RESTful Security

2009-05-21 W2SP

Dan Forsberg
Nokia Research Center &
Helsinki University of Technology (HUT)
dforsber@gmail.com

Web security concerns

Architectural mismatch

RESTful architectural style vs. Traditional web security

- Stateless
- Simple URIs
- Very caching friendly

- Stateful
- TLS pipes
- No caching

Performance and scalability

- TLS re-encrypts data for every client and session, no caching

Liability, privacy, and copyright

- Personal data needs to be protected by the service provider

But on the other hand: privacy concerns without TLS

- Personal data transferred and cached unencrypted on the web

Split access control and data protection

Encrypt the data and do access control for the decryption keys

1. Encrypt files
2. Provide keys over TLS session for authenticated and authorized users
3. Use key hierarchies for robust key management (details in the paper)

➔ Flexible access control – RESTful ☺

- Separates **data access** (encrypted data) and **data location** (public URI) from **access control** (access to decryption keys)
- Separates **access control enforcement time** (encryption procedure) from **access times** (access to decryption keys & decryption procedures)

Next steps?

Implementation level?

- Browser, plugin/middleware, application, ...

How to identify that data is encrypted? Key identity?

- HTTP header, MIME type, filename part, application level, ...

Let users control access to keys?

- Application policy for semi-automatic key sharing between users

Combine with OAuth?

- Access Token becomes the decryption key..

Comments? Questions?
Thank you! ☺