P2PNS: A Secure Distributed Name Service for P2PSIP

Ingmar Baumgart

Mobile P2P 2008, Hong Kong, China
Outline

- Decentralized VoIP (P2PSIP)
- Peer-to-Peer name service (P2PNS)
  - Architecture
  - Two-stage name resolution
- P2PNS security
  - Attacks on nodeID generation
  - Attacks on message forwarding
  - Attacks on DHT layer
- Conclusion
Peer-to-Peer SIP

- **What is P2PSIP?**
  - Using a peer-to-peer network instead of centralized servers for SIP user registration and location lookup

- **Why P2PSIP?**
  - Cost reduction (no servers needed)
  - Scalability
  - Reliability (No single point of failure, self healing)
  - Failover for server-based SIP networks (in emergency cases)
  - NAT traversal

→ Skype (largest VoIP provider in the world) also uses P2P technologies, but no open standard
Server-based SIP vs. P2PSIP

Call setup with server-based SIP:

1. Alice registers with server:
   - "REGISTER alice@tm.uka.de => 141.31.93.13"

2. Bob sends INVITE request:
   - "INVITE alice@tm.uka.de"
   - Contact: 141.31.93.13

Call setup with P2PSIP:

1. Alice registers with server:
   - "REGISTER alice => 141.31.93.13"

2. Bob sends INVITE request:
   - "INVITE alice"
   - Contact: 141.31.93.13
Decentralized VoIP (P2PSIP)

Main task in P2PSIP:
- Resolve AoR to current IP address
- Challenge: Many security issues in a completely decentralized network

Our approach: Generic distributed name service P2PNS *(IETF draft-baumgart-p2psip-p2pns-00)*
**Peer-to-Peer Name Service (P2PNS)**

- Distributed name resolution for:
  - P2PSIP, decentralized DNS, HIP, decentralized IM (XMPP)
- Same task in all scenarios:
  - Resolve a name (AoR, domain name, HIT) to the current transport address (IP, port)
- P2PNS interface:
  - `register(name, transport address)`
  - `resolve(name)`
- Name cache on top of KBR/DHT P2P layer
- Focus on security in completely decentralized networks:
  - Unique usernames
  - Prevent identity theft
P2PNS Architecture

- Modular architecture based on Common API:
  - Key Based Routing (KBR)
    - Task: Message routing to nodeIDs
  - Distributed Hash Table (DHT)
    - Task: Distributed data storage
  - Name Cache
    - Task: Caching of AoRs
  - P2PSIP proxy:
    - Connects legacy SIP UAs to the P2PNS service
Key-based Routing (KBR)

- Message routing to nodeIDs
- Provided by structured overlay networks
  - Kademlia, Chord, Koorde, Broose, Pastry
- Main idea:
  - Each node has a nodeID
  - Overlay routing table with nodeIDs of overlay neighbours
  - Efficient lookup of keys and nodeIDs in $O(\log N)$
Distributed Hash Table (DHT)

- Distributed storage of (key, value) tuples
- Uses the KBR layer to determine responsible nodes for data storage
  - Locate a node with a nodeID close to H(key)

![Diagram of Distributed Hash Table]

- H("sip:baumgart") = 2313
- Node stores the mapping (sip:baumgart, NodeID)
- NodeID ranges:
  - 0-1000
  - 1001-2000
  - 2001-4000
  - 4001-7000
  - 7001-10,000
  - 10,001-21,000
  - 21,001-40,000
  - 40,001-65,536
  - 65,536-10,000
  - 10,001-21,000
  - 21,001-40,000
  - 40,001-65,536
DHT security is expensive

- Malicious nodes can modify or delete locally stored data items
- Countermeasure: Replicate data items on k nodes and use majority votes
- Modifying data items in a DHT is expensive
- DHT usage for P2PSIP
  - Usual approach:
    - DHT stores AoR→IP mapping
  - P2PNS approach:
    - Two-stage name resolution based on KBR and DHT services
Two-Stage Name Resolution

1.) Resolve AoR $\rightarrow$ NodeID (DHT layer)
2.) Resolve NodeID $\rightarrow$ IP (KBR layer)

Motivation:
- Modification of data records on DHT is expensive (due to security mechanisms)
- (AoR, NodeID) binding is static: No modification needed if IP address changes (ID/Loc split)
- IP address changes are efficiently handled on KBR layer
Example: P2PNS user registration

1. REGISTER(To:U)

2. REGISTER(U)

3. JOIN(NodeID_X)

4. PUT(U, NodeID_X)

User U

Peer X

SIP

Name Cache

DHT

KBR

register()

resolve()

put()

get()

route()

lookup()

Example:

- User U registers with a SIP server.
- The SIP server then joins the DHT with a unique node ID (NodeID_X).
- The node ID is used to store the user's name in the Name Cache.
- The DHT provides a route to the user's name, which is then stored in the cache.
- Finally, the user's name is put into the KBR for lookup.
Example: P2PNS user lookup

1. INVITE(To:U)
2. RESOLVE(U)
3. GET(U)
4. LOOKUP(NodeID_X)
5. INVITE(To:U)
6. INVITE(To:U)

User V

Peer Y

Name Cache

DHT

KBR

register()
resolve()

put()
get()

route()
lookup()
P2PNS security threats

- **Attacks on routing (KBR)**
  - NodeID generation
    - By carefully choosing a node ID an attacker can control access to target objects
  - Message forwarding
    - Malicious nodes along the route between sender and target node can modify or drop messages to a key
  - Routing table maintenance
    - DoS attack by distribution of faulty routing table updates

- **Attacks on data storage (DHT)**
  - Malicious nodes can modify or delete locally stored data items
Attacks on nodeID generation

- Eclipse attack: By carefully choosing a nodeID an attacker can control access to target objects
- Sybil attack: A single node can join the network with several nodeIDs
- Countermeasure:
  - Make nodeID generation expensive
  - Limit free nodeID selection
Secure NodeID generation

- **Common approach:** NodeID = SHA1(IP+port)
  - **Problems:**
    - Sybil attack still possible if an attacker controls several IP addresses
    - Constantly changing nodeIDs on dial-up connections

- **Better:** NodeID = SHA1(public key)
  - Public key can be used to authenticate node messages
  - Sybil attack and choose of a specific nodeID still feasible
    - Use in combination with crypto puzzles to make creation of new nodeIDs expensive
    - Use a offline CA to generate nodeIDs (if available)
Attacks on message forwarding

- Malicious nodes along the path between sender and target node can modify or drop messages to a key.

- Countermeasure: Parallel lookup over disjoint paths increases the lookup success ratio:
  \[ P(\text{lookup success}) = 1 - (1 - (1 - m)^h)^d \]

- Most important security properties of KBR protocols:
  - Average path length \( h \)
  - Number of disjoint paths \( d \)
Effect of disjoint paths on lookup success

Even with 25% adversarial nodes 99% lookups succeed in a Kademlia network with 10000 nodes
Secure DHT layer

- Data records must only be modified by the owner of a record
  - Modification requests are signed with $k_{\text{priv}}$
- Only store a single record for each key
  - Unique usernames
- Data records are replicated on $k$ nodes
  - Query all replica in parallel and use majority votes
  - Joining nodes pull all replica in their key range
P2PNS implementation

- Unmodified SIP UAs
- Added P2PNS support to OpenSER SIP proxy
- Overlay Framework OverSim (http://www.oversim.org/)
  - Provides P2PNS service to the P2PSIP proxy
Future work

- **KBR protocol selection**
  - Several promising candidates:
    - Kademlia, Broose, Pastry
    - Focus on low latency and security

- **Evaluation of DHT replication strategies**

- **Standardization**
  - Generic P2P protocol
  - Common interface for KBR/DHT service

- **Bootstrapping**

- **NAT traversal**
Conclusion

- P2PNS provides generic name resolution for:
  - P2PSIP, DNS, Jabber, HIP
- Modular architecture based on Common API
- Focus on security in completely decentralized environments
- Two-stage name resolution reduces communication costs in dynamic networks
Thank you for your attention!

Any questions?