5. Übungsblatt

Übung Peer-to-Peer-Systeme und Sicherheit (SS2009)

3. Juli 2009
Abgabetermin: Fr. 17.7.
Übungstermin: Do. 24.7.

Dipl.-Inform. Heiko Niedermayer
Lehrstuhl für Netzarchitekturen und Netzdienste
Technische Universität München

Regeln:
Es sind insgesamt 5 Übungsblätter mit je 10 Punkten vorgesehen. Ein Blatt kann von 3 Studenten gemeinsam bearbeitet werden. Für die Übungscredits wird erwartet, dass Aufgaben für 30 Punkte sinnvoll bearbeitet wurden und dass 1x erfolgreich eine Aufgabe vorgerechnet worden ist.

Aufgabe 1 (2 Punkte) Peer-to-Peer-Sicherheit
In dieser Aufgabe sollen Sie durch Gegenbeispiel drei Aussagen widerlegen.

a) Authentisierung per Email-Adresse verhindert sicher einen Sybilangriff.
b) Bei Kademlia ist kein Eclipse-Angriff möglich.
c) Daten in einem Peer-to-Peer-Netzwerk können nicht gelöscht werden.

Aufgabe 2 (2 Punkte) Anonymität und Sicherheit
In dieser Aufgabe gibt es wieder Verständnisfragen, die mit dem Wissen aus der Vorlesung beantwortet werden können sollten.

a) Per HTTPS (HTTP über SSL) werden heute Webseiten geschützt. Wenn Sie per HTTPS einer statischen Webseite ansurfen, in welchen Fällen kann ein beobachtender Angreifer auf die Webseite und deren Inhalte schließen?
b) Bei der Verwendung eines anonymen wie auch zensurresistenten Systems kann es passieren, dass Sie unliebsame oder auch kriminelle Daten speichern und bedienen. Mit welchen Methoden versuchen die Systeme Ihnen zu helfen, sich als unschuldig erklären zu können?

Aufgabe 3 (2 Punkte) Mixe
Mixe verwenden die Verwirbelung der Paketreihenfolge, um Beobachtern die Verfolgung von Paketen zu erschweren. Natürlich sind diese auch neu verschlüsselt („Onion Routing“), so dass der Paketinhalt nicht alles wieder verrät.
Ein *Schwellwert-Mix* sammelt dabei eingehende Pakete und sendet diese erst in einer zufälligen Reihenfolge heraus, wenn der Schwellwert S an Paketen vorhanden ist. Dieses Aussende-Ereignis nennen wir „Feuern“

a) Angenommen, pro Millisekunde komme mit p=25 % ein Paket an. S sei 10. Wie lange dauert es nun von einem Feuern des Mixes zum nächsten? Geben Sie die Verteilungsfunktion an.
b) Wie groß ist die Anonymitätsmenge (Anonymity Set)?

Kommen wir nun zum *Timed-Mix*. Ein Timed-Mix wartet immer eine Zeit T, um dann alle bis dahin erhaltenen Pakete in zufälliger Reihenfolge zu verschicken.

c) Wann deckt ein Timed Mix „aus Versehen“ ein Paket auf?

Aufgabe 4 (2 Punkte) Mixe
Ein *Pool-Mix* ist ein Mix mit einem Gedächtnis. Diese Gedächtnis wird Pool genannt und hat eine Größe von P Nachrichten. Die Funktionsweise ist nun wiefolgt. Der Mix wartet bis er S+P Pakete zwischengespeichert hat. In diesem Moment wählt er davon S Pakete zufällig aus und sendet diese in zufälliger Reihenfolge. Die restlichen P Pakete verbleiben im Mix.

a) Wie lange kann eine Nachricht in so einem Mix verweilen?
b) Wie wirkt sich das auf Anonymitätsmenge aus?
c) Wie könnte ein starker Angreifer versuchen, einen Pool-Mix anzugreifen, um dennoch Pakete verfolgen zu können?

--- bitte wenden ---

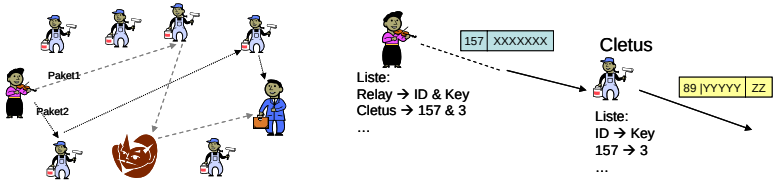
Aufgabe 5 (2 Punkte) Angriffe durch Abzählen

Die reine Verwendung von Mixen oder einfachen Anonymisierungsnetzen hat ihre Schranken. Wir wollen dies hier durch Angriffe diskutieren.

a)

Angenommen, alle hier abgebildeten Personen kommunizieren über den Mix. Stellen wir uns den als einfachen Schwellwertmix vor. Sie sind hier aber „Angreifer“ und überwachen das Netzwerk und loggen alles brav mit.
Jemand hat ein geheimes Dokument an Alice versandt, eine Datei der Größe 111,87 MB! Wie können Sie den Delinquenten (Bob) trotz des Mixes identifizieren?

b)



Anstelle des Mixes sollen nun alle Personen ein Peer-to-Peer-Netzwerk bilden. Dabei seien auch Sie als *Angreifer mit einem Knoten dabei, der jetzt aber nur Verkehr über sich selber beobachten kann*. Die Idee dieses Peer-to-Peer-Netzwerkes sei das Verschleiern des Verkehrs durch Onion-Routing über zufällige Pfade pro Paket. Um effizient Verschlüsselung per Onion-Routing betreiben zu können, muss jeder Sender mit jedem anderen Knoten einen für seine Pakete spezifischen gemeinsamen Schlüssel haben. Dies wurde auf einem Weg erreicht, so dass der Sender mit seiner echten Identität (z.B. IP:Port) jeweils dem Knoten nicht bekannt ist. Hierzu habe jeder Knoten für die Entschlüsselung eingehender Pakete eine individuelle Liste Zufalls-ID und Schlüssel. Der Sender habe entsprechend für seine eigenen Pakete eine Liste mit Relay-Knoten, Zufalls-ID und Schlüssel. Das rechte Bild zeigt dabei die Arbeitsweise eines Relays, das wir hier mal Cletus nennen. Cletus empfängt von irgendwoher ein Paket. Die Kennung sagt, es ist ID 157, was dem Key 3 entspricht. Mit diesem entschlüsselt er das Paket und erkennt, wohin es weitergeschickt werden muss. Durch die Entschlüsselungsoperation hat sich das Paket insgesamt verändert. Frei zu erkennen ist die ID 89, die der nächste Relay-Knoten braucht, um den Key zu bestimmen. Hier versteckt sich Alice also nicht hinter der gleichen ID. Am Ende musste Cletus noch etwas Padding „ZZ“ einfügen, damit das Paket gleichgroß bleibt und die Länge nicht den Abstand vom Sender oder zum Ziel verrät.

Aufgabe: Als Angreifer wollen wir natürlich wissen, wer hinter den jeweiligen Zufalls-IDs steckt. Nehmen Sie dazu beispielhaft ein Netz aus 8 Knoten an, wovon einer Alice, einer Bob und einer der Angreifer sind. Betrachten Sie nun rein den Verkehr von Alice zu Bob. Senden Sie 25 Nachrichten über je 2 möglichst zufällig (gleichverteilt) gewählte Hops (verwenden Sie ggf. Zufallsgenerator/Software). Zählen Sie dabei am Angreifer, wie oft er wen als Vorgänger und wie oft er wen als Nachfolger sieht. Lässt sich aus den Daten ein Angriff ableiten?