Admin Traffic Security

Tim Quinn
asarch Review
Aug. 11, 2010

Agenda

- Overall requirements/goals
- Steady-state
  - Admin client ↔ DAS
  - DAS ↔ instance
  - Some implementation notes
- Bootstrapping
- Some possible to-do items

General Requirements/Design Goals

- Command-line compatibility with GlassFish 2
- Elective – admin security not required
- When elected:
  - *Never* send sensitive information in the clear
  - Secure all traffic among clients, DAS, instances
  - Prevent remote admin client access directly to instances

Secure All Traffic

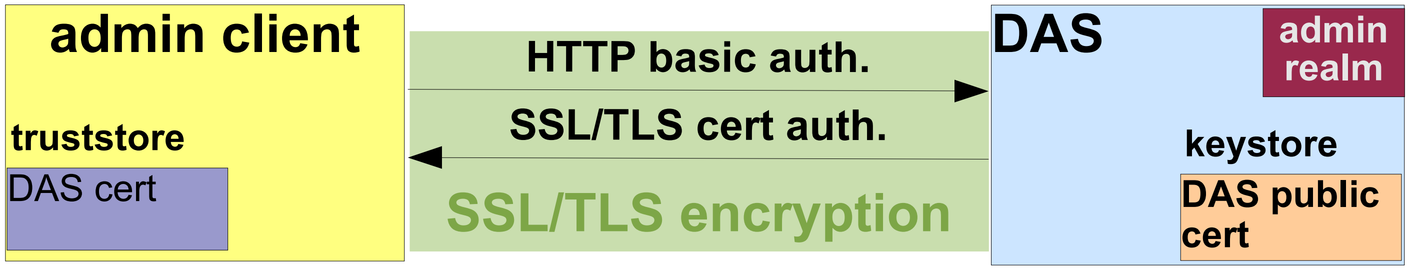
- Integrity, confidentiality: SSL/TLS encryption
- Authentication
  - Clients – at HTTP level using HTTP header

    ```
    Authorization: Basic [encoded user:password]
    ```

    (but only over secure connection, unlike GlassFish 2)

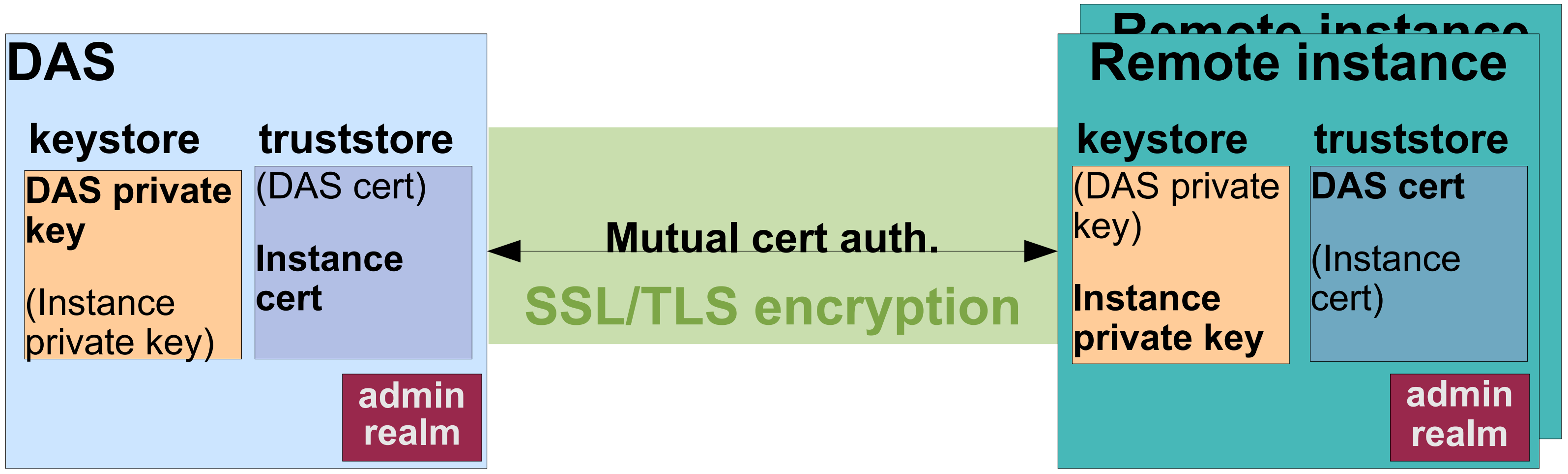
  - Servers (DAS, instances) – SSL/TLS level using certificates
- Authorization – authenticated Principal must be in admin-realm

Admin client ↔ DAS



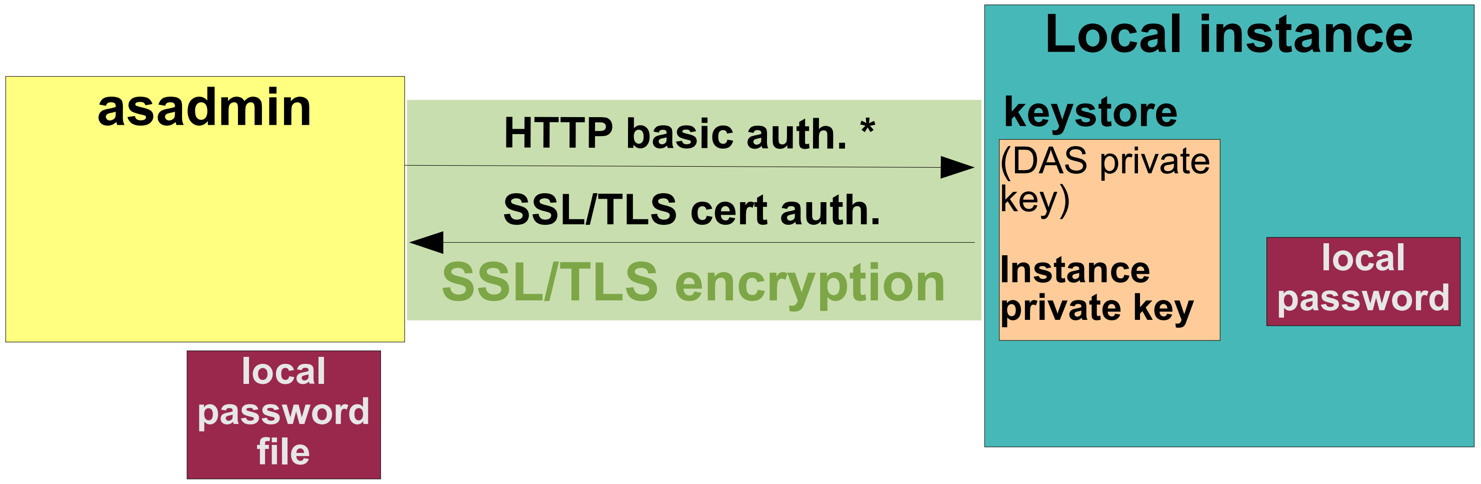
- DAS presents its public cert; client displays to user who accepts/rejects
  - asadmin always stores cert in ~/.asadmintruststore
  - Browser asks user whether to store in its truststore
- Client provides HTTP authentication
  - DAS challenges if necessary
  - DAS authorizes against admin-realm

DAS ↔ Instance



- Mutual cert authentication
- Receiver authorizes Principal from Grizzly request against admin-realm

asadmin client ↔ Local Instance



- Secure connection over SSL/TLS
- Instance presents its cert
- *asadmin sends empty user, local password from file
- Instance matches supplied password with in-memory password

Enabling/disabling

- Two new commands

```
enable-secure-admin-traffic
        [ – dasalias=alias (default s1as)]
        [ --instancealias=alias (default gf-instance)]


disable-secure-admin-traffic
```

- (Is there a good, single-command substitute?)
- Sent to all running instances
- Affects all configurations in domain
- Affects default-config so future new instances use correct settings
- Changes Grizzly config, adds instance alias to DAS admin realm

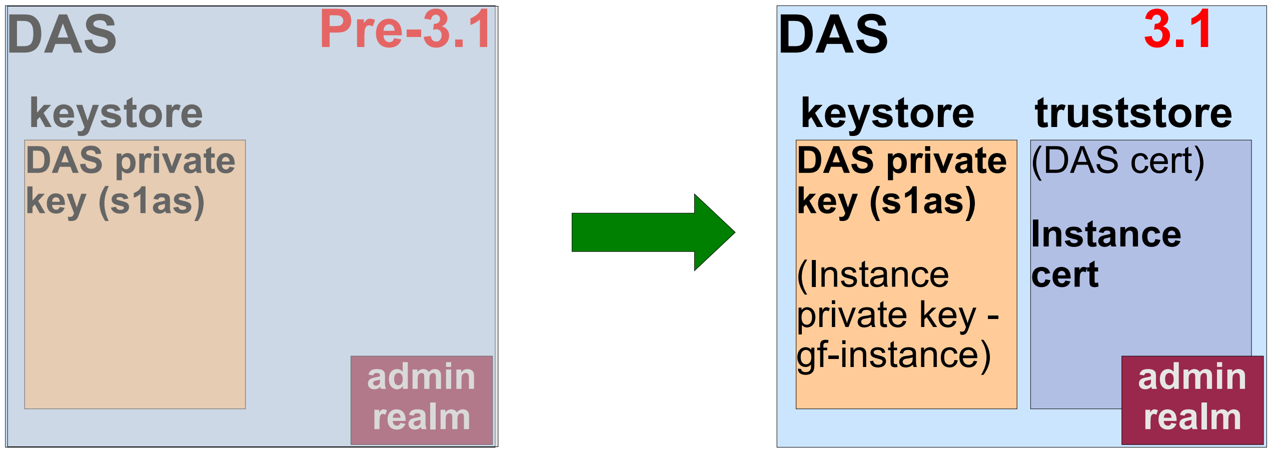
Some Implementation Notes

- Grizzly config
  - Port unification – http, https handled by one port
  - Redirection – http → https
  - SSL/TLS cert-based authentication
    - DAS: s1as (DAS self-signed cert)
    - Instance: gf-instance (instance self-signed cert)
    - DAS & instance: client-auth=want (not need)
- Instances
  - Have exact copy of DAS keystore, truststore
  - Have copy of private admin FileRealm containing only DAS principal (included in sync operations)

Bootstrapping

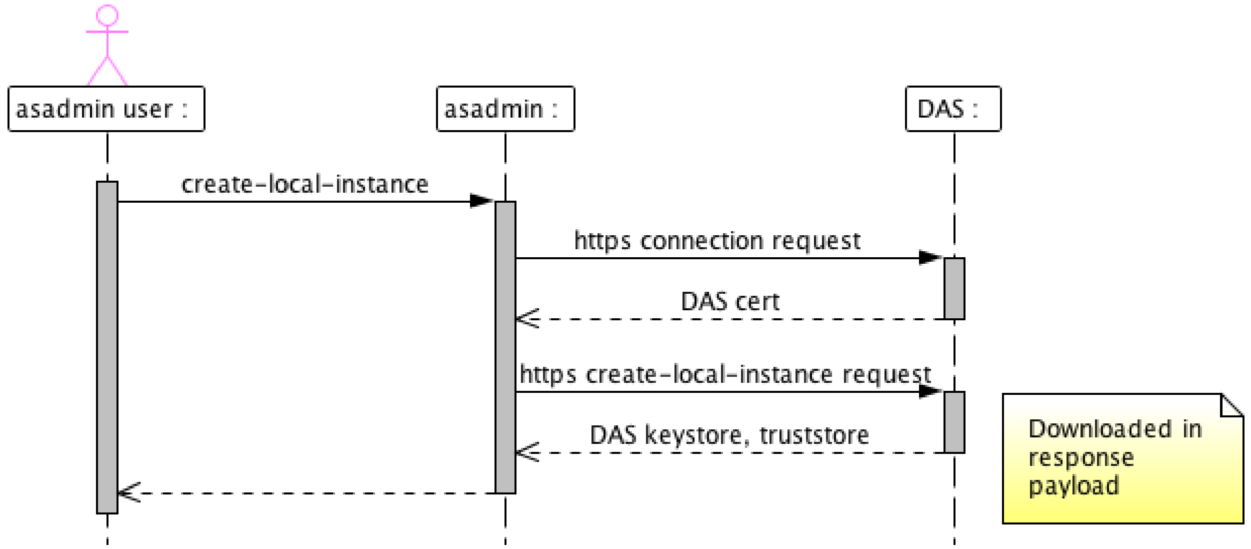
- DAS
- Create instance locally
- Create instance remotely

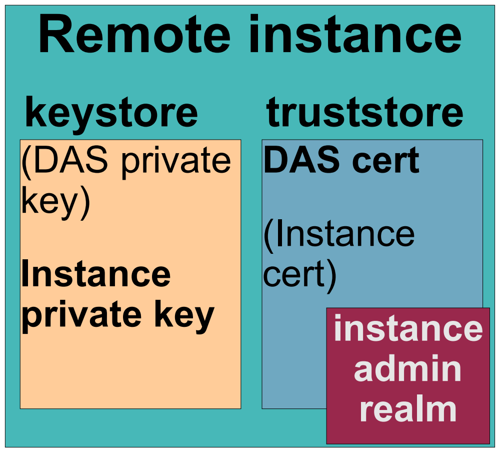
Bootstrapping DAS



- During build/create-domain:
  - Create truststore, add s1as public cert to truststore
- During initial domain start-up (or “slightly later”):
  - Generate self-signed key pair for instances to use
    - Save private key in keystore with alias gf-instance (e.g.)
    - Save public cert in truststore with alias gf-instance
    - Add gf-instance to admin realm

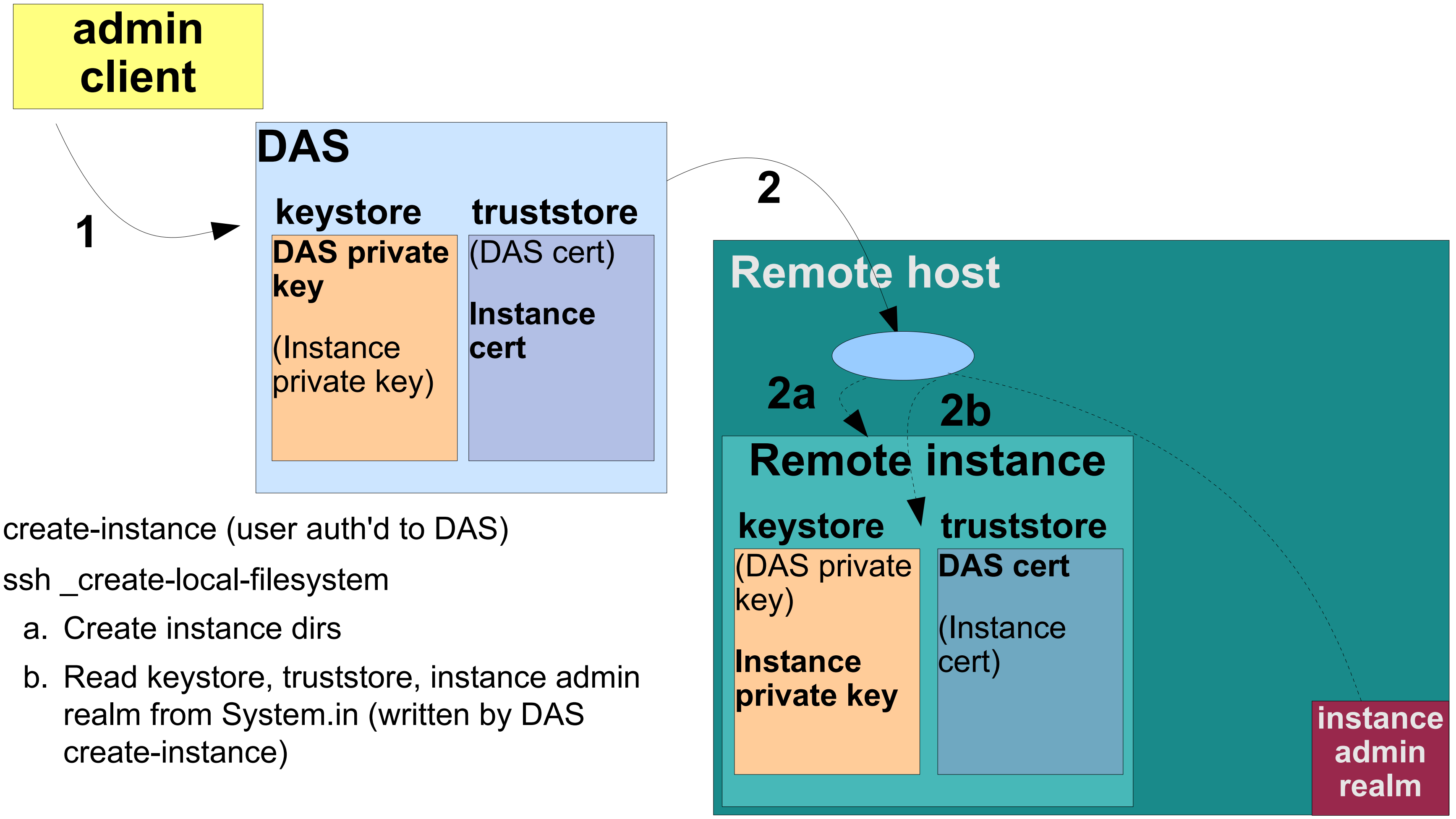
Bootstrapping
Create instance locally





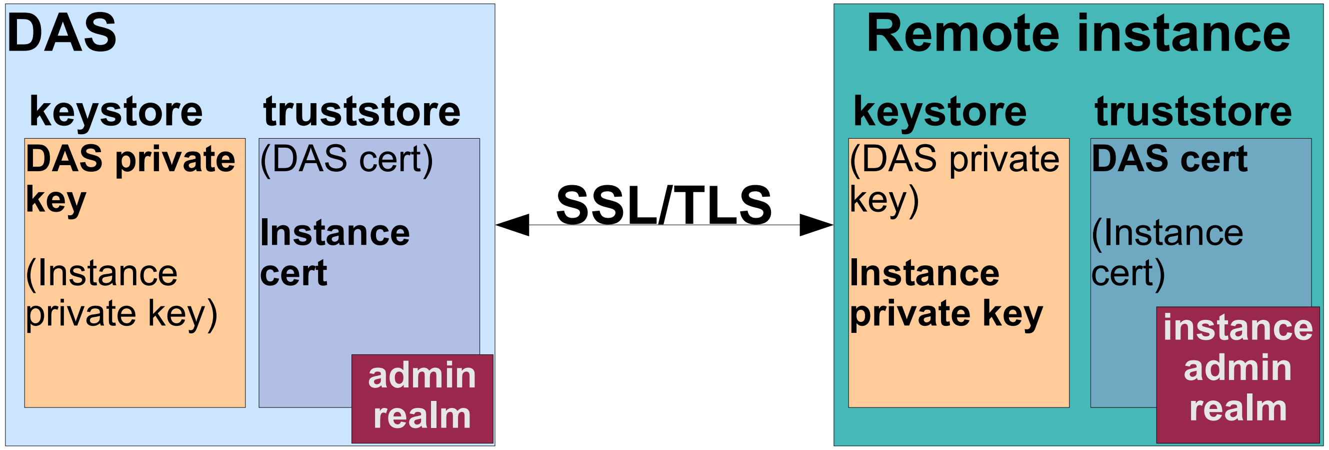
- Command response payload: keystore, truststore, inst. admin realm
- When instance starts it has correct keystore, truststore for mutual auth with DAS

Bootstrapping
Create instance remotely



1. create-instance (user auth'd to DAS)
2. ssh _create-local-filesystem
   a. Create instance dirs
   b. Read keystore, truststore, instance admin realm from System.in (written by DAS create-instance)

Bootstrapping
Create instance locally or remotely



Whether by create-local-instance or create-instance;

- Correct keystore, truststore, private admin realm in place on instance
- At start-instance time:

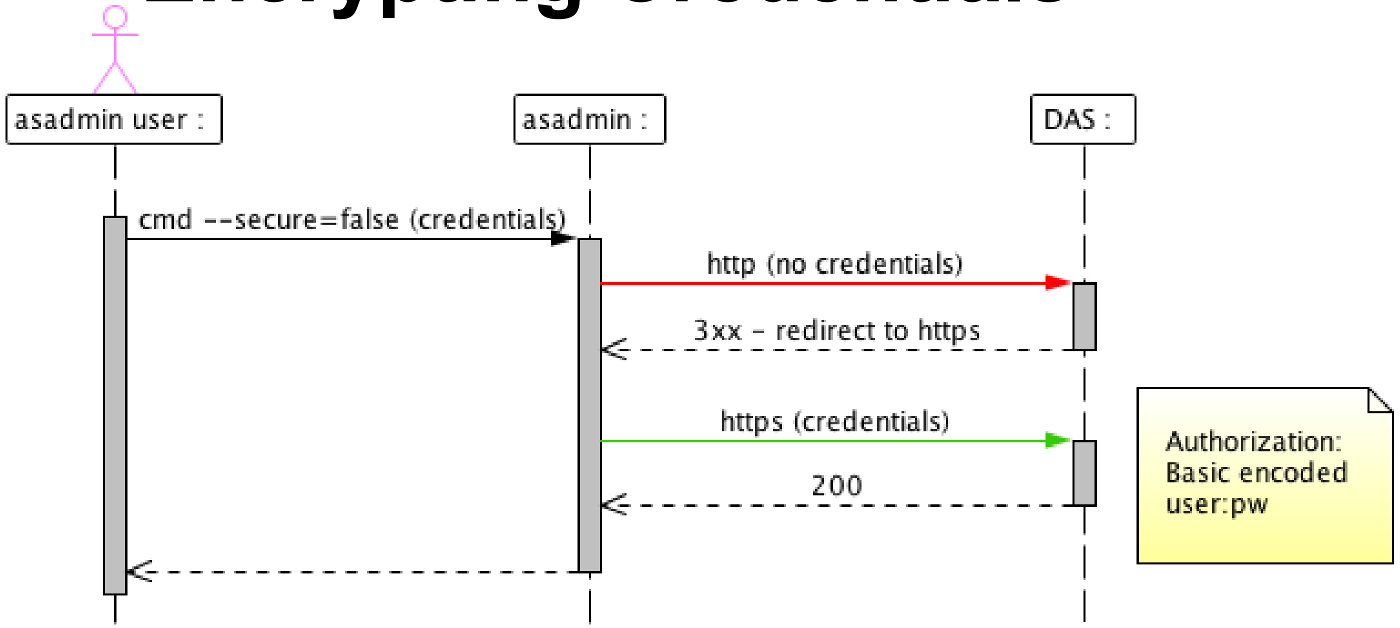
  DAS ↔ instance mutually authenticate

Some To-do Items...

- Open questions
  - Best way to deliver data with create-instance (stdin? Buffer?)
  - Better command name(s)?

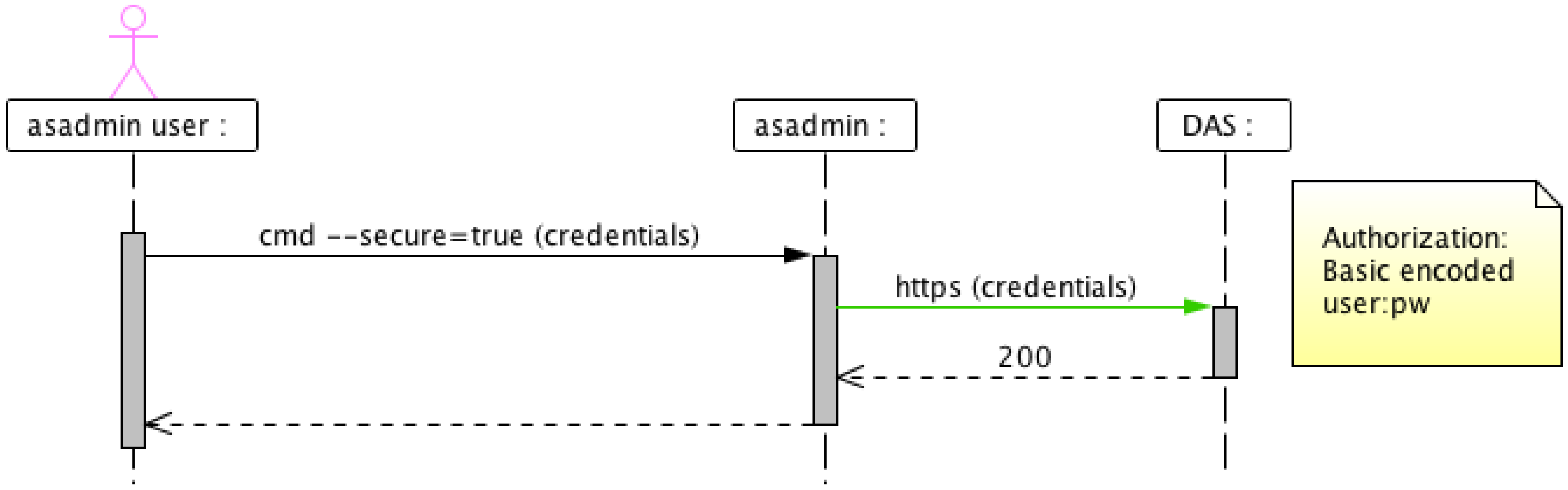
Questions

asadmin ↔ DAS --secure=false Encrypting Credentials



- User specifies credentials on command line
- asadmin withholds creds (connection is insecure)
- DAS insists on SSL, redirects to https
- asadmin follows redirection, sends credentials with resent request (connection is now secure)

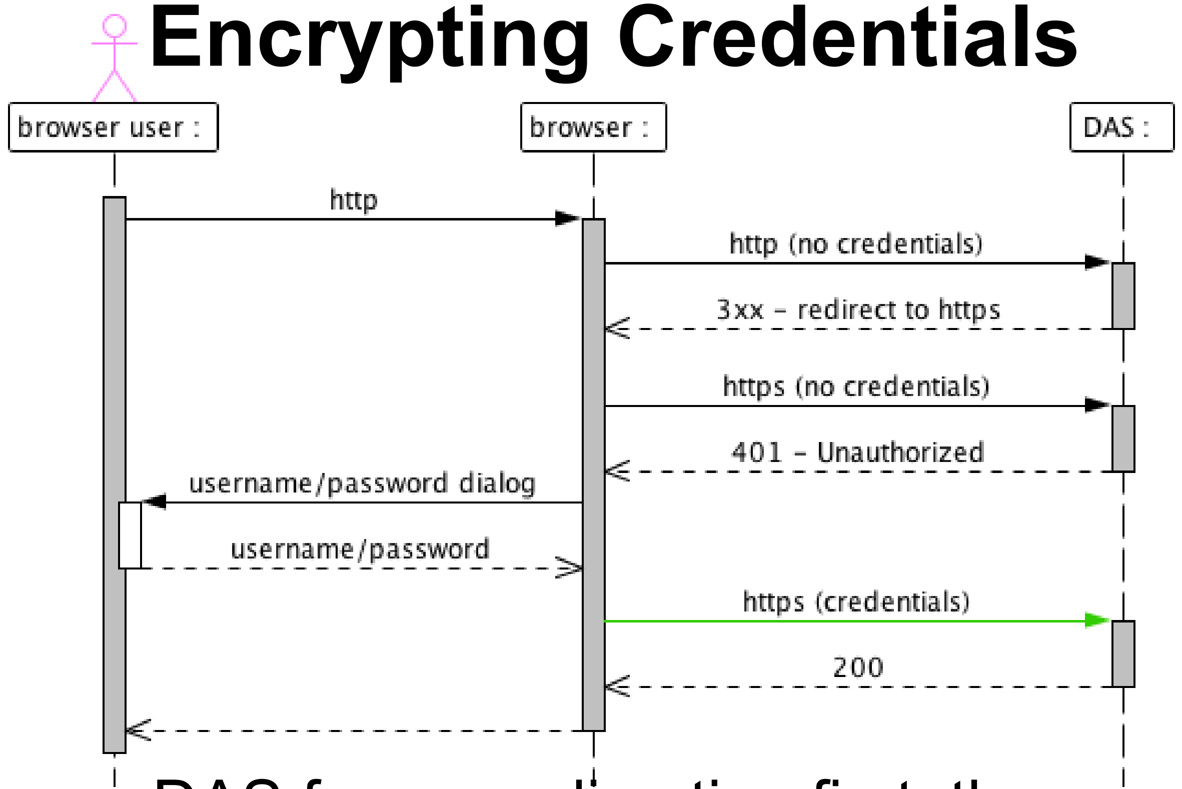
asadmin ↔ DAS --secure=true Encrypting Credentials



asadmin user :
asadmin :
DAS :
cmd --secure=true (credentials)
https (credentials)
200
Authorization: Basic encoded user:pw

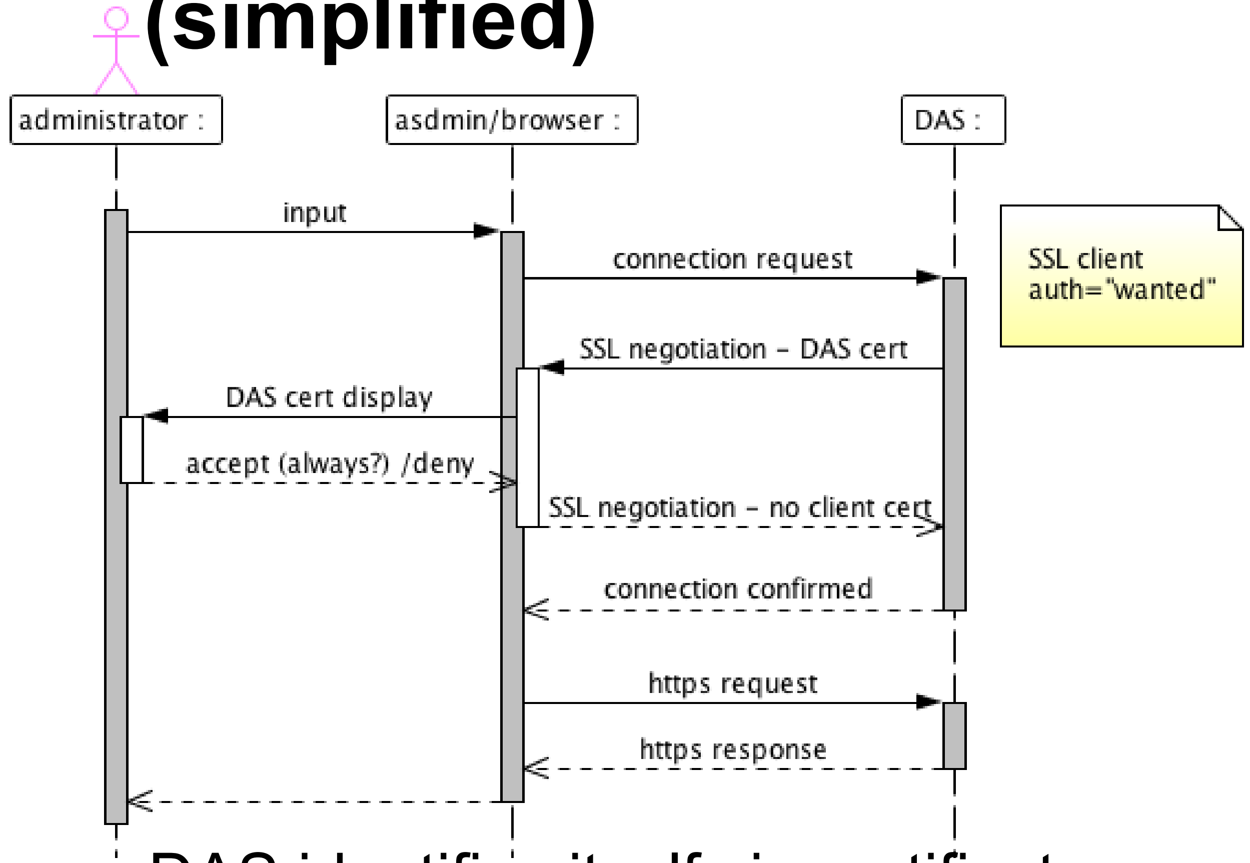
- User specifies credentials *and* secure connection
- asadmin initiates https itself, sends creds on initial request

Browser ↔ DAS Encrypting Credentials



- DAS forces redirection first, then...
- ...browser follows redirection (still no credentials)...
- ...DAS challenges for credentials
- ...browser prompts for, collects, then sends creds

A Brief Aside: SSL negotiation (simplified)

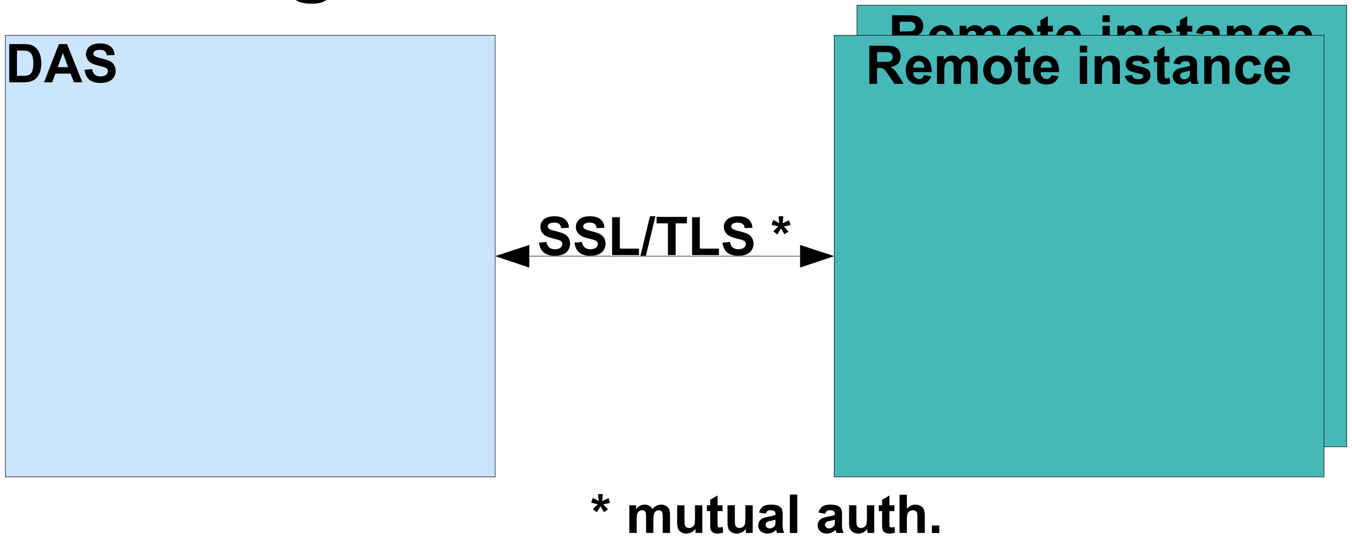


- DAS identifies itself via certificate
  - End-user accepts, perhaps "for always"
- Client *does not* typically identify using cert

DAS ↔ Instance
High-level requirements

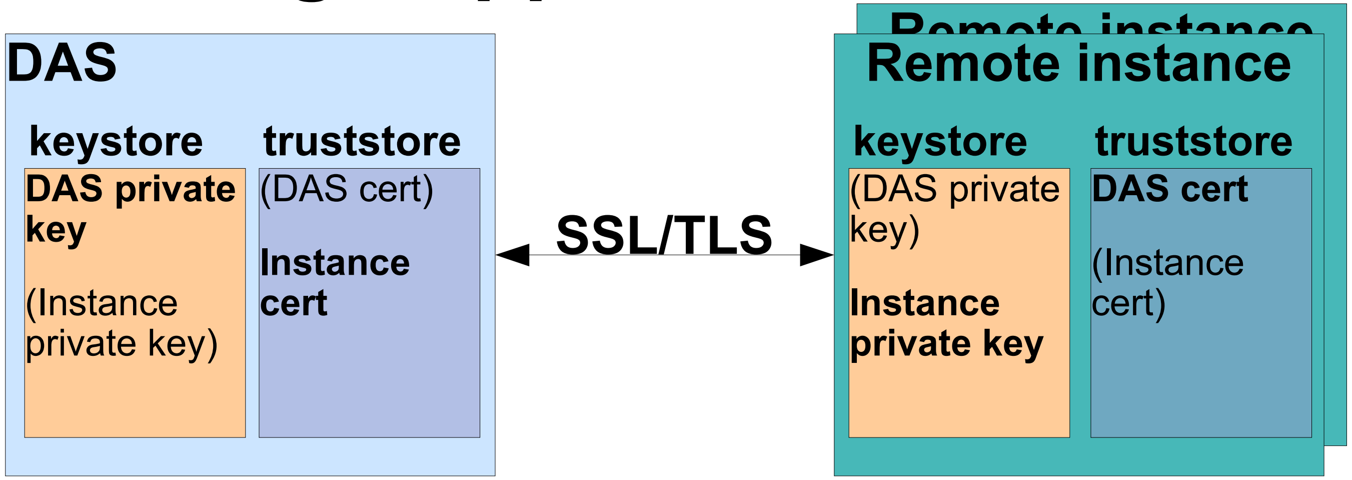
- Secure traffic between DAS, instances
- Do not store admin password in clear
- Help prevent rogue direct connections
  - Admin client ↔ instance
  - Instance ↔ instance
  - DAS ↔ DAS

DAS ↔ Instance Design Goals



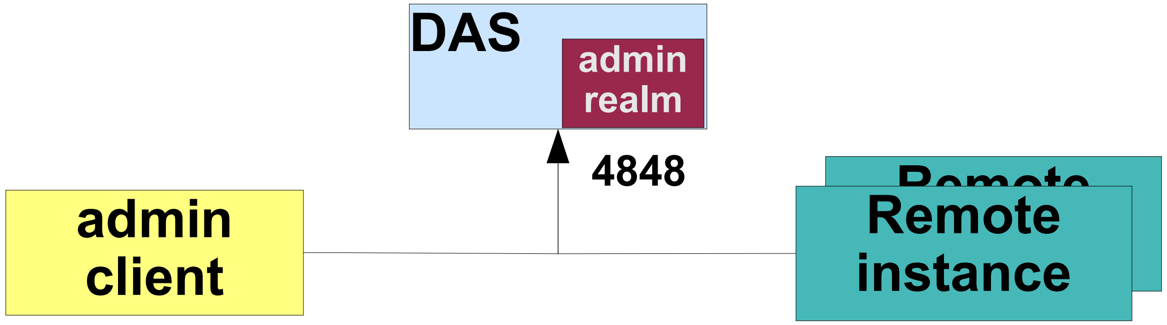
- SSL/TLS mutual authentication
- Cert-based, not username/password-based

DAS ↔ Instance Design Approach



- DAS, instance use copies of same keystore, truststore
  - Avoids problems with DAS → instance sync
- DAS authenticates w/ one cert, instances use one other
- DAS trusts instance cert, instance trusts DAS cert

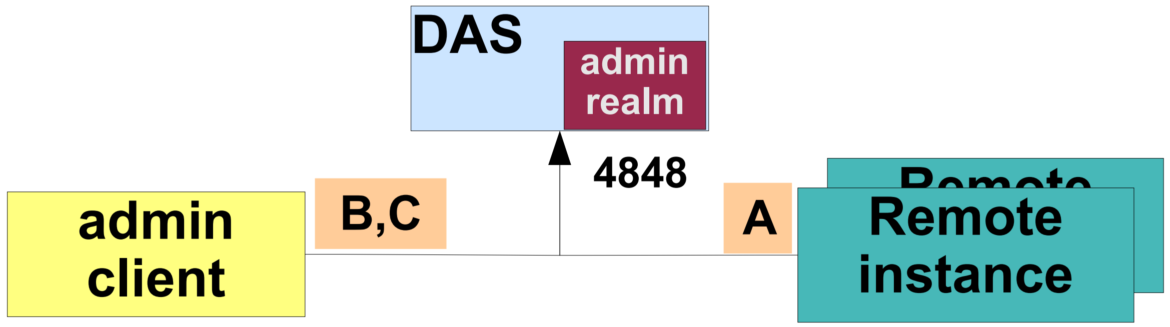
DAS ↔ Instance
One DAS admin port



Grizzly configuration

- Port unification – one port serves both http,https
- Redirection: http://das:4848 → https://das:4848
- SSL: client auth=”want” (not “need”)

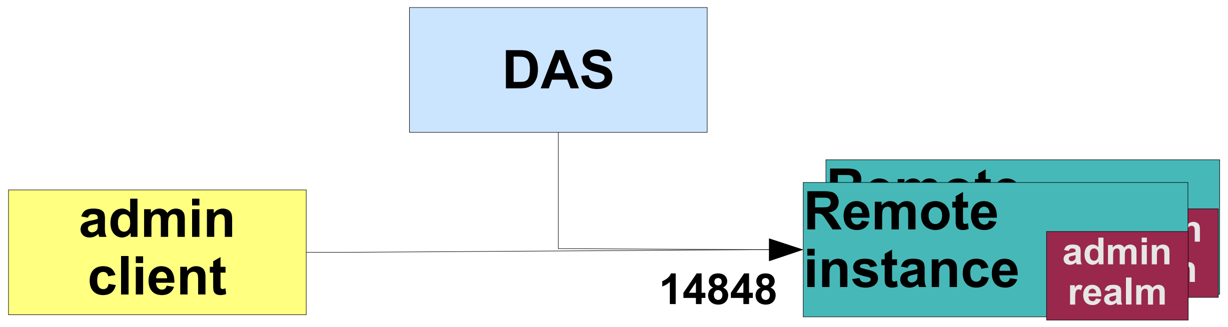
DAS AdminAdapter Logic



Accepts message if any one of the following is true:

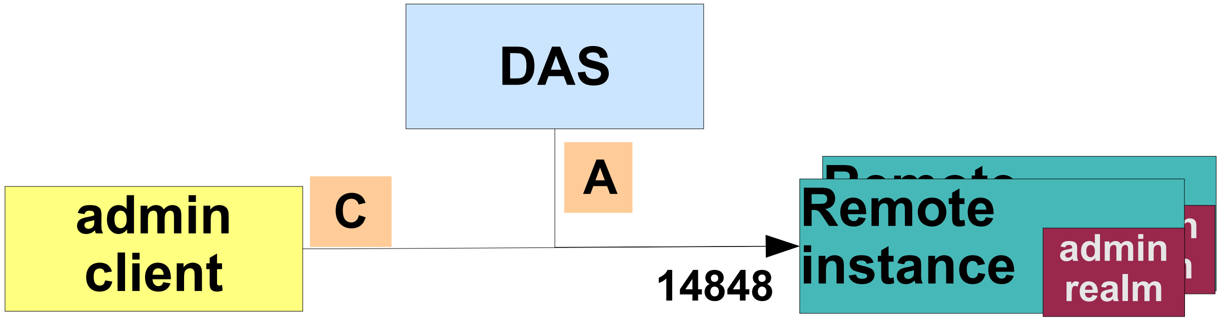
- **A**. Principal from request:
  non-null, != itself, in admin realm
- **B**. HTTP Authentication header specifies valid admin user/pw (issues challenge if header absent)
- **C**. Password == provisioned local password

DAS ↔ Instance
One Instance admin port



- Grizzly configuration: ***exactly*** as on DAS
  - Uses copies of same keystore, truststore
  - Port unification, redirection, client auth=”want”

Instance AdminAdapter Logic



Accepts message if any one of the following is true:

- **A**. Principal from request:
  non-null, != itself, in admin realm
- **B**. ~~HTTP Authentication header specifies valid admin user/pw (issues challenge if header absent)~~
- **C**. Password == provisioned local password

Authentication Summary

This ↓	Authenticates to		
	Any Client	**DAS**	**Instance**
Any Client		username/pw	XX
Local asadmin		username/pw; local password (if on DAS host)	local password
DAS	SSL server auth	X	SSL mutual auth
Instance	SSL server auth	SSL mutual auth	X

rovisioned
password in HTTP authorization