragen richtig zu lösen.

Unkontrollierter Kontrollkanal

Als ob es nicht schon genug Probleme gäbe, senden einige VPNs auch ihre Daten zur Sitzungssteuerung über den nicht besonders sicheren Tunnel, den sie selbst aufgebaut haben. Da der Kontrollkanal nicht über TCP läuft, kann er noch nicht einmal auf diesen Minimalschutz vertrauen. VPN-Sitzungssteuerung ist aber wesentlich sensibler als gewöhnliche Daten, da ein erfolgreicher Angriff auf eine einzige Steuerungsnachricht sämtliche Nutzdaten kompromittieren kann. SSL, SSH und IPsec führen daher einen vollständigen Handshake durch, wenn sie Kryptoparameter ändern, und jagen nicht einfach einen neuen Schlüssel durch den Tunnel.

Fehlt jeglicher Replay-Schutz, dann kann ein Angreifer eine alte Steuerungsnachricht wieder einspielen und das VPN dazu bringen, einen bereits kompromittierten Schlüssel erneut einzusetzen. Da die meisten VPNs Pakete mit falschen Prüfsummen einfach ignorieren, genügt ein invertiertes Bit, um Schlüsselaustausch-Botschaften zu annullieren. Das VPN arbeitet dann weiter mit dem kompromittierten Key. Da die meisten VPNs die tatsächliche Paketlänge nicht verschleiern (also kein Padding hinzufügen), erkennen Angreifer die Managementpakete an deren Länge und manipulieren sie zielsicher.

Die Paketlänge verschleiern (Abbildung 6) verhindert auch, dass andere Pakete mit konstanter Länge als solche erkennbar sind, beispielsweise TCP-ACKs, ARP-Requests oder DHCP-Discover-Nachrichten. Bei ICMP könnten Angreifer sogar Nachrichten und deren Antwort abfangen, auf den Inhalt schließen und eigene Pakete einschleusen **[8]**. Um besonders sensible Daten wie die Passworteingabe zu verschleiern, füllen Protokolle wie SSH die Pakete oft auf eine feste Länge auf oder senden zusätzlichen Datenmüll, der das Vorhandensein wertvoller Daten verbirgt.

Verschlüsselung abschalten

Weitere Angriffe auf Schlüsselaustausch-Botschaften sind möglich, wenn ein Protokoll die Integrität der Pakete nicht ausreichend schützt. Der nahe liegende Angriff versucht die Bits so zu manipulieren, dass der Schlüssel nur noch aus Nullen besteht. Bei einigen Algorithmen ist die Attacke noch wesentlich einfacher. Es genügt, ein beliebiges Bit im Schlüssel zu verändern, egal welcher Wert dabei entsteht. Er muss sich nur vom Original unterscheiden. Dieser An-

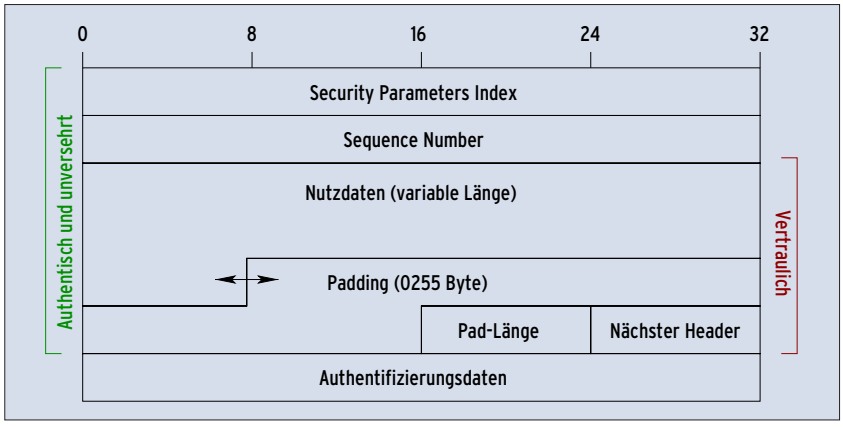

Abbildung 6: Am Header eines ESP-Pakets sind wesentliche Sicherheitsmerkmale zu erkennen. Beispielsweise schützt die Sequence-Nummer vor Replay-Angriffen, das Padding verschleiert die Länge der Nutzdaten.

griff funktioniert bei DES und 3DES, weil deren Schlüssel Parity-Bits enthalten. Diese Idee stammt aus dem Jahr 1970, als DES entwickelt wurde. Laut Spezifikation soll ein System nur Schlüssel verwenden, bei denen alle Paritätsbits stimmen. Sind sie falsch, deutet das auf einen Übertragungsfehler hin.

Ausnutzen lässt sich dieser Umstand, weil die meisten VPN-Implementierungen die Rückgabewerte ihrer Krypto-Funktionen nicht prüfen und einfach davon ausgehen, dass sie immer arbeiten. Beispiele solch nachlässiger Programmierung sind Tunnelvision, VPE und Zebedee. Der Angreifer ändert einfach ein Bit eines Pakets, das einen Schlüssel transportiert. Wenn die VPN-Software den manipulierten Key verwenden will, scheitert dies wegen der falschen Parity. Die Software merkt davon nichts, weil sie den Rückgabewert ignoriert, und arbeitet blindlings weiter. Da das Laden des echten Keys scheiterte, verwendet sie nun einen Null-Schlüssel.

Es handelt sich hier nicht nur um einen Fall schlechter Programmierung. Bei einer vernünftig entworfenen Sicherheitsapplikation darf es gar nicht dazu kommen, dass sie einen sabotierten Schlüssel lädt. Sie muss die Manipulation vorher entdecken. Der DES-Schlüssel ist nur ein einfaches Beispiel, es gibt noch eine Menge ähnlicher Angriffe.

Tabelle 1: Abkürzungen

3DES	Triple-DES	IP	Internet Protocol
ACK	Acknowledge	IPsec	Internet Protocol Security
AES	Advanced Encryption Standard	IV	Initialisierungsvektor
AH	Authentication Header	MAC	Message Authentication Code
ARP	Address Resolution Protocol	PGP	Pretty Good Privacy
CAST	Carlisle Adams, Stafford Tavares	PPTP	Point-to-Point Tunneling Protocol
CBC	Cipher Block Chaining	QoS	Qualit of Service
CERT	Computer Emergency Response Team	RC4	Rivest Cipher 4
CFB	Cipher Feedback	RSA	Rivest Shamir Adleman
CRC	Cyclic Redundancy Check	SHA1	Secure Hash Algorithm 1
DES	Data Encryption Standard	SIGMA	Sign-and-MAC
DHCP	Dynamic Host Configuration Protocol	SSH	Secure Shell
DoS	Denial of Service	SWIFT	Society for Worldwide Interbank Funds Transfer
ECB	Electronic Codebook		
EDI	Electronic Data Interchange	TCP	Transmission Control Protocol
ESP	Encapsulating Security Payload	TLS	Transport Layer Security
HMAC	Keyed-Hashing for Message Authentication	UDP	User Datagram Protocol
		VPN	Virtuelles Privates Netz
ICMP	Internet Control Message Protocol	WEP	Wired Equivalent Protocol
IKE	Internet Key Exchange	XOR	Exklusiv-Oder (Antivalenz)

Weitere Probleme

Neben dem Ärger mit den VPN-Daten und der Sitzungssteuerung bieten vor allem der erste Handshake und die Authentifizierung ein weites Angriffsfeld. Dies Thema ist deutlich komplexer und füllt ganze Sicherheitsbücher (Kasten „Weitere Quellen“). Zu allem Überfluss verwenden die meisten VPN-Implementierungen selbst gestrickte und kaum dokumentierte Handshake-Mechanismen. Eine vernünftige Evaluierung müsste daher erst das Design per Reverse-Engineering der Quellen ermitteln.

Zu den Fehlern im Protokolldesign gesellen sich bei vielen Programmen auch noch Mängel bei der Implementierung. Mache VPNs generieren ihre Schlüssel anhand der Prozess-ID, der aktuellen Uhrzeit oder mit der »rand()«-Funktion und machen es einem Angreifer damit sehr leicht, den Schlüssel zu erraten. Gute, nicht vorhersehbare oder nachträglich berechenbare Zufallswerte zu finden ist ein weiteres Thema, das Bücher füllen würde.

Beliebt sind auch die klassischen C-Programmierfallen: Wer Eingabedaten nicht oder nicht ausreichen prüft, handelt sich Buffer Overflows ein oder rechnet mit Zahlen, die außerhalb des erwarteten Wertebereichs liegen. Bruce Schneier und Mudge stießen auf Fehler dieser Kategorie, als sie Microsofts PPTP-Implementierung analysierten. Dabei wollten sie nur Protokollfehler untersuchen und nicht den Server angreifen. Sie hatten enorme Schwierigkeiten, ihre Theorien über Protokollfehler praktisch zu belegen, da bei den meisten Experimenten der Server abstürzte [9].

Nur ein ausführlicher und detaillierter Code-Audit könnte all diese Probleme aufdecken. Bei jeder Schwachstelle wäre dann zu klären, ob es sich um einen Fehler im Protokolldesign handelt oder schlicht um einen Bug in der Software.

Bitte hier angreifen

Einige VPNs bieten mehrere Betriebsmodi, meist ist einer auf Sicherheit optimiert und ein anderer verspricht besseren Durchsatz auf Kosten der Sicherheit. Das ist völlig unnötig, der zusätzliche

Weitere Quellen

Wer tiefer in das Thema einsteigen will, findet in Charlie Kaufman, Radia Perlman und Mike Speciners Buch „Network Security: Private Communication in a Public World“ eine gute Informationsquelle. Es dient als allgemeine Referenz für Krypto- und Security-Fragen, beschreibt viele Sicherheitsmechanismen, erklärt, warum so viele Mechanismen unsicher sind, und empfiehlt passende Lösungen.

Bruce Schneier und Niels Ferguson gehen in „Practical Cryptography“ schrittweise den Entwurf eines SSL/SSH-ähnlichen Protokolls durch. Sie beschreiben jede Stufe, die in einem sicheren Design nötig ist, um potenzielle Schwächen zu vermeiden. Und „Lessons Learned in Implementing and Deploying Crypto Software“ vom Autor dieses Artikels – unter [http://www.cs.auckland.ac.nz/~pgut001/pubs/usenix02 .pdf] – beleuchtet einige Fallen für Entwickler, wenn sie sichere Krypto-Bausteine unsicher zusammensetzen.

Bei IPsec-VPNs fällt die Nachlese mager aus. Die RFCs enthalten kaum Informationen dazu, warum das Protokoll so ist, wie es ist. Die meisten Bücher zu IPsec geben auch nicht mehr her. Wer mit dem hohen Traffic auf der IPsec-Mailingliste zurechtkommt, findet in deren Archiv einen reichen Erfahrungsschatz. [http://www.vpnc.org/ietf-ipsec/]

Weiteres Hintergrundmaterial ist kaum zu finden, zumal sich Details im Laufe der Zeit häufig ändern. „The SIGMA Family of Key-Exchange Protocols“ – unter [http://www.ee.technion.ac.il/~hugo/sigma.html] – gibt einen lesenswerten Überblick über die Aufgaben, die das Key-Management in IPsec erledigt.

Overhead für guten Schutz der Nachrichten- und Datenfluss-Integrität ist unbedeutend im Vergleich zum allgemeinen VPN-Overhead. Der Durchsatz-optimierte Modus sollte besser „Bitte hier angreifen“-Modus heißen. Jeder halbwegs intelligente Angreifer wird den sicheren Modus ignorieren und sich sofort auf den anderen stürzen.

Setzt ein Opfer sein VPN im sicheren Modus auf, bremst der Angreifer die Online-Anbindung mit einer von tausenden Standard-DoS-Techniken. Sobald das Opfer auf den „Bitte hier angreifen“-Modus umschaltet, stoppt der Saboteur den DoS. Aus Sicht des Opfers hat der sichere Modus Schuld am miserablen Durchsatz, da das Problem jetzt behoben ist. Beide sind glücklich: Angreifer und Opfer. In besseren Protokollen (etwa SSL) sind Mechanismen eingebaut, die vor solchen Tricks schützen.

Der Downgrade-Angriff ist auch eine jener Standardtechniken, die bei Attacken gegen die Authentifizierungs-Mechanismen von Windows zum Einsatz kommt. Warum sich mit den neuen, besseren Mechanismen herumschlagen, wenn man das Opfer leicht dazu bringen kann, in die hoffnungslos unsichere Prä-Windows-3.1-Ära zurückzufallen.

Ein vergleichbarer Angriff wurde Mitte der 1980er vom Swift-Netzwerk berichtet. Die „Society for Worldwide Interbank Funds Transfer“ bewegt täglich etwa 6 Billionen US-Dollar. Ihre Link-Verschlüsselung funktionierte über Testleitungen völlig problemlos. Jeder Versuch im Produktivbetrieb führte aber zu korrumpierten Daten und verlorenen Paketen. Nach einigen Wochen gab die Bank auf und sandte alle Finanztransaktionen im Klartext. Nun traten keine Übertragungsfehler mehr auf.

Reine Zeitfrage

Einige Probleme sind sehr subtil. Zum Beispiel können zwei sichere Subsysteme zusammengenommen zu einem unsicheren Protokoll führen. Sind sowohl der Schutz der Vertraulichkeit als auch der Integritätsschutz eines VPN jeweils korrekt entworfen und implementiert, kann ihre Kombination dennoch zu Problemen führen. Das VPN wird die Nachricht erst mit dem MAC versehen und dann verschlüsseln.

Der Angreifer manipuliert eine Nachricht und wartet ab, wie der Empfänger reagiert. Lehnt er das Paket rasch ab, hat er die Änderung schon beim Entschlüsseln entdeckt. Dauert es etwas länger, dann musste der Empfänger erst noch den MAC prüfen. Der Angreifer weiß jetzt, ob seine Aktion die Verschlüsselung oder den MAC trifft.

Derzeit gilt dies allgemein als hypothetische Schwäche – was nicht heißt, dass nicht zehn Minuten nach Veröffentlichung dieses Artikels jemand das Problem erfolgreich ausnutzt. Noch schlimmer: Er könnte einen Exploit entwickeln, zwei Jahre nachdem die verwundbare Software im großen Stil installiert wurde. Falls so etwas einem der etablierten Protokolle wie SSL oder SSH passiert, wird das CERT ein Advisory veröffentlichen und alle SSL- und SSH-Hersteller werden das Loch stopfen.

Die Analyse, durch die dieses Timing-Problem bekannt wurde, ist jünger als SSH und SSL **[10]**. Trotzdem ist SSH nicht anfällig und die SSL-Implementierungen enthalten inzwischen Workarounds. Sollte die Schwäche auch in anderen Protokollen enthalten sein, wird das aber niemand bemerken, da Kryptologen die Protokolle jenseits des Mainstream kaum beachten.

Die Tatsache, dass sich so viele VPN-Applikationen durch einfachste Angriffe aushebeln lassen, belegt, wie gefährlich der Mangel an eingehender Kryptoanalyse ist. Dass keiner der Guten eine Lücke bemerkt, ist kein Garant dafür, dass die schwarzen Schafe im Internet ihre große Chance ignorieren. Entdeckt jemand eine neue Angriffstechnik, dann müssen sich die großen Drei dagegen bewähren (SSL, SSH, IPsec). Meist findet dabei einiges an Fremdbestäubung zwischen den Protokollen statt, um alle Varianten zu immunisieren. Die neue Tech-

Mahnende Worte

Ein sicheres VPN-Protokoll entwerfen ist schwer, sehr schwer sogar. Ich würde mich selbst unwohl dabei fühlen – und ich verdiene meinen Lebensunterhalt mit Kryptographie und anderen Sicherheitsfragen. Einige der weltweit besten Kryptologen und Security-Ingenieure benötigten Jahre, um IPsec zu entwickeln. Und das nicht nur, weil Komitees immer langsam arbeiten.

Wann immer die IPsec-Väter dachten, sie hätten ihr Design im Trockenen, entdeckte jemand eine neuen Attacke auf einen Aspekt des Protokolls. Manchmal mussten sie völlig neue Sicherheitskriterien und Analysetechniken entwickeln, um die Sicherheit der IPsec-Bestandteile zu beurteilen. Sie entdeckten und lösten dabei Probleme, von denen vorher niemand auch nur geträumt hatte.

Die Standards für die Schlüsselaustausch-, Schlüsselvereinbarungs- und Authentifizierungs-Protokolle haben sich in den letzten Jahren deutlich weiterentwickelt, das ist eine der Errungenschaften der Arbeit an Protokollen wie IPsec, SSL und SSH. Dabei hat sich herausgestellt, dass authentifiziertes Schlüssel-Agreement überraschend kompliziert ist: Es mutet zwar einfach an, ist aber richtig schwer, wenn man es richtig machen will. Sogar die grundlegenden Sicherheitsmodelle, anhand derer ein Protokoll zu bewerten ist, sind noch Gegenstand intensiver Forschung.

Protokolle wie IPsec (ohne IKE), SSL und SSH sind nicht sinnlos komplex, wie mancher VPN-Neuling denken mag. Im Gegenteil, noch einfacher geht es kaum, wenn das Protokoll sicher sein soll.

Sichere VPN-Lösungen

Wer eine sichere VPN-Software sucht, sollte sich an erster Stelle den IPsec-Lösungen widmen. Unter Linux kommt hier vor allem Freeswan in Betracht **[13]**, [http://www.freeswan.org/]. Leider ist diese Applikation für viele Anwender recht schwer zu verwenden, offenbar noch schwerer als die meisten anderen IPsec-Implementierungen. Genau diese Schwierigkeiten haben zu vielen VPN-Lösungen jenseits von IPsec geführt. Die Situation wird sich in Zukunft bessern, da neuere Linux-Kernel eine Kame-basierte IPsec-Implementierung mitbringen **[14]**.

Von allen dem Autor bekannten IPsec-freien VPN-Applikationen hatten nur OpenVPN (**[15]**, [http://openvpn.sourceforge.net/]) sowie Yavipin [http://yavipin.sourceforge.net/] keine offensichtlichen Schwächen. Interessant wäre, wie die Fehlerrate bei Closed-Source-Software ausfällt.

OpenVPN nutzt ein SSL-basiertes Protokoll für seinen Kontrollkanal (vergleichbar zu IKE in IPsec); der Datenkanal ist ähnlich aufgebaut wie ESP in IPsec. Yavipin ist zwar komplett neu entworfen, es stammt aber von jemandem, der sein Handwerk beherrscht, und ist offenbar sauber implementiert. Beides sind reine Userspace-Applikationen und deutlich leichter aufzusetzen als IPsec.

nik führt ihr Leben dann nur noch in den Proceedings einer Krypto- oder Security-Konferenz.

Wie in einem biologischen System passen sich die großen Drei an und werden resistent, während schwächere Stämme wie SSHv1 oder SSLv1 aussterben. Noch während SSLv1 erstmals vorgestellt wurde, knackte jemand im Publikum dieses Protokoll. Daher hat kaum einer je von dieser Fehlgeburt gehört. Viele der Eigenschaften von SSLv1 erinnern übrigens an die weiter oben vorgestellten unsicheren VPN-Varianten. Weniger bekannte Protokolle können in ihrer Isolation durchaus jahrelang überleben, bis sie erstmals mit der Außenwelt in Kontakt kommen. Bereits der erste Angriff könnte sie dann aber völlig vernichten.

Standards sind gut

Das sind noch nicht alle Gründe, die für die Standardprotokolle sprechen. Die erwähnten Probleme mit der Implementierung von nicht dokumentierten, selbst gestrickten Entwürfen gibt es bei Standardprotokollen nicht. Hier ist unmissverständlich klar, welcher Fehler dem Protokoll und welcher der Implementierung zuzuschreiben ist. Fehler in der Implementierung eines Standardprotokolls fliegen meist sehr schnell auf, da diese Version dann inkompatibel mit allen anderen ist. Doch bei Entwürfen, die nicht zu dieser Gruppe gehören, ist das nicht so einfach.

Ein konkreter Fall: Bei der Untersuchung eines dieser VPN-Programme tauchte der Verdacht auf, dass es den ersten Sitzungsschlüssel ungeschützt übertragen könnte, denn zu diesem Zeitpunkt im Protokollablauf gibt es noch keinen Sitzungsschlüssel, mit dem der neue Key hätte verschlüsselt sein können. Aber auch eine knappe Stunde Wühlen in einem Berg von Spaghetti-Code brachte die genauen Protokolldetails nicht ans Tageslicht. Bei einem Standardprotokoll wäre das nicht passiert. Zum einen würde es nie einen Verschlüsselungsschlüssel übertragen. Vor allem aber wäre jede Implementierung, die es dennoch versucht, inkompatibel zu allen anderen, der Fehler würde also sehr schnell auffliegen.

Der Kasten „Sichere VPN-Lösungen" gibt einen Leitfaden, welche Produkte aus Sicht des Autors zu empfehlen sind. Im Kasten „Tipps für VPN-Entwickler" finden Unerschrockene noch einige Ratschläge, wie sie ihre eigene VPN-Software sicher implementieren. *(fjl)* ■

Tipps für VPN-Entwickler

Wer tatsächlich seine eigene VPN-Software entwickeln oder VPN-ähnliche Funktionen zu einer vorhanden Applikation hinzufügen muss, sollte sich beim Kontrollkanal auf die Expertise der SSL- und SSH-Entwickler stützen und für den Datenkanal bei IPsec spicken (genau das macht OpenVPN, [15]). Für den SSL- und SSH-Teil ist genug freier Code verfügbar: OpenSSL [http://www.openssl.org], OpenSSH ([12], [http://www.openssh.org]) oder Cryptlib [http://www.cs.auckland.ac.nz/~pgut001/cryptlib/]. Letztere stammt vom Autor dieses Beitrags; die Bibliothek implementiert unter anderem SSL, SSH und ein Menge mehr.

Für den IPsec-Anteil bieten sich das Kame-Framework [14] der neueren Kernel oder »iPsec_tunnel« an [http://ringstrom.mine.nu/ipsec_tunnel/]. Letzteres ist eine ESP-Implementierung mit nur 38 KByte Sourcecode. Allerdings scheint ihr noch einiges zu fehlen, etwa der Schiebefenster-Algorithmus zum Schutz vor Replay-Angriffen.

Auf der VPN-Theory-Mailingliste versuchen derzeit Teilnehmer, Standards für IPsec-freie VPNs zu definieren. Wer will, kann die Diskussion über Gmane verfolgen [http://news.gmane.org/gmane.network.vpn.theory].

Infos

[1] Performance of Freeswan: [http://www.freeswan.org/freeswan_trees/freeswan-2.02/doc/performance.html]

[2] Bodo Möller, „TLS insecurity (attack on CBC)", Beitrag auf der IETF-TLS-Mailingliste im September 2001. Auch zitiert in Bodo Möller, „Security of CBC Ciphersuites in SSL/TLS": [http://www.openssl.org/~bodo/tls-cbc.txt]

[3] Wei Dai, „An attack against SSH2 protocol": Posting in der Newsgroup »sci.crypt« im Februar 2002

[4] Phil Rogaway, „Problems with Proposed IP Cryptography", April 1995: [http://www.cs.ucdavis.edu/~rogaway/papers/draft-rogaway-ipsec-comments-00.txt]

[5] Ariel Futoransky, Emiliano Kargieman und Ariel Pacetti, „An attack on CRC-32 integrity checks of encrypted channels using CBC and CFB modes": CORE SDI S.A, 1998

[6] Steven M. Bellovin, „Problem Areas for the IP Security Protocols": Proceedings of the 1996 Usenix Security Symposium, August 1996, S. 205

[7] Stephen Kent, Randall Atkinson, „RFC 2406: IP Encapsulating Security Payload (ESP)", Nov. 1998

[8] Nikita Borisov, Ian Goldberg und David Wagner; „Intercepting Mobile Communications: The Insecurity of 802.11": Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, 2001, S. 180

[9] Bruce Schneier, Mudge, „Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol (PPTP)": Proceedings of the 5th ACM Conference on Communications and Computer Security, Nov. 1998, S. 132; [http://www.schneier.com/paper-pptp.html]

[10] Hugo Krawczyk, „The order of encryption and authentication for protecting communications (Or: how secure is SSL?)": Proceedings of Crypto '01, Lecture Notes in Computer Science No. 2139, Springer Verlag, August 2001, S. 310

[11] Achim Leitner, „Transport-Sicherung – verschlüsselt und authentifiziert kommunizieren mit TLS": Linux-Magazin 04/02, S. 50

[12] Karl-Heinz Haag, Achim Leitner, Artikelreihe zu OpenSSH. Linux-Magazin 05/02, S. 56; 07/02, S. 70 und 09/02, S. 72

[13] Ralf Spenneberg, „Standard-Tunnel – VPN Linux 2.4 und Freeswan 2.01": Linux-Magazin 10/03, S. 24

[14] Ralf Spenneberg, „Sicherer Transport – VPN mit Linux 2.6 und IPsec": Linux-Magazin 05/03, S. 60

[15] Achim Leitner, „Offener Tunnel – VPN ohne Kernel-Modifikation mit OpenVPN": Linux-Magazin 10/03, S. 29

Der Autor

Peter Gutmann forscht am Department of Computer Science der University of Auckland (Neuseeland) über Design und Implementierung von kryptographischen Sicherheitsarchitekturen. Er half beim Entwickeln von PGP und schrieb eine Reihe von Veröffentlichungen und RFCs zum Thema Sicherheit und Verschlüsselung (unter anderem den „X.509 Style Guide for Certificates"). Außerdem ist er Autor von Cryptlib (eines Open-Source-Security-Toolkits) und des Buches „Cryptographic Security Architecture Design and Verification" (Springer Verlag, 2003). In seiner Freizeit stochert er nach Löchern in jedem Sicherheitssystem und jedem Mechanismus, der ihm auffällt, und schimpft über PKIs.