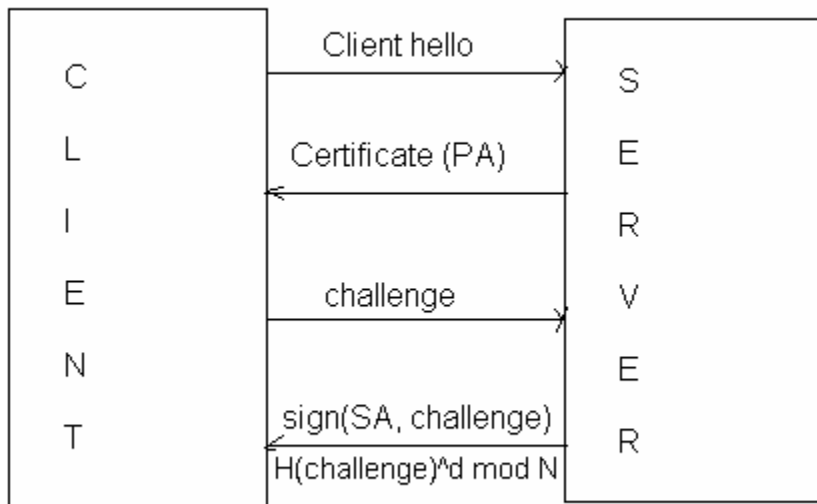


Denial of Service (Continued)

Properties of zombie nets:

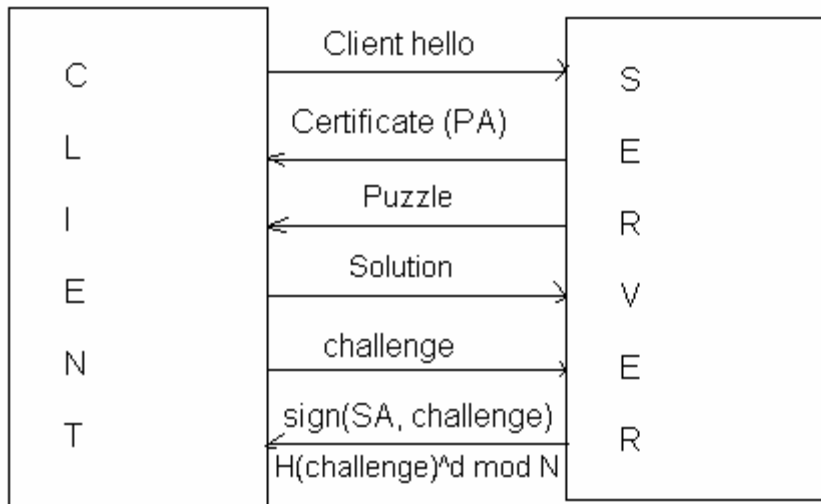
- One does not have to do much to execute a DoS attack – simply access the server providing the service.
- With 100k zombies, the attack can be made to look like “legitimate” traffic (just a lot of it) which will exhaust the system’s resources in time.
- It’s impossible to distinguish from all that traffic who the attacking computers are and who legitimate users are. It all looks the same

Client Puzzles & TLS



TLS not only encrypts transfers, but also authenticates the server. Moreover, responding to the client’s challenge requires big exponentiation which causes a certain asymmetry between the work the client does and that of the server.

To avoid DoS from clients that intend to exhaust the server’s resources by repeatedly asking for expensive RSA decryptions, it is necessary to fix that computational asymmetry. How about forcing the client to solve a puzzle in order to prove that they are legitimate? The diagram below expresses this change.



Properties of puzzle:

- hard to solve
- easy to generate
- easy to grade (check that the client gave the right answer)

Generate puzzles (use hashing):

Server sends y and x' where $h(x) = y$ and x' differs from x only in L least significant bits (L an integer)

Solve the puzzle:

On average it will take the client $2^{(L-1)}$ operations to solve the puzzle by trying all possible combination to arrive at x .

Now, we have changed the asymmetry somewhat; the client has some more work to do, and if they fail to do their work, we won't have to do the expensive decryption.

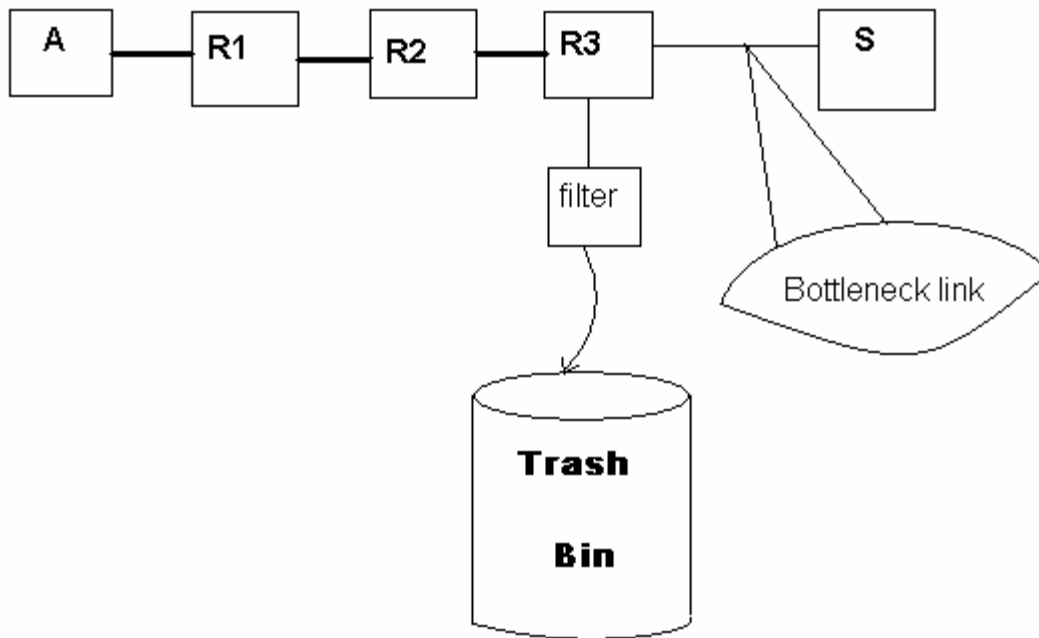
Note: we have not eliminated the problem, we've only made it harder for clients that want to flood the server with decryption requests to do so.

A numeric example:

If before puzzle, it took one client 1ms to request decryption from server, and it took server 10 ms to do decryption, all client has to do is generate 10 requests in 10 ms and that would suffice to lock up the server and make it unavailable to legitimate users.

Assuming that the puzzle changes the time it takes the client between RSA decryption requests to 1000ms, the attacker only has to command a small zombie net of 100 clients to again overwhelm the server with request.

IP Trace Back (Smarter Networks)



If we know the source address of the attacker A, we can put a filter in place that would discard all traffic with A as source address.

What if they are faking the source address?

One possible solution:

Let ?? be a packet from A, as ?? travels through the network, each Router tags the packet
e.g.:

A	R1	R2	R3	S gets ?? R1 R2 R3
??	?? R1	?? R1 R2	?? R1 R2 R3	

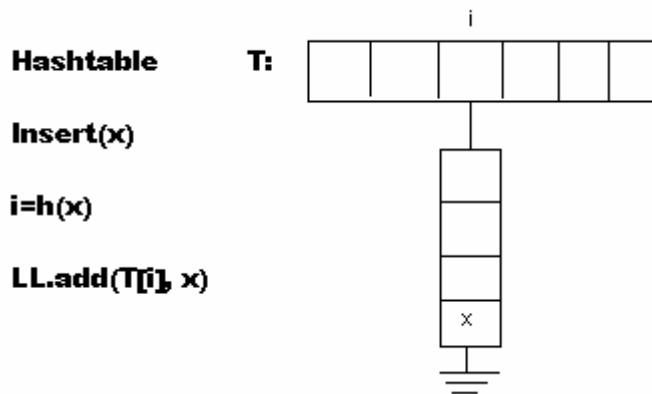
If S feels that it's under a DoS attack and all packets are tagged (passed through) R1, S can request that R1 start dropping packets (at least the ones with S as the destination)

Egress Filtering:

A router with Egress filtering can drop packets that look fake from leaving its internal network. For more info on Egress filtering, visit this link:

<http://www.sans.org/y2k/egress.htm>

Computational DoS



Theorem (first try): If T contains n cells and we insert n objects, then insertion/look-up costs are $O(1)$.

Many networks maintain a similar hashtable for current connections

If h is fixed and known, attacker can force lots of collisions. Attacker can send a lot of connections to cell i and the associated linked list, and then send packets that force the server to down all the way to x every time which would be a lot of work for the server.

Solutions:

→ Use random h.

At startup, pick random h from some family of hash functions H.

Theorem (final): If T contains n cells and we insert n objects, and $h \in_R H$ (h picked randomly from H), and H is 2-universal, then insertion/look-up costs are $O(1)$

Theorem: A family of hash functions, H, is 2-universal if, given $(x, h(x))$ for some x, you have no information about which hash function h_i is.

$H(x) = ax + b$ ($a \neq 0$) is one such family.