Proxychain: Developing a Robust and Efficient Authentication Infrastructure for Carrier-Scale VoIP Networks

Italo Dacosta and Patrick Traynor
Performance, Scalability and Security

• Finding the right balance between performance/scalability and security is a well-known challenge.

• Robust but computationally expensive security mechanisms are difficult to deploy in production environments
  – S-BGP, DNSSEC

• Weaker but more efficient security mechanisms are generally broken and abused
  – WEP, IKE Aggressive mode
Another Example: SIP Authentication

- Session Initiation Protocol (SIP)
  - Establishes, manages and terminates sessions between two or more clients
  - Generally associated with VoIP
- RFC 3261 recommends several security mechanisms: Digest authentication, SSL/TLS, IPsec and S/MIME
- However, **Digest authentication** is typically the only one employed
  - Weaker but more efficient
SIP Digest Authentication

- Challenge-response authentication protocol
- Based on cryptographic hash operations (MD5)
- De facto authentication mechanism in SIP
SIP Dialogs with Digest Authentication

INVITE
407 Response [realm, n]
INVITE [H(H(uid||realm||pwd)||n||H(method||URI))]

uid
H(uid||realm||pwd)
INVITE

Request to DB always required

uid
H(uid||realm||pwd)
INVITE
Problems with Digest Authentication

• Inefficient in scenarios with a remote authentication service or database
  – RTT added to each authentication operation
  – One request to the database per authenticated SIP message
  – High load in the database if it is shared by multiple SIP servers

• Considered a weak authentication protocol
  – E.g., No mutual authentication
Our Scenario: A Nationwide VoIP Provider

P = SIP Proxies
DB = Authentication database
The Problem: Digest Authentication

Performance in Our Scenario

- No Authentication: ≈ 24,000 cps (no auth.)
- Digest Authentication: ≈ 1,160 cps (Digest auth.)
Our Proposed Solution

• Reduce the number of requests to the database by **caching temporary authentication credentials** in the proxies

• Use **hash chains** to build these temporary credentials
  – Take advantage of hash chains properties

• **Caching Digest auth. credentials reduces security!**
Hash Chains Background

- Sequence of **one-time** authentication tokens
- Created by applying a **cryptographic hash function** to a secret value $r$ multiple times

$$H^n(r) = H(\ldots H(H(r))\ldots)$$
Methodology

• Design and implementation of new SIP authentication protocol: **Proxychain**
• Experimental evaluation
  – Call throughput
  – Bandwidth utilization
  – CPU utilization
• Results analysis
Proxychain Design Goals

- **Efficiency**
  - Faster authentication operations
- **Scalability**
  - Support larger number of users and proxies
- **Security**
  - Provide more security guarantees
Proxychain SIP Dialogs

1. Alice (UAC) sends an INVITE message containing the AP key.

2. The Proxy forwards the INVITE message to Bob (UAS).

3. The Proxy sends a 407 Response message to Alice, containing the AP key, and a HMAC value for the new session.

4. Alice sends an INVITE message to Bob, containing the AP key, HMAC, and a new session ID.

5. Bob sends an INVITE message to Alice, containing the AP key, HMAC, and the new session ID.

6. No request to DB is required.

Secure Channel:

A, P

H^l(tk_A), i, n_{DA}, n_{DP}, tk_P

INVITE

INVITE

INVITE [n_{AP}]

INVITE [A, B, i, HMAC(tk_P, A||B||i), H^{i-1}(tk_A)]

INVITE [A, B, i-1, HMAC(tk_P, A||B||i-1), H^{i-2}(tk_A)]

INVITE
Proxychain implementation

• Modifications to proxy, database and client software
  – Implemented in C language
  – Relatively small when compared to original code base

• Total credential size (MD5): 134 bytes
  – Only ≈26 MB of proxy’s memory required for storing 200,000 users credentials
Experimental Setup

• **Planetlab** for obtaining real RTT values

• **GT Emulab testbed** for database and proxies
  – **OpenSIPS** for proxies
  – **MySQL** for the database

• Nine high-capacity servers for generating SIP call traffic
  – **SIPp** as the SIP traffic generator
Results: Call Throughput

- No Authentication: ≈ 24,000 cps (no auth.)
- Digest Authentication: ≈ 1,160 cps (Digest auth.)
- Proxychain: ≈ 19,700 cps (Proxychain)
Results: Database CPU Utilization

- Digest authentication
- Proxychain

DB saturation (dual core machine)

MySQL % CPU utilization vs Time (sec)
Results: Scalability

Maximum usable throughput (cps)

# of proxies

Digest authentication.
Proxychain

\[ y = 3243.9x + 416.5 \]

\[ R^2 = 0.998 \]
Results: INVITE and BYE Authentication

Proxychain (INVITE) 
Proxychain (INVITE and BYE)

≈ 19,700 cps (INVITE)
≈ 12,000 cps (INVITE+BYE)
Discussion: Performance and Scalability

• Proxychain reduces the effects of network latency, allowing higher call throughput

• Lower load to the database allows more scalability and lower HW requirement
Discussion: Performance and Scalability

• Hash chains allow **constant storage space**
  – Dynamic reprovisioning (future work)

• **Key assumption:** each proxy caches most of its users’ credentials (>75%)
  – Pre-fetching mechanism
  – Cache eviction policies (future work)
Discussion: Security

• **Security improvements** over Digest authentication and hash chain protocols
  – **Efficient mutual authentication**, additional security verifications

• Protection against passive and active attackers
  – Stealing credentials from a proxy **does not allow user impersonation** (only affects mutual authentication)
Conclusions

• Proxychain simultaneously provides a robust, scalable and efficient authentication mechanism for carrier-scale SIP providers without additional HW

• Even non-carrier level infrastructures with centralized authentication service can benefit from Proxychain

• The key concepts behind Proxychain can be applied to authentication protocols in other domains
Questions?

Contact: idacosta@gatech.edu