SSO, LDAP, SAML, DN, RDN…etc

Over the past week, I've been working on the database and storage design. I was originally assigned the role of tester (with nothing to test) and programmer (with nothing to program yet), and Tim was internally given the job of 'Database Design'. Ideally, he would have collaborated with Eeren (the interface designer) to give us a nicely designed storage backend. But of course, real life is never ideal. So the ball has been thrown back at me.

I was originally given a database that had been normalized by Jing, Eeren, Tim (3 people to normalize one db…) It consisted of separate tables for supplier and customer invoices, refunds, and orders. An interesting bit was how it tracked both orders and sales in one table. I'd personally considered that, but logically, sales and purchases were different.

So I scaled a lot of the original model down, then blew it back up to accommodate discounts, product combos, multiple payments over time to settle a single bill, varying prices, multiple VAT bands (normalized to another table) and so on. Two areas I changed conceptually is that there's no longer the distinction between a customer and a supplier (it's very possible for someone to both sell and buy something from you, e.g. a business working in the same sector as you), and the merging of organizations and customers into a single 'contacts' entity. This would allow us to buy from and sell to both individuals and organizations. Having said that, I'm now in the process of scaling this back down again to simplify the model for others in my group to write the front-end.

We had to provide email systems for the hypothetical business, so it was a choice between running our own SMTP/POP/IMAP server and using something like Google Apps. A major downside for Google Apps is that they no longer offer free plans; however I decided to design the system with data portability in mind, and cater for integration with Google Apps anyway. This resulted in much of the data being moved to a LDAP server, which I've since installed (OpenLDAP), set up, and populated with appropriate fields. The main data being stored on this server are 1) employee user account data, 2) shared contacts data (for both suppliers and customers), and 3) authorization data specifying which user are part of certain groups, which can be used to determine access levels. The first two data sets can be synchronized with Google Apps using Google Apps Directory Sync.

All three data sets are unlikely to change massively (unlike transaction details etc), which makes them suited for a directory server rather than the conventional SQL database. Furthermore, use of an LDAP provider will provide much more scalability in the future, allowing integration with enterprise-level IT services if the business grows to that size, plus all the benefits of an LDAP-compliant directory (fast read times, potential for replication, strictly defined standard attributes and objectClasses for interoperability).

The email provider is still undecided at this point, but having designed the backend with an LDAP-compatible server, we could opt to use a number of external providers which can automatically synchronize with our user account listings (such as Google Apps Mail, Zoho Mail, Novell etc.) Of course, if we were to bring email in-house, account synchronization is still possible through the PHP functions for interacting with LDAP servers. An area I also looked into was SSO support (read: for integration with Google Apps). However that required a lot more research on SAML, and user synchronization with Google Apps was a prerequisite anyway.

The final area I looked into was calendars. After much mulling over how to store calendar data (and whether it should also go into the LDAP server), I had a look at the webcal and iCalendar specifications. It turned out that all a calendar subscription consisted of was a listing of events in text format. The implementation I'm opting for is therefore is to store dates (such as payment deadlines and delivery times) in the database as part of the sales and purchase order information, then provide a URI which would fetch the dates from the database and header them as an iCalendar-compliant text stream, which can then be subscribed to by calendar providers (read: Google Apps Calendar).

Alphabet Soup

Email

Last week, I considered how to implement an email service to our hypothetical business. The easy option was to tie it in with Google Apps or Zoho Mail (or other readily available email providers). The fun, challenging, and morally correct option would be to set up our own server with a RoundCube webmail frontend (sorry SquirrelMail, your interface reminds me of Windows 95).

The choices boiled down to the most popular options on the market that were FOSS were Dovecot and Courier IMAP for IMAP/POP access, and Postfix, Exim, and Courier as the MTA (SMTP server). This shortlist was scientifically based on the tried and tested method of listing the top 10 Google hits. From this, I flipped a coin (figuratively and literally), and decided to go for a combination of Dovecot and Postfix. They also happened to have some of the best support for LDAP interfacing and prettiest documentation, though that did not have any impact on my decision.

Having looked at the multitude of HowTo guides, a number of them suggested using for SSO. Of course, using an LDAP backend would already allow a single identity, but an SSO implementation would be much nicer to show off, and complement my alphabet soup.

Kerberos

So I set off on my quest to open the next Russian doll. Having heard that LDAP could be integrated with OpenLDAP, I followed a number of these guides. Although I installed both successfully, a number of factors deterred me from ultimately realizing my dream:

- I probably spent something like 3 hours connecting to a Kerberos KDC, assuming it was due to a DNS error (and of course the extra time I had to wait for the DNS configuration to propagate), only to eventually realize that the daemon hadn't automagically started itself (duh!)
- After almost three full days of testing and playing around 'integrating' Kerberos with LDAP, I eventually realized there were three ways that this could be done, and they were all for different reasons and with differing results.
    1. Use the LDAP server (technically called the DSA, I think, since LDAP only refers to the protocol) as the principal database for my MIT Kerberos installation. This I actually achieved with relative ease. It was after I finally got this working that I started to wonder what use it would be. (none?!)
    2. Use Kerberos (GSSAPI) to authenticate the LDAP session itself with the inbuilt SASL support offered by OpenLDAP. I'd spent half a day researching this before I realized it wasn't what I needed, since the PHP code would only ever authenticate to LDAP using hardcoded default administrator credentials.

3. Use Kerberos for user authentication and OpenLDAP for user authorization, i.e. pass-through authentication. This uses Cyrus SASL to connect MIT Kerberos and OpenLDAP.

I was half way through implementing option 3 before realizing the complexities of having to keep the Kerberos principal database and the LDAP user directory in sync. Since I didn't have any dedicated subsystem for executing atomic CRUD operations on my now-divided user database, I would have to manually enforce this in the PHP code. Furthermore, the benefits of Kerberos was much diminished upon finding out its limited client side implementation in browsers, and that I would have to integrate and test an unofficial apache2 extension (mod_auth_kerb) with the PHP code and RoundCube. If I had (a lot!) more time, Kerberos would have added that final sparkle, and allowed, say, an end user's employee to log in once to an in-store computer, and go straight onto the web interface without logging in again. Or SSO for the main PHP system and the webmail (CodeIgniter and RoundCube have different methods of handling user sessions and cookies, so getting SSO to work using a domain-based cookie would've been a pain).

LDAP

So we are (rather, I am) back to square one, and I'm now trying to get Dovecot working, authenticating against a LDAP directory of users. So far little luck, as the Dovecot POP3 and IMAP daemons keep quitting silently whenever I start the service. Hopefully I'll have better news to report next week. Having said that, I now understand how all the authentication magic works (sort of… hmmm), so once I figure out why Dovecot isn't behaving, it shouldn't be difficult to hook up Postfix to use Dovecot SASL for authentication (instead of GSSAPI/Kerberos, another popular configuration which prompted me to set up Kerberos in the first place), configure RoundCube webmail to these servers, and start on the models on the user management.

In Other News

I had a wee chat with our TA, who suggested to my horror (gasp!) that small businesses might not actually need all the bells and whistles offered by my Dovecot/Postfix/RoundCube cocktail. He recommended us to scale our project down, ensure the core elements were functional, before adding any fancy stuff. If I still can't get Dovecot to work, I'd give Courier IMAP a shot. If all goes to hell, I'll resort to external providers such as 25mail.st (have I mentioned Google Apps?), but hopefully it won't come to that. I would very much like to get it running locally so everything can authenticate against the LDAP directory and use the shared contacts and user data there.

For other areas of our site, if our code conformed to any recognizable MVC conventions, it shouldn't be too hard at a later date to add modules for an online store or something fancy like custom libraries for complete API integration with other services (e.g. Twitter, MailChimp, Facebook).

Final Thoughts

If Labour (or was it the Tories who ordered the actual cancellation?) can scrap a £11bn NHS IT project after 9 years, I'm not doing too bad cutting back my wish list. Nobody's being made redundant, and no man-hours has been spent on those bits I'm cutting anyway. Meh.

As for a curious member of my group asking why I opted to use LDAP – it sounds much cooler than plain old MySQL!

D for Database

Since my last post, I've been flooded with questions about my decisions (no I haven't, but so to speak), why LDAP is cooler than MySQL, and my latest strategic review.

D for ~~Database~~ Directory

There's the Wikipedia article which is a nice read (and I won't link to). Within the context of our project, here we go:

- Scalability – in the unlikely event of the company suddenly supersizing, you can start enslaving/replicating more LDAP servers to balance the load. Not quite as relevant, but reliability is something to consider for any business IT infrastructure.
- L for Lightweight – although it won't make much difference in our scenario due to the limited scale of deployment, LDAP has relatively low bandwidth requirements, reducing pressure on the network.
- D for Directory – the data we're storing are contacts and user data, information which are generally suited for directories. Indeed, RFC2798 specifies the inetOrgPerson schema, what is now arguably an industry standard for storing a person object in many organizations. The number of write operations are very limited after the initial population, and like many directory services, read operations are faster than they would be on a RDB. The ultimate comparison boils down to using a directory verses a database, and for the data we are concerned with here, a directory is a better choice conceptually.
- A for Access – because of the proliferation of LDAP instances out in the wild, many other services (including my much adored Google Apps and Zoho Mail) can natively access LDAP directories, allowing them to synchronize the user details from our in-house server. Although we are not going to require enterprise-level infrastructure anytime soon, it's also good practice to future proof anything we build. Five years down the line, the end user might have 100 staff members (ok, so they'll probably perform an upgrade well before this, but anyway). If they wanted to change to a new front-end, or any other commercial-grade package, an LDAP server would integrate quite nicely, allowing their new system to access the same data with minimal migration and data integrity issues.
- P for Protocol – this is purely a philosophical point, but LDAP is based on the ITU/ISO specifications relating to directory services, a common protocol for operations (specifically the X.500 series), and the relevant schemas (such as RFC2798, RFC2307, RFC4524) that is now in common use in enterprises. This complements our aim of using FOSS and open standards where possible, as opposed a closed database which we'd have to write separate APIs for (since we'd be using our own custom abstractions of users to store in the database).

Closed for Business

Drawing another comparison with the NHS project, we've both realized that certain features of the bucket list are unfeasible. In their case, it was due to the financial black hole. In our case, a lack of time. It's much more ideal to cut our losses now than to risk any more man-hours on something that we might have to end up scrapping anyway.

We're in the middle of creating the views and the controllers for business employees, but yet to link them up with the data storage. A prerequisite for an online store would be for the employees to modify the data records on the system, and since that hasn't been completed yet (and I have my doubts as to whether we'd have adequate time and resources for the online store), I decided to divert our resources on completing the core elements of the system rather than risk having an half-finished stock management system and a broken web store.

The PHP system is built on top of CodeIgniter, a MVC-oriented framework. If all our code has followed the recommended conventions, separating methods into the appropriate classes and custom libraries/modules, and we manage to complete the employee interface, there's no reason why we can't work on a customer front end. That could range from an API for other software to work with our backend to a fully-fledged online store, depending on time constraints. I've put all related tasks (research and plugin installations etc.) on the back burner indefinitely, but that does not rule out bringing it back if we manage to complete the core elements with time to spare. The design of the database structure and framework organization has been chosen with considerations for modifications and additions such as this, but these addons are useless without the necessary management interface for the staff being implemented first.

Onwards and Upwards

After much debugging dovecot, and being provided with a very detailed error 89, I've finally figured out the source of the problem. Contrary to as what may appear to be the most common issue of missing closing curly braces, it was broken SSL settings that kept killing dovecot. In a rage of fury, I disabled it altogether. Now onwards and upwards, carpe diem!

The Joy of Logs and Documentation

Having been on a hiatus for a fortnight, one of which wasn't the most productive, and the other very frustrating, we now appear to be making steady (albeit slow) progress. This will be a rather short post, just to update on my recent problems in the areas I've been working on, and what works.

RoundCube

The email client can now send emails successfully. The login (indirectly) is based on the LDAP user list. Shared contacts are available under one of two categories – 'Business Contacts' which queries all the contacts in the LDAP directory and 'Internal Contacts' which shows people within the organization, i.e. other employees. It has primitive write capabilities for some of the fields for the business contacts. However, not all the fields are available (only names, email address, and phone number) due to a problem with the field mapping. A related problem has already been noted in the original comments, and certain field mappings have already been acknowledged as broken. Thankfully, this will not be a cause for concern since contacts will primarily be managed in our main frontend.

OwnCloud

OwnCloud 5, released only weeks prior, was installed under the 'owncloud' subdomain. Again, this now authenticates against the LDAP server. Although OwnCloud has a large range of features, we are only using this primarily for file storage. It has capabilities for Calendars and Contacts, but both of these require preexisting WebDAV servers, and the latter does not yet provide support for LDAP-based address books.

Dovecot/Postfix

The first step consisted of setting up Dovecot IMAP. Since each user will not have a corresponding UNIX account, I set them up as virtual users. The settings required were not difficult to set in itself, but the way the configuration files were stored were counter intuitive. At the top level, the /etc/dovecot/dovecot.conf is the primary configuration file. At the top of the file, this references /usr/share/dovecot/protocols.d which sets the enabled protocols, but this is overridden in one of the sub config files in /etc/dovecot/conf.d/ at the end of the document. Furthermore, half of the config files within conf.d are automatically included, whereas the rest sub-included (mostly under 10-auth.conf).

I opted to use a static userdb since each user wouldn't require any specific settings nor their own PAM account. Once IMAP login was set up, it was relatively straightforward to tell Postfix to use Dovecot as a SASL authentication provider, mostly because the postfix configuration was all done in one file. From then on, due to the loose security nature of SMTP, I could send emails from my account in RoundCube fairly easily, even though they often ended in the spam of my other email accounts I used to test.

I then had the dilemma of using LDA vs LMTP to receive messages. I opted for LMTP since it appears to be a newer technology (more scalable), and would not spawn a new process for every message received. What really took my interest was how LMTP would allow for 'dynamic address verification', so Postfix wouldn't need to check if a user existed from the LDAP directory before transferring it to Dovecot (which would do the same thing again). Sadly, the documentation didn't make it clear that it wouldn't allow authentication binds (I may be wrong, but this is what seems to be the case). (It actually does say so, but not where it mattered, in the tutorial where it tells you how to set up dynamic verification with Postfix, which I read and reread for hours.) Dovecot can authenticate against LDAP using 'password lookups' or

‘authentication binds’. While the former is faster, the latter is more secure since the actual authenticating is done in the LDAP implementation.

There are two areas which I’ve to integrate Dovecot with Postfix. The first was the authentication which would allow Postfix (indirectly) and Dovecot (directly) to check usernames and logins with the LDAP provider. The second is to allow the MTA (Postfix) to receive mail for the MDA (Dovecot, which annoyingly also calls the LDA). On the second integration, I spend almost a week debugging the multiple problems. These consisted of:

- Just generally not working. (I went through the process of testing if the LMTP server was actually working/listening, and if postfix was even trying to connect to an LMTP/LDA server)
- The LMTP daemon not starting. (It was enabled at the start of the configuration file, but somewhere in the subsequent sub-config files, another instance of the setting disabled it again. I also tried opening the ports, and switched between listening ‘inet’ and ‘unix’, to little avail. Setting up a log file wasn’t useful since the problem turned out to be on postfix’s side, and there were minor file read/write permissions problems with it.)
- Emails were always getting bounced with user not found errors. (It turned out this was because Postfix wasn’t even connecting to the LDA)
- Emails were getting bounced for other reasons. (Trying to debug the above problem, I started meddling with smtpd_recipient_restrictions and the likes, which also broke the SMTP email sending.)
- Using the virtual_transport without reading the related documentation (which stipulated that virtual_mailbox_domains must also be declared for it to work)
- Declaring a new service in postfix/master.cf for LDA whilst listening for LMTP connections in Dovecot. (I also ‘hacked’ the master.cf entry to attempt to initiate the LMTP service instead of LDA, without realizing that LMTP operates as an always-on server spawned from the Dovecot master process while the LDA is run once per message when Postfix receives it. It turned out that a separate wasn’t required at all since lmtp is an available option by default and virtual_transport adequately does the job)
- Ran into repeated "User unknown in local recipient table" errors. (Ended up disabling local_recipient_maps entirely.)

As of now, it can send emails, but can’t receive any. Since I now get “passdb doesn't support lookups” errors, I will try to convert to password lookups instead of authentication binds this week. This will save multiple requests to LDAP from both Postfix and Dovecot, giving a (minor) performance boost.

*Update 2 hours later:

Having changed from authentication binds to password lookups, the log files then showed Postfix receiving the email, then passing it to Dovecot to lookup. However, it kept looking for the user@domain.com instead of just 'user'. Fortunately, there's a convenient setting (auth_username_format) which allows Dovecot to automatically strip out the appended domain name if it exists. With everything working perfectly, I appended 'reject_unverified_recipient' to 'smtpd_recipient_restrictions' for Postfix as recommended. I then ran a short email conversation with another email address, including one where the account was receiving an email cc'ed rather than directly sent to it. Except for a few minutes delay and mixed up email orders, the short test went smoothly.

Presentation

Yesterday was the presentation to industry reps for the project. While the marking criteria suggested that this would take the form of a presentation of report on how the group worked, we found out a few days before that it would be more of a pitch, to mark the USPs of our project.

We originally had multiple versions, which we then trimmed off a lot of content, and made way for some screenshots. If anyone was there listening to my presentation of the development and evaluation of the project, they might have found my speaking too fast – I was trying to stick to the 10 minute limit religiously, and in our trials (before we trimmed off some fat), we were almost 3 min over the limit.

Anyway, below are the original slides (slightly adapted, since the version used in the actual presentation was an offline backup), complete with the speakers notes.