P2P Systems, Security and Overlays

Presented by
Vishal Misra

thanks to
Dan Rubenstein
Columbia University

Security in Structured P2P Systems

- Structured Systems assume all nodes "behave"
  - Position themselves in forwarding structure to where they belong (based on ID)
  - Forward queries to appropriate next hop
  - Store and return content they are assigned when asked to do so
- How can attackers hinder operation of these systems?
- What can be done to hinder attacks?

Attacker Assumptions

- The attacker(s) participate in the P2P group
- Cannot view/modify packets not sent to them
- Can collude

Classes of Attacks

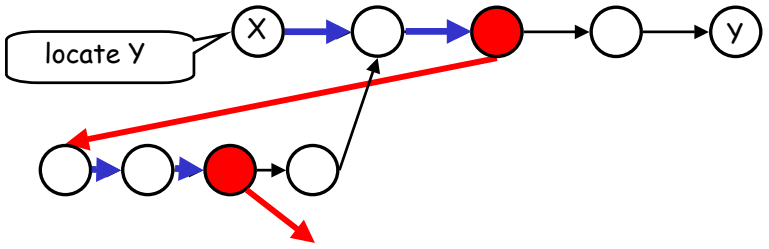
- Routing Attacks: re-route traffic in a "bad' direction
- Storage/Retrieval Attacks: prevent delivery of requested data
- Miscellaneous
  - DoS (overload) nodes
  - Rapid joins/leaves

Identity Spoofing

- Problem:
  - Node claims to have an identity that belongs to other node
  - Node delivers bogus content
- Solution:
  - Nodes have certificates signed by trusted authority
  - Preventing spoofed identity: base identity on IP address, send query to verify the address.

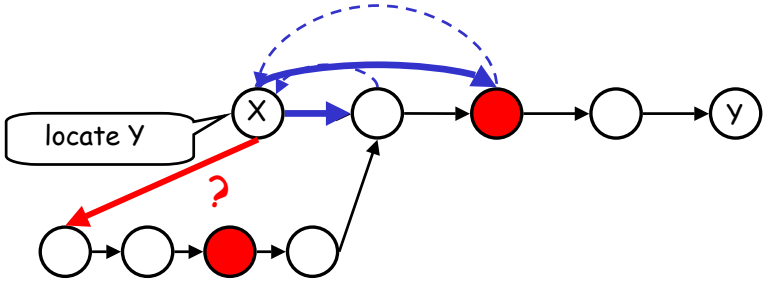
Routing Attacks 1: redirection

- Malicious node redirects queries in wrong direction or to non-existent nodes (drops)



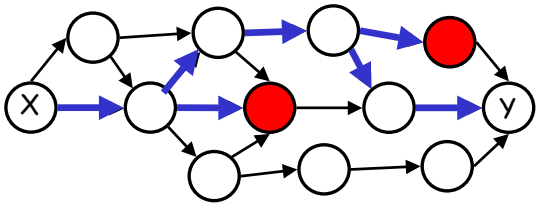
Suggested Solution: Part I

- Use iterative approach to reach destination.
  - verify that each hop moves closer (in ID space) to destination



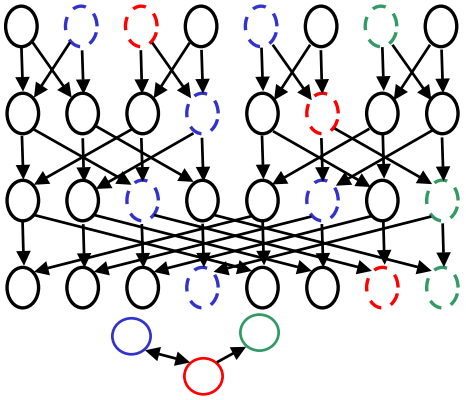
Suggested Solution: Part II

- Provide multiple paths to "re-route" around attackers



Choosing the Alternate paths: e.g., a CAN enhancement

- Use a butterfly network of virtual nodes w/ depth log n – log log n

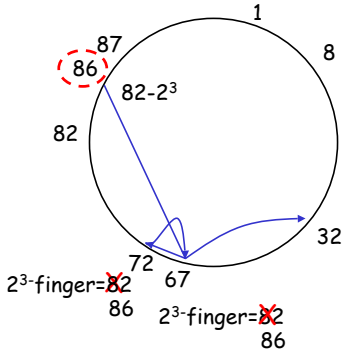


- Use:
  - Each real node maps to a set of virtual nodes
  - If edge (A,B) exists in Butterfly network, then form (A,B) in actual P2P overlay
  - "Flood" requests across the edges that form the butterfly
- Results: For any ε, there are constants such that
  - search time is $O(\log n)$
  - insertion is $O(\log n)$
  - # search messages is $O(\log^2 n)$
  - each node stores $O(\log^3 n)$ pointers to other nodes and $O(\log n)$ data items
  - All but a fraction ε of peers can access all but a fraction ε of content

Routing Attack 2: Misleading updates

- An attacker could trick nodes into thinking other nodes have left the system
- Chord Example: node "kicks out" other node
- Similarly, could claim another (non-existent) node has joined
- Proposed solution: random checks of nodes in P2P overlay, exchange of info among "trusted" nodes

Malicious node 86 "kicks out" node 82



e.g., for i=3

Routing Attack 3: Partition

- A malicious bootstrap node sends newcomers to a P2P system that is disjoint from (no edges to) the main P2P system
- Solutions:
  - Use a trusted bootstrap server
  - Cross-check routing via random queries, compare with trusted neighbors (found outside the P2P ring)

Storage/Retrieval Attacks

- Node is responsible for holding data item D. Does not store or deliver it as required
- Proposed solution: replicate object and make available from multiple sites

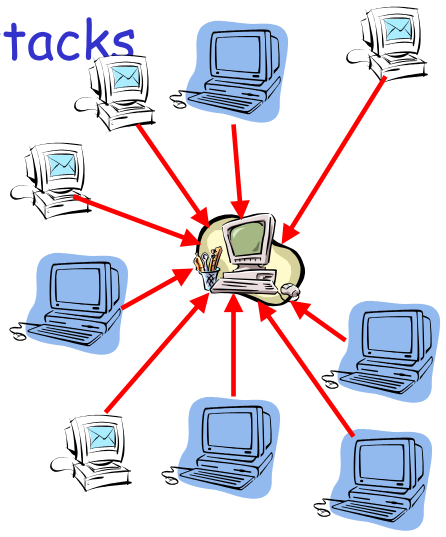
Miscellaneous Attacks

- Problem: Inconsistent Behavior - Node sometimes behaves, sometimes does not
- Solution: force nodes to "sign" all messages. Can build body of evidence over time
- Problem: Overload, i.e., DoS attack
- Solution: replicate content and spread out over network
- Problem: Rapid Joins/Leaves
- Solutions: ?

SOS: Using DHTs to Prevent DoS Attacks

To perform a DoS Attack:

1. Select Target to attack
2. Break into accounts (around the network)
3. Have these accounts send packets toward the target
4. Optional: Attacker "spoofs" source address (origin of attacking packets)



Goals of SOS

- Allow moderate number of legitimate users to communicate with a target destination, where
  - DoS attackers will attempt to stop communication to the target
  - target difficult to replicate (e.g., info highly dynamic)
  - legitimate users may be mobile (source IP address may change)
- Example scenarios
  - FBI/Police/Fire personnel in the field communicating with their agency's database
  - Bank users' access to their banking records
  - On-line customer completing a transaction

SOS: The Players

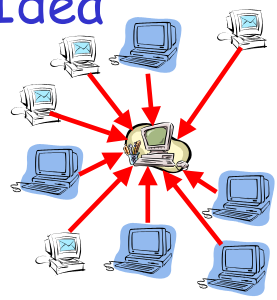
- Target: the node/end-system/server to be protected from DOS attacks

- Legitimate (Good) User: node/end-system/user that is authenticated (in advance) to communicate with the target

- Attacker (Bad User): node/end-system/user that wishes to prevent legitimate users' access to targets

SOS: The Basic Idea

- DoS Attacks are effective because of their many-to-one nature: many attack one

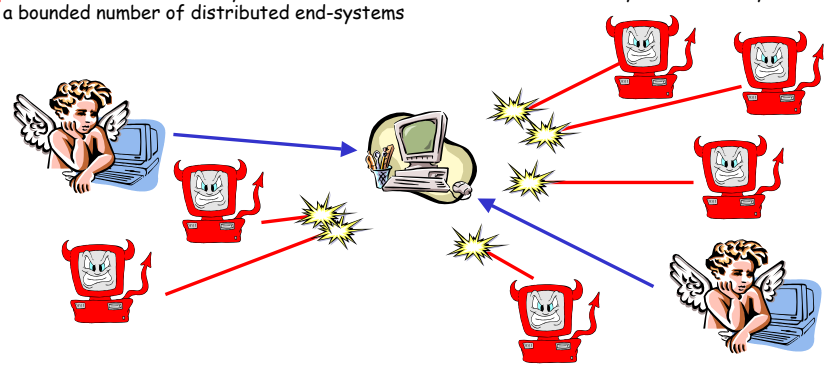


- SOS Idea: Send traffic across an overlay:
  - Force attackers to attack many overlay points to mount successful attack
  - Allow network to adapt quickly: the "many" that must be attacked can be changed

Goal

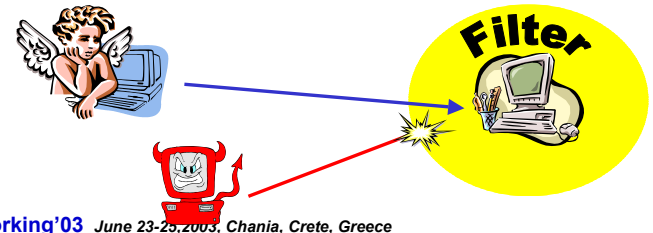
- Allow pre-approved legitimate users to communicate with a target
- Prevent illegitimate attackers' packets from reaching the target
- Want a solution that
  - is easy to distribute: doesn't require mods in all network routers
  - does not require high complexity (e.g., crypto) ops at/near the target

Assumption: Attacker cannot deny service to core network routers and can only simultaneously attack a bounded number of distributed end-systems



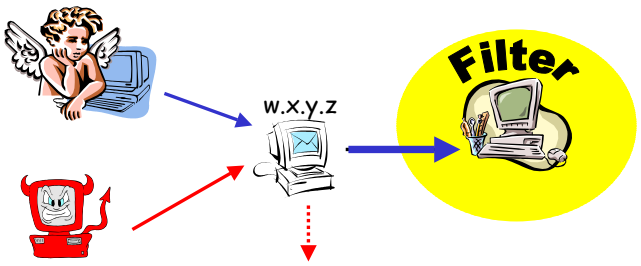
SOS: Step 1 - Filtering

- Routers "near" the target apply simple packet filter based on IP address
  - legitimate users' IP addresses allowed through
  - illegitimate users' IP addresses aren't
- Problems: What if
  - good and bad users have same IP address?
  - bad users know good user's IP address and spoofs?
  - good IP address changes frequently (mobility)? (frequent filter updates)



SOS: Step 2 - Proxies

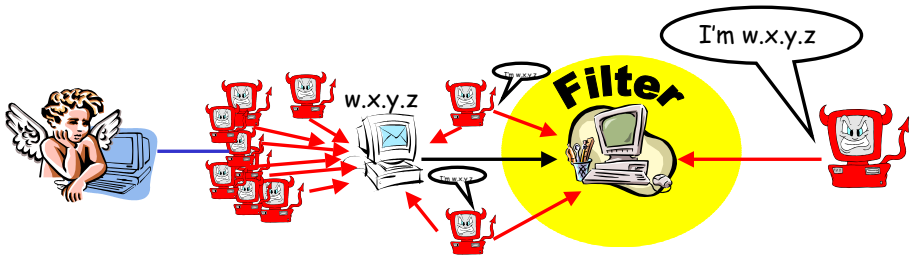
- Step 2: Install Proxies outside the filter whose IP addresses are permitted through the filter
  - proxy only lets verified packets from legitimate sources through the filter



Problems with a known Proxy

Proxies introduce other problems

- Attacker can breach filter by attacking with spoofed proxy address
- Attacker can DoS attack the proxy, again preventing legitimate user communication



SOS: Step 3 - Secret Servlets

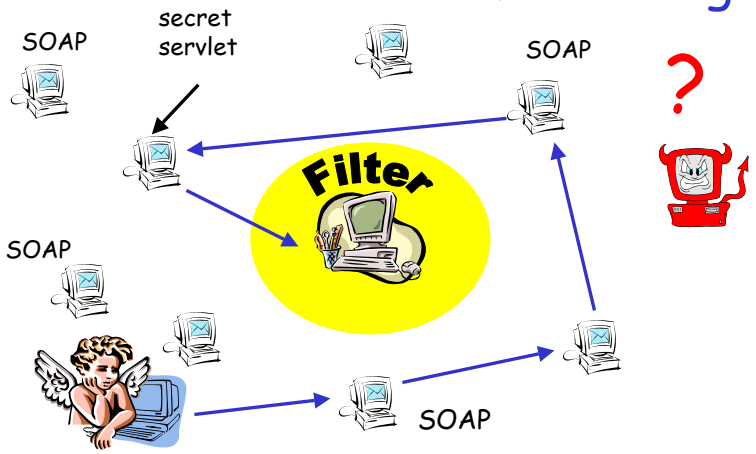
- Step 3: Keep the identity of the proxy "hidden"
  - hidden proxy called a **Secret Servlet**
  - only target, the secret servlet itself, and a few other points in the network know the secret servlet's identity (IP address)

SOS: Steps 4&5 - Overlays

- Step 4: Send traffic to the secret servlet via a network overlay
  - nodes in virtual network are often end-systems
  - verification/authentication of "legitimacy" of traffic can be performed at each overlay end-system hop (if/when desired)
- Step 5: Advertise a set of nodes that can be used by the legitimate user to access the overlay
  - these access nodes participate within the overlay
  - are called **Secure Overlay Access Points (SOAPs)**

User → SOAP → across overlay → Secret Servlet → (through filter) → target

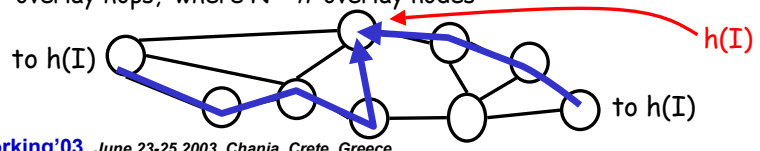
SOS with "Random" routing



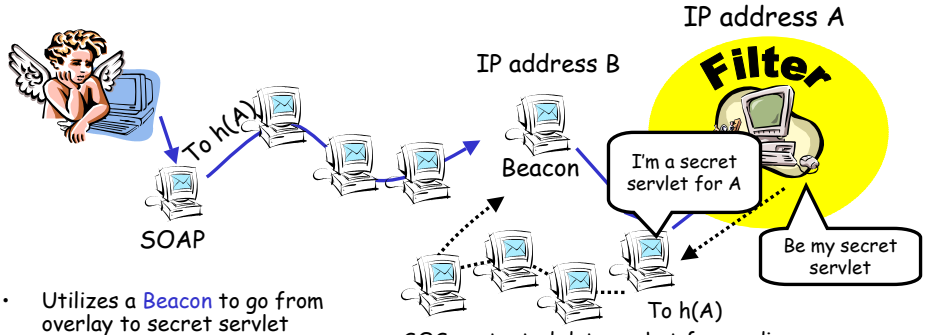
- With filters, multiple SOAPs, and hidden secret servlets, attacker cannot "focus" attack

Better than "Random" Routing

- Must get from SOAP to Secret Servlet in a "hard-to-predict manner": But random routing routes are long (O(n))
- Routes should not "break" as nodes join and leave the overlay (i.e., nodes may leave if attacked)
- Current proposed version uses DHT routing (e.g., Chord, CAN, PASTRY, Tapestry). We consider Chord:
  - A distributed protocol, nodes are used in homogeneous fashion
  - identifier, I, (e.g., filename) mapped to a unique node h(I) = B in the overlay
  - Implements a route from any node to B containing O(log N) overlay hops, where N = # overlay nodes



Step 5A: SOS with Chord



- Utilizes a Beacon to go from overlay to secret servlet
- Using target IP address A, Chord will deliver packet to a Beacon, B, where h(A) = B
- Secret Servlet chosen by target (arbitrarily)
- Servlet informs Beacon of its identity via Chord

SOS protected data packet forwarding

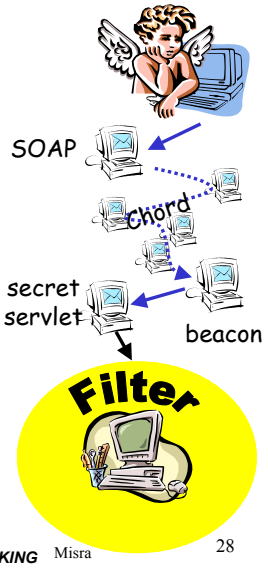
1. Legitimate user forwards packet to SOAP
2. SOAP forwards verified packet to Beacon (via Chord)
3. Beacon forwards verified packet to secret servlet
4. Secret Servlet forwards verified packet to target

Adding Redundancy in SOS

- Each special role can be duplicated if desired
  - Any overlay node can be a SOAP
  - The target can select multiple secret servlets
  - Multiple Beacons can be deployed by using multiple hash functions
- An attacker that successfully attacks a SOAP, secret servlet or beacon brings down only a subset of connections, and only while the overlay detects and adapts to the attacks

Why attacking SOS is difficult

- Attack the target directly (without knowing secret servlet ID): filter protects the target
- Attack secret servlets:
  - Well, they're hidden…
  - Attacked servlets "shut down" and target selects new servlets
- Attack beacons: beacons "shut down" (leave the overlay) and new nodes become beacons
  - attacker must continue to attack a "shut down" node or it will return to the overlay
- Attack other overlay nodes: nodes shut down or leave the overlay, routing self-repairs



SOS Summary

- SOS protects a target from DoS attacks
  - lets legitimate (authenticated) users through
- Approach
  - Filter around the target
  - Allow "hidden" proxies to pass through the filter
  - Use network overlays to allow legitimate users to reach the "hidden" proxies
- Preliminary Analysis Results
  - An attacker without overlay "insider" knowledge must attack majority of overlay nodes to deny service to target

Future directions with Overlays?

- More sophisticated routing
- Analogy:
  - Routes -> Frequency
  - DDoS -> Jamming
- Spread Spectrum Overlay Routing?
- Malicious overlay node detection using route PN sequences?