Enterprise desktop: improving client side in the age of Samba AD and FreeIPA

Principal Software Engineer, Red Hat // Samba Team

Alexander Bokovoy

SambaXP'16

Enterprise desktop?

Centralized identity management system

There are now several free software identity management systems with the focus on managing operating systems' environments:

- Samba AD
- FreeIPA
- [many other LDAP+Kerberos based projects]

Enterprise desktop

- a client enrolled to a centralized identity management system
- a tool to solve business tasks
- a subject to centrally defined access controls

Enterprise desktop

Typical enterprise desktop includes agents for

- identity services: POSIX attributes for users and groups via NSSWITCH

Enterprise desktop

Typical enterprise desktop includes agents for

- identity services: POSIX attributes for users and groups via NSSWITCH
- authentication services: logon details, mostly via PAM interface

Enterprise desktop

Typical enterprise desktop includes agents for

- identity services: POSIX attributes for users and groups via NSSWITCH
- authentication services: logon details, mostly via PAM interface
- authorization services: mostly via PAM interface or application-specific ones

Practical case: Fedora and FreeIPA

FreeIPA client defaults to use SSSD as an agent

- `nss_sss` is referenced in `/etc/nsswitch.conf` on Fedora by default

Practical case: Fedora and FreeIPA

FreeIPA client defaults to use SSSD as an agent

- `nss_sss` is referenced in `/etc/nsswitch.conf` on Fedora by default
- `pam_sss` use is configured at the enrollment time for `system-auth` PAM service which is included to all PAM configurations

Practical case: Fedora and FreeIPA

FreeIPA client defaults to use SSSD as an agent

- `nss_sss` is referenced in `/etc/nsswitch.conf` on Fedora by default
- `pam_sss` use is configured at the enrollment time for `system-auth` PAM service which is included to all PAM configurations
- SSSD performs host-based access control to PAM services using rules stored centrally in FreeIPA

Practical case: Fedora and FreeIPA

FreeIPA client defaults to use SSSD as an agent

- `nss_sss` is referenced in `/etc/nsswitch.conf` on Fedora by default
- `pam_sss` use is configured at the enrollment time for `system-auth` PAM service which is included to all PAM configurations
- SSSD performs host-based access control to PAM services using rules stored centrally in FreeIPA
- OpenSSH is configured to look up public keys for users and hosts via SSSD

Practical case: Fedora and FreeIPA

FreeIPA client defaults to use SSSD as an agent

- `nss_sss` is referenced in `/etc/nsswitch.conf` on Fedora by default
- `pam_sss` use is configured at the enrollment time for `system-auth` PAM service which is included to all PAM configurations
- SSSD performs host-based access control to PAM services using rules stored centrally in FreeIPA
- OpenSSH is configured to look up public keys for users and hosts via SSSD
- SUDO is configured to look up SUDO rules in FreeIPA via SSSD

Practical case: Fedora and FreeIPA

FreeIPA client defaults to use SSSD as an agent

- `nss_sss` is referenced in `/etc/nsswitch.conf` on Fedora by default
- `pam_sss` use is configured at the enrollment time for `system-auth` PAM service which is included to all PAM configurations
- SSSD performs host-based access control to PAM services using rules stored centrally in FreeIPA
- OpenSSH is configured to look up public keys for users and hosts via SSSD
- SUDO is configured to look up SUDO rules in FreeIPA via SSSD
- automount can be configured to use SSSD to deliver the mount maps

Practical case: Fedora and FreeIPA

FreeIPA client defaults to use SSSD as an agent

- `nss_sss` is referenced in `/etc/nsswitch.conf` on Fedora by default
- `pam_sss` use is configured at the enrollment time for `system-auth` PAM service which is included to all PAM configurations
- SSSD performs host-based access control to PAM services using rules stored centrally in FreeIPA
- OpenSSH is configured to look up public keys for users and hosts via SSSD
- SUDO is configured to look up SUDO rules in FreeIPA via SSSD
- automount can be configured to use SSSD to deliver the mount maps
- SSSD provides locator and localauth plugins to MIT Kerberos to discover domain controllers and map Kerberos principals to POSIX user names for trust operations

Practical case: Fedora and FreeIPA

FreeIPA client defaults to use SSSD as an agent

- `nss_sss` is referenced in `/etc/nsswitch.conf` on Fedora by default
- `pam_sss` use is configured at the enrollment time for `system-auth` PAM service which is included to all PAM configurations
- SSSD performs host-based access control to PAM services using rules stored centrally in FreeIPA
- OpenSSH is configured to look up public keys for users and hosts via SSSD
- SUDO is configured to look up SUDO rules in FreeIPA via SSSD
- automount can be configured to use SSSD to deliver the mount maps
- SSSD provides locator and localauth plugins to MIT Kerberos to discover domain controllers and map Kerberos principals to POSIX user names for trust operations
- SSSD supports offline logon, logon with smart cards, logon with Kerberos proxy

Practical case: Samba AD domain member

- A pure winbindd-based setup or a hybrid SSSD+winbindd configuration

Practical case: Samba AD domain member

- A pure winbindd-based setup or a hybrid SSSD+winbindd configuration
- Pure winbindd: `nss_winbind` and `pam_winbind` for identity and authentication

Practical case: Samba AD domain member

- A pure winbindd-based setup or a hybrid SSSD+winbindd configuration
- Pure winbindd: `nss_winbind` and `pam_winbind` for identity and authentication
- Pure winbindd: Limited authorization capabilities (account lock)

Practical case: Samba AD domain member

- A pure winbindd-based setup or a hybrid SSSD+winbindd configuration
- Pure winbindd: `nss_winbind` and `pam_winbind` for identity and authentication
- Pure winbindd: Limited authorization capabilities (account lock)
- Hybrid approach:

Practical case: Samba AD domain member

- A pure winbindd-based setup or a hybrid SSSD+winbindd configuration
- Pure winbindd: `nss_winbind` and `pam_winbind` for identity and authentication
- Pure winbindd: Limited authorization capabilities (account lock)
- Hybrid approach:
- `nss_sss` and `pam_sss` for identity and authentication

Practical case: Samba AD domain member

- A pure winbindd-based setup or a hybrid SSSD+winbindd configuration
- Pure winbindd: `nss_winbind` and `pam_winbind` for identity and authentication
- Pure winbindd: Limited authorization capabilities (account lock)
- Hybrid approach:
- `nss_sss` and `pam_sss` for identity and authentication
- winbindd is used by Samba, configured to trust SSSD-provided ID range

Practical case: Samba AD domain member

- A pure winbindd-based setup or a hybrid SSSD+winbindd configuration
- Pure winbindd: `nss_winbind` and `pam_winbind` for identity and authentication
- Pure winbindd: Limited authorization capabilities (account lock)
- Hybrid approach:
- `nss_sss` and `pam_sss` for identity and authentication
- winbindd is used by Samba, configured to trust SSSD-provided ID range
- SSSD supports offline logon, logon with smart cards, logon with Kerberos proxy

Practical case: Samba AD domain member

- A pure winbindd-based setup or a hybrid SSSD+winbindd configuration
- Pure winbindd: `nss_winbind` and `pam_winbind` for identity and authentication
- Pure winbindd: Limited authorization capabilities (account lock)
- Hybrid approach:
- `nss_sss` and `pam_sss` for identity and authentication
- winbindd is used by Samba, configured to trust SSSD-provided ID range
- SSSD supports offline logon, logon with smart cards, logon with Kerberos proxy
- SSSD does authorization using GPO and/or account lock

That’s all behind the scenes, what would user see?

How *enterprisey* are we?

Let’s score by a password

Let’s score by a password

A typical workflow for every laptop reboot

1. Sign into a local system account (enter a password)

Let’s score by a password

A typical workflow for every laptop reboot

1. Sign into a local system account (enter a password)
2. Jump onto virtual private network (enter a password or more)

Let’s score by a password

A typical workflow for every laptop reboot

1. Sign into a local system account (enter a password)
2. Jump onto virtual private network (enter a password or more)
3. Obtain initial Kerberos credentials (enter a password)

Let’s score by a password

A typical workflow for every laptop reboot

1. Sign into a local system account (enter a password)
2. Jump onto virtual private network (enter a password or more)
3. Obtain initial Kerberos credentials (enter a password)
4. Use corporate applications (enter a password?)

Can we do better than this?

how far are we from

- Sign into a corporate environment
- Use corporate applications

?

Let’s try to login!

Demo (first 40 seconds):
`http://talks.vda.li/2016/05/SambaXP/freeipa-logon-1FA-2FA.webm`

What was that?

- The system is configured to be a client for FreeIPA

What was that?

- The system is configured to be a client for FreeIPA
- SSSD handles login and Kerberos keys

What was that?

- The system is configured to be a client for FreeIPA
- SSSD handles login and Kerberos keys
- Login to the system is verified over public network using a proxy for Kerberos protocol

What was that?

- The system is configured to be a client for FreeIPA
- SSSD handles login and Kerberos keys
- Login to the system is verified over public network using a proxy for Kerberos protocol
- Established VPN connection based on Kerberos ticket

What was that?

- The system is configured to be a client for FreeIPA
- SSSD handles login and Kerberos keys
- Login to the system is verified over public network using a proxy for Kerberos protocol
- Established VPN connection based on Kerberos ticket
- **Credentials were entered only once**

Kerberos proxy

Available on the client side with Microsoft Active Directory and MIT Kerberos 1.13

- protocol is called MS-KKDCP
- transparent for Kerberos library users

Kerberos proxy is implemented by FreeIPA 4.2, OpenConnect Server 7.05, and as a standalone server

- Requires HTTPS connection, set up by default in FreeIPA 4.2, very easy to use (one line change on the client)
- Allows to obtain tickets from anywhere
- SSSD 1.12+
- Real-life example: GNOME project uses KDC proxy to allow GSSAPI authentication in SSH for GNOME developers

VPN and Kerberos

OpenConnect client supports GSSAPI negotiation

- Fedora 22+ works out of the box

OpenVPN does not support GSSAPI negotiation

- to do since 2005, ignored by upstream

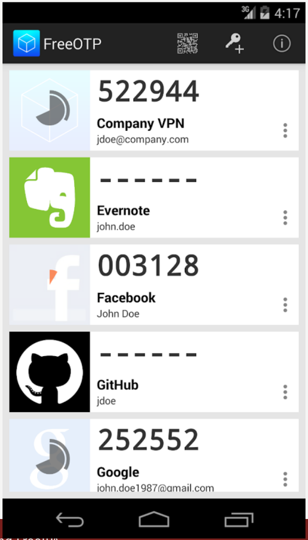
Support for GSSAPI in IPSEC is coming

Two-factor authentication

FreeIPA 4.x supports 2FA natively

- Yubikey, FreeOTP client for Android and iOS, any HOTP/TOTP compatible software and hardware
- Two-factor authentication is enforced on Kerberos level
- Performs pre-authentication before issuing a ticket
- Authentication Indicators are in Kerberos 1.14
- Pre-authentication modules can say *how* tickets were issued

FreeOTP: Android and iOS



522944
Company VPN
jdoe@company.com

Evernote
john.doe

003128
Facebook
John Doe

GitHub
jdoe

252552
Google
john.doe1987@gmail.com

Demo

Demo (starting from 40s):
`http://talks.vda.li/2016/05/SambaXP/freeipa-logon-1FA-2FA.webm`

What was that?

1. One time password token was programmed to Yubikey and added for the user in FreeIPA

What was that?

1. One time password token was programmed to Yubikey and added for the user in FreeIPA
2. SSSD handles login and notices OTP pre-authentication support in Kerberos conversation

What was that?

1. One time password token was programmed to Yubikey and added for the user in FreeIPA
2. SSSD handles login and notices OTP pre-authentication support in Kerberos conversation
3. Login to the system is verified over public network using a proxy for Kerberos protocol

What was that?

1. One time password token was programmed to Yubikey and added for the user in FreeIPA
2. SSSD handles login and notices OTP pre-authentication support in Kerberos conversation
3. Login to the system is verified over public network using a proxy for Kerberos protocol
4. Kerberos ticket is obtained, first factor is provided by SSSD to GDM for unlocking GNOME passwords and keys storage (SeaHorse)

What was that?

1. One time password token was programmed to Yubikey and added for the user in FreeIPA
2. SSSD handles login and notices OTP pre-authentication support in Kerberos conversation
3. Login to the system is verified over public network using a proxy for Kerberos protocol
4. Kerberos ticket is obtained, first factor is provided by SSSD to GDM for unlocking GNOME passwords and keys storage (SeaHorse)
5. **Credentials were entered only once**

If Kerberos credentials are available, what can we do with them?

- Authenticate with GSSAPI against almost anything

If Kerberos credentials are available, what can we do with them?

- Authenticate with GSSAPI against almost anything
- Obtain SAML assertion for other web services (and more)

If Kerberos credentials are available, what can we do with them?

- Authenticate with GSSAPI against almost anything
- Obtain SAML assertion for other web services (and more)
- Use to access networking file systems

If Kerberos credentials are available, what can we do with them?

- Authenticate with GSSAPI against almost anything
- Obtain SAML assertion for other web services (and more)
- Use to access networking file systems
- Display properties of the available tickets

If Kerberos credentials are available, what can we do with them?

- Authenticate with GSSAPI against almost anything
- Obtain SAML assertion for other web services (and more)
- Use to access networking file systems
- Display properties of the available tickets
- Renew the ticket granting ticket (TGT)

If Kerberos credentials are available, what can we do with them?

- Authenticate with GSSAPI against almost anything
- Obtain SAML assertion for other web services (and more)
- Use to access networking file systems
- Display properties of the available tickets
- Renew the ticket granting ticket (TGT)
- Choose which Kerberos principal is in use

Authenticate with GSSAPI

Epiphany, the GNOME Web Browser, in GNOME 3.18:

- GSSAPI support is no more, depends on libsoup support

Authenticate with GSSAPI

Epiphany, the GNOME Web Browser, in GNOME 3.18:

- GSSAPI support is no more, depends on libsoup support
- libsoup has been dragging since 2009, bug #587145

Authenticate with GSSAPI

Epiphany, the GNOME Web Browser, in GNOME 3.18:

- GSSAPI support is no more, depends on libsoup support
- libsoup has been dragging since 2009, bug #587145
- WebkitGtk is unusable for SAML/OAuth2 interactions involving Kerberos

Authenticate with GSSAPI

Epiphany, the GNOME Web Browser, in GNOME 3.18:

- GSSAPI support is no more, depends on libsoup support
- libsoup has been dragging since 2009, bug #587145
- WebkitGtk is unusable for SAML/OAuth2 interactions involving Kerberos
- One cannot use Google apps with GSSAPI in Gnome Online Accounts

Authenticate with GSSAPI

Epiphany, the GNOME Web Browser, in GNOME 3.18:

- GSSAPI support is no more, depends on libsoup support
- libsoup has been dragging since 2009, bug #587145
- WebkitGtk is unusable for SAML/OAuth2 interactions involving Kerberos
- One cannot use Google apps with GSSAPI in Gnome Online Accounts
- No single sign-on with GSSAPI from GNOME applications using WebkitGtk to authenticate

Can we do better than this?

Demo of single sign-on in Epiphany

Demo (until 1m42s): `http://talks.vda.li/2016/05/SambaXP/freeipa-ipsilon-googleapps-signon.webm`

What was that?

Tomáš Popela (Red Hat), David Woodhouse (Intel), and Guido Guenther (Debian) worked to fix `libsoup` and `WebkitGtk`

We logged into my FreeIPA server's Web UI

The code is in GNOME 3.20 (March 2016) and is in Fedora 24 beta (released on May 10th)

By default, all HTTPS sites advertising `WWW-Authenticate: Negotiate` authentication method will be probed with GSSAPI

Why all this is important?

- Multi-factor authentication moves to Web-based flows (Azure, Windows 10)
- Corporate portals are used to authenticate when accessing external resources
- Captive portals prevent Internet access before logon
- You need to be able to be on VPN before logon to your system or have Kerberos proxy working, or have multi-factor authentication working (full circle loop now)
- WebkitGTK+ is embedded in many applications

Diversion: Running a browser before logon?

- Yes, effectively, a sandbox with a locked-down web engine

Down the rabbit hole...

Diversion: Running a browser before logon?

- Yes, effectively, a sandbox with a locked-down web engine
- But network profile (access point) needs to be selected first

Down the rabbit hole...

Diversion: Running a browser before logon?

- Yes, effectively, a sandbox with a locked-down web engine
- But network profile (access point) needs to be selected first
- This means Network Manager has to run before logon

Down the rabbit hole...

Diversion: Running a browser before logon?

- Yes, effectively, a sandbox with a locked-down web engine
- But network profile (access point) needs to be selected first
- This means Network Manager has to run before logon
- This means Network Manager needs to access user-specific data before logon

Down the rabbit hole...

Diversion: Running a browser before logon?

- Yes, effectively, a sandbox with a locked-down web engine
- But network profile (access point) needs to be selected first
- This means Network Manager has to run before logon
- This means Network Manager needs to access user-specific data before logon
- A complete re-arrangement of logon UX

Down the rabbit hole...

Ok, anything for users, not admins?

Single sign-on to Google Apps

Demo (starting from 1m42s): `http://talks.vda.li/2016/05/SambaXP/freeipa-ipsilon-googleapps-signon.webm`

What was that?

- Ipsilon as an Identity Provider (IdP) taking user information from FreeIPA

What was that?

- Ipsilon as an Identity Provider (IdP) taking user information from FreeIPA
- Ipsilon project: `http://ipsilon-project.org/`

What was that?

- Ipsilon as an Identity Provider (IdP) taking user information from FreeIPA
- Ipsilon project: `http://ipsilon-project.org/`
- Google Apps as a Service Provider (SP) talking to FreeIPA via Ipsilon's IdP

What was that?

- Ipsilon as an Identity Provider (IdP) taking user information from FreeIPA
- Ipsilon project: `http://ipsilon-project.org/`
- Google Apps as a Service Provider (SP) talking to FreeIPA via Ipsilon's IdP
- Users from FreeIPA can logon to Google Apps if admin did pre-create accounts for them

What was that?

- Ipsilon as an Identity Provider (IdP) taking user information from FreeIPA
- Ipsilon project: `http://ipsilon-project.org/`
- Google Apps as a Service Provider (SP) talking to FreeIPA via Ipsilon's IdP
- Users from FreeIPA can logon to Google Apps if admin did pre-create accounts for them
- At no point Google has access to FreeIPA users' credentials

What was that?

- Ipsilon as an Identity Provider (IdP) taking user information from FreeIPA
- Ipsilon project: `http://ipsilon-project.org/`
- Google Apps as a Service Provider (SP) talking to FreeIPA via Ipsilon's IdP
- Users from FreeIPA can logon to Google Apps if admin did pre-create accounts for them
- At no point Google has access to FreeIPA users' credentials
- SSSD is the key component here, the same setup will work for a SSSD enrolled to Samba AD

What was that?

- Ipsilon as an Identity Provider (IdP) taking user information from FreeIPA
- Ipsilon project: `http://ipsilon-project.org/`
- Google Apps as a Service Provider (SP) talking to FreeIPA via Ipsilon's IdP
- Users from FreeIPA can logon to Google Apps if admin did pre-create accounts for them
- At no point Google has access to FreeIPA users' credentials
- SSSD is the key component here, the same setup will work for a SSSD enrolled to Samba AD
- GNOME Online Accounts now configured to access Google Apps' services

What was that?

- Ipsilon as an Identity Provider (IdP) taking user information from FreeIPA
- Ipsilon project: `http://ipsilon-project.org/`
- Google Apps as a Service Provider (SP) talking to FreeIPA via Ipsilon's IdP
- Users from FreeIPA can logon to Google Apps if admin did pre-create accounts for them
- At no point Google has access to FreeIPA users' credentials
- SSSD is the key component here, the same setup will work for a SSSD enrolled to Samba AD
- GNOME Online Accounts now configured to access Google Apps' services
- Setup recipe:
  `https://ipsilon-project.org/doc/example/google-apps.html`

What does GSSAPI support open for use in GNOME Online Accounts?

- Single sign-on is the primary feature

What does GSSAPI support open for use in GNOME Online Accounts?

- Single sign-on is the primary feature
- Automated credentials renewal

What does GSSAPI support open for use in GNOME Online Accounts?

- Single sign-on is the primary feature
- Automated credentials renewal
- Automated token/assertion renewal for SAML/OpenID

What does GSSAPI support open for use in GNOME Online Accounts?

- Single sign-on is the primary feature
- Automated credentials renewal
- Automated token/assertion renewal for SAML/OpenID
- No need to store passwords locally (secure kiosks?)

Visualize

GNOME Online Accounts could show Kerberos ticket properties

- Ticket time validity, flags (forward, renewal)
- Authentication indicators
- Existing service tickets in the credentials cache and allow to remove them selectively
- Allow automatic ticket renewal if KDC permits it

Consume

Choose between different Kerberos principals

- MIT Kerberos supports kernel keyring (1.12+) and directory-based (1.11+) storage of credentials
- Multiple Kerberos principals can be stored and used at the same time
- Only a single principal can be defined as “primary” for each Kerberos realm in the collection of credentials

Kerberos ticket renewal

- SSSD supports automatic Kerberos ticket renewal for single factor cases
  - Renewing 2FA tickets requires UI interaction triggered by expiry time
  - Automatic ticket renewal requires permission from KDC, visible as a ticket flag
- GNOME Online Accounts could integrate with SSSD in prompting for credentials (multiple factors) as in 2FA case needed information could be provided via SSSD InfoPipe/AuthPipe

Better Kerberos in browsers

- Firefox Kerberos setup isn't nice
  - needs about:config manipulation
  - DNS domains associated with Kerberos realm could be discovered via DNS SRV records, prompted for confirmation once
- FreeIPA used to provide an extension to automate Firefox setup
  - Extension was generated locally for for each FreeIPA deployment to provide configuration details
  - not anymore: Firefox removed ability to provide non-publicly available extensions since version 43
- There are about dozen bugs related to GSSAPI support in Firefox, gradually being fixed by Red Hat Firefox team together with Firefox upstream

Better Kerberos in browsers

- Chromium/Chrome
  - Have bugs for processing of `WWW-Authenticate: Negotiate` when Kerberos credentials are not available
  - On Linux only allows to configure Kerberos use through command line or statically system-wide, poor user experience
- A fixed libsoup/WebkitGtk allows to always use GSSAPI if server advertises `WWW-Authenticate: Negotiate` over HTTPS
  - no need to configure anything in Epiphany
  - could be further confined with a user confirmation similar to how passwords are managed at the first logon
- Konqueror browser in KDE allows to always use GSSAPI if server advertises `WWW-Authenticate: Negotiate` over HTTPS

Better Kerberos in browsers

- GSSAPI flow is synchronous, needs better UI interaction to avoid hogging down other tabs
  - still major issue for many browsers
  - Bug #890908 is on the way to be fixed in Firefox
    `https://bugzilla.mozilla.org/show_bug.cgi?id=890908`
- But asynchronous GSSAPI flow would do wonders too!

Any *practical* use of it?

Single sign-on at home

Demo: `http://talks.vda.li/2016/05/SambaXP/freeipa-ipsilon-owncloud-signon.webm`

What was that?

- I set up Ipsilon to authenticate against my FreeIPA server

What was that?

- I set up Ipsilon to authenticate against my FreeIPA server
- I set up Owncloud instance and created a simple application to do login via Ipsilon SAML

What was that?

- I set up Ipsilon to authenticate against my FreeIPA server
- I set up Owncloud instance and created a simple application to do login via Ipsilon SAML
- Successfully logged-in users get created in Owncloud if they belong to a certain group in FreeIPA

What was that?

- I set up Ipsilon to authenticate against my FreeIPA server
- I set up Owncloud instance and created a simple application to do login via Ipsilon SAML
- Successfully logged-in users get created in Owncloud if they belong to a certain group in FreeIPA
- No need to enter password if Kerberos credentials are available

What was that?

- I set up Ipsilon to authenticate against my FreeIPA server
- I set up Owncloud instance and created a simple application to do login via Ipsilon SAML
- Successfully logged-in users get created in Owncloud if they belong to a certain group in FreeIPA
- No need to enter password if Kerberos credentials are available
- **Credentials were entered only once**

Better support for SAML in GNOME Online Accounts

GNOME Online Accounts in GNOME 3.20 supports single sign-on with a catch

- WebDAV protocol doesn't really work well with `mod_auth_mellon` as SAML client
- Have to use separate Owncloud end-point for non-SAML logon

There is a plan to fix GNOME VFS to support SAML negotiation so that Nautilus would be able to re-negotiate when accessing WebDAV shares

How *enterprisey* our home could become?

Very very *enterprisey*

Demo: `http://talks.vda.li/2016/05/SambaXP/freeipa-ipsilon-trusted-domain-owncloud-signon.webm`

What is that?

- FreeIPA has a cross-forest trust to an Active Directory forest

What is that?

- FreeIPA has a cross-forest trust to an Active Directory forest
- Ipsilon is configured to accept all valid users provided by FreeIPA

What is that?

- FreeIPA has a cross-forest trust to an Active Directory forest
- Ipsilon is configured to accept all valid users provided by FreeIPA
- Active Directory users are valid ones, with fully qualified user names to differentiate them from IPA users

What is that?

- FreeIPA has a cross-forest trust to an Active Directory forest
- Ipsilon is configured to accept all valid users provided by FreeIPA
- Active Directory users are valid ones, with fully qualified user names to differentiate them from IPA users
- Active Directory administrator signed into Owncloud as a normal user

What is that?

- FreeIPA has a cross-forest trust to an Active Directory forest
- Ipsilon is configured to accept all valid users provided by FreeIPA
- Active Directory users are valid ones, with fully qualified user names to differentiate them from IPA users
- Active Directory administrator signed into Owncloud as a normal user
- **Credentials were entered only once**

What benefits do we get by becoming *enterprisey* with free software?

1. Control your own infrastructure

What benefits do we get by becoming *enterprisey* with free software?

1. Control your own infrastructure
2. Improve user experience by reducing number of password/logon interactions

What benefits do we get by becoming *enterprisey* with free software?

1. Control your own infrastructure
2. Improve user experience by reducing number of password/logon interactions
3. Profit?

Questions?