Denial-of-service resistant authentication

Tuomas Aura, Pekka Nikander
Helsinki University of Technology
`Tuomas.Aura@hut.fi`
`Pekka.Nikander@hut.fi`

Jussipekka Leiwo
Vrije Universiteit
`leiwo@cs.vu.nl`

Security Protocols Workshop 2000

Denial-of-service attacks

- **Open network**
  vs. multi-user OS, TCB, classical DOS models (Gligor&Yu83,Millen92)
- **Server resource exhaustion**:
  memory (SYN attack), processor, communications (ping flooding)
- Does **authentication** help?
- **Network DOS**:
  Cookies (SYN, Photuris, IKE)
  Stateless connections (Aura&Nikander97)
  DOS-resistant KAP (Hirose&Matsuura99)
  Gradually strengthening authentication (Meadows99)
  Increasing the cost of junk mail (Dwork&Naor92)
  Client puzzles (Juels&Brainard99)

Client puzzles

The server asks the client to solve a puzzle before it creates a protocol state or does expensive computations.

C		S
		S does not save state data or do expensive computations.
	← Puzzle	
C commits its resources into solving the puzzle.	Solution →	
		S verifies the solution. **S may now commit resources to expensive parts of the authentication.**

Properties of a good puzzle

1. Creating a puzzle and verifying the solution is inexpensive for the server.
2. The cost of solving the puzzle is easy to adjust from zero to impossible.
3. The puzzle can be solved on most types of client hardware (although it may take longer with slow hardware).
4. It is not possible to precompute solutions to the puzzles.
5. **While the client is solving the puzzle, the server does not need to store the solution or other client-specific data.**
6. **The same puzzle may be given to several clients. Knowing the solution of one or more clients does not help a new client in solving the puzzle.**
7. **A client can reuse a puzzle by creating several instances of it.**

The puzzle

The client solves X (and Y) from the following equation by brute force:

$$h(C, N_S, i_C, X) = \overbrace{000\ldots000}^{\text{the } k \text{ first bits of the hash}} \underbrace{Y}_{\text{the rest of the hash bits}}$$

h	=	a cryptographic hash function (MD4, MD5 or SHA)
C	=	the client indentity
N_S	=	the server's nonce, sent to many clients, changed periodically
i_C	=	the client's counter
X	=	the solution of the puzzle
$000\ldots000$	=	the k first bits of the hash value; must be zero
Y	=	the rest of the hash value; may be anything

Conclusion

- **At any point in the protocol,**
  the cost of to the client is greater than the cost to the server.
  $\Longrightarrow$
  The cost of an attack is greater than the damage caused by it.
- Drawback: extra work for the client.