



Falko Dressler Tobias Limmer

Netzwerksicherheit WS 2007/08

4. Übungsblatt

27.11.2007

Zufallszahlen spielen in der Kryptographie eine grosse Bedeutung – nahezu alle kryptographischen Algorithmen und Protokolle basieren darauf, dass bestimmte Eingabedaten des Algorithmus nicht vorherbestimmbar sind. Ist diese Eigenschaft nicht gegeben, werden prinzipiell sichere Algorithmen angreifbar. In dieser Übung wird Ihnen die Beschreibung eines Authentifizierungsprotokolls vorgegeben. Allerdings wird in der verwendeten Implementierung kein sicherer Zufallszahlengenerator verwendet. Ihre Aufgabe ist nun, das Protokoll zu analysieren und die durch die unsicheren „Zufallszahlen“ entstandene Sicherheitslücke auszunutzen.

Beschreibung des Protokolls: Das Protokoll wird verwendet, um sich auf einen Server (nachfolgend S genannt) verbindende Clients (C) zu authentifizieren. Nach erfolgter Authentifizierung werden Nutzdaten von S zu C gesendet. Die Authentifizierung läuft wie folgt ab:

Schritt	Client (C)	Server (S)
1.		\xrightarrow{REQ}
2.		Berechne $ID_i = ID_{i-1} \times 107$
3.		Berechne $R = T_i \oplus K_{SC} \oplus ID_i$
4.		\xleftarrow{R}
5.	Berechne $A = T_i \oplus K_{SC}$	
6.		\xrightarrow{A}
7.		Überprüfe $A == T_i \oplus K_{SC}$
8.		\xleftarrow{DATA}

Anmerkungen zum Protokoll:

- \oplus stellt den XOR-Operator dar.
- i wird bei jeder neuen Anfrage um 1 inkrementiert.
- K_{SC} ist ein nur C und S bekannter geheimer Schlüssel.
- T_i besteht aus zwei Bytes: das höherwertige Byte stellt die aktuelle Tageszeit (UTC) in Stunden dar, das Niederwertige die Anzahl der Minuten der aktuellen Stunde.
- T_i , R , A , K_{SC} und ID_i sind 2 Bytes lang. Eventuelle Überträge bei Berechnungen werden verworfen.
- REQ wird als nullterminierter String übertragen und muss den Text **REQ** enthalten. $DATA$ ist ein nullterminierter String, dessen Länge nicht 1023 Zeichen überschreitet.
- Alle Werte werden in der Little-Endian Bytereihenfolge gespeichert. Falls Sie auf i386 Rechnern arbeiten, müssen Sie sich also nicht um Bytekonvertierung kümmern.

Aufgabenstellung:

1. Beschreiben Sie, wie das dargestellte Authentifizierungsprotokoll angegriffen werden kann, sodass der Server den Angreifer fälschlicherweise als berechtigten Client erkennt, welcher den geheimen Schlüssel K_{SC} besitzt.
2. Programmieren Sie einen Client, welcher die in der vorigen Aufgabe beschriebene Methode implementiert. Es wird ein Server auf dem Rechner `faii7d2.informatik.uni-erlangen.de`, Port 20000 zur Verfügung gestellt. Dieser realisiert das beschriebene Protokoll, lässt aber nur eine Verbindung pro Sekunde zu und beendet laufende Authentifikationen nach maximal 2 Sekunden. Falls Ihr Angriff darauf basiert, mehrere Authentifikationen direkt nacheinander durchzuführen, müssen Sie beachten, dass sich andere Gruppen gleichzeitig auf den Server verbinden können. Versuchen Sie nun mit Hilfe Ihres Programms die Daten zu erlangen, welche nach erfolgreicher Authentifizierung übertragen werden.
3. Wie verändert sich die Sicherheit des Protokolls, wenn ID bei jeder neuen Authentifizierungsanfrage mit Hilfe eines kryptographisch sicheren Pseudozufallszahlengenerator gesetzt wird? Warum?

Technische Hinweise:

- Die Aufgaben sind in C/C++ oder Java zu erstellen.
- Eine Vorlage für den Aufbau eines TCP/IP Sockets unter Linux befindet sich auf der Webseite der Vorlesung.
- Der Code muss auf dem Betriebssystem Debian/Linux im CIP-Pool der Informatik mit der dort installierten Version des GNU C/C++-Compilers kompilieren und lauffähig sein.
- Es ist ein Makefile oder ein ausführbares Skript zu bereitzustellen, welches temporäre Daten löschen (Target „clean“) und eine Binärdatei (Target „all“) erstellen kann.
- Es ist eine Dokumentation im ASCII-Format zu erstellen, welche die Ausgaben während des Angriffs dokumentiert und über Besonderheiten des Programms berichtet.

Abgabe der Lösungen per Mail an `tobias.limmer@informatik.uni-erlangen.de` bis Montag, den 3.12.2007 als *Archiv* im Format `.tar.(gz|bz2)` oder `.zip`. Als Betreff der Mail `Netsec Übung 4, Gruppe X` angeben.