Identity Management Experiences

Jon Colombo & Keith Awcock

Introduction

Identity Management is attracting attention from regulators, authorities and auditors. This is forcing Information Security Professionals to confront the problem of how to keep track of large numbers of rights across numerous systems and applications. They have to be able to evidence all the Accounts (logins) anybody holds, prove that profiles associated with them are correct, show that every Account on every system is owned by an authorised person and guarantee that the authorisations have not lapsed. This is a major exercise in record keeping. This paper aims to help practitioners understand the problems they will encounter so they can make informed decisions. It will be followed by one describing the architecture of a technology developed to address the problems.

Background

Four years ago we set out to create a record of people, their Accounts and profiles across over 100 systems and applications. The aims were to:

1. Centralise administration and enforce common record keeping standards. In particular the intention was to ensure timely deletions for leavers and identify all unauthorised modifications to Accounts.
2. Identify people who have acquired dangerous combinations of rights, remove these and prevent future reoccurrence.
3. Provide a foundation for a log analysis system for correlating people's activities across multiple systems.

Other significant factors were:

- The large variety of systems. There were no common management standards, different security models, security requirements, reporting levels and platforms. Importantly bureau services and other systems outside our direct control were also in scope. Any reliance on 'agent' technology was impracticable.
- Our reporting line was such that there was no possibility of implementing a single staff ID across all the business units that would be managed by the administration function.
- There was a problem getting management buy-in to security issues, exacerbated by memories of big projects that had petered out elsewhere in the organisation. Together these meant that there was neither money, nor political will to implement a big security project.
- Resources were limited and had to be tightly focused on Identity Management. Side issues, such as data cleaning would not be dealt with.
- There was a total lack of commercial products on the market that could cope with the requirements. (To some extent this has since improved).

These factors meant that the only feasible route was to develop our own architecture. A lot was learnt.

Model

The approach followed was to borrow the principle of double entry bookkeeping from accountancy. All that was required was data on:

- Who should be on each system – i.e. those users who have gone through the official process to get access rights and who have not had them rescinded, or left the organisation – and what rights they have been authorised to use.
- Who actually is on each system and what they can do/are doing. This information can be obtained in two ways; an output of the user list from the system, or, less sure but nevertheless effective, by checking the logs for the system to see which users are using it.

A simple reconciliation would produce reports of discrepancies. In reality, a more complex model is needed – see Figure 1 (next page).

The 'should' side of the equation is best considered in two parts: authorisations to use the system, and removals of rights.

1. *Authorisations* – hopefully there will be records of those who have been authorised to use each system through some form of Account application process – 'Authorisation Data'. Typically these will consist of the original application forms, or the administrators records derived from them.
2. *Removals* – identifying those who should be removed from a system is usually a much harder task. Most people will not apply to have an Account removed when it is no longer required 'because I might need it sometime', but this may represent a control risk. 'People Data' will be needed to deduce

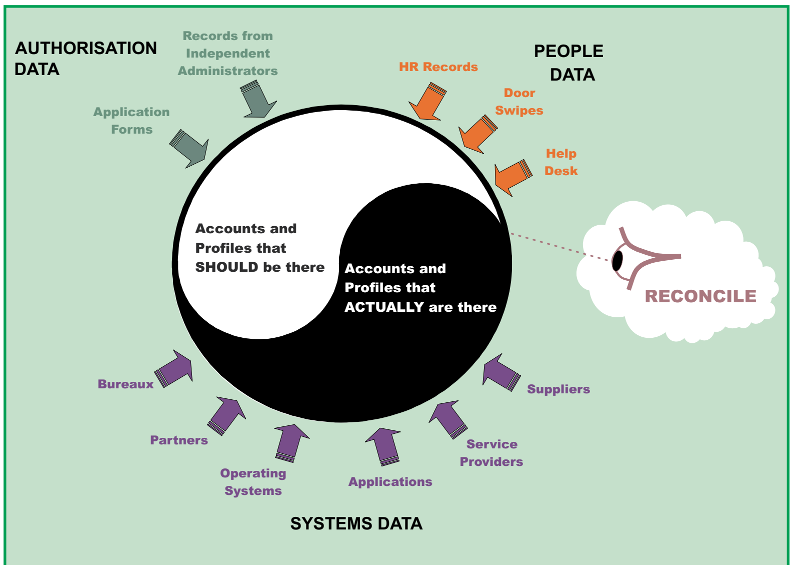


Figure 1 - **The Model**

where authorisations should have expired. i.e. where the owner has left the organisation or their role has changed significantly. 'People Data' may be obtainable from the Human Resources (HR) department. It may be necessary to supplement it from other sources if the systems under administration have users that do not come under HR's remit.

'Systems Data' will come directly from the systems under administration. The variety of systems and power relationships are important as they will determine the ability to influence content and delivery.

Once both sides of the equation are assembled, then reconciliation can take place. Reconciliation should identify not just Accounts that lack corresponding entries, but also variations in profiles. i.e. where Accounts have gained (or lost) rights without authorisation. If reconciliation is frequent, then it provides a crude IDS. Intruders can create additional accounts or modify existing ones to ease their return. Such changes will become readily apparent.

As data is assembled, searches for people with dangerous combinations of rights becomes more realistic. Many business processes rely on technical measures to enforce segregation of duties. e.g. in cheque processing, a complementary pair would be 'input' – create the cheque, and 'release' – sign it. Someone may bypass such controls if they acquire two Accounts. Complementary pairs of Accounts are usually on a single system. However, in a large complex environment with many interfaces it may be possible for dangerous combinations to exist across systems – input on System A that feeds into System B, release using a System B Account.

Pitfalls

Once serious planning started, pitfalls emerged and more became apparent as the work progressed. In hindsight it is clear that these will be common to all such projects.

Personnel Records

The HR department's leavers' records could only be relied on to a certain extent, simply because HR have different priorities.

- HR may have a different perspective on the definition of 'staff', one that only includes people on the payroll and 'official' temps. Security administration needs to record consultants, visitors, secondee's, third party hardware engineers, staff visiting from overseas – anyone who may use corporate computer systems.

- During busy periods, HR 'batch up' leavers and joiners, only entering them into their system just before the payroll run.

- Leaving dates tend to reflect end of payments. i.e. end of contract, not last day in the office.

- HR may not renew temp contracts on their database until the paperwork is complete – so end-dates appear and disappear.

Data produced by an HR department will reflect HR priorities, not security ones. Information Security will not be able to change these, but by getting to know the people in HR and understanding their procedures, workarounds can be found.

Other people can help fill the holes in HR data. Door-swipe and Help-desk are good starting places. The owners are delighted to trade information and as the data store is designed to reconcile many systems, the extra overhead is small.

Leavers Information

If all the information needed is not available from one source it will be necessary to stitch together several sources. This can create problems:

- *Overlapping data* – there are frequent temp-to-perm employees, occasional perm-to-temp and even temp/perm to consultant, or visitor. If data on each comes from different databases, tracking peoples changes in status can be challenging.

- *Accuracy of data* – if HR say someone is leaving, can they be believed implicitly? Usually not. What if Premises Security think so? How about the Help desk? Can the telephone list be trusted? If they disagree, who should be believed?

It is essential to build a system that can cope with levels of trust using the most trusted source of data available. i.e. for leavers the order may be:

1. HR's records of permanent staff
2. Premises Security records for all staff
3. HR's records of temporary staff.

If HR believe someone has left, but Premises Security do not, then the decision rests on whether the person is permanent or not.

Names
Matching names from different data sources requires careful thought:

- *Acronyms* – some people use nicknames, abbreviations, or local spellings of their name (William = Bill, Parakesh Kokoran = PK, Stefan = Stephen). They may vary their use of these depending on the formality of the situation. Thus, HR records may have the full name, but Account applications have the alternative. Matching names becomes very difficult.

  Consider having both forename and 'also known as' fields and use both when matching.

- *Same names* – it is surprising how often there are people who share a forename/surname combination in even small organisations (John Brown, Paul Smith, etc.).

  Introduce mechanisms and processes that hunt for these and split them out.

- *Name changers* – in a reasonably sized population there is a constant flow of people changing their surnames (usually marriage or divorce) their forenames, or even both at the same time. How can a computerised system figure out that the person who used to be called Penny Green is now called Penelope Blenkinsop? Don't rely on all records changing the names simultaneously. There will always be a long cross-over period.

  Again, good relations with HR help, but there must also be a well thought out process. Also consider having both surname and 'also known as' fields and use both when matching.

- *Returnees* – is the Barry Smith in Finance the same person as the Barry Smith who worked there a few months ago, and does it matter? (Answer: Yes, when working through the historical backlog of systems). Should he get his old Accounts back, or have new ones?

  No simple answer here, but good relations with HR can help provide tip-offs on individual returnees.

Depth of Model
Auditors may look at the data model and demand that not only must the authorisation for each person on the system be recorded, but also that their profiles (authorities, groups, permissions etc.) must be tracked. This logic can spiral out of control. The next demand may be to record the menu options that attach to each group; the next the screen options that the menu options give access to; then the field options on each screen, etc. We have seen this implemented elsewhere and it created a maintenance nightmare.

Keep the data model as simple as possible. Two layers is ideal. The mapping process during take-on may be harder, but payback will be in operational efficiency.

Differing Security Models
Any reconciliation system has to take into account the fact that no two manufacturers will design their security models in the same way, even when they do the same job. One may have compulsory grouping of users, each belonging to only one group, another may allow membership of multiple groups, whilst another may even create a 2 or 3 way group/rights matrix.

Do not try to emulate all possible security models in the data store. Keep it simple and map other security models to it.

Differing Administration Standards
Individuals can apply different approaches and differing standards – for many systems the records of previous administrators may be haphazard, or non-existent.

The security team may have to create the 'should' records from scratch, so factor this into work plans. A sensible estimate is around 30% of systems taken on.

'Generic' and System Accounts
Most systems have some shared Accounts. For any system over a couple of years old, their precise use, history and ownership details are likely to be vague.

Tracking the ownership of these is surprisingly time consuming. Factor this into work calculations.

Poor Typing
Even where names are maintained that should nicely link into HR records, a single misplaced character can throw computerised matching. Is Jon Colombo the same person as Jon Columbo, or Jon Colmobo? With an unusual name, many will not see a mis-spelling. The administrators may not see the spelling mistake and Jon may be so used to having his name mis-spelt, he does not raise it as a problem.

Do not rely solely on the 'Name' field in administration records as a key for imports. More complex filtering is needed.

Naming Conventions
If systems have been administered in different departments, they may use different identifiers. i.e. Keith Awcock can be "AwcockK", "KAwcock", "k.awcock@us.com", even "Kevin" or "Q000220". As conventions may change over time, there may be oddities lurking even in well run systems.

Do not rely on a simple conversion schema to make imports work, instead look for other keying information.

Different Worlds/Different Languages?
A mainframe analyst will use the same security term, (i.e. profile, or group) as a network or Unix specialist or owner of an application. They may even use it in roughly the same context. Unfortunately, each will have subtle but important differences in practice.

Use metaphors to describe security models during analysis. A useful one is that of a House – an Account maps to a key (rights) to the front door and may let people into room(s) in the house and deeper into cupboards, boxes, envelopes, all of which require additional rights.

Account Inheritance
Inevitably on some systems ownership of accounts will change. e.g. an account that is required to run a particular report will be passed on as staff move.

Start and end dates should record ownership of Accounts, not creation and deletion – thus 'FinanceRpt' was owned by Jon Downing from 2/9/00 to 2/9/01 and by Keith Hibbert from 3/9/01.

Account Evolution
Accounts change over time. As people progress through the organisation, the privileges they have on systems will change. Someone moving from Finance must lose their ability to pay cheques, but may still need to run reports.

Privileges need their own start and end dates to document the evolution of accounts.

Key Recycling
Administrators will say that Account names are always unique, so why not use them as a key on data imports? Consider the problem over time.

Always ask: "Jon Colombo leaves, so his 'JColombo' is deleted. What Account will Jane Colombo get when she joins in 2 years time?"

Allow Non-Real People
Although anathema to good security, shared, generic, and system Accounts do exist on most systems and have to be tracked; Looking at commercial products now available, these are not always handled easily.

Make sure the design can cope. Assigning such Accounts to 'virtual people' is a good way to group and track them.

Transmission Mechanisms
Regular, reliable and secure deliveries of data are needed. Details will include personal information and sensitive details about the configurations of machines. Delivery mechanisms for data from applications, mainframes, mini's, networks, databases etc. have to be derived.

Do not underestimate the amount of time, negotiation, patience and sheer persistence that this will require. Also consider the security of data at all points on the route, as although this is a security application, those providing the data are not security people.

Time Dependence of Data Capture
Do not assume that data will always be there on time. The agreement may have been that it would be there at 8 o'clock each night, dropped by an automated process, but even automated processes go wrong.

The system has to be able to deal with non-delivery. Simply stopping all other imports and waiting for the data to appear is not an option, but to keep retrying for a reasonable period may be.

Sequence Matters in Data Capture
Some imports work better if they happen after others. E.g. application imports should occur after 'People Data' ones. That way new joiners will already have trusted records in the datastore to link to. So their new Accounts will not create anomalies.

Transmission Hiccups
Inevitably, from time to time key deliveries will fail (i.e. HR reconciliation data). If this happens the reconciliation engine may continue loading up data, creating erroneous entries in the database. If there is automatic reporting, then it could send garbage out right across the company, or worse, terminate accesses for whole departments. Although the fault was someone else's, your credibility is dented.

Try to build in automatic 'blocking' of reports when bad data is suspected.

"Fiddlers"
Related to, but actually a worse problem than the hiccups, is the fact that sooner or later, for almost every source, someone will make a change to any data you rely on. Examples include:

- *Files disappear* – "Oh, you still need that do you" – the run breaks so it is quickly noticed.
- *Updates stop, but the last export is still available* – "Oh, we don't use that any more, we copied it here and change that one now."

– this can take weeks to spot, as it could just have been a quiet period.

- *Get reformatted* – "I just moved the columns around, was that a problem?" – With luck no, the run breaks, without it, very definitely. E.g. updating department numbers with ID's that passed the validation routines.
- *Change their content* – (saving the best for last) "I moved these departments into another database, it seemed tidier, so marked them as left", or "Oh, we've changed the way we do that, now when people are seconded overseas we just mark them as leavers". The effects of this can be dramatic.

The answer is to build a system that reports every glitch so reaction can be quick until the pattern is understood, thereafter it can be filtered out automatically.

Philosophy of Approach

Some basic, early decisions helped navigate the minefield, and their adoption is recommended:

Adaptation

Adapt the system and approach to the environment, not the other way around – for instance, make a point of engineering all interfaces so that data is presented through a buffer layer over which Information Security has control. When the owner of a system changes the data format, the problem can be resolved without altering the core system.

Automate, But Do So Incrementally

Automate and adapt processes incrementally. Often the optimum route will only become apparent part-way along it – the interface started simple then gradually ways to simplify workflow appeared. These functions could not have been predicted without using the system.

Report Errors Only Once

If reports churn out the same problems day after day, they mask new and real ones when they appear.

Be Prepared to Throw Away Work

This may seem counter-intuitive, but when dealing with this level of complexity, throwing away parts of the solution and implementing anew arrives at efficient answers and seems to turn out cheaper than adapting.

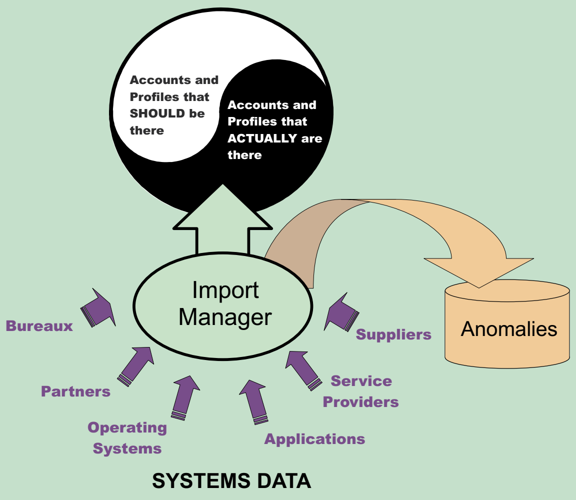
Requirements

Given the above, there are some general requirements that a system to centralise security administration needs:

No Agent Technology

Installing 'agents' to interface with secure or obscure applications will increase costs dramatically. More importantly there are often sound or-

A Word About Filtering



Many of the 'pitfalls' arising from dirty data were overcome by sophisticated filtering. Data is collected from each system using a separate *Collection and Collation* (C&C) routine, which transports the data and creates a file of a standard format. The C&C files are fed into a single *Import Manager* which matches incoming data to that already in the database, and applies filtering rules. The Import Manager reports any discrepancies it finds into an *Anomalies Database* where they can be dealt with by the administrators. It is important that each discrepancy is only reported once, and that each 'Anomaly' shows just a single discrepancy.

C&C files contain two types of data.

- *Account information* – System, Account name etc. – the data that interests the administrators.
- *Keying information* – used to match incoming records, this is other information that is available on the system being imported, first name, surname etc. Ideally, this will include a unique identifier that will never change – a 'key'. Failing that, a 'fingerprint' is created using information on the system about the Account that is unlikely to change. Typically this will be a concatenation of personal data fields available. e.g. account name, contact number and department. Occasionally these will get changed, but most of the time they provide a workable unique key.

Filtering is applied to both types of data, it takes the form of a cascade of questions about the record being imported:

1. Does it have a 'key' that matches one already on the database? Use it.
2. Does it have a 'key' that does not match one already on the database? – go to step 5.

3. Does it have a 'fingerprint' that matches one already on the database? Use it.

4. Could the Account have already been processed, but something has changed on the system, so the 'fingerprint' no longer matches. Match on Account name, and some other data supplied by the system.

5. Assume the Account is a new one. Does it have a parallel on another system that shares the same naming conventions as the system being imported? Match on Account name, and some other data supplied by the system.

6. Is there information from the system that allows the new Account to be assigned directly to a person on the database? Assign it to the person.

… else, create a new person, and assign the Account to them.

The order of questions is generally optimum for best speed and accuracy of import. At each stage, the Import Manager can: import the new data, import it and create an Anomaly, or not import it and create an Anomaly instead. However, both filtering and reporting rules have to be configurable to adapt to normal activity on the system.

ganisational reasons why agents cannot be installed. Consider using reports to 'dead-letter drops'. The administered system outputs a report at a specific time to a specific place. The administration system picks it up from there.

Intelligent Updating of Datastore
The import engine, needs a large (and complex) array of filtering options. The only safe assumption is that data will be 'dirty'. Imports need tuning individually to automatically correct and mitigate many of the problems encountered in individual data sources. The only realistic way is to have adjustable options to compliment the way the owner of the data source works – see insert.

Intelligent Scheduler
Tasks have to be run or spawned in the correct order, at the right time and there must be an interface to set up jobs. It also needs to stop tasks if there is a possibility that the data they depend on has not arrived correctly (i.e. do not send out 'leavers reports' if any of the HR imports are suspect!).

Imports Different Data Shapes
It needs to pick up data from different systems using different transmission mechanisms. It cannot assume that all systems will deliver to it in a pre-set format.

Incremental Data Importing
This is surprisingly tricky. It needs to 'Stitch' data together to allow incremental imports from sources that may write one big file, or dozens of little ones. No matter what the source looks like the datastore should just get the next instalment.

Out of Sequence Data Importing
The import mechanism must check the sequence of data deliveries for each system and when necessary react to and cope with out of sequence data files. As the earlier files in the sequence may contain required data, simply ignoring them is not an appropriate option.

Partial File Importing
It must handle imports where files do not contain all users for a system – some systems only create partial reports of users. For these an automatic "if it's not there, the Account must be closed" rule cannot be applied.

Report All Import Anomalies
Where the import mechanism cannot automatically decide what to do, or trust in the incoming data quality is low, it must be configurable. It can: do nothing and set an alert for a person to see, ("I couldn't figure this out – can you?") or update the datastore and still set the alert ("I've done this, but am not sure it was correct – can you check?")

It Must Journal Its Own Activity
This is useful internally, important for debugging, but most of all provides a full record of work undertaken for audit purposes.

Datastore Comparison Reporting
A good reporting engine is essential for all sorts of reasons. e.g. reconciliation reports, comparing system to administration records to leavers records.

Choice of Output Formats
Printer, email and workflow engines are essential. Print-outs for filing, emails sent directly to managers who requested a report. and the workflow engine to dealing with and recording issues in a manageable way.

Automated Housekeeping
Inevitably databases need maintaining (removing orphan records, out of date ones, re-indexing files, etc.). Automation of these is important to reduce errors.

Provide Operations Tools
There has to be some way to quickly check that all the imports, reports and housekeeping tasks have run to plan each day. Consider a software 'status board' showing the health of each task when last run. e.g. Red, Amber, Green.

Conclusion

Implementing a central security administration control system is becoming a more visible element of a well run information security infrastructure. Setting one up is possible, and it is practicable – but do not believe those that say it is easy. Be careful of what can not be controlled and do not try to take large steps too soon. This paper has documented many pitfalls, but every organisation will encounter its own.

Jon Colombo started work as an archaeologist with degrees from London and Oxford. From there he moved into IT, gaining an MBA from City University in 1993. For the last 10 years he has worked entirely in Information Security, setting up InfoSec functions at United Friendly Insurance and at WestLB AG, London, where he still works. He is a qualified CISSP.

Keith Awcock, graduated as an engineer from Kings College, London in 1989, moved directly into software engineering with Phillips. For the last 5 years has specialised in Information Security, working at WestLB AG in London. He is a qualified CISSP.