Rechnernetze I
Übungsblatt 6

Anne Martens, Felix J. Oppermann

19. Juni 2006

1 TCP - Sicherheit

a) Zunächst wird dem Paket zum Zweck der Prüfsummenberechnung ein Pseudoheader vorangestellt der einige Informationen aus dem IP-Protokoll enthält. Die enthaltenen Daten sind: Internetadressen von Sender und Empfänger, TCP-Protokollnummer und der Länge des TCP-Datagramms.

Da erweiterte Paket wird im folgenden in 16-Bit Blöcke unterteilt die jeweils als Zahlen im 1-Komplement interpretiert werden. Diese Zahlen werden addiert und das Ergebnis in das Prüfsummenfeld geschrieben. Der Empfänger führt die gleiche Berechnung durch und überprüft ob das Ergebnis übereinstimmt. Das Verfahren kann nur Ein-Bit-Fehler sicher erkennen.

Einige Zwei-Bit-Fehler können auf diese Weise nicht erkannt werden. Da es in der 1-Komplement Addition keinen Übertrag gibt, beeinflussen Fehler immer nur eine Stelle des Ergebnisses. Ein zweiter Fehler in einem anderen Summanden an derselben Stelle führt dazu, dass die Stelle des Ergebnisses wieder korrekt ist. Sobald also eine gerade Anzahl von Bitfeldern an einer Stelle in den Summanden auftreten, kann dieser Fehler nicht erkannt werden.

Ein Beispiel aus TCP wäre der Fall, in dem das 18. Bit in der Quittungsnummer des Headers sowie das zweite Bit des Vorrang-Feldes verfälscht werden würde. Dieser Fehler würde nicht erkannt werden. Ebenso würde die Verfälschung des ersten Bits des Quellports sowie des ersten Bits des Optionen-Feldes nicht erkannt.

b) Arten von Fehlern

- Verlorenes Paket: In diesem Fall erhält der Sender keine Bestätigung für das gesendete (und verlorene) Paket. Nach Ablauf eines Timers wird das Paket dann nochmals gesenedt.
- Verfälschtes Paket: Ein verfälschtes Paket kann vom Empfänger anhand der Prüfsumme erkannt werden. Ist dies der Fall, wird das Paket nicht bestätigt, so dass der Sender das Paket nach Ablauf seines Timers erneut sendet. Manche Fehler (s.o.) können nicht erkannt werden, in diesem Fall wird das Paket an obere Schichten weitergeleitet
- Verlorene Bestätigung: In diesem Fall erreicht die Bestätigung eines Pakets den Sender nicht, so dass bei diesem der Timer abläuft und er das Paket nochmal sendet. Der Empfänger erkennt das Paket als bereist erhalten, sendet erneut eine Bestätigung und verwirft das Paket selbst.
- Dupliziertes Paket: Dies kann durch Fehler der unteren Ebenen auftreten. In diesem Fall wird wie oben die Bestätigung ein weiteres Mal gesendet und das Paket verworfen. Der Sender verwirft dann auch die duplizierte Bestätigung.
- Falsche Reihenfolge: Die Pakete werden von TCP gepuffert und in der richtigen Reihenfolge an die oberen Protokollebenen weitergegeben.

c) i. Auf Grund der Umlaufhöhe von Satelliten ist bei der Satellitenverbindung mit einer Round-Trip-Zeit von ca. 0.5 sek zu Rechnen (siehe auch Aufgabe 1c, Übungsblatt 3). Die Round-Trip-Zeit der Crossconnect-Verbindung wird nur geringfügig durch die Signallaufzeit in der Leitung beeinflusst. Hier überwiegt der Einfluss der von der Hard- und Software zum Antworten benötigten Zeit. Es ist mit einer Round-Trip-Zeit im Bereich weniger Millisekunden zu Rechnen. Leider steht uns kein Satellit zum Überprüfen der Round-Trip-Zeit einer Satellietenverbindung zur Verfügung. Die Messung der Round-Trip-Zeit ein Crossconnect-Verbindung musste in Ermangelung eines verfügbaren zweiten Rechners zwischen Rechner und Router erfolgen. Es ergab sich eine Durchschnittliche Round-Trip-Zeit von ca. 0.03 ms.

ii. Da TCP im bei Fehlerhaften Paketen keine aktive Benachrichtigung kennt wird bei Fehler mit dem Versenden des Ersatz-Pakets immer für die Zeit Timeout gewartet. Ist diese Zeit zu groß gewählt, so treten unnötige Wartezeiten auf, die die Verbindung unnötig verlangsammen. Ist die Zeit zu klein gewählt, so werden Pakete unnötig mehrfach gesendet.

d) Retransmission Timeout. Die Abkürzungen im RFC sind wie folgt gewählt:

- Round Trip Time RTT
- Smoothed Round Trip Time SRTT
- Retransmission timeout RTO
- Smoothing factor ALPHA (z.B. 0,8 bis 0,9)
- Delay variance BETA (z.B. 1,3 bis 2,0)
- Upper bound on the timeout UBOUND (z.B. 1 minute)

- Lower bound on the timeout LBOUND (z.B. 1 second)

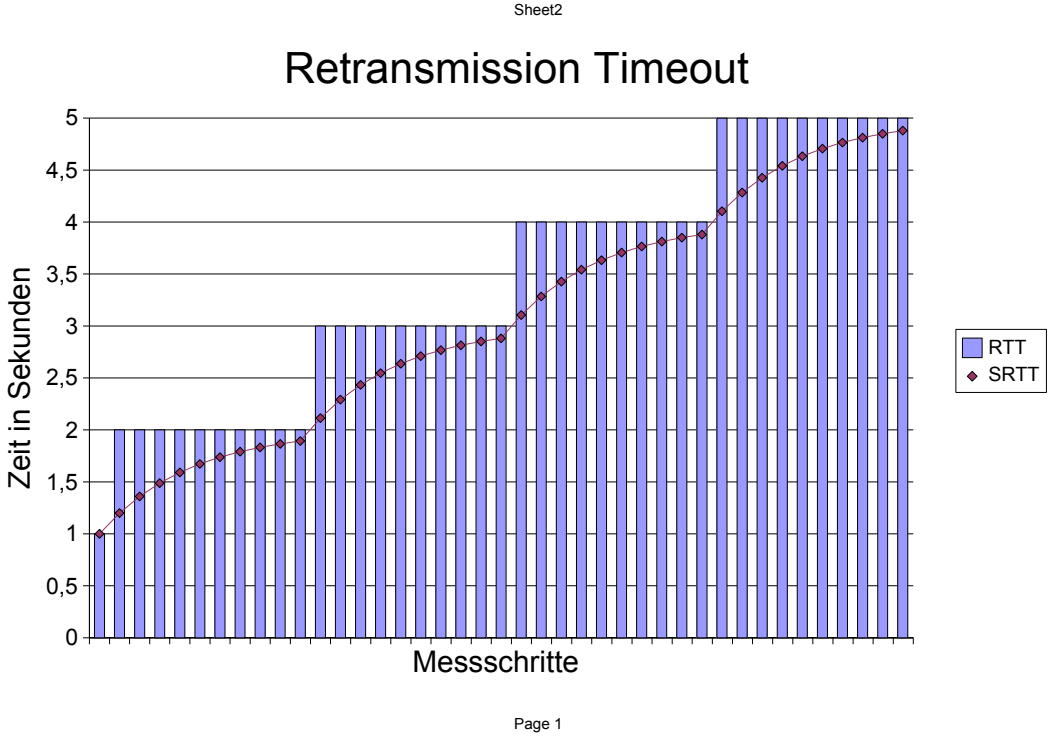
Zunächst wird die SRTT aus der vorhergehenden SRTT und der aktuellen RTT berechnet:

$$SRTT = (ALPHA * SRTT) + ((1 - ALPHA) * RTT)$$

Darauf basierend das RTO wie folgt:

$$RTO = min[UBOUND, max[LBOUND, (BETA * SRTT)]]$$

Das RTO hängt also von den bisherigen RTT, den Konstanten ALPHA und BETA sowie den Grenzen UBOUND und LBOUND ab. Für das folgende Diagramm haben wir ALPHA = 0,8 und BETA = 1,3 sowie UBOUND = 60 s und LBOUND = 1 s gewählt. Der RTT wurde von 1 s stufenweise erhöht und blieb jeweils 10 Schritte lang konstant.



e) Probleme
Ein Rechner B reserviert einen Bereich im Speicher, in dem Daten zur halboffenen Verbindung abgelegt werden, nachdem er die Anforderung zu einer Verbindung von Rechner A erhalten hat, und sendet eine Bestätigung. Wenn der anfragende Rechner A nun nicht mehr reagiert, verbleiben die Informationen im Speicher des Rechners B und werden erst nach Ablauf eines Timers gelöscht. Wenn zuviele solcher halboffenen Verbindungen während der Laufzeit des Timers aufgebaut werden, ist der Speicherplatz für TCP Verbindungen des Rechners B voll und B kann keine weiteren TCP Verbindungen annehmen. Auf diese Weise können Attacken gegen den Rechner B ausgeführt werden, indem viele solcher halboffenen Verbindungen erzeugt werden.

Lösungsmöglichkeiten
Die beste Lösung ist es, während des Verbindungsaufbaus keine Daten auf dem Rechner B zu speichern. Dies kann dadurch gelingen, dass alle benötigten Daten zum Verbindungsaufbau zurück an den Rechner A geschickt werden, der diese in einem zweiten Schritt wieder zurückschickt und sie bestätigt bekommt. Der Verbindungsaufbau würde damit auf ein 4-Wege-Handshake ausgweitet werden. Dieser Ansatz wird in SCTP verwendet.

Unter der Annahme, dass das TCP Protokoll weiter verwendet und nicht verändert werden soll, gibt es allerdings kaum Möglichkeiten. Eine vorgeschaltete Firewall (z.B. in einem Router) könnte erkennen, dass

von einem Rechner im Netz viele Anfragen für neue Verbindungen erzeugt werden, und dieser Rechner könnte gesperrt werden und seine Anfragen nicht weitergeleitet. Allerdings können so nur Attacken die von einem oder wenigen Rechnern ausgeführt werden, entdeckt werden, wenn viele Rechner an einer Attacke beteiligt sind, kann die Attacke nicht erkannt werden.

f) Das Protokoll TCP stellt ausschließlich einen nicht formatierten Byte-Strom zur Verfügung. Eine explizite Strukturierung diese Datenstroms ist durch das Protokoll nicht vorgesehen und bleibt der Anwendung überlassen. Diese kann jedoch problemlos Strukturierte Daten übertragen, da die Ordnung des Byte-Stroms garantiert wird. Einzig die Umwandlung zwischen den unstrukturierten und strukturierten Daten bleibt der Anwendung selbst überlassen.

2 SCTP

a) Das Stream Control Transmission Protocol (SCTP) vereint Eigenschaften der Protokolle TCP und UDP. Es ist wie TCP vorrangig stromorientiert. SCTP stellt dabei im Gegensatz zu TCP mehr potentiel mehr als einen Datenstrom pro Übertragungsrichtung bereit. Zusätzlich zu den definierten Datenströmen können jedoch auch unabhängig einzellne Packete verschickt werden. Hier verhält sich das Protokoll Paketorientiert wie UDP. Anders als TCP kann bei SCTP über Heartbeats explizit kotrolliert werden ob die Verbindung abgerissen ist. Als weiterer Unterschied zu TCP verwendet SCTP einen vier Wege Handshake. Dieser verhindert SYC-Flood-Angriffe, da keinen Statusverbindungen über unvollständige Verbindungsversuche gehalten werden müssen. Die Algorithmen zur Flusskontrolle in SCTP entsprechen weitgehend jenen von TCP.

b) Anders als TCP verwendet SCTP einen vier Wege Handshake um eine Angriffsmöglichkeit per SYC-Flooding zu verhindern. Der Verbindungsaufbau erfolgt dabei in den folgenden Schritten:

- Zunächst sendet der Client an den auf Verbindungsanfragen wartenden Server ein INIT-Paket.
- Der Server beantwortet diese Paket mit einem INIT-ACK-Paket, welches auch die Statusinformationen der Verbindung enthält. Der Server muss keine Informationen zwischenspeichern.
- Das INIT-ACK-Paket beantwortet der Client mit einem COOKIE-ECHO-Paket, welches erneut die vom Server erhaltenen Statusinformationen enthält.
- Als letzter Schritt beantwortet der Server das COOKIE-ECHO-Paket mit einem COOKIE-ACK-Paket. Sobald der Client diese Paket empfangen hat ist die Verbindung vollständig aufgebaut.

3 Namensauflösung im Internet

a) **Uniform Resource Identifier (URI)** URI sind eine eindeutige Zeichenfolge die eine bestimmte Ressource repräsentiert. Eine URI besteht jeweils aus einem URI-Schema und einem Schema-Abhängigen String die durch einen Doppelpunkt getrennt sind. Beispiel: `file:/home/felix/documents/uni/2006SS/compilerbau/uebung/uebung04/uebung4.tex`

Uniform Resource Locator (URL) URL sind eine Unterart von URI. Mit URL wird eine Ressource über das zum Zugriff verwendete Protokoll identifiziert. Beispiel: `http://felix-oppermann.de/index.html`

Uniform Resource Name (URN) Bei URN handelt es sich um URI mit dem Schema ,,urn". In diesem Schema sind ebenfalls durch einen Doppelpunkt von Ressourcen String weiter Namesräume definiert. Jedes Namensraum kann dabei ein eigenes Identifiaktionsverfahren für seine Ressourcen verwenden. Es können dabei sowohl bestehende Indentifikationsbezeichner wie ISBN-Nummer oder völlig neue Bezeichner verwendet werden. Die Identifikation durch URN ist Ortsunabhängig. Beispiel: `urn:ISBN:3-8272-0062-7`

b) Die durch Punkte getrennten Elemente im Domain-Namen beschreiben von links nach rechts umfassendere Einheiten. Weiter links stehender Name im Domain-Namen ist also spezifischer als weiter rechts stehtender Name. Die Namen bilden eine Hierachie.

Im Domain-Name `www.felix-oppermann.de` bezeichent ,,www" einen bestimmten Rechner der sich in der Domain ,,felix-oppermann" befindet. Diese Domain ist wiederum Teil der Länderdomain Deutschlands.

c) Domain-Namen sind für Menschen einfacher zu verstehen und merken als IP-Nummern. DNS-Namen dienen also dem leichteren Zugriff auf Ressourcen im Internet. Darüber ermöglich das DNS-System einen Austausch von IP-Adressen bei von System, da der DNS-Name trotz einer veränderten IP-Adresse beibehalten werden kann.

Lokale Namensauflösung Bei der lokalen Namensauflösung ist dem Rechner selbst die Zuordnung des DNS-Namen zur IP-Adresse bekannt. Die Auflösung erfolgt über einen Eintrag in der Hosts-Datei. In der Regel ist dies zum Beispiel für den Namen ,,localhost", der den eigenen Rechner bezeichnet, der Fall. In der Hosts-Datei findet sich ein Eintrag, welcher dem Namen ,,localhost" die Adresse 127.0.0.1 zuordnet.

Rekursive Namensauflösung Beim der rekursiven Namensauflösung versucht der angefragte Nameserver den Request selbst aufzulösen. Als Beispiel stelle ein Client die Anfrage nach der Adresse ,,www.anne-martens.de". Diese Anfrage wird zunächst durch den Nameserver des Client an den für ,,de" zuständigen Nameserver weitergeleitet. Dieser kennt den für die Domain ,,anne-martens" zuständigen Nameserver. Bei der rekursiven Namensauflösung kontaktiert der Nameserver nun selbstständig diesen zweiten Nameserver um die IP-Adresse für den Rechner ,,www" der Domain zu ermitteln. Dieser Antwortet mit der IP-Adresse, welche jetzt auf dem umgekehrten Weg zurück übermittelt wird. Als letztes sendet der Nameserver des Clients die gesuchte IP-Adresse an den anfragenden Rechner.

Iterative Namensauflösung Beim der iterativen Namensauflösung antwortet der Nameserver nur mit dem nächsten zuständigen Nameserver. Die schrittweise Namensauflösung bleibt dem Nameserver des Clients überlassen. Als Beispiel stelle ein Client erneut die Anfrage nach der Adresse ,,www.anne-martens.de". Diese Anfrage wird zunächst vom Nameserver des Clients an den für ,,de" zuständigen Nameserver weitergeleitet. Dieser kennt den für die Domain ,,anne-martens" zuständigen Nameserver und übermittelt dessen Adresse an den anfragenden Nameserver. Dieser stellt nun an diesen Rechner erneut eine Anfrage nach der Namensauflösung für den Rechner ,,www", welche der für ,,anne-martens.de" zuständige Nameserver mit der IP-Adresse des Rechners beantwortet. Diese IP-Adresse kann dann an den Client zurückgegeben werden.

d) Die ersten vier Einträgen orden jeweils einen Domain-Name einer IP-Adresse zu. Da die Namen nicht voll qualifiziert sind wird jeweils die Domain des eigenen Rechners angehängt. Durch das Zeichen wird markiert, dass dieser Eintrag die Domain ohne einen vorangestellten Rechnernamen meint. Die nächsten vier Eintrage spezifizieren Alias-Namen. Der Name in der ersten Spalte wir jeweils als Alternative Name für den Namen in der letzten Spalte definiert. Bei den nicht voll qualifizierten Namen wird auch hier immer die eigenen Domain angehängt.

4 TFTP

a) **Übertragen von Dateien** Das tftp-Protokoll eignet sich nur zum Übertragen von Dateien. Diese können sowohl gelesen oder geschrieben werden. Ein Auflisten von Verzeichnissen ist nicht möglich.

Paketorientiert Das Protokoll ist paketorientiert. Es setzt normalerweise auf UDP auf.

Bestätigung jedes Pakets Jedes Paket wird einzeln bestätigt. Ein senden des nächsten Pakets erfolgt erst nach Empfang der Bestätigung. Dies macht eine Flusskontrolle unnötig.

Keine Authentifizierung Im Gegensatz zu ftp stellt tftp keine Authentifizierungsmechanismen zur Verfügung.

Keine Kompression Kompression und Verschlüsselung werden durch das Protokoll nicht unterstützt.

b) Korrekte Übertragung einer Datei mit einer Länge von 1200 Oktetten
Wir nehmen an, dass die Datei gelesen werden soll. Zunächst wird die Verbindung aufgebaut, indem eine Anforderung zum Lesen der Dateien gesendet wird. Die Datei wird in Blöcken von 512 Oktetten übertragen. Zunächst werden zwei Blöcke der Größe 512 Byte übertragen und jeweils mit einer Quittung mit der gleichen Blocknummer vom Empfänger quittiert. Erst nachdem der Sender die Quittung des ersten Paket erhalten hat, sendet er das zweite Paket mit der nächsten Blocknummer. Nach dem Empfang der zweiten Quittung wird das letzte Paket mit 176 Byte Größe gesendet und vom Empfänger quittiert. Nachdem der Empfänger ein wenig wartet, kann die Verbindung abgebaut werden.

c) Auch hier wird zunächst die Verbindung aufgebaut. Der erste Block wird gesendet und vom Empfänger quittiert. Der zweite Block wird gesendet, trifft aber nicht beim Empfänger an. Nachdem ein Timer beim Sender abgelaufen ist, ohne dass dieser eine Bestätigung erhalten hat, wird der Block erneut gesendet, erreicht dieses Mal auch den Empfänger und wird quittiert. Nach dem Empfang der zweiten Quittung wird das letzte Paket mit 176 Byte Größe gesendet und vom Empfänger quittiert. Nachdem der Empfänger ein wenig wartet, kann die Verbindung abgebaut werden.

d) Der Ablauf ist sehr ähnlich wie (c). Auch hier wird zunächst die Verbindung aufgebaut. Der erste Block wird gesendet und vom Empfänger quittiert. Der zweite Block wird gesendet, trifft beim Empfänger an, der eine Quitting zurücksendet. Da die Quittung beim Sender nicht ankommt, läuft dort der Timer ab,

und der Block wird erneut gesendet. Der Empfänger antwortet auf den Erhalt des duplizierten Pakets mit einer erneuten Quittung, verwirft aber das Paket selbst. Nach dem Empfang der zweiten Quittung wird das letzte Paket mit 176 Byte Größe gesendet und vom Empfänger quittiert. Nachdem der Empfänger ein wenig wartet, kann die Verbindung abgebaut werden.

e) Bei der Übertragung einer Datei der Länge 1536 Bytes werden 4 Pakete gesendet. Dabei enthält das letzte Paket keine Daten. Der Grund ist, dass eine TFTP Verbindung erst durch das Senden eines Pakets, das weniger als 512 Bytes enthält, beendet wird. Da die Größe der Datei glatt durch 512 teilbar ist, sind die Datenteile ersten drei Pakete 512 Byte lang. Es muss also zum Verbindungsabbau noch ein Paket einer geringeren Länge geschickt werden, da keine Daten mehr vorhanden sind, hat dieses die Länge 0 Byte (plus Header).