GSS-Proxy:

Better privilege separation

Simo Sorce
Principal Software Engineer, Red Hat
February 2013

Outline

- Introduction to GSS-API
- Using GSS-API
- Introduction to GSS-Proxy
- GSS-Proxy components
- Kernel upcalls and GSS-Proxy
- Privilege separation and GSS-Proxy
- Automatic credential handling

Introduction to GSS-API

- GSS-API = Generic Security Service API
- Abstraction layer introduced to simplify use of Kerberos for client-to-server interaction by hiding low level kerberos API into a 'mechanism'.
  - Not limited to Kerberos.
  - Enables applications to use a consistent API with multiple authentication protocols, to set up communication channels
  - Also provides integrity (authentication/signing) and confidentiality (encryption/sealing) services.
  - If the underlying protocol allows it, also provides delegation capabilities
  - Analogous to Windows SSPI (and interoperable with it)

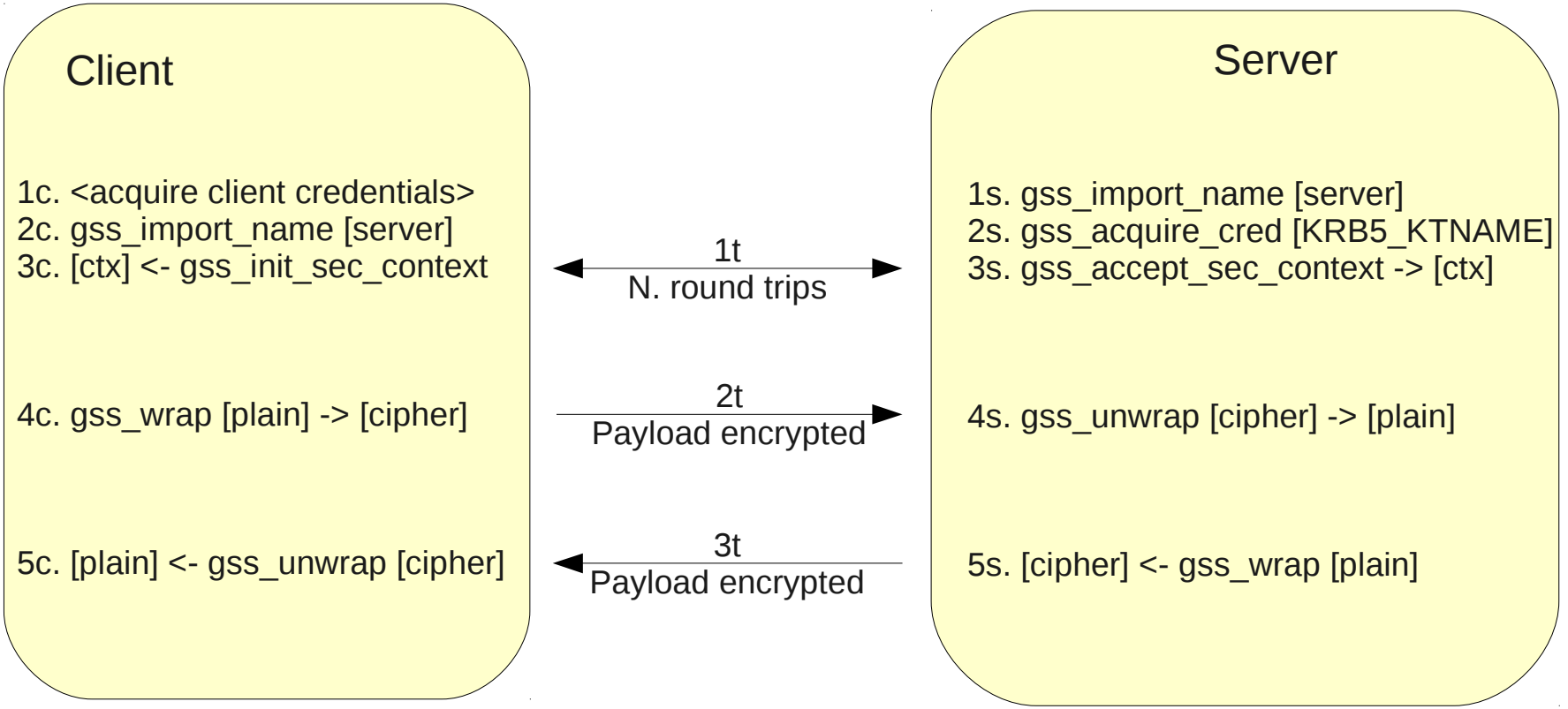
Applications using GSSAPI

- Enterprise applications that want to offer Single Sign On capabilities (generally through Kerberos)
- Examples:
  - LDAP/IMAP/SMTP/... + SASL/GSSAPI/Krb5
    - SASL = Simple Authentication and Security Layer
  - SSH + GSSAPI
    - GSSAPI/Krb5 used for auth only (also avail. Keyex patches)
  - HTTPS + SPNEGO
    - GSSAPI/SPNEGO/Krb5 or NTLMSSP)
  - NFS + RPCGSS (Secure NFS)
    - GSSAPI/Krb5

Using GSS-API

1. Acquisition of credentials
   - Generally 'default' credentials are used
     - in the krb5 case obtained via kinit (password or keytab)
2. Establishment of security context
   - gss_init_sec_context() / gss_accept_sec_context()
     - Depending on the underlying protocol multiple round trips may be used to complete context establishment.
3. Exchange of messages using security context
   - Messages can be signed and/or sealed using the established security context. eg. gss_wrap/gss_unwrap
4. Disposal of security context

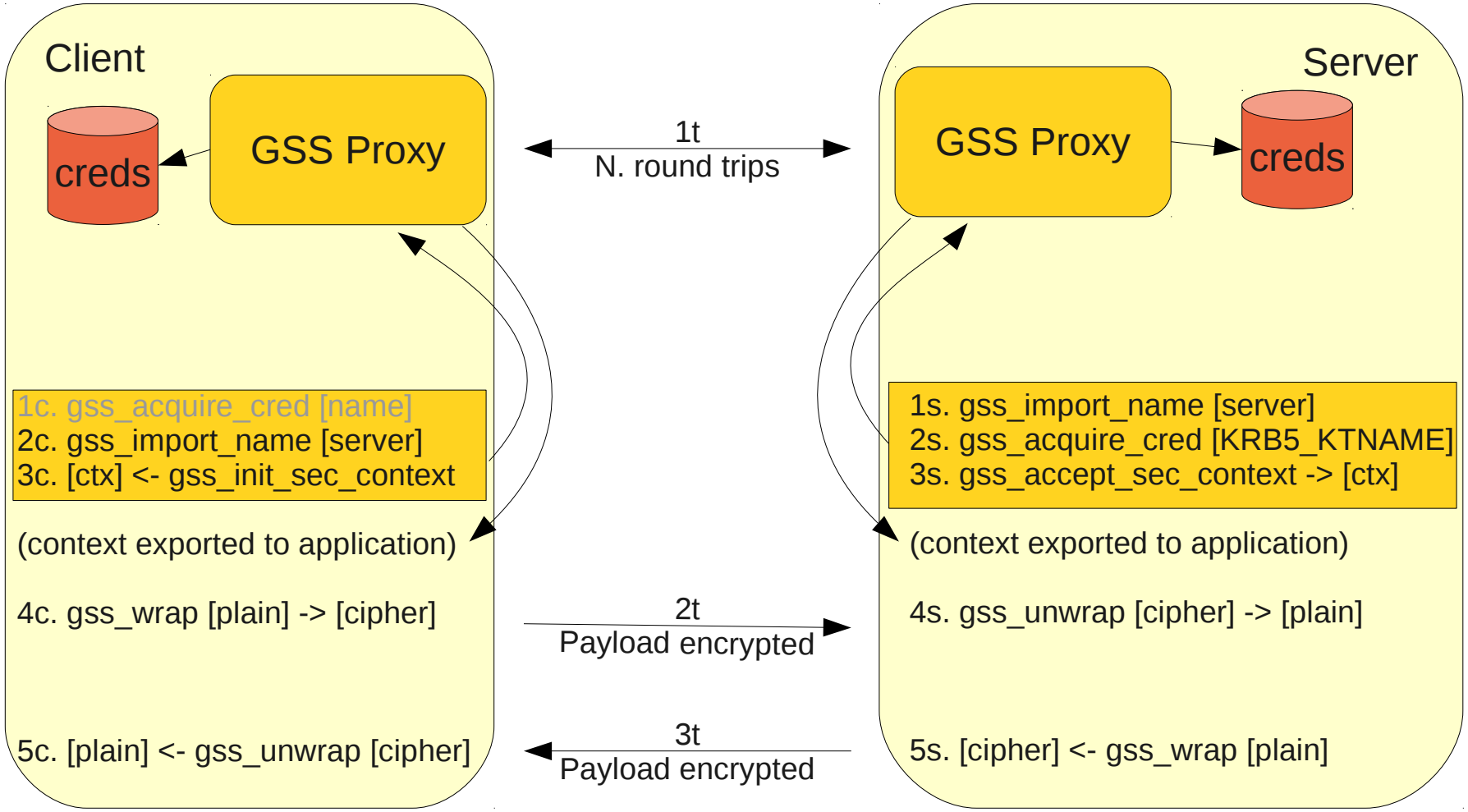
Connection using GSS-API



Why a GSS Proxy ?

- Standard GSS-API assumes direct access to credentials and long term keys by the application
  - A proxy allows to implement privilege separation
  - Application can use credentials w/o access to long term secret
- GSS-API is an extensive library and is not usable directly by the kernel
  - Allows to use the full GSS-API from the kernel by turning a local API into a local IPC
- Potential for developing an ssh agent
  - avoid full delegation of credentials
  - keep SSO working when jumping through multiple hosts

Connection using GSS-API with GSS-Proxy



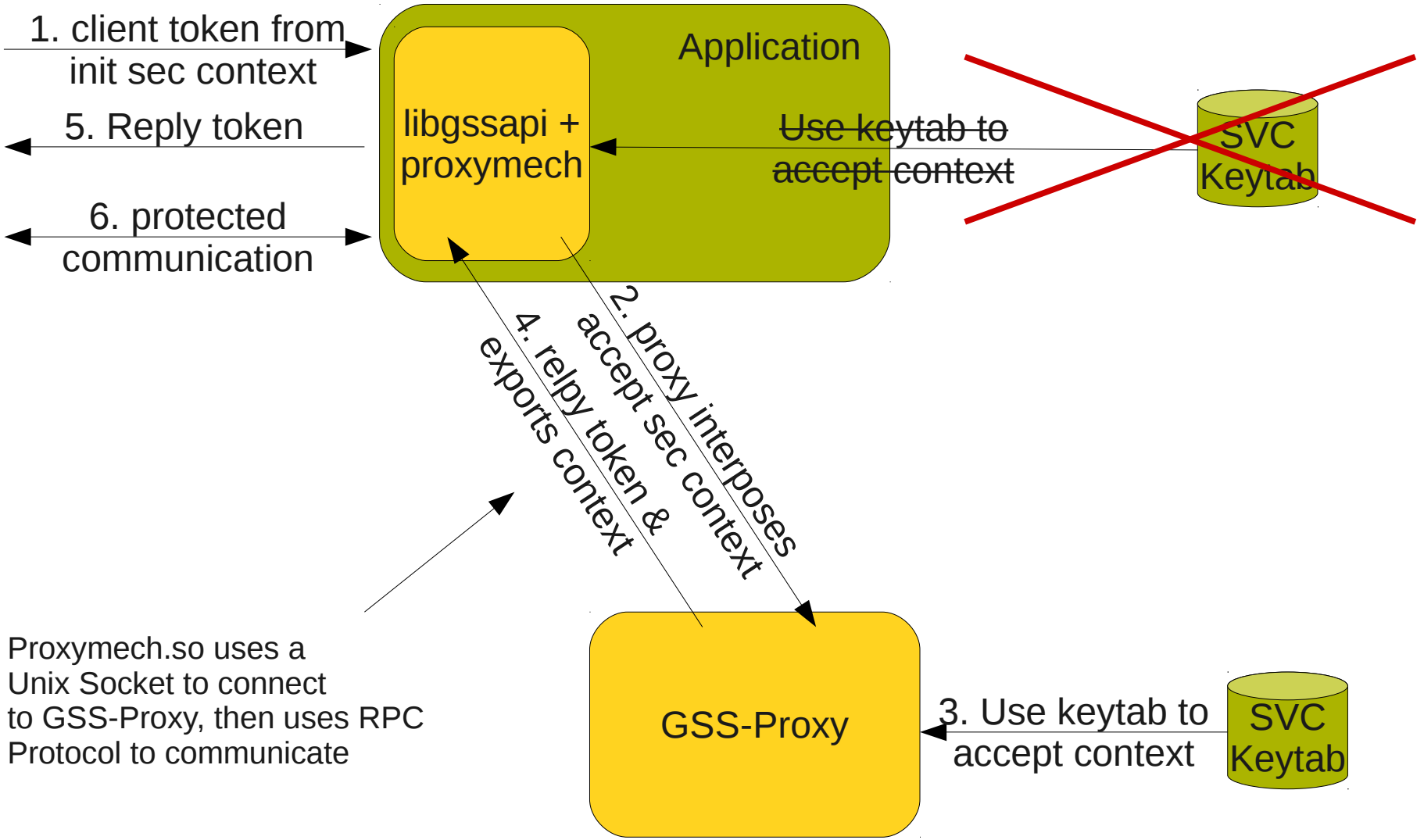
GSS-Proxy anatomy

- GSS Proxy is actually 3 things in one.
  - A service daemon
    - the 'gssproxy' binary - listens on unix sockets
    - A stateless, event driven server
  - A GSSAPI mechanism plugin (shared object)
    - proxymech.so - a gssapi 'interposer' mechanism
    - Requires special interposer plugin support (only in MIT 1.11)
  - A communication protocol
    - An XDR based RPC protocol (see gss_proxy.x file)
    - RPCs ops are compounded to reduce latency

Privilege separation

- For services that use keytabs to accept contexts
  - Keytab not available directly to the application
  - Proxymech.so intercepts KRB5 mechanism and proxies calls to GSS-Proxy
  - GSS-Proxy establishes the context on behalf of the application and then exports the context with only the session keys to the application
- If the application is compromised credentials can be used, but not stolen.
  - Multiple applications can use the same keytab w/o compromising each other
  - In future the GSS-Proxy can be augmented with policies that limit what the credentials can be used for.

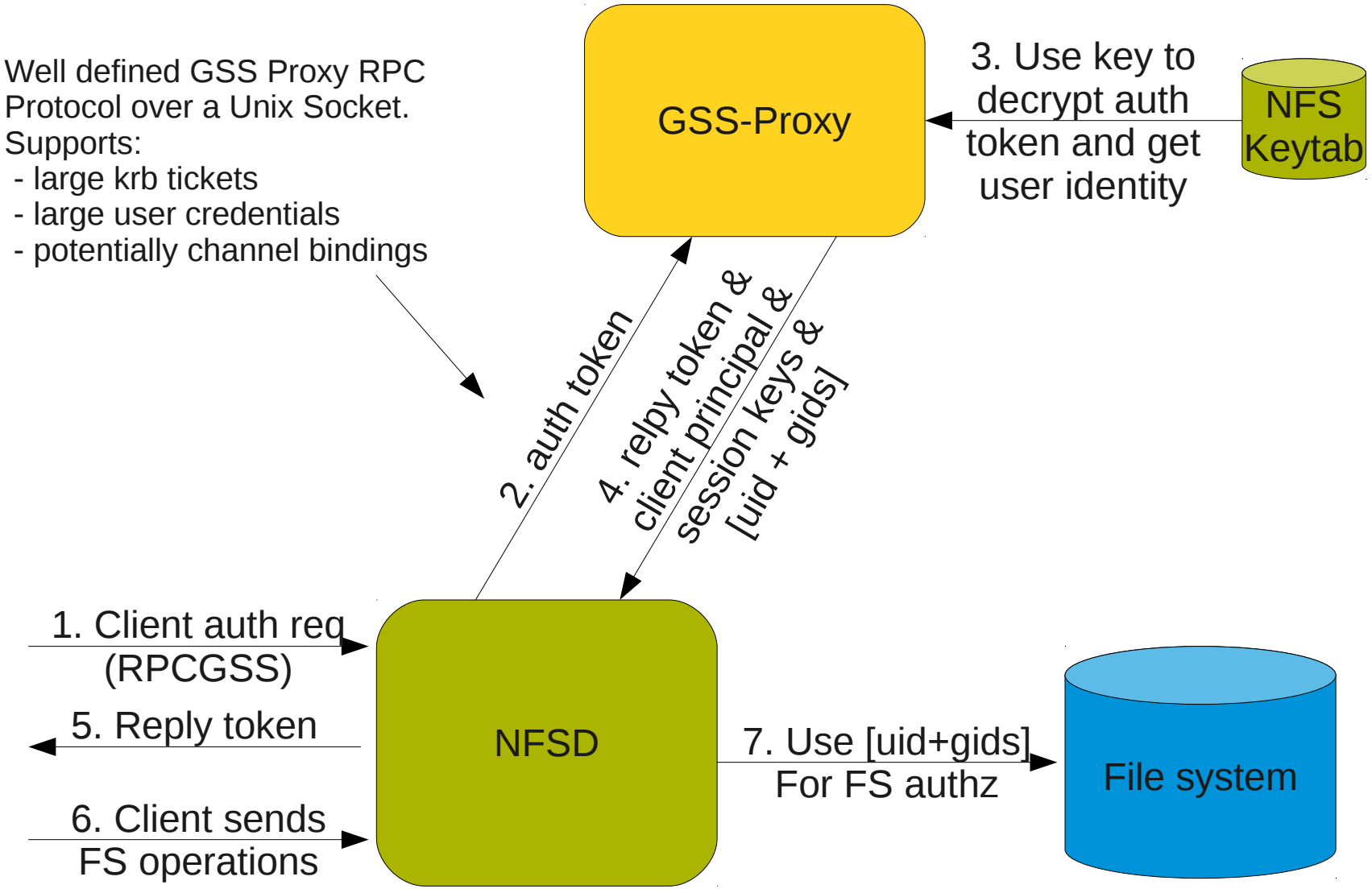
Privilege separation



Kernel upcall

- First prospect user of GSS Proxy: kernel NFSD
  - Current NFS server uses a bad hand crafted protocol for upcalls that is limited to less than a memory page (~ 2KiB)
  - Prevents context establishment with large tickets
    - such as when a large MS-PAC is attached to a ticket
  - Kernel patches have been created to let the kernel speak the GSS-Proxy protocol on a unix socket
    - Still not upstream due to minor integration issues caused by new support for containers
- The GSS Proxy establishes the security context
  - Exports a 'lucid' context to the kernel
  - Also sends user creds (uid + list of secondary gids)

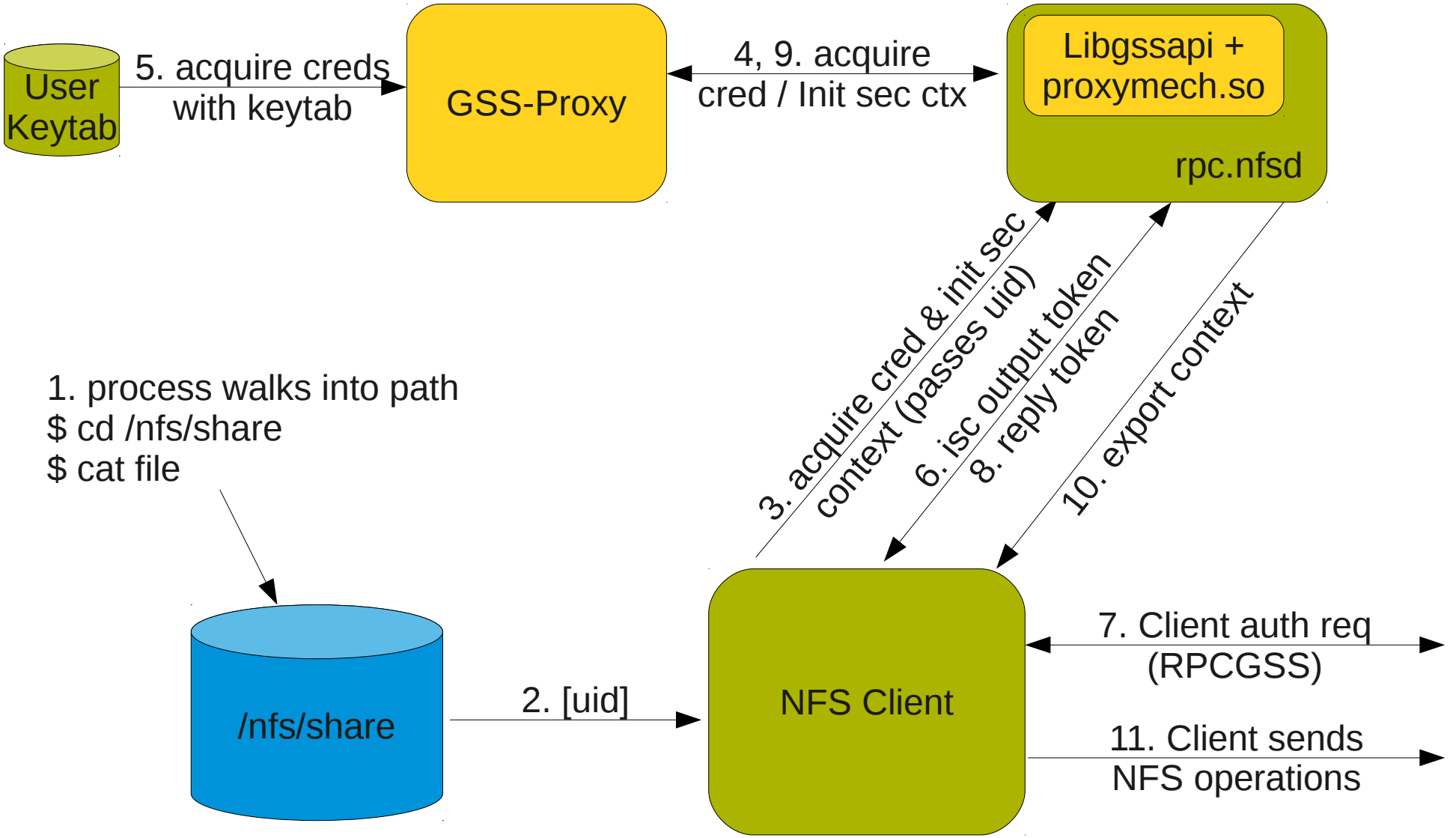
Kernel NFSD and GSS-Proxy



Automatic Credential handling

- The Secure NFS client case
  - Secure NFS relies on RPCGSS and Kerberos
  - A user needs krb5 credentials to access the NFS share
  - Some applications run as users but have no reason to use Kerberos outside of the need to access NFS
  - GSS-Proxy can use a keytab stored in a special area to acquire credentials on behalf of the application user so that Secure NFS access is allowed
  - Applications need no modification nor fragile cron jobs need to be created, process is transparent

NFS Client with GSS-Proxy provided autocred



Grab the PAC and run (more on priv. sep.)

- MS Active Directory attaches user credentials to krb5 tickets
  - PAC (Privilege Access Certificate)
  - The PAC is signed with the KDC and the SVC keys
- It is extremely useful to use this information
  - it is complete and avoids* the need to search info via LDAP
  - Pass the MS PAC to SSSD to prime its caches
- Problem: the receiving service can forge it.
  - The SVC signature is done with the SVC long term key
  - Potential for cache poisoning if the service is compromised
- GSS-Proxy is trusted
  - privilege separation prevents forgery from the service

Future possibilities

- GSS Agent
  - Current ssh+GSSAPI requires to export full credentials set to target host in order to use your krb5 creds there
    - Exposes TGT to the target machine
    - Still much better than sending your password
  - Like ssh-agent, the GSS-Proxy protocol could be used to only forward access to credentials
- Pros:
  - TGT remains on user machine
  - GSS-proxy forwards only session keys
  - No contamination of local target machine cred cache
- Cons:
  - Works only with pure GSSAPI applications, can't do direct krb5 calls

Call to action

- Please stop building applications that accept exclusively a “simple” user/password for authentication
  - Even (or especially) web apps
- It is very nice if you can support Kerberos SSO
  - Use GSSAPI, not Krb5 API directly
  - Alternatively use SASL (gives you PLAIN, GSSAPI, EXTERNAL, .., auth)
- For web applications:
  - use apache and mod_auth_kerb (RFC4559)
  - implement the RFC on your own.
  - use form based auth as a fallback.