USING DIRECTORY SERVICES FOR APPLICATION INFORMATION

Abstract
This paper will discuss the work done at UMBC in the past year to create an institutional directory service and the applications that now utilize the directory. In particular this paper will highlight the following points:

- The business factors driving this initiative
- How we got involved in developing directory services
- The directory development team and process
- Development and deployment of new applications using the directory service
- Creation of a single sign on web authenticator
- Applications that were developed on top of these services
- Integrating WebCT and Blackboard course management tools

First, I would like to provide some information about the University of Maryland Baltimore County (UMBC). UMBC is located in the Baltimore/Washington corridor 10 miles from downtown Baltimore. UMBC was established in 1966 and has a FTE enrollment of 10,800. We have a Carnegie classification of Research Extensive and our mission is focused on providing a strong undergraduate education in the liberal arts with a focus on the sciences, engineering, and public policy at the graduate level. UMBC has experienced rapid growth over the last 10 years, especially in research funding and enrollment. UMBC is known nationally for the Meyerhoff Scholars Program, where minority students are mentored and prepared for PhD programs and for our heavy emphasis of IT majors. Almost 40% of our students major in Computer Science, Computer Engineering, Information Systems, or Imaging and Digital Arts. Academically, UMBC is in the upper quartile of public universities with an average SAT of 1190.

Because of our young age and size, small by research universities, UMBC has focused on development of flexible central services and has decentralized services such as IT to departments. As a result, we can move quickly when we see an opportunity and do not have the same issues of negotiating turf found in some larger, less centralized universities.

Business Factors Driving the Development of Directory Services

UMBC has made a push to start our first online program in 1998. Once you start dealing with students located in nationally and internationally, which never set foot on your campus, you realize the necessity of making campus services self-service through the web and easy to use. In the spring of 1999, as part of our IT planning work, we met with
faculty in this online program and others using technology enhanced learning tools to identify their needs. This resulted in three priorities:
Revamp our existing web-based administrative services and create a web portal that would provide all services in one place for students.
Revamp our account generation process, which was online but required students to show ID when they picked up their account, to allow students to create their account over the web and never have to set foot on campus.
Better integrate our course management tools with our institutional systems to eliminate the work of creating and distributing accounts to students and enrolling students into their appropriate course. Creation and distribution of accounts was identified as the major impediment by faculty using these course tools in large sections.

Associated with this planning activity was a review of our administrative business systems. Our SIS and HR systems were developed on campus under a HP 3000 environment and were now showing their age. While we had good transaction capabilities we had limited tools for reporting and the systems were not integrated well in terms of data definitions. As such, it was felt that UMBC should begin investigating other options such as SCT, Oracle, and Peoplesoft for replacing these systems now that we were ready for Y2K. We anticipated bringing up a new HR system in the 2002 and a new SIS a year later.

In fall 1999 we launched our web portal myUMBC. This was extremely successful and we began to get inquiries for providing access to myUMBC for alumni, prospective students, and parents. We knew this would create a challenge to us to revamp our authentication and authorization services. We rely heavily on a Kerberos infrastructure for authentication to campus services. This system supplied authentication services for computer lab workstations, modems, portal, email, shell access, and a number of other services on campus. The issue for us was that many services we provide were automatically granted once you had a Kerberos entry and we knew that we didn’t want alumni dialing into our modems if they were given access to myUMBC. As a result, we had to develop a finer level of control for accounts and develop a set of global attributes that could be queried by any application service to determine whether to allow that person access. At the same time LDAP directory services were becoming more widespread and we saw that as a way to solve the larger problem of access control.

At the fall 1999 Internet 2 meeting I attended a session on Middleware given by Ken Klingenstein and it was mentioned that Internet2 would be looking for schools to work on a “Middleware” implementation project coordinated by Internet2. UMBC applied for one of the slots and was selected to participate in December 1999.

Directory Services Development and Planning
Once selected by I2 to participate this gave reason to create a team and begin planning our next steps. As Director, I had written the application to Internet2 and begin the executive sponsor and evangelist for the middleware project on campus. We created a team of about a dozen people that represented all facets of IT (e.g. helpdesk, faculty support, database administration, SIS, HR, networks, LANS, and Unix systems administration).

The project team was charged with developing the project phasing, timeframes, and development plans for integrating our legacy data into the directory. For those schools entering into this area this phase was one of the more problematic. One challenge facing the team was that each group had a different take on directory services and how it should be implemented. Our LAN group wanted Novell’s NDS selected as the campus directory, our applications group wanted Oracle directory server and the Network group wanted a product supported by Cisco such as active directory. In developing our plans we created a group headed by the Director to review products and make a selection. Factors the group had to consider were: cost, cross-platform support, flexibility, scalability, and reliability. In the end we decided that iPlanet’s directory server had the best mix of features but the cost was too high. Our second choice was Innosoft’s directory product. Luckily as we were negotiating with Innosoft we found out that Netscape would be buying the company and customers would be given Netscape’s iPlanet directory server in lieu of Innosoft’s product. In late March 2000 we purchased the Innosoft directory server and we were send the iPlanet directory server in April.

(Note: Sun iPlanet now provides a low cost directory server to education customers).

In April 2000 we ordered three Sun E-250 servers to run our directory service. Each server has 2GB of RAM and has served exceptionally well the past year.

A second issue that arose was the question of what system would be authoritative and whether we would let users update the directory with information where the directory service was not considered authoritative. Some in the group felt this is the best way to utilize directory services and by providing an easy to use way for people to update their information we would have better data. Implicitly this makes sense but the challenge faced is one of getting buy-in from the campus data stewards to do this. While we were certain we could reliably sink our directory data back to our enterprise databases we also knew that the data stewards were very concerned about this issue and it would slow up development while we negotiated access issues. As a result we decided in April 2000 that the directory would only be authoritative and allow updates for data that the directory is responsible to maintain (e.g. username, email, account information). All other data contained by the directory that is updated will come from the system of record. Updates to that data will be handled through our web portal to the authoritative system and then distributed back to the directory. With that decision in place the data stewards gave us permission to move forward with no further review.

Once these decisions were made we went about architecting the directory. A number of good resources now exist for doing this, notably Michael Gettes LDAP directory recipe.
Working within the I2 middleware group provided feedback from others that were involved or had already done this type of work. As such, we went with a flat hierarchy that had all people in a single level and we assign attributes to let us know what roles those people have at the university. A second decision we made was to load all students who had ever applied to UMBC, approximately 275,000 people, so that we could see how well the directory service could handle a large number of objects.

The directory architect worked with our database administrators and SIS and HR managers to identify what data from our systems should be moved over. We decided early on we would focus on consolidating data such as addresses across these systems. When duplicate data, such as address, was identified what to use by examining update timestamp to determine what data should be placed in the directory.

One decision we spent a great deal of time on was the question of whether we should create a people registry under Oracle and then use that registry to update the database. There are some advantages to this approach, especially if your data sources are coming from disparate databases. In our case, creating this would have been quite easy; we had moved all of our institutional data moved over to Oracle as part of our portal project. To support the portal project we wanted access to the portal to not impact our transactional databases. As a result, UMBC had purchased software that supports replicating data from HP 3000 Image databases to Oracle. We had used this software to create a real-time shadow of our SIS data running on Oracle. In the spring of 2000 we had rewritten our HR system from SQLserver to Oracle and all that data was now available on Oracle. Since all the data we would need was already on our production Oracle database we did not see the value added of creating the people registry.

Instead, we worked with our DBA to do a one-time bulk load and to develop database change logs, each time a record is updated it is written to the change log. We have a process running on this server that constantly reads the change log and updates the directory automatically. As a result, our directory is updated within seconds of an update to the transactional system.

Finally, during this phase of development we decided that since we would not allow people to directly update their directory data through off-the-shelf directory tools we needed to develop our own system to control this. This would mean that each application that used the directory was responsible for maintaining security. This had the potential to be a nightmare and so we looked at how we could authenticate to the directory and control access. The result was the development of Webauth.

Providing Access to the Directory – WebAuth

UMBC has been utilizing Kerberos with great success for the last decade. As a result, we have developed a great deal of expertise and respect for the system. In addition, all of our users presently have Kerberos principals and passwords and we began discussion web
application security in light of “wouldn’t it be nice if we could develop a Kerberos like system for the web.”

WebAuth works on a principle very similar to Kerberos, in that it relies on an initial authentication to establish a Ticket Granting Ticket (TGT) that is stored in the browser. The TGT is later used to generate Service Tickets (ST), which are presented to web services as authenticators. These tickets are stored as Cookies on the clients’ web browser, so no other browser plugins or software is necessary to use WebAuth, with the exception of a cookie-enabled browser.

An Example Session

1. User is browsing web site, and enters an authenticated portion.
2. Instead of content, or error message being displayed, a Redirect is returned, sending the user's web browser to the authentication server. Encoded in the redirect is enough information go 'get back' to where the user was browsing, and the service that needs to be authenticated.
3. At the Authentication Server, one of two things happen. Either the user has already authenticated to and received a TGT, in which case this is verified, and the appropriate ST is issued, or
   The user is presented with a screen asking them for their credentials, after which a TGT and ST are issued.

Using the information encoded into the redirect, the user is transferred back to the originating site, with authentication credentials in hand, and their browsing experience continues.

Ticket Format

The ticket has the following format:

cookiestring = base64(3deskey(ticket, md5sum(ticket)))

ticket is the plaintext ticket string, and skey is a service key -- a shared secret between a particular service, and the webauth authorization server.

The format of the string ticket is a set of colon-separated text fields:

- The service name. (e.g. webct)
- The requesting IP address (the IP that this ticket should be valid from -- where the user's browser is running)
- Time issued -- a UNIX timestamp (integer)
- Lifetime -- in seconds
- Username -- the authenticated username, DN, etc.

What can also follow are multiple optional name/value pairs (delineated by equal (=) signs), which are currently usually:
• **authentic** - authentication type used to get this ticket
• **ptype** - principal type -- e.g. 'username', 'ssn', etc.

These can be infinitely extended in different authentication modules, and should not affect compatibility with older clients that do not know about specific fields.

**WebAuth Usage**

WebAuth requires an Apache module that replaces the Apache Basic Auth module with one using WebAuth. With WebAuth in place we now had a mechanism that could be used to authenticate people and be used across a wide range of web-based applications without requiring the user to reenter a password each time. In addition, by application we can enforce different levels of security (e.g. type of authentication, IP address must match, time to live, etc.)

One example of the flexibility of webauth is that we use this for students to generate their account. In this initial instance we can’t require students to enter a username and password because they don’t have one; however they do know their SSN and Pin number (the pin is formed by using part of their birth date). A student that wants to generate an account goes to our page accounts.umbc.edu, enters their ssn/pin combination, is authenticated by WebAuth for one service, account generation. The person is then redirected to the account generation website where the person can generate their account. If they tried to use WebAuth for other functions when they were authenticated with just a ssn/pin they wouldn’t be granted access.

UMBC has developed interfaces for Java and Perl to utilize webauth. For more information on WebAuth please consult the online information, [http://umbc.edu/~jack/cumrec/webauth-info.htm](http://umbc.edu/~jack/cumrec/webauth-info.htm). Please note, WebAuth is not the only application to do this and Internet2 is working on a project with IBM named Shibboleth that will provide an application very similar to WebAuth.

**UMBC Directory Applications**

**Online Web-Based Account Generation System**

Our first application that we brought up was a new web-based account generation system. This application was released August 2000, four months after receiving the iPlanet directory software. This application allows people to select their username, enforces they select a strong self-selected password, and allows them to setup a number of global account preferences such as custom email names, mail forwarding, and Unix shell preference to name a few. This system also provides our Helpdesk the ability to generate an account for a user, update their preferences, and increase disk quotas.

This account generation system met the second goal of creating an easy to use account generation system that allows a student anywhere in the world to come in and create their
account and begin work at UMBC. This system also provided a proof-of-concept of WebAuth as a mechanism to authenticate web-based applications.

While WebAuth and the new account generation system have been well received it was not without some pain. In retrospect, we probably should of waited a few more months and not launched this at the start of the fall semester. While the application worked well it did create a number of changes and this caused some confusion on campus that could have been lessened if we had done this changeover in January. The decision to go was predicated on the belief that UMBC would launch a new online masters degree in the spring 2001 timeframe with high enrollment projections. As such, we wanted to begin work on integrating the course management tools as quickly as we could.

Blackboard 5 Integration

UMBC purchased Blackboard 5, level 3 in July 2000. The faculty developing a new online graduate program in Information Systems would use this software. Because this program was a high priority on campus we focused our initial work on integrating Blackboard 5 with our existing systems. Level 3 contracts require a minimum amount of consulting support from Blackboard, as such, there is a formal project-planning meeting to outline objectives and develop a project plan. UMBC met with Blackboard in October 2000 to plan our work. Phase 1 of the integration was integrating users and utilizing our assigned username/password combination on blackboard. We developed a plan to load usernames from our directory on a daily basis, more so during the first 10 days of the semester, and integrate our webauth system into Blackboard 5. To minimize cost, UMBC wrote java classes that Blackboard could call to handle the redirect and extract the ticket information. Blackboard consultants modified the Blackboard product to make these calls in lieu of their regular authentication.

UMBC had hired a consultant to write the Java classes and this person failed to deliver the code in the required timeframe. We then went to a different consulting company in the area and got this done. The result of the initial failure was that we were one month late in delivering our code to Blackboard and didn’t get this operation till two weeks before our spring semester. Since Blackboard 5 had a few features that some faculty wanted very badly we allowed classes to “beta-test” Blackboard 5 with the new authentication. We had 15 active classes use this in the Spring 2001 semester.

One problem that was encountered was that WebAuth verifies the IP address of the client is the same as the one that got the initial ticket granting ticket (TGT). In going live we had sporadic complaints that people could not authenticate properly. All of these users were connecting from off-campus; however they used many different ISP’s. That said, many people connecting from off-campus seemed to have no problem. This problem was traced back to the practice that some ISP’s allow connections to an SSL server, such as WebAuth, to go through unaltered but regular HTTP requests go through a cache server. Since Blackboard using regular http requests the IP address of the computer would match the ISP’s caching server and not the IP address inside the WebAuth ticket. To get around
this problem we had to modify our WebAuth implementation to make the IP check optional. Once this was done, connections worked fine.

A second problem we encountered was how to provide Guest access. Inside Blackboard faculty can allow generic guest access to their course. This doesn’t require a password and users can view the course materials. Blackboard consulting probably should have put this test into the code they provided but didn’t think of it. As a result, we were faced with the issue we couldn’t provide guest access to Blackboard. The end result is that we wrote a special hook into WebAuth to treat guest as a special account name and not require a password check. If a user logs in as guest they are returned back to blackboard with a guest username and can then enter the course.

The last issue we faced in this implementation is that some faculty wanted to have the option of allowing non-UMBC people to take their courses. This is still to be resolved, for now it is being treated partially as a policy question: should the institution allow people not associated with the campus to take these courses without paying? We have requested our faculty oversight committee review this issue. In the meantime, we grant temporary accounts to those individuals that faculty want to be entered into their course. However, faculty must make a formal request and give us the name and contact information of the person in question.

Presently we update the Blackboard user and class databases through the batch update facilities build into Blackboard. Information is extracted from the directory and used to update the Blackboard server. We would like to get this to be done closer to real time but we want to first see if this goal is worth the consulting cost. Students who register after the semester starts have the option of enrolling themselves in the course (if the faculty member allows this) and so this is not a pressing problem at the moment. It is likely we will hold off on this till we have completed our Peoplesoft SIS implementation.

WebCT Integration Efforts

The effort to integrate WebCT 3.1 with our directory systems and WebAuth was much easier than we anticipated. One reason is the WebCT 3.1 utilizes a number of technologies we had strength in: Perl, Apache, and Unix. For authentication WebCT utilizes the Apache basic auth module. We replaced that standard Apache module with our Apache module that utilizes WebAuth. Will a few modifications to WebCT to make sure the redirection worked this were working within a business week.

WebCT has defined interfaces for updating the DBM files that hold the user list and course information. We have written scripts to extract the user list and course registrations from the directory and update WebCT 3.1.

We did not see the same issues with WebCT because we are just now putting this into production for the summer. We had planned to introduce this for our spring semester but because of an early semester crisis with one of our AFS file servers it was decided to stay with the existing WebCT server for courses this spring.
As a result of the Blackboard implementation we have addressed the two issues we saw with Blackboard and expect this implementation go smoothly this summer.

Like Blackboard, in a perfect world we would like to see students added to the WebCT course roster as soon as the student added the course. Students who register after the semester begins do have the option of enrolling themselves in the course (if the faculty member allows this) and so this is not a pressing problem at the moment. It is likely we will hold off on this till we have completed our Peoplesoft SIS implementation.

**MyUMBC Portal**

Our myUMBC portal has been very successful; the challenge we face is we have more demand for new applications than we have staff to write the applications. Because of the way myUMBC was implemented new applications that wanted to use the myUMBC authentication system had to developed by the myUMBC team. What was needed was a way to extent our authentication framework but still integrate applications into myUMBC. To do this we have developed a beta version of myUMBC that authenticates to WebAuth and gets a TGT cookie for the user as part of logging into myUMBC. With this TGT the user can now connect to any application that utilizes WebAuth for authentication services. The benefit of this is that a student can now log into myUMBC, and connect automatically to Blackboard, WebCT, or any other application that uses WebAuth. This change will go into production this summer. Once into production departments and other groups that want to create WebAuth applications can utilize our authentication infrastructure.

**Results to Date and Work to be Done**

The directory service has had 1 hour of unplanned downtime the first 9 months of production. The fact we have two replicated servers provides a level of reliability that parallels that of existing Kerberos Domain Controllers (KDC), which measure downtime in minutes per year. Over the summer we will implement multiple WebAuth servers to function as backup should there be a problem with a WebAuth server. This is necessary because WebAuth is becoming as fundamental to our computing environment as the KDC or DNS. We have to make sure that WebAuth maintains the same level of reliability that we expect from all critical production services.

Faculty using the beta-version of Blackboard 5 that utilized the WebAuth authentication have expressed support for this work and reported that the students found it much easier to get started in the course. Looking beyond formal course tools as part of our WebAuth plans we will be providing faculty with the ability to easily setup web-based ACL’s to control access to their course. Faculty will specify a group ACL, any person in that directory-defined group will be given access to the resource.
Using a directory allowed us to utilize our institutional data in an academic context. The staff that did this would never be able to directly access and update our SIS tables. Through the use of the directory and WebAuth we have added a wide range of new self-service web-based applications. These applications have been well received and will become easier to create through this new framework. In one year, the directory service, coupled with the development of WebAuth, has changed the way we approach the development of web-based services at UMBC. We have seen the benefits of a single sign on (SSO) for web-based applications and are moving forward to better integrate our administrative and academic applications.