Client Puzzles
A Cryptographic Defense Against Connection Depletion Attacks

Most of slides come from Ari Juels and John Brainard
RSA Laboratories

The Problem

How to take down a restaurant

Saboteur
Restauranteur
Saboteur vs. Restauranteur

Table for four
at 8 o’clock.
Name of Mr. Smith.

O.K.,
Mr. Smith

Saboteur

Restauranteur

No More Tables!

Saboteur

An example: TCP SYN flooding

“TCP connection, please.”

“O.K. Please send ack.”

Buffer
TCP SYN flooding has been deployed in the real world
- Panix, mid-Sept. 1996
- Others

Similar attacks may be mounted against e-mail, SSL, etc.

Some defenses against connection depletion

Problem: Legitimate clients must keep retrying

Throw away requests
IP Tracing (or Syncookies)

Hi. My name is 10.100.16.126.

Problems:
- Can be evaded, particularly on, e.g., Ethernet
- Does not allow for proxies, anonymity

Digital signatures

Problems:
- Requires carefully regulated PKI
- Does not allow for anonymity

Connection timeout

Problem: Hard to achieve balance between security and latency demands
Our solution: client puzzles

Intuition

Table for four at 8 o'clock. Name of Mr. Smith. Please solve this puzzle.

O.K., Mr. Smith

A puzzle takes an hour to solve
There are 40 tables in restaurant
Reserve at most one day in advance

Intuition

Suppose:
- A puzzle takes an hour to solve
- There are 40 tables in restaurant
- Reserve at most one day in advance

A legitimate patron can easily reserve a table
Intuition

Would-be saboteur has too many puzzles to solve

The client puzzle protocol

Client

Service request $M$

Server

Buffer

O.K.

What does a puzzle look like?
Puzzle basis: partial hash inversion

Pair $(X', Y)$ is $k$-bit-hard puzzle

Puzzle basis: (Cont’d)
- Only way to solve puzzle $(X', Y)$ is brute force method. (hash function is not invertible)
- Expected number of steps (hash) to solve puzzle: $2^k / 2 = 2^{k-1}$

Puzzle construction

Client

Service request $M$

-secret $S$

Server
Puzzle construction

Server computes:
- secret $S$
- time $T$
- request $M$

Hash

Puzzle

Hash

Image $X$

Hash

Image $Y$

Sub-puzzle

- Construct a puzzle consists of $m$ k-bit-hard sub-puzzles.
- Increase the difficulty of guessing attacks.
- Expected number of steps to solve: $m \times 2^{k-1}$.

Why not use $k+\log m$ bit puzzles?

- $(k+\log m)$-bit puzzle
  - Expected number of trials $m \times 2^{k-1}$
- But for random guessing attacks, the successful probability
  - One $(k+\log m)$-bit puzzle
    - $2^{m \log m}$ (e.g., $2^{8 \log 8}$)
  - $m$ k-bit subpuzzles
    - $(2^m)^n = 2^{mn}$ (e.g., $2^{8m}$)
Puzzle properties
- Puzzles are stateless
- Puzzles are easy to verify
- Hardness of puzzles can be carefully controlled
- Puzzles use standard cryptographic primitives

Client puzzle protocol (normal)

\[ M_i^1 : \text{first message of } i\text{th execution of protocol } M \]

\[ \text{Client} \]
- \[ M_i^1, \text{"Puzzle"} \]
- \[ \text{"No puzzle"}, M_i \]

\[ \text{Server} \]
- \[ \text{Registers permission of } M_i \]

Client puzzle protocol (under attack)

\[ \text{Client} \]
- \[ M_i^1, \text{"Puzzle"} \]
- \[ \text{"Yes, puzzle": } P, t \]

\[ \text{Server} \]
- \[ \text{Validates that C, T, } \]
- \[ \text{Compared: } (t + j, \text{timestamp of puzzle}) \]
- \[ \text{Verifies that } \text{solution} \text{ is correct} \]
- \[ \text{Registers permission of } M_i \text{ to } P_j \]

P - puzzle with m sub-puzzles
j - timestamp of puzzle
T - time to receive solution
P - end time of puzzle
Where to use client puzzles?

Some pros
Avoids many flaws in other solutions, e.g.:

- Allows for anonymous connections
- Does not require PKI
- Does not require retries -- even under heavy attack

Practical application
- Can use client-puzzles without special-purpose software
  - Key idea: Applet carries puzzle + puzzle-solving code
- Where can we apply this?
  - SSL (Secure Sockets Layer)
  - Web-based password authentication
Conclusions

Puzzle and protocol description

Rigorous mathematical treatment of security using puzzles -- probabilistic/guessing attack

Contributions of paper

- Introduces idea of client puzzles for on-the-fly resource access control
- Puzzle and protocol description
- Rigorous mathematical treatment of security using puzzles -- probabilistic/guessing attack

Questions?