

U-Bolt: Campus Identity Integration for Decentralized Systems

David Champion

Computation Institute, Enrico Fermi Institute

US ATLAS, UC3

University of Chicago





UC3 Identity

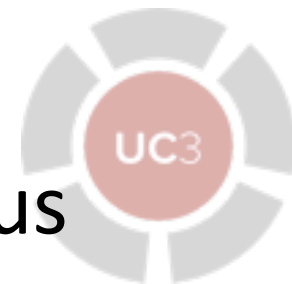
Where it started



UC3 Identity Goals



- UC3 is an open platform for connecting research to distributed HTC resources across campus.
 - » Condor cluster that can flock to other Condor clusters on campus
 - » 4-5 other facilities on campus, upwards of 10,000 job slots accessible
- Users could be anyone on campus.
- Users should be validated as legitimate campus personnel.
- Shared facilities should have a common basis for identifying owners of data and users of resources.





- We want to get potential users online:
 1. *quickly*
 - » minimally operational within 60 minutes
 2. *simply*
 - » use existing connection tools and identity frameworks
 3. *cheaply*
 - » no new username and passwords
 - » no complicated registration process, when the University already knows all users
- Campus identity is the obvious solution, but not all the pieces were there. To integrate our local access, we needed to improvise.





UC3 Onboarding Demo






University of Chicago Computing Cooperative (UC3)

<https://uc3.uchicago.edu>

Old Apple Yahoo! Google Maps YouTube Wikipedia News Popular sessions g com Saved Tabs UC3

University of Chicago Computing Cooperative (UC3)



University of Chicago Computing Cooperative

[Log In / Web Connect](#) [UC3 News](#)

What is UC3?

UC3 is an open computing framework for connecting users to shared Distributed High-Throughput Computing (DHTC) resources, both on- and off-campus.

[Click for details](#)

Who may use UC3?

Anyone with a University of Chicago [CNetID](#) may register to use UC3.

[Click for details](#)

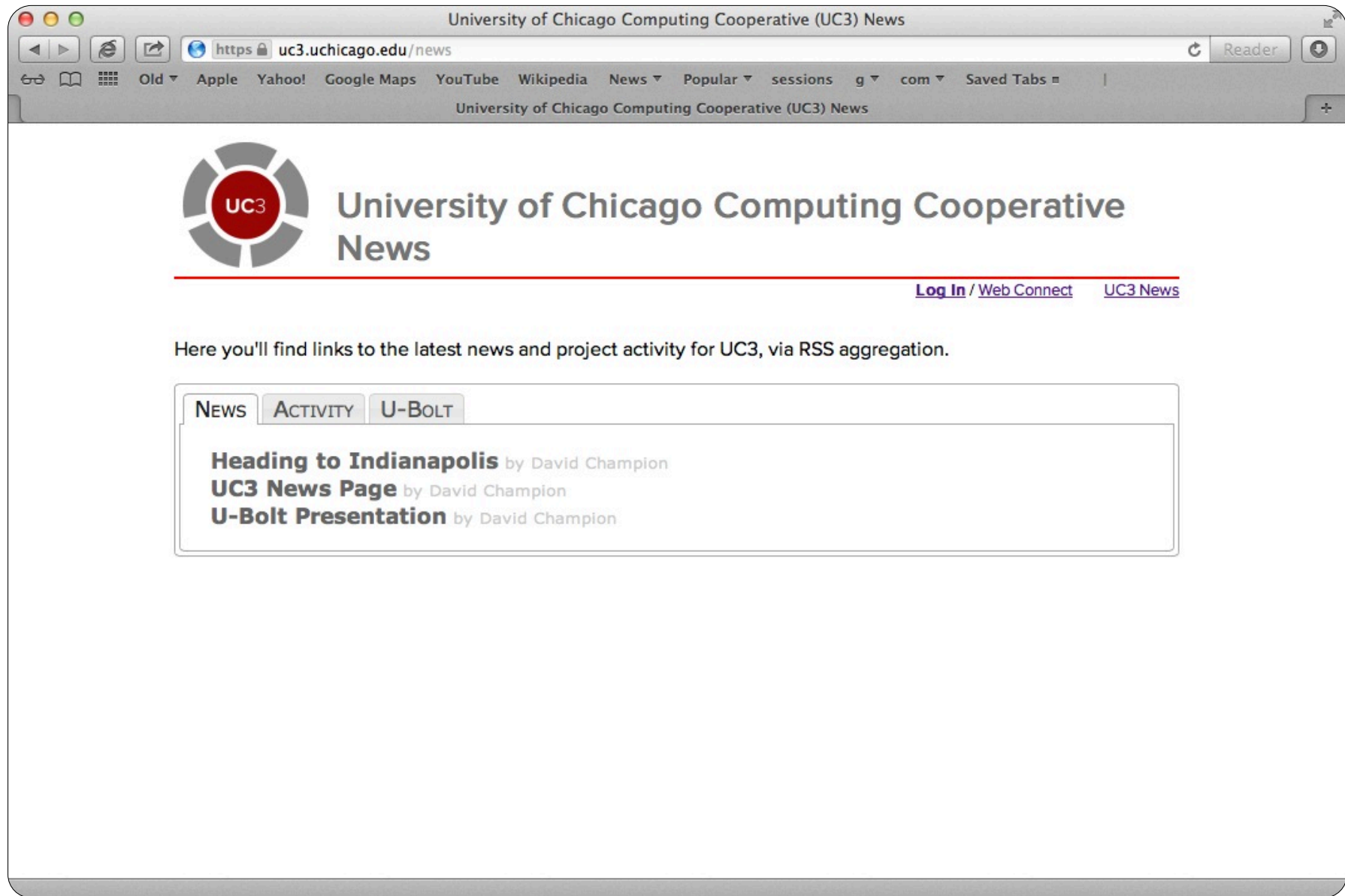
How can you use UC3?

Upon receiving authorization for UC3, you may log in to uc3-sub.uchicago.edu using your CNetID and password.

[Click for details](#)

Further Reference

- [Job Submission Reference](#)



Login (Campus Credentials)




University of Chicago Computing Cooperative (UC3) News

uc3.uchicago.edu/login

Old Apple Yahoo! Google Maps YouTube Wikipedia News Popular sessions g com Saved Tabs

University of Chicago Computing Cooperative (UC3) News

 Univ
New

To view this page, you must log in to this area on uc3.uchicago.edu:443:

CNet Authentication

Your login information will be sent securely.

Name:

Password:

☐ Remember this password in my keychain

[In / Web Connect](#) [UC3 News](#)

Here you'll find links to the

NEWS ACTIVITY U-I

Heading to Indianapolis by David Champion

UC3 News Page by David Champion

U-Bolt Presentation by David Champion

Go to "https://uc3.uchicago.edu/login"

User Registration




UC3 - Profile

https://uc3.uchicago.edu/register

Old Apple Yahoo! Google Maps YouTube Wikipedia News Popular sessions g com Saved Tabs

UC3 - Profile

 **Update UC3 Profile**

Registered as [David Champion \(dgc\)](#) [Tutorial](#) / [Sponsor](#) / [Web Connect](#) [Log Out](#) [UC3 News](#)

Please take a moment to update your profile and to tell us more about your interests. The better we know you, the better we can support your work.

Your profile information

Name:	David Champion (dgc)
E-Mail address:	dgc@uchicago.edu
Affiliation:	staff
Department:	Computation Institute
Member since:	Aug. 15, 2012, 11:27 a.m.

Research:

I research everything

UC3 interests:

just curious

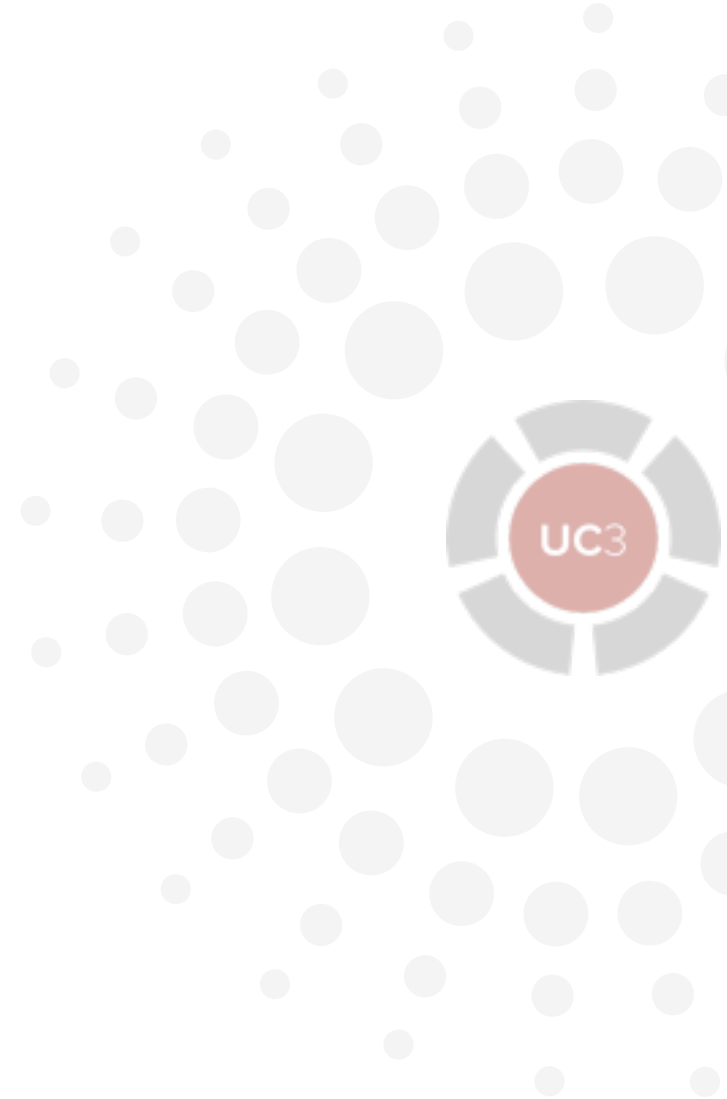
Software needs:

zork



One Hour

(ideal)




Return after Registration: Quick Start




UC3 - Quick Start Tutorial

https://uc3.uchicago.edu/framer/tutorial

UC3 - Quick Start Tutorial

 **Quick Start Tutorial**

Registered as [David Champion \(dgc\)](#) [Tutorial / Sponsor / Web Connect](#) [Log Out](#) [UC3 News](#)

 **UC3 Quickstart** [Tasks](#) [Edit](#) [Add](#) [Tools](#)

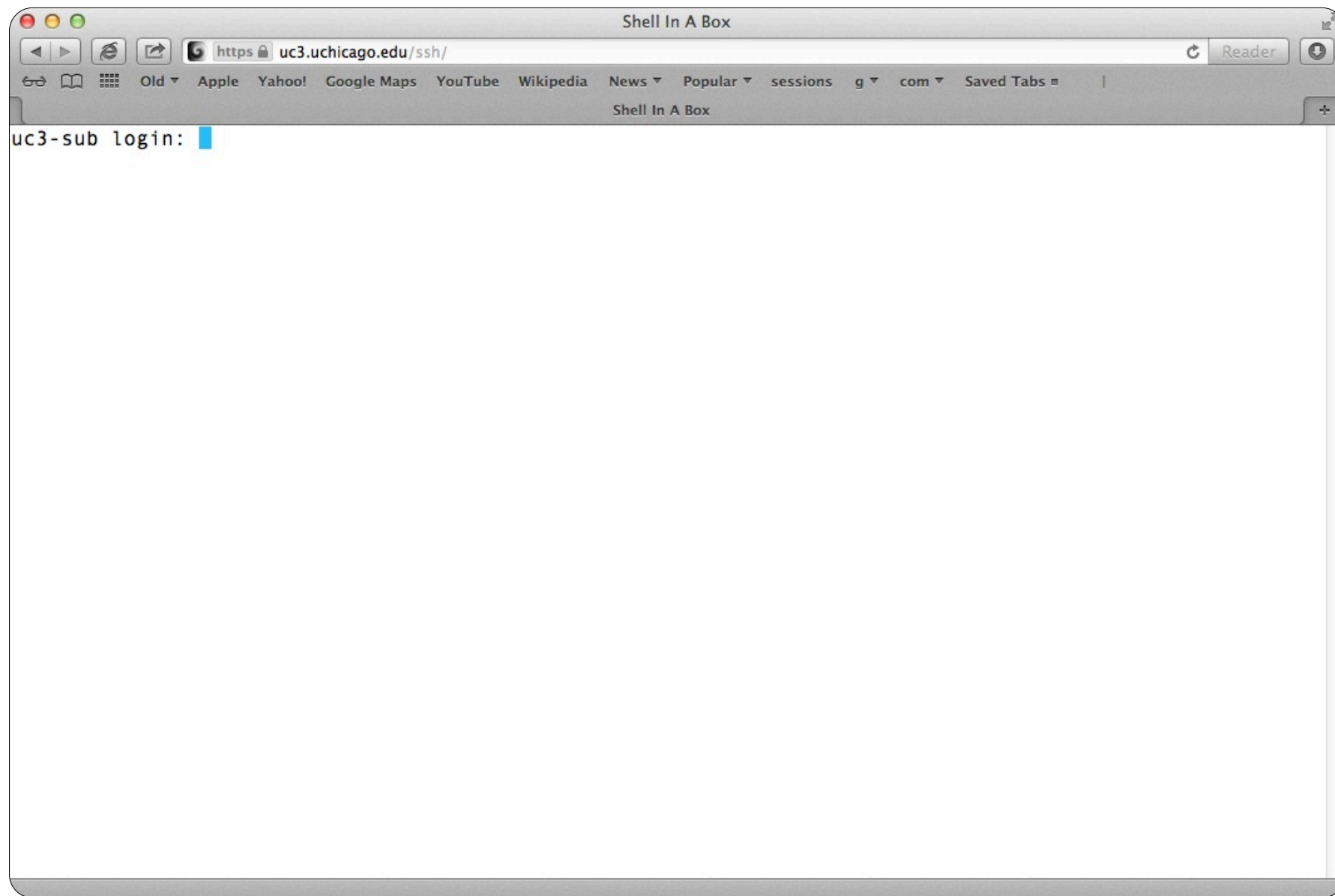
Added by [Marco Mambelli](#), last edited by [David Champion](#) on Mar 06, 2013 ([view change](#)) [show comment](#)

Table of Contents

- 1 Login to UC3
- 2 Set up the tutorial
 - 2.1 Manual setup
 - 2.2 Pretyped setup
- 3 Create a workload
- 4 Check job status
- 5 Check the job output
- 6 Now scale up
- 7 Flocking to a specific environment
- 8 Deleting jobs
- 9 Getting help

This is a quick start page which should take only a few minutes to go through. For more complete information go to the [Job Submission](#) page or other guides linked from [UC3 Home](#).

Return after Registration: Submit!



Login Failure



```
dgc — 80x24 — 888
bash ttys007 09:39:41 ~ [3/0]:
bash ttys007 09:39:42 ~ [3/0]: ssh uc3-sub.uchicago.edu
You must be a uniqueMember of cn=uc:org:uc3:users,ou=groups,dc=uchicago,dc=edu to login.
Connection closed by 128.135.158.243
bash ttys007 09:39:54 ~ [4/0]:
```





What is Campus Identity Integration?

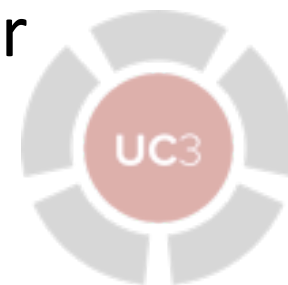
(And why do we care?)



Basic Questions About Identity



- What is identity, fundamentally?
 1. a token whose meaning is shared between a user or user agent and a resource controller
 - » My userid is *wjclinton*.
 2. identity can, but need not, make claims about your individual self
 - » My userid is *wjclinton*. My birthday is August 19. I am number 42.
 - » My userid is *CN=38f97c01-ccbe-4ad1-a6d7-72bebe31249b*.
 3. identity represents you or your agent in a transaction with a service provider
 - » As *wjclinton*, I demand that you release the codes.



- What does identity allow, in practice?
 1. a *provable* assertion of entitlement
 - » I, *wjclinton*, claim to have access to this computational facility. The evidence of my claim is this well-guarded secret.
 2. a *shared token* whose meaning is agreed upon between a user agent and a resource controller (service provider)
 - » You grant *wjclinton* rights. If we agree that I am *wjclinton*, then give me those rights.
 3. links to other attributes of a person or agent
 - » Since we agree that I am *wjclinton*, you may trust that I am reachable at a known e-mail address and phone number.
- *Identity Management (IdM)* is solving these problems and managing necessary data flows.

- *Isolated identity* refers to an identity store that is disconnected from other consumers and providers
 1. it does not provide identity to anything but itself
 2. it does not provide service to anyone identified externally
- Examples:
 1. UNIX /etc/passwd (usually)
 2. Apache htpasswd
 3. Samba smbpasswd
 4. Mac, Windows local users





- Advantages of isolated identity service
 1. **local control**: no external authority controls who may have identity in your service
 2. **flexibility**: because you control it, you may create multiple distinct identity types (individual user, workgroup, VO, glide-in agent)
 3. **low latency**: new users can be created and given privileges without significant delay
 4. **independence**: your service does not rely upon external providers to grant access



- Disadvantages of isolated identity
 1. **obligation**: no one else is going to help you maintain your identity system(s)
 2. **high latency**: it's another hoop for a prospective user of your service to jump through before being active
 3. **redundancy**: users have already provided ID to your greater institution; why must they do it again for you?
 4. **difficulty**: users must remember another password (and perhaps also username), or manually keep them in sync
 5. **inefficiency**: reduplication of effort in constructing, maintaining, and disabling accounts at various life cycle

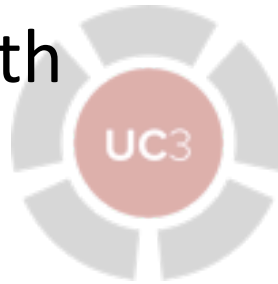


- Many or most UNIX (Linux, etc) login servers operate using isolated identity systems
 1. local /etc/passwd for each system or site
 2. configuration management only takes you one step up
 - + assists with synchronization of the passwd file across many systems
 - does not distance your team from the maintenance obligation
 - does not address user's concerns (obstacles to enablement)
 - still reduplicates labor across the institution
- So what do we do, then?





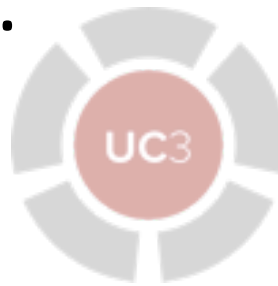
- *Centralized identity* services permit identity to be unified across the larger organization — the company or institution
 1. puts core identity management in the hands of a distinct team who can negotiate eligibility and life cycle with central resources (HR, Student Systems, Provost, Research VP)
 2. publishes this information to all consumers
 3. all consumers in sync with one another
 - » common identifiers lower barriers to intra-institutional resource sharing
 - » separate organizations can know that they're talking about the same user — ID becomes a shared token in a larger context than otherwise



Campus Identity Integration (CII)



- *Campus Identity* is centralized identity for the campus.
- *Campus Identity Integration* is addressing how to make campus identity work at the local scope.
 1. offset isolated identity disadvantages with central identity advantages
 2. use mixed identity services to hang onto the advantages of isolated identity management
- Central IdM provides this service, but cannot solve your localized concerns.

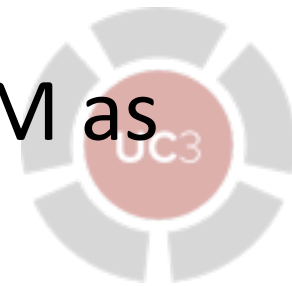


1. Identity roles provided by central IdM should be visible and meaningful locally.
2. Authentication (authN) using these identities should be possible using centralized authN credentials.
3. Resource authorization (authZ) policy should be managed locally.
 - a. Central IdM may, however, act as a disburser of authZ policy that is defined by a resource manager (you).





4. The resource manager (you) should have liberty to augment central identity with local identity (for VO, etc).
5. The resource manager should have liberty to supersede identity attributes from central IdM as necessary, to ensure correct behavior and sustainability in the resource environment.



- All this is doable out of the box, provided a sufficiently enabling campus directory.
 - » Today, this generally means an LDAP or Active Directory (AD) service.
 - » LDAP provides both directory service and authentication service using only OpenLDAP client software.
 - » AD is LDAP + Kerberos + Microsoft magic sprinkles. OpenLDAP client software provides directory service, while Samba's winbind provides authentication.
- Out of box success requires *complete* provisioning of the attributes required by the posixAccount object class.





- Why is this a challenge?

1. Anecdotally, *almost nobody* provides this completely.
2. When they provide it partially, they come up short in different ways.
 - » Some central IdM services do not publish all users, or give clients only limited views.
 - » Some IdM services don't incorporate posixAccount at all, making them no more than authentication services. Don't expect posixAccount from an AD, for example.
 - » Some IdM services provide posixAccount, but put useless data in some attributes.
 - » Some IdM services provide posixAccount, but put no data some attributes.
 - » Your central IdM probably provides no useful groups service at all.
3. It's each resource manager's individual burden to address — few common tools exist.



Where do we begin with CII?



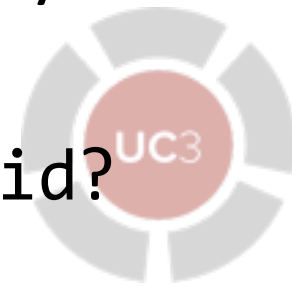
- It's important to understand first where central IdM is coming from:
 1. They have a large and disparate user base, and must fit their service curve to widely scattered data points.
 2. Administrative and business applications will usually carry more weight than instruction or research.
 - » At core they are a business service, called upon to enable integrations and cost management that make all other lines of work possible.
 - » Don't let this frustrate you; it's unavoidable and they didn't make that decision.
 3. Central IdM probably cannot contribute directly to solving your problems.
 - » They're really busy too, and you'd be surprised in what aggravating ways they have to spend their time.



- Where central IdM is coming from (cont'd):
 4. They are, however, *interested* in your problems, and are probably happy to discuss them in the abstract.
 5. They are also interested in enhancing their service, provided that you present a cohesive case:
 - » why you need the change or new capability;
 - » why this won't harm any current application or use of their service;
 - » why your request constitutes an overall improvement to the institution as a whole, and not just to your “business unit”.
 6. In short: they can be a good partner, but they can never work for you.
 - » There are too many other people they also work for.
 - » You're going to be doing a lot by yourself — but keep them informed!



1. Understand what your central IdM currently provides to you.
 - a. Services: LDAP? AD? Other?
 - b. Extent of service: all institutional constituents? Only certain classes?
 - c. Specific attributes *provisioned* and *provided*: cn, uid? posixAccount attributes? (which ones?)
 - d. Is a service DN (bind DN) required? If so, can that DN read the attributes that you need?
 - e. Preferably, can your service bind as an authenticating user, then as that user retrieve required attributes?



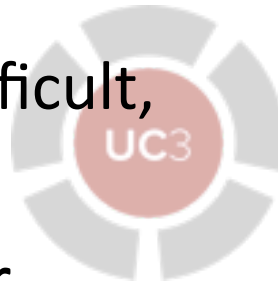
2. Ask IdM about potential enhancements:

- a. what are they willing (and able) to add or change?
- b. what is adequate demonstration of need?
- c. what time frame do they need to accomplish this — and is that sufficient to meet your needs?



3. Minimum requirements:

- a. bind as user; read user attributes as user
- b. cn or uid contains a unique identifier
- c. all users in a single directory service
 - » it's technically possible to work around this, but it's difficult, and there are multiple risks to negotiate
- d. everything else is on you, and not all tools exist for making the translation (but they are feasible to create)



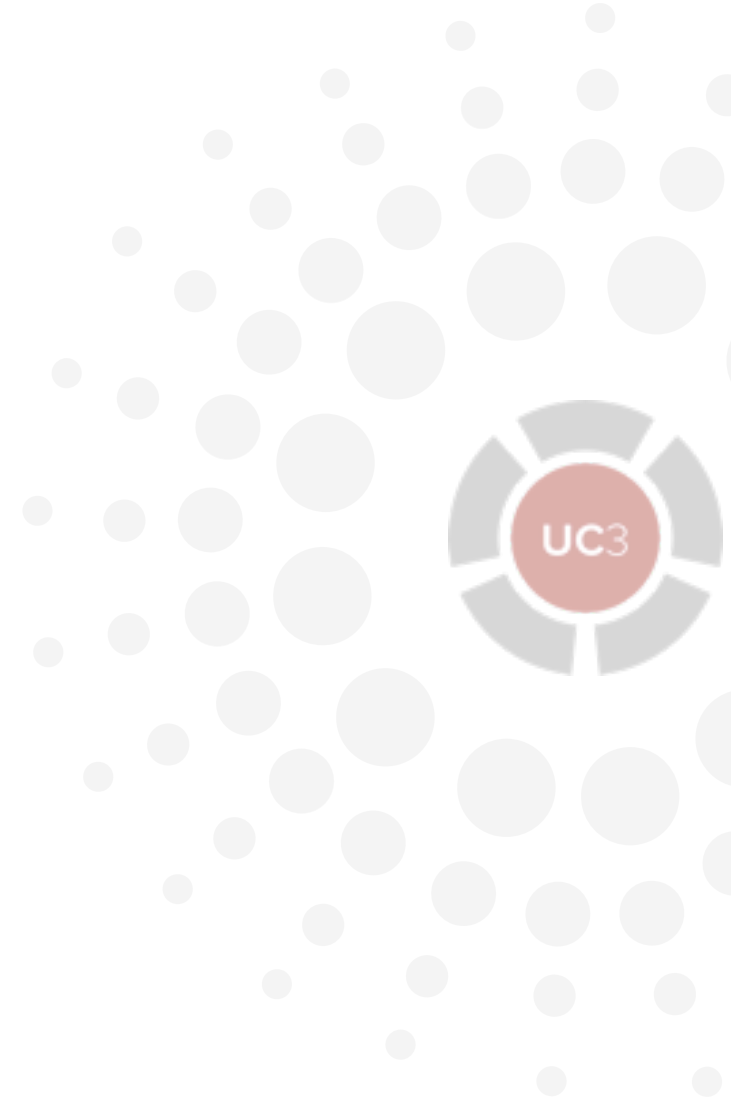
4. Ideal scenario (cumulative):

- a. all of the `posixAccount` MUST attributes: `cn`, `uid`, `uidNumber`, `gidNumber`, `homeDirectory`
- b. two of the `posixAccount` MAY attributes: `loginShell`, `gecos`
- c. sensible and distinct values for `uidNumber` and `homeDirectory`
- d. group mapping from `gidNumber` to group names



Your situation is probably somewhere between minimum and ideal.

U-Bolt



What is U-Bolt?



- U-Bolt is identity integration middleware for the campus
- it aims to be a flexible toolkit for addressing identity integration problems for distributed environments embedded within, or with access to, larger campus infrastructures

What U-Bolt Provides



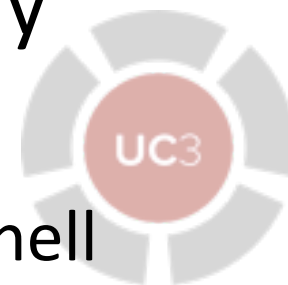
- U-Bolt currently consists of two NSS modules to address problems arising from limitations in a campus LDAP environment
 1. `nss_identity` to provide forward and reverse group mapping for artificial groups not in LDAP
 2. `nss_filter` to provide unique home directory mapping when LDAP does not
 - » `nss_filter` also allows optional mapping of `pw_gecos` and `pw_shell`
- It is a work in progress; contributions are welcomed



Measures of Completion



- U-Bolt is already a success in that it has addressed identity integration for our site
- Major objective is to be able to piggyback on any LDAP or AD authentication service without any attribute visibility whatsoever:
 - no uid • no gid • no gecos • no home directory • no shell
- However, when such attributes *are* visible, we should use them



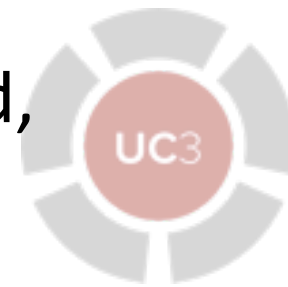


Integrating Your Site



Checklist of required and preferred components:

- ☐ bind as user; read user attributes as user
- ☐ cn or uid contains a unique identifier
- ☐ all users in a single directory service
- ☐ all of the posixAccount MUST attributes: cn, uid, uidNumber, gidNumber, homeDirectory
- ☐ two of the posixAccount MAY attributes: loginShell, gecos
- ☐ sensible values for uidNumber and homeDirectory
- ☐ group mapping from gidNumber to group names



Mitigating Reality: A Rough Guide



What ideals are you missing, and what do you do?

Missing feature / Problem	How you handle it	Works Today?
<ul style="list-style-type: none">• cn or uid missing• cannot bind as user	talk to IdM - you need one of these	
<ul style="list-style-type: none">• loginShell is not present or not useful	supersede with nss_compat	✓
<ul style="list-style-type: none">• gecos is not present or not useful	configure nss_ldap to use cn or displayName instead	✓
<ul style="list-style-type: none">• gidNumber does not map to a named group	not strictly required, but you can map it in the nsswitch stack using nss_identity	✓
<ul style="list-style-type: none">• homeDirectory is not sensible or not unique	mapped through multiple layers of nsswitch: nss_compat to supersede pw_home, with nss_filter to perform user substitutions	✓
<ul style="list-style-type: none">• uidNumber is not sensible or not unique	You can't work with this. Ignore it and resolve as if there's no uidNumber.	↩
<ul style="list-style-type: none">• no gidNumber	either supersede with nss_compat, or treat like missing uidNumber	
<ul style="list-style-type: none">• no uidNumber	need to manufacture this on demand; this will in turn require stateful user caching	✗

Solving loginShell



Problem:

- loginShell is not present or not useful in LDAP

Solution:

1. use nss_compat to supersede locally

```
# /etc/nsswitch.conf
passwd: files compat
passwd_compat: ldap
```

```
# /etc/passwd
root:x:0:0:System Administrator:/root:/bin/sh
bin:x:1:0:::/bin/false
alice:x:2049:500:Alice:/home/alice:/bin/tcsh
+:::/:/bin/bash
```

Problem:

- gecocos is not present or not useful in LDAP

Solution:

1. configure nss_ldap to use cn or displayName instead

- » usually nss_ldap shares configuration with openldap and pam_ldap
- » configuration file is typically /etc/ldap.conf or /etc/openldap/ldap.conf
- » see nss_ldap(5), RFC2307

```
# /etc/ldap.conf
# For generic LDAP, map cn onto gecocos
nss_map_attribute gecocos cn
# For Active Directory, map displayName onto gecocos
nss_map_attribute gecocos displayName
```

Solving gidNumber



Problem:

- gidNumber does not map to a named group in LDAP

Solution:

1. map in nsswitch using nss_identity

» see UC3 case study

```
# /etc/nsswitch.conf
passwd: files ldap
group: files identity
hosts: files dns
...
```



Solving homeDirectory



Problem:

- homeDirectory is not sensible or not unique

Solution:

1. supersede in nsswitch using nss_compat and nss_filter

» see UC3 case study

```
# /etc/nsswitch.conf
passwd: files filter
passwd_filter: compat
passwd_compat: ldap
```

```
# /etc/passwd
+:::::/home/&:
```



Solving gidNumber



Problem:

- gidNumber is not present in LDAP

Solution:

1. this may be a small problem that can be solved with nss_compat supersession

```
# /etc/group  
users::1001:
```

```
# /etc/passwd  
+:::1001:::
```

2. otherwise this is akin to solving uidNumber; see below

Future Challenge: Solving uidNumber



Problem:

- uidNumber is not present in LDAP

Solution:

1. forward development in U-Bolt will address this:
 - a. an nss_http module bind to an HTTP-based directory service
 - » GET /ubolt/0.1/passwd/byuid/2052
{'name': 'dgc', 'passwd': '!', 'uid': 2052, 'gid': 2052, 'gecos': 'David Champion', 'dir': '/home/dgc', 'shell': '/bin/bash'}
 - b. U-Bolt will provide a plugin-based reference implementation that can be tuned or extended to meet local needs
 - c. service will provide stateful storage for manufactured data
 - d. can be run locally or centrally

Why the HTTP approach?



- We want to simplify client configuration as much as possible, while providing solutions to any problem sites are likely to encounter
 1. There are two approaches to this:
 - a. let the client talk to an extant DS (e.g. LDAP) and teach that extant DS to incorporate complexity
 - » this either involves a lot of continuous feed processing — the kind of thing we'd need extensive cooperation from IdM to do — or hacks to provide configurable backends to an LDAP/NIS frontend
 - b. invent a shim for the client that lets us talk to a DS that anyone can implement, or that they can borrow from us and adjust
 2. The latter is simpler and more maintainable in the long term than trying to graft complex dynamic backends onto code projects (e.g. OpenLDAP) managed by an upstream host.

Why the HTTP approach?



- We need a protocol for the exchange between the nss module and the service. HTTP is:
 1. widely implemented
 - » anyone can build their own service, or use our reference implementation
 2. scalable as needs change
 - » can run as a local standalone service, or under Apache, etc.
 3. easily extensible
 - » structure, scope, and hierarchy already present in standard HTTP WS idioms



Questions?

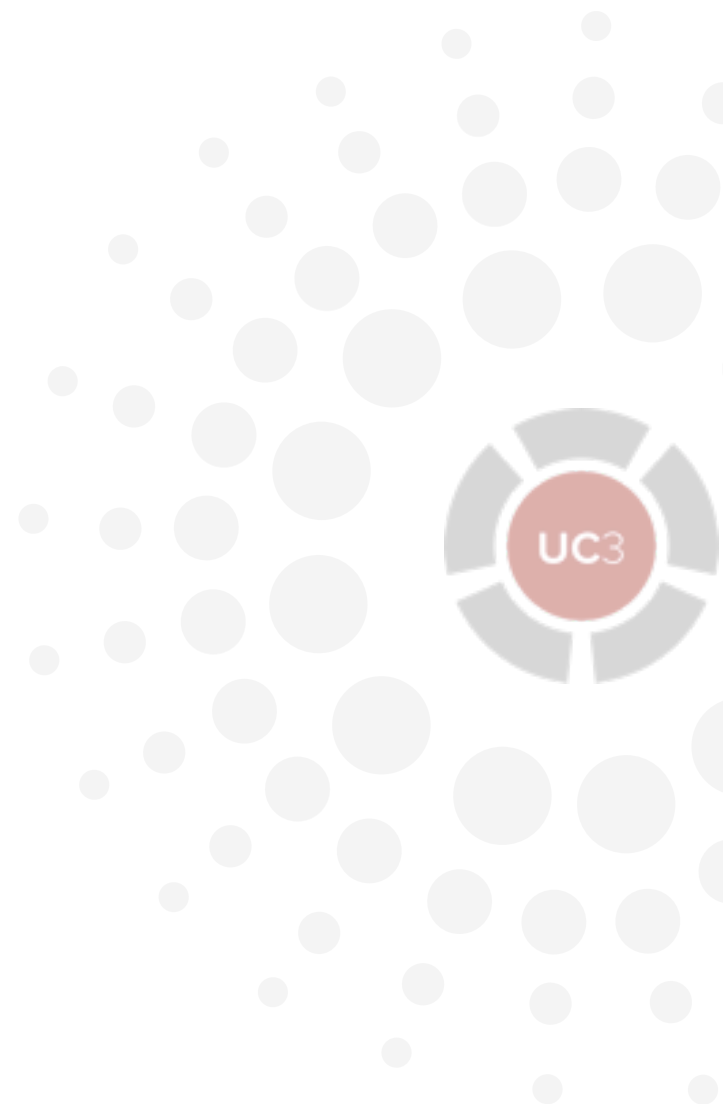
<https://uc3.uchicago.edu/>

<https://uc3.uchicago.edu/news>

<https://uc3.uchicago.edu/ubolt>

<https://github.com/DHTC-Tools/ubolt>

Contact: uc3-support@uchicago.edu, dgc@uchicago.edu





The remaining slides are part of the full topical slide set, but not part of today's high-level talk. They are included as a technical reference.



Account Services Technical Overview

How does it work?





1. Two key service departments:

a. Nameservice Switch (NSS)

- *identity* service (ID): what users exist?
- *directory* service (DS): what are a user's attributes (uid, home directory, shell, groups, etc)
- limited *authorization* controls
 - » existence = access
 - » valid shell = access

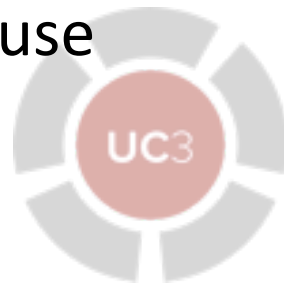




1. Two key service departments (cont'd):

b. Pluggable Authentication Modules (PAM)

- *authentication* service (authN): is this user (agent) who she (it) claims to be?
- *authorization* service (authZ): is this user permitted to use this system?





2. NSS and PAM are both generalized frameworks implemented within libc
3. Both are extensible under a plugin architecture
 - a. plugins are dynamic objects (DLL/DSO) that load into the address space of the process that needs their service
 - b. that process can be *any* program that calls into the NSS or PAM framework
 - c. most commonly that process for NSS is nscd, which proxies/caches lookups for other processes
 - d. PAM modules are loaded directly by e.g. sshd, httpd

Study Example: Basic local ssh login



1. sshd prompts for username and password
2. sshd looks up user via `getpwnam()`, a libc function
 - a. `getpwnam()`, passes request to nsswitch (nss) framework
 - b. nss checks `/etc/nsswitch.conf`
 - c. nsswitch loads `libnss_files.so` to resolve request
 - d. `libnss_files.so:_getpwnam_r()` resolves name via `/etc/passwd`
 - e. struct `passwd` is constructed (almost like `/etc/passwd`)

```
passwd: files  
hosts: files dns
```

```
alice:$1$YeNsbWdH$wvOF...:2049:500:Alice:/home/alice:/bin/bash
```

Study Example: Basic local ssh login



3. sshd asks PAM to validate user using the password
 - a. PAM checks `/etc/pam.conf`, `/etc/pam.d/*` to find the applicable module stack
 - b. `pam_unix.so` retrieves struct `passwd` from NSS; it contains a DES, MD5, or SHA hashed password

```
alice:$1$YeNsbWdH$wvOF...:2049:500:Alice:/home/alice:/bin/bash
```
 - c. `pam_unix.so` validates the password by hash compare, and returns success value to sshd
4. if authN succeeded, other PAM modules may refuse login on a policy basis (authZ)

Use Case: Moving to LDAP/AD



1. If your site has all the requirements of the ideal scenario, *all you need may be to convert to LDAP!*

» Active Directory is a special case of LDAP, and can also work.

2. NSS must be augmented:

- a. nsswitch *stacks*: if a user is not found in the first listed plugin, it falls through to the second, third, fourth...

```
# /etc/nsswitch.conf
passwd: files ldap
group: files ldap
hosts: files dns
...
```

- b. users not listed locally can now be found in LDAP (configuring LDAP access not described here)



3. PAM must be updated:

a. `/etc/pam.d/system-auth` (Red Hat derivatives); `/etc/pam.d/sshd` (other):

```
auth        required      pam_env.so
auth        sufficient     pam_unix.so nullok try_first_pass
auth [default=ignore success=done] pam_ldap.so use_first_pass
auth        requisite      pam_succeed_if.so uid >= 500 quiet
auth        required      pam_deny.so
```

- this tells PAM to try standard UNIX authentication (hashed passwords); if that doesn't work, then try binding to LDAP as the candidate user using the password presented

b. if you need Active Directory authentication, use `pam_winbind.so` instead of `pam_ldap.so`, and separately configure Samba's `winbindd`.



4. What about home directories?

- a. if your users' home directories are pre-created (e.g. reside on an extant network share), you're done
- b. otherwise, you can add PAM configuration to create home directories on demand. Two options:
 - `pam_mkhomedir.so`: copies a skeleton home from `/etc/skel` when home is absent
 - ▶ maximally efficient, completely rigid
 - `pam_exec.so`: executes an arbitrary script as root for each login; this script can check whether a home is needed and create it in any arbitrarily complex fashion
 - ▶ very inefficient, infinitely flexible

Use Case: Restricted LDAP integration



1. Mixing local and directory-sourced users is an old problem.

a. old NIS (SUN) system managers will recall this pattern:

```
root:x:0:0:System Administrator:/root:/bin/sh
bin:x:1:0:::/bin/false
alice:x:2049:500:Alice:/home/alice:/bin/tcsh
+bob:::::
+:::::/bin/nologin
```

b. the + lines means: map into this system's user list a user who appears in my (NIS) directory service, and treat them as though they were local

c. a blank field is inferred from the directory

d. an explicit field supersedes the one in the directory

Use Case: Restricted LDAP integration



2. nss_compat brings this to a modern nsswitch —
whether using NIS or any other directory service

a. suppose the following:

```
# /etc/nsswitch.conf
passwd: files ldap
group: files ldap
hosts: files dns
...
```

```
# /etc/passwd
root:x:0:0:System Administrator:/root:/bin/sh
bin:x:1:0:::/bin/false
alice:x:2049:500:Alice:/home/alice:/bin/tcsh
+bob:::::
+:::::/bin/nologin
```

b. all LDAP users are now known to the local system, but
only “bob” may log in

How can we tap into this?



Your site is probably not so lucky. But how can you extend this to make it work for you?

- NSS and PAM together provide an extremely flexible mechanism for semi-arbitrary governance of identity, authentication, and macro-level authorization
- you can insert authNZ policies and mechanisms into PAM using a simple API
- you can insert directory services or wrappers for directories into NSS using an even simpler API
- the plumbing is there; all you need is code
(but documentation and examples are very limited)

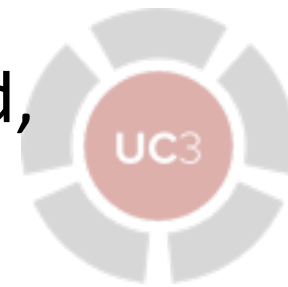


Case Study: UC3



Recall our required and preferred components:

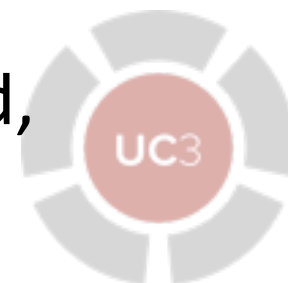
- ☒ bind as user; read user attributes as user
- ☒ cn or uid contains a unique identifier
- ☒ all users in a single directory service
- ☐ all of the posixAccount MUST attributes: cn, uid, uidNumber, gidNumber, homeDirectory
- ☐ two of the posixAccount MAY attributes: loginShell, gecos
- ☐ sensible values for uidNumber and homeDirectory
- ☐ group mapping from gidNumber to group names





Recall our required and preferred components:

- ☒ bind as user; read user attributes as user
- ☒ cn or uid contains a unique identifier
- ☒ all users in a single directory service
- ☒ all of the posixAccount MUST attributes: cn, uid, uidNumber, gidNumber, homeDirectory
- ☒ two of the posixAccount MAY attributes: loginShell, gecos
- ☒ sensible values for uidNumber and homeDirectory
- ☐ group mapping from gidNumber to group names





Authentication and authorization:

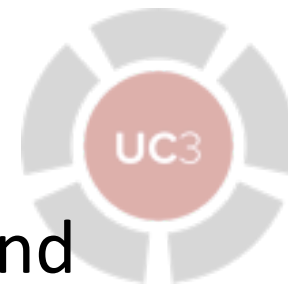
- ☑ bind as user; read user attributes as user
 - ☑ cn or uid contains a unique identifier
 - ☑ all users in a single directory service
-
- ➡ We can use pam_ldap for authentication
 - ➡ Authorization works through pam_ldap also — more on this later



UC3's Advantages



- ☑ all of the posixAccount MUST attributes: cn, uid, uidNumber, gidNumber, homeDirectory
 - ☑ two of the posixAccount MAY attributes: loginShell, gecos
 - ☑ sensible values for uidNumber
- ➔ nss_ldap gets us most of the way to an identity and directory solution



Two problems to solve:

1. no POSIX groups in LDAP

» user's gidNumber does not reverse map to any group name.

2. homeDirectory value is an artifact of a past age:

`/nfs/harper/ha0/usernameA`

`/nfs/harper/ha1/usernameB`

`/nfs/harper/hb0/usernameC ...`

» We want to replace with a simple `/home/username`.

Each of these is a case for an NSS plugin.

UC3: Solving gidNumber



The problem:

1. In ouchicago LDAP, gidNumber == uidNumber.
2. In ouchicago LDAP, there are no POSIX groups.

```
$ id -a
uid=2052(dgc) gid=2052
$ getent group 2052
(no result)
$ ls -ld ~
drwxr-x--x 135 dgc 2052 421 2013-02-21 12:03 /nfs/harper/hc0/dgc
```

UC3: Solving gidNumber



The solution:

1. `nss_identity.so` is an NSS module to manufacture such groups on demand. Pseudocode:

a. `getgrgid(gid=2052) ⇒`

(i) `gid = uid = 2052`

(ii) `pw = getpwuid(uid=2052)`

(iii) `return new group(gid=2052, name=pw.name)`



UC3: Solving gidNumber



2. Added to nss stack:

```
# /etc/nsswitch.conf
passwd: files ldap
group: files identity
hosts: files dns
...
```

3. Done!

```
$ id -a
uid=2052(dgc) gid=2052(dgc)
$ getent group 2052
dgc::2052:dgc
$ ls -ld ~
drwxr-x--x 135 dgc dgc 421 2013-02-21 12:03 /nfs/harper/hc0/dgc
```

UC3: Solving homeDirectory



A little more complex. The problem:

1. In ouchicago LDAP, homeDirectory is a valid and distinct value, and technically works, but
2. It's ugly, unpredictable, a bit surprising and confusing to users.

```
$ cd; pwd
/nfs/harper/hc0/dgc
$ getent passwd dgc
dgc:x:2052:2052:David Champion:/nfs/harper/hc0/dgc:/bin/bash
```

3. We can't just stack onto NSS, because we receive and use the rest of the nss_ldap response.



The solution:

1. `nss_compat.so` is an NSS passthrough or proxy module. We can use this technique. (Twice!)

```
# /etc/nsswitch.conf  
passwd: files compat  
passwd_compat: ldap
```

- a. `getpwnam(name="dgc")` checks with `nss_files`
- b. `getpwnam(name="dgc")` cascades to `nss_compat`
- c. `nss_compat` makes a "back door" call through `nss_ldap`
- d. `nss_ldap` returns `dgc`'s LDAP POSIX account
- e. `nss_compat` updates this data and returns it

UC3: Solving homeDirectory



1. `nss_filter.so` is an NSS passthrough or proxy module that we compound with `nss_compat.so`. This is unusual, but it solves our problem exactly.

```
# /etc/nsswitch.conf
passwd: files filter
passwd_filter: compat
passwd_compat: ldap
```

```
# /etc/passwd
root:x:0:0:System Administrator:/root:/bin/sh
bin:x:1:0:::/bin/false
alice:x:2049:500:Alice:/home/alice:/bin/tcsh
+:::::/home/&:
```



2. Logic flow:

- a. `getpwnam(name="dgc")` checks with `nss_files`
- b. `getpwnam(name="dgc")` cascades to `nss_filter`
- c. `nss_filter` makes a "back door" call through `nss_compat`
- d. `nss_compat` makes a "back door" call through `nss_ldap`
- e. `nss_ldap` returns `dgc`'s LDAP POSIX account, with `pw_dir="/nfs/harper/hc0/dgc"`
- f. `nss_compat` replaces with `pw_dir="/home/&"`
- g. `nss_filter` replaces `/home/&` with `/home/dgc` and returns the full `passwd` entry

UC3: Solving homeDirectory



2. Added to nss stack:

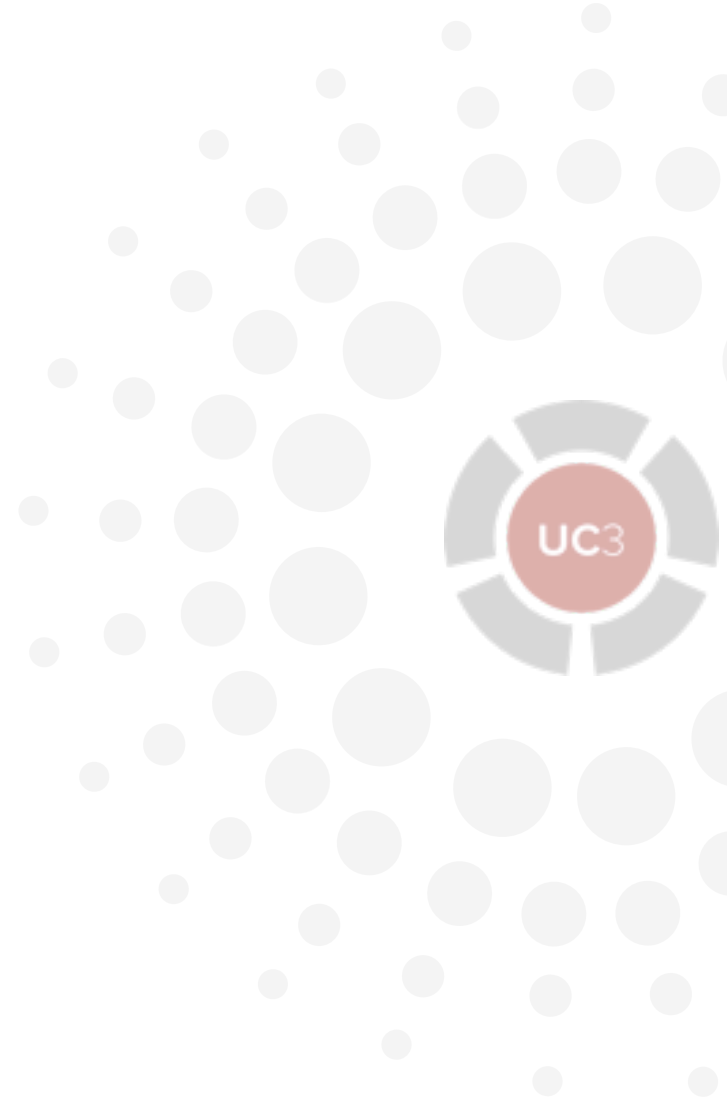
```
passwd: files filter  
passwd_filter: compat  
passwd_compat: ldap
```

3. Added to /etc/passwd:

```
+::::::/home/&:
```

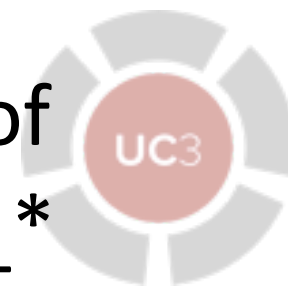
4. Done!

```
$ cd; pwd  
/home/dgc  
$ getent passwd dgc  
dgc:x:2052:2052:David Champion:/home/dgc:/bin/bash
```





- Federated identity is an interesting goal, but different scope and requirements — we're primarily addressing directory service, not authentication needs
- It's probably possible to perform some or all of nss_compat's functions in ldap.conf with nss_* parameters; this might save some headache
 - » However, nss_http would obviate this too and still be more flexible





- where do nss and pam modules install?
 1. `/lib/security/pam_xyz.so[.2]`
 2. `/lib64/security/pam_xyz.so[.2]`
 3. `/usr/lib/libnss_xyz.so[.2]`
 4. `/usr/lib64/libnss_xyz.so[.2]`

