A Multipurpose Delegation Proxy for WWW Credentials

Tobias Straub
Johannes Buchmann

Computer Science Department
Technische Universität Darmstadt
`tstraub@gkec.tu-darmstadt.de`

Thilo-Alexander Ginkel

TG Byte Software GmbH
Bensheim
(now with SAP Germany)

EuroPKI Workshop 2005 – Canterbury

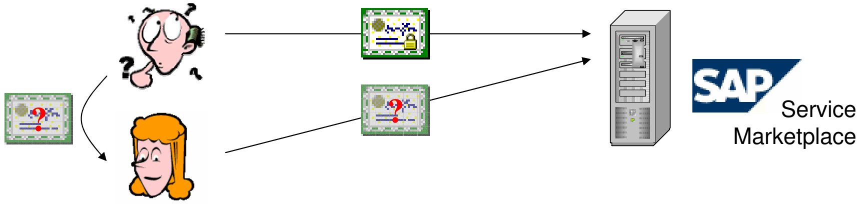
Motivation

- Most client-server applications on the Internet rely on HTTP as communication protocol – e.g.
  - online stores / banking
  - web-based e-mail
  - virtual market places, enterprise portals ...
- Same technology also used on intranets as it is ubiqitous and cost-effective

=> User authentication requires credential management – often burdensome

Scenario

- Logon at an online portal via SSL/TLS using an X.509 user certificate for authentication

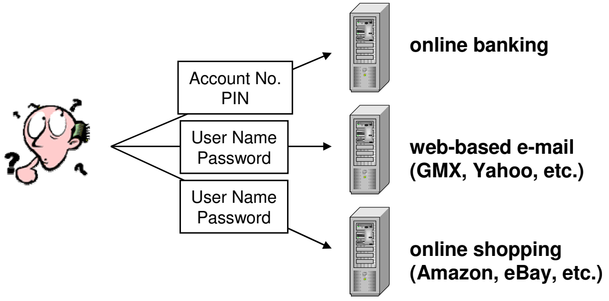


Service Marketplace

- Challenges:
  - (temporary) delegation to a proxy
  - group usage w/o revealing the credential
  - logging / restrict / withdraw access

Alternative Scenario

- Managing personal account data



Account No.
PIN

User Name
Password

User Name
Password

online banking

web-based e-mail (GMX, Yahoo, etc.)

online shopping (Amazon, eBay, etc.)

- Challenges: limited memory capacity, change policy, minimum complexity rules

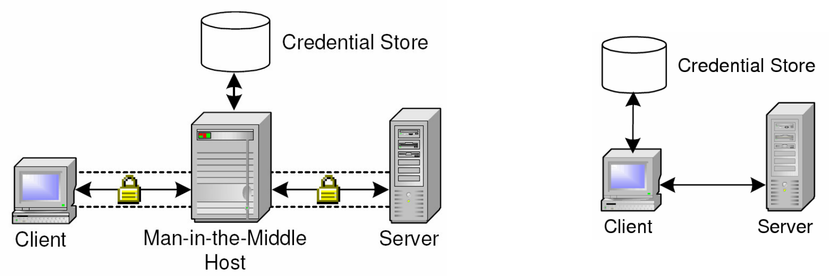
Requirements

1. Let proxy impersonate credential owner w/o need to reveal secret.
2. Protect credential from unauthorized access.
3. Keep track of actual credential usage and restrict it.
4. Support common WWW authentication mechanisms (Basic/Digest Auth., form-based, client certificate-based)
5. Easy-to-use, small footprint solution, operate transparently.

System Architecture

Examined 4 variants:

- 3 man-in-the-middle (MITM) approaches where client authenticates towards gateway and gateway authenticates towards target host
- 1 *client-side solution* where client has access to credential DB



Application Server Variant

- Gateway = *machine providing remote login facilities*, e.g.
  - VNC (full-screen desktop)
  - X11 (forwarding single browser windows)
- Credentials managed by browser running on gateway

Drawbacks:

- no proper delegation or policy enforcement,
- low protection level for credentials,
- user experience,
- network latency

HTTP Server Variant

- Gateway = *web server*
- calling particular URL initiates retrival of credential and authentication to target host for actual URL, e.g. go to http://gateway/amazon.com instead of http://amazon.com
- response returned to client seems to come from gateway

Drawbacks:

- MITM has to do hyperlink-rewriting to redirect subsequent requests to gateway, not directly to target host
- This is difficult for JavaScript and Macromedia Flash
- Java applets won't run properly due to Sandbox restrictions.

HTTP Proxy Variant

- Gateway = *intelligent HTTP proxy*
- Well-known approach on the Web
- Proxy works as forwarding agent acting both as server and client => HTTP-based credentials supported naturally

Drawback:

- SSL normally tunneled through proxy => breaking up end-to-end security (i.e. confidentiality)

Client-Side Architecture

- Standard *web browser enhanced* with additional functionality to access centralized credential store.
- Shouldn't be too difficult ...
- Alternatively fit up client’s TCP/IP stack or socket library
- Allows seamless integration and end-to-end security (confidentiality in this case)

Drawback:

- Credentials are revealed to browser => malicious client could steal them.

Comparison

- For each variant**, we** assessed:
  - *Compatibility and Standard Compliance*,
  - *Transparency*,
  - *Usability*,
  - *Security Characteristics*,
  - *Deployment Costs*
- (see paper for details, please)

	App. Server	HTTP Server	HTTP Proxy	Client-Side
Compatibility	○	○	+	○/−*
Transparency	+/−−*	○	+	++/○*
Usability	○	○	++	+
Security	−	+	+	−
Deployment	+/−−*	++/○*	○	−−

- Decided to implement HTTP proxy architecture
  – considered it superior to other MITM architectures
  – lower estimated developement costs compared to client-side variant

TLS Authentication Proxy Prototype

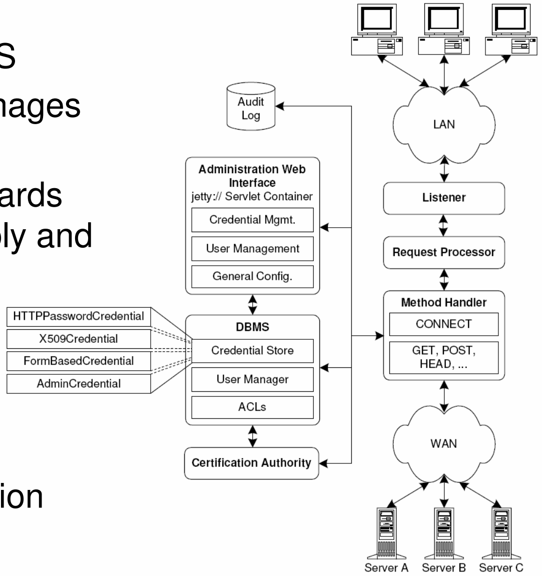
– proxy for HTTP and HTTPS

– stores credentials and manages access rights (ACL)

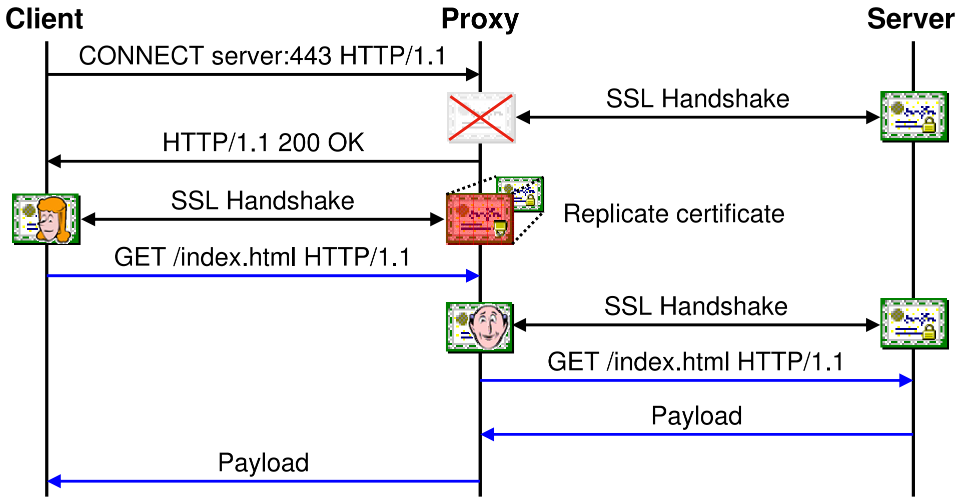
– authentication method towards proxy can be chosen flexibly and independently of actual authentication method

– credential usage is logged

– Web interface provided for administration and delegation



Data flow for an SSL target web site with certificate-based authentication



Client

Proxy

Server

CONNECT server:443 HTTP/1.1

SSL Handshake

HTTP/1.1 200 OK

SSL Handshake

Replicate certificate

GET /index.html HTTP/1.1

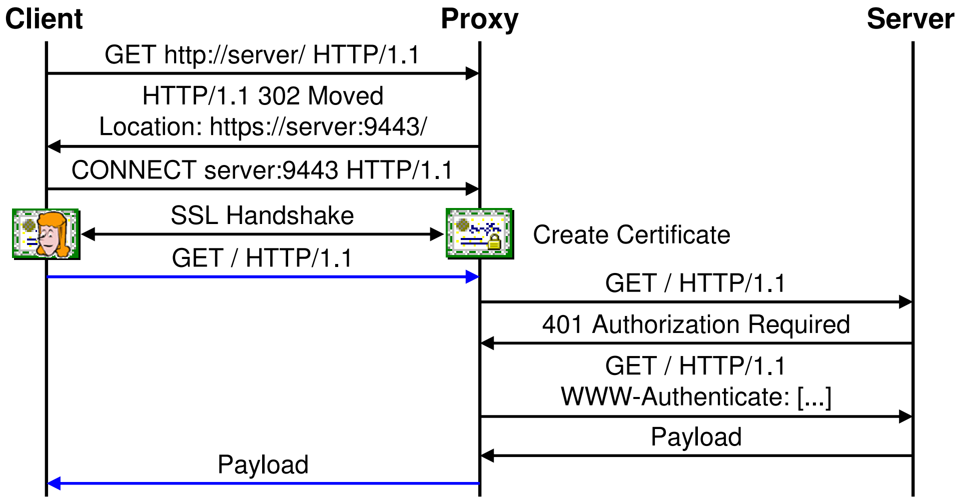
SSL Handshake

GET /index.html HTTP/1.1

Payload

Payload

Data flow for an HTTP target web site



Client

Proxy

Server

GET http://server/ HTTP/1.1

HTTP/1.1 302 Moved
Location: https://server:9443/

CONNECT server:9443 HTTP/1.1

SSL Handshake

Create Certificate

GET / HTTP/1.1

GET / HTTP/1.1

401 Authorization Required

GET / HTTP/1.1
WWW-Authenticate: [...]

Payload

Payload

Session Management

- Basic/Digest Authentication: each request has to be authenticated
  - TLS Authentication Proxy does not wait for "401" responses => Basic Authentication speeded up
  - Not possible for Digest Authentication as it is a challenge-response scheme
- Usually session management used for form-based authentication
  - HTTP Cookies fully supported by prototype, can re-authenticate automatically to avoid time-outs

Authorization Issues

- Proxy identifies end-users by distinguished name in their certificate, PKCS#12 tokens issued on-the-fly
- Access rights modeled by database table indexed by pair (user ID, credential ID)
- Constraints defined on fine-grained basis (time frame for access, maximum number of total/daily logins, restriction to a subset of web pages …)
- Flag indicates whether delegation is allowed, currently no restriction on number of sub-delegations

Deployment

1. Key and certificate distribution:
   => ship root certificate within PKCS#12 file

2. Web browser configuration:
   – Web Proxy Auto-Discovery Protocol
   – obtain Proxy Auto-Configuration file from the network automatically or manually

Privacy Issues:

– PAC file adjusted to only route traffic through proxy that requires credential
– URLs of such hosts obscured by hash function (PAC file is JavaScript code)

Conclusions

- Work motivated by requirement to delegate X.509 credentials
- TLS Authentication Proxy offers transparent credential management and proxy authentication
- Users authenticated w/o knowing credential
  => allows delegation and group usage
- Pseudo SSO: target host is unaware of what's going on
- Zero footprint solution, reasonable deployment costs
- Central storage for credentials instead of spread all over clients, however Single Point of Failure/Attack

Thank you for your attention!

Questions?

Contact: Tobias Straub / TU Darmstadt

`tstraub@gkec.tu-darmstadt.de`

`www.informatik.tu-darmstadt.de/GK/participants/tstraub/`