



NOTICEBORED

The creative information security awareness service

Information Security Awareness Briefing Pack for Software Developers

SUMMARY

This briefing pack contains a set of double-sided security awareness notelets to help those involved in managing and delivering IT system developments fulfill their information security obligations.

The blue technical notelets introduce common information security controls, explain generic control requirements and outline the options available to satisfy those requirements.

The green development process notelets outline information security issues that ought to be taken into account during most software developments (including 'end user computing' projects such as spreadsheet programs).

Please contact the Information Security Manager for further assistance in relation to integrating information security into the systems development life cycle.

NOTICEBORED

Introduction and contents

This briefing pack contains a suite of short double-sided briefing papers (security awareness notelets) designed to assist those involved in managing and delivering software developments to meet their information security obligations:

- The blue-colored technical series introduces technical information security controls commonly required in application systems. After outlining the generic control requirements, the notelets describe security controls typically used to satisfy those requirements.
- The green-colored development process series outlines information security issues that should be taken into account during most software development projects, including 'end user computing' developments such as spreadsheets and desktop databases.

<p>Technical security controls</p> <ol style="list-style-type: none">1. Technical security controls overview2. Antivirus and other malware controls3. Audit trails, records and logs4. Backups and archives5. Business- and safety-critical systems6. Change and configuration management7. Cryptography8. Database security9. High-availability systems10. Information security policies11. Integrity controls12. IT operations13. Logging and alerting14. Logical access controls15. Network security16. Operating system security17. Passwords	<ol style="list-style-type: none">18. Physical IT security19. Security administration20. Security-relevant laws and regulations21. User authentication <p>Development process controls</p> <ol style="list-style-type: none">22. Software development security overview23. Planning & managing security24. Information security risk analysis25. Security architecture and design26. Selecting and procuring secure systems27. Securing the development environment28. Secure coding practices29. Testing information security controls30. Implementing security controls31. Supporting and maintaining security32. Retiring redundant systems33. Security aspects of 'end user computing'
---	--

More specific information and feedback

The notelets are necessarily generic so please contact the Information Security Manager for further information and advice on information security matters. Specialist assistance is available, either within the organization or from external sources. The Information Security Manager would also appreciate your feedback and contribution to this awareness briefing pack – if there is something missing from the pack or a better way of putting something, please get in touch.

NOTICEBORED

Security Awareness Notelet

Technical information security controls overview

Introduction

This is an overview of the technical controls typically used to reduce information security risks. The controls are listed according to the phase in which they are effective:



The controls outlined below are merely examples. Other notelets explain the controls further.

Preventive controls

Preventive controls are designed to prevent information security breaches from occurring in the first place. They are generally the most cost-effective form of control since they minimize the direct impact costs of breaches plus the associated costs to investigate and clean up after incidents.

- Antivirus software prevents malicious software (malware) from entering the organization and/or prevents it from spreading and damaging systems and data integrity;
- Firewalls act as network filters, blocking unwelcome traffic such as worms and hacking attacks. Intrusion Prevention Systems actively respond to potential attacks e.g. by closing network ports;
- High availability systems are specifically designed to be extremely resilient, that is resistant to various incidents. So-called dual-live, live-live or fault-tolerant systems duplicate critical elements of the system, and may even use 'voting' processes to trap processing errors for example in some safety-critical applications;
- Logical access controls prevent certain users and programs from unauthorized access to certain data, functions, programs, systems or networks e.g. denying access to system administration utilities, or denying all but read-only access to web pages;
- Encryption is a further refinement of logical access controls, involving the use of public key and symmetric key encryption methods to obscure confidential data from those not holding the correct keys, to protect the integrity of data, and to provide nonrepudiation;
- Authentication methods such as login processes involving username, password/PIN code, security tokens and possibly biometrics give us confidence that the person interacting with a computer is in fact the person it should be, not a colleague, imposter, data thief or fraudster. Similar methods are used to prevent the introduction of unauthorized programs and data sets;
- Well-written applications apply business logic rules that, for example, prevent any one person from both raising and authorizing a purchase order. Whilst this is really a governance rather than an information security issue, information security controls are required to prevent users bypassing the business logic and circumventing the process controls.

Detective controls

Detective controls are designed to identify actual breaches as soon as possible in order that the organization can respond appropriately and limit the extent of any damage.

- Logs and alerts can be set to raise the alarm and record the details in auditable records if systems experience actual security breaches, near misses or early warning signs. This controls category also includes fire/smoke/water/intruder alarms, CCTV monitoring, power and over-temperature monitors in the computer room;
- Intrusion Detection Systems look for patterns of suspicious network or systems use that indicate the possibility of a hacking attack, and raise the alarm accordingly;
- Data integrity controls are designed to identify and possibly react to integrity failures such as incomplete data entry, invalid values and out-of-range values. Parameter checks and criteria may be applied when data are entered, processed and/or output;
- System integrity controls include database referential integrity checks, software fingerprinting and self-checking routines when systems boot and software loads;
- Once again, it is worth pointing out that well-written applications typically incorporate detective controls at the business logic layer, such as the use of “suspense-” or “hold-files” to identify non-matching transactions pending management review.

Corrective controls

Corrective controls are designed to help the organization recover from information security incidents as efficiently as possible, clean up quickly and return to normal operations. Corrective actions normally involve steps to improve preventive and detective controls in order to reduce the chances of breaches recurring.



- Known good data may be restored from backups or archive copies taken before the incident, replacing that lost or damaged by the breach;
- Automated systems/network failover and redundant configurations aim to maintain IT services even if certain elements of the IT infrastructure fail;
- Contingency arrangements may be put into effect, including IT Disaster Recovery and Business Continuity Planning methods;
- Insurance cover is available for all sorts of risk, but the price reflects the level of risk. Policies usually include general statements to the effect that the owner of an insured asset must take ‘all reasonable steps’ to minimize loss, and quite often specific controls are required.

Selecting, implementing and using appropriate controls

In all three categories, there is normally a choice of information security control options to address a given risk. Choosing suitable controls is where skilled business analysts and security architects demonstrate their art, balancing the cost of controls against the business requirements and anticipated benefits. Furthermore, the selected controls must be properly designed, developed, tested, implemented, used, managed and maintained in order to be and remain effective.

For more information

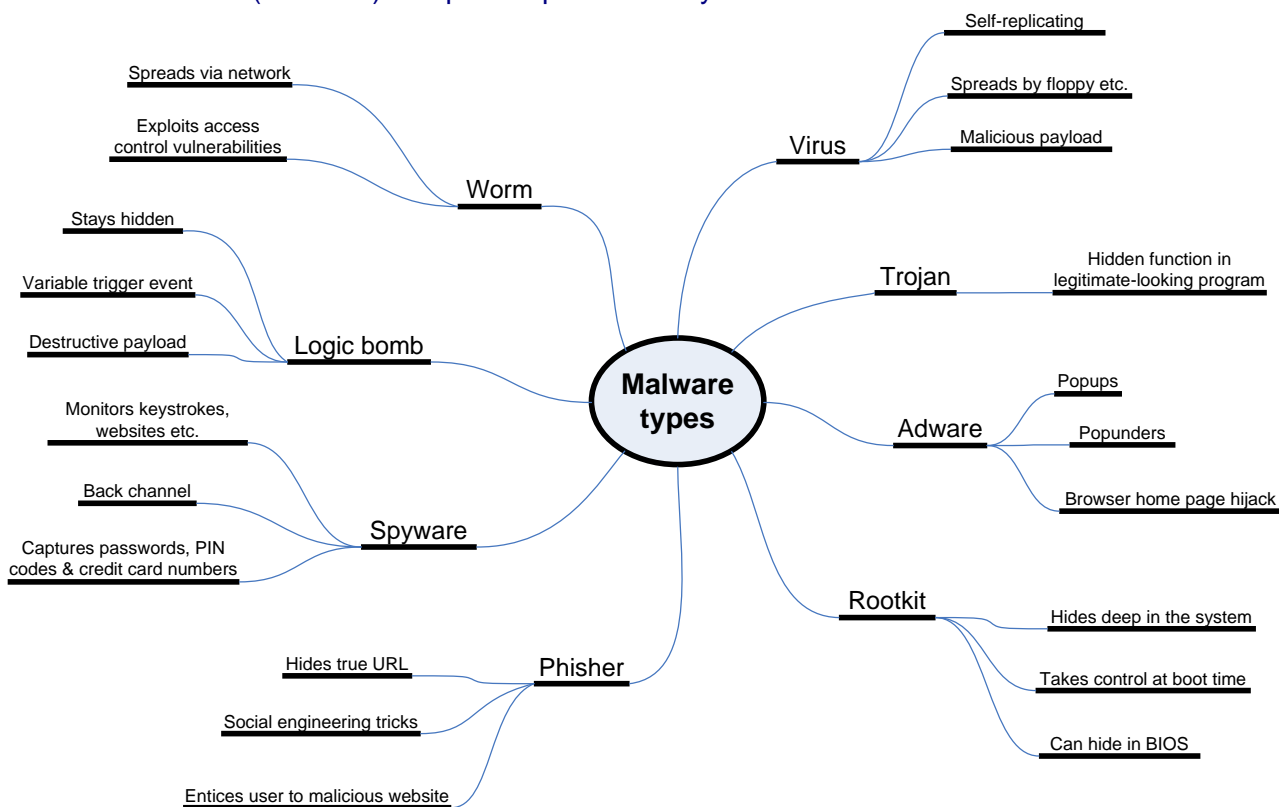
For further advice on the information security controls outlined in this paper, check the other briefings in this series or contact the Information Security Manager.



Antivirus and other malware controls

Introduction

Malicious software (malware) comprises quite a variety:



Worms and viruses have traditionally been the main malware threats but key-logging Trojans and rootkits are an emerging cause of serious security incidents. Adware and spyware programs waste system and network resources, and are annoying at best. The main impacts of malware infections include:

- The direct effects of virus and worm payloads e.g. loss of data, loss of IT services;
- Business disruption and consequential costs;
- Costs incurred in investigating and resolving specific incidents;
- Loss of privacy, identity theft and fraud (identity theft impacts both the individuals whose identities are stolen and the banks etc. that are defrauded);
- Adverse publicity and possibly legal action resulting from incidents e.g. if highly visible customer-facing systems fail or if personal data are compromised;
- A broad increase in the cost base as a result of having to select, implement, operate and maintain all the malware controls;
- Intangible costs relating to the general distrust of computer systems and networks, and conceivably some reluctance to undertake certain types of IT-supported business processes due to the chance of malware incidents.

Malware controls

Antivirus software

Hundreds of new malware variants are detected every month so antivirus vendors are constantly updating their programs. Unfortunately, malware authors are also constantly seeking new ways to avoid detection; self-modifying code and encryption are just two of the cunning techniques used. Signature-based antivirus programs are only as good as their latest virus signatures implying that they need to be updated frequently, and that the vendor has a highly competent research lab efficiently churning out reliable signatures. So called 'heuristic scanners' learn about the system on which they are installed and pick up clues about potential malware from unusual activities such as sending a rapid succession of emails (is the system is being used to distribute spam?), but are more likely to create false alarms. Nevertheless, antivirus software is the most important technical control against malware, particularly viruses, adware and spyware, and should generally be installed on all client and server systems. Consider using antivirus products from more than one vendor.



Other malware controls

Despite what some antivirus vendors imply, their software is generally insufficient to minimize the malware risk. Here are some other controls worth considering:

- **Rootkit detectors** – specialized software that aims to identify rootkits despite the cloaking mechanisms they use to hide themselves. As fast as the rootkit detector programs are evolving, so are the rootkits, a classic arms race like the early days of virus detection;
- **Prohibition** of the use of floppy disks, CD-ROMs, USB flash memory sticks, iPods *etc.* in corporate IT systems by means of policies and procedural measures and/or by technical system access/configuration controls;
- **Procedural controls** such as avoiding malicious websites, being wary of emails from unknown senders, not opening executable email attachments *etc.* Raising employee awareness of the existence of malware, and advising them on how to operate the controls *e.g.* how to recognize and react to a possible virus infection, are themselves important controls to prevent the damage caused by malware. Application user guides, help text and training courses are often good opportunities to reinforce procedural controls like these;
- **Network intrusion detection systems** and **firewalls** can identify and prevent worms and some other forms of malware *e.g.* the 'phone home' features of crude spyware;
- **Code reviews** and other forms of testing can potentially identify malware such as logic bombs, backdoors or Trojans, deliberately or accidentally inserted into new systems but don't forget that most forms of malware rely on concealment, making it difficult to spot the signs of infection;
- **Contingency measures** include having reliable backups of data, programs and system files; rapid response teams to deal efficiently with incidents; contact with internal/external sources of specialist assistance; ability to isolate infected network segments.

For more information

For further advice on the malware controls in this notelet *e.g.* help to identify malware risks, specify control requirements and test controls, contact the Information Security Manager.

NOTICEBORED

Security Awareness Notelet

Audit trails, records and logs

Introduction

Auditing literally means listening. It involves the gathering and assessment of factual information commonly known as “audit evidence”, including data from computer systems and information from the users and managers.

Auditors need to refer to information regarding the business processes and systems under review such as completed data-entry forms, system-generated reports and, of course, the people involved in doing or managing the relevant business processes. Computer auditors often use data analysis tools to examine computer records. Furthermore, all auditors normally interview staff in the business areas under review and may use other observational techniques to examine business processes in action.

What is an audit trail?

An audit trail is a chronological sequence of audit records, each of which contains evidence directly relating to the execution of a business process or system function. Computer systems maintain audit records mostly as simple data files or databases, whereas human beings normally use logbooks and the like. Common examples are:



- Visitor logs and databases showing who accessed the facilities, and when;
- System access logs showing who logged-on to the computer network or systems, and when;
- Application access logs showing who ran monitored applications, and when;
- Internal application audit logs containing details about the operation of application functions, including snapshots of important data and information about the execution of key controls e.g. who authorized certain transactions, what data were authorized and when this happened.

By analyzing information from several such sources, it should be possible for instance to trace a sequence of events such as a user arriving at a building (using information from the card-entry system or CCTV pictures), authenticating to a computer system (from the network and system login records), running an application (from the application access records) and processing certain data (from the application audit records). Similarly, using just one of the logs, an auditor or manager could identify a series of logged activities, such as entering, reviewing, matching, authorizing and paying an invoice in a computerized accounting system, in effect tracing the sequence of data items relating to a business process.

Audit trails help maintain system security. They are an important deterrent control against fraud and theft if the fraudsters and thieves know or believe that their activities are being recorded. In some cases, audit records may also be used to recover lost transactions or reverse inappropriate or erroneous transactions.

Important audit log design considerations

The specific information that has to be recorded in an audit log depends on the particular circumstances and purposes for which it will be used. Timestamps and user IDs are conventional but the transactional data and activity records should be specified by the users through the normal business analysis and architectural design process. There may be legal or regulatory obligations for some systems, especially in the financial world.

Audit trails are most likely to be used for routine management reporting and audit purposes, but on rare occasions may be needed to investigate serious incidents. It is therefore sensible to assume that audit logs may end up being used as evidence in court, implying that the data files must be suitably protected against tampering *i.e.* unauthorized modification such as deleting or inserting records. Electronic audit logs must be access-controlled to prevent unauthorized write access or deletion. A fraudster or hacker intent on breaking system controls may deliberately interfere with the auditing process or communications. Audit configuration files must also be tamper-proofed and should be alarmed where there is a significant risk of unauthorized changes.

Situations where the logs fill up, possibly as a result of deliberately being loaded with junk data, should be taken into account where this is a possibility, in order to prevent a hacker or fraudster covering his tracks. Similarly, thought needs to be given during the design process to backing-up and archiving audit logs, and of course to the need for reporting functions so that the logged data can be analyzed by authorized users.

Following a chronological sequence of events requires reliable timestamps. This is a key reason why computer systems should use good real-time clocks or align themselves automatically with authoritative external time sources such as NTP servers on the Internet or GPS systems. 'Keepalive' functions may be advisable to raise the alert if someone deliberately interrupts communications between logging generators and log recording systems, or in case of simple failures and errors. Just imagine standing before the judge trying to explain why two systems differ by an hour as a result of one of them failing to take account of daylight savings!

Given the deterrent value of audit logs, details about precisely what information is recorded, how it is stored and how it is reviewed should arguably be kept reasonably confidential whereas the fact that activities are logged should be made obvious. System security designs, including audit functionality, should be restricted on a need-to-know basis. Warning notices about activities being logged may be worthwhile when users login, or may be included in system and process documentation.

Other factors relating to audit

Like Information Security Management and Risk Management, Internal Audit should be informed about significant new system development or procurement projects at an early stage, and given the opportunity to contribute. Prompt notification is obligatory in the case of business critical systems and those with strategic, regulatory, legal or health and safety implications. Whereas all three functions can be considered risk, security and control specialists, Internal Audit's unique advantage is their professional independence.

For more information

The auditors would naturally appreciate the opportunity to contribute to the specification of audit functions. For further advice on the information security controls outlined in this paper, contact the Information Security Manager or visit the department's intranet website. Other awareness briefings in this series expand on aspects such as access controls and backups.

NOTICEBORED

Security Awareness Notelet

Backups and archival

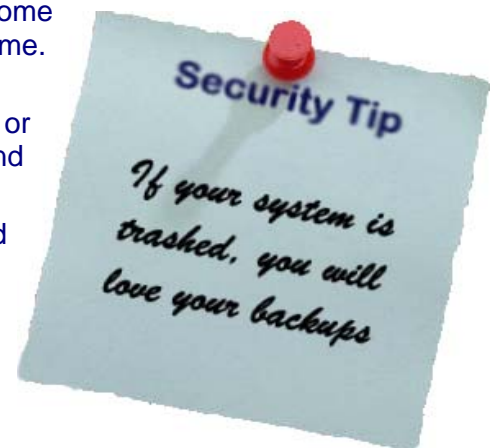
Introduction

Systems and processes for backups and archival should be designed and delivered along with a new system. Don't wait for go-live day, or until the first incident, to sort this out!

Backups

You may not truly appreciate the need for backups until some disaster befalls your data, seemingly at the worst possible time. Typical examples:

- Your system is infected by a virus that deletes the data, or maybe the only way to disinfect it is to reformat the disk and reinstall programs and data from scratch;
- Someone (probably you, possibly a hacker or a disgruntled colleague) or something (such as a program bug) damages or deletes critical data files;
- The hard drive suddenly stops working and has to be replaced, or the entire system is lost, stolen or damaged beyond repair in a fire, flood or electrical surge.



Types of backup

Backup types range from real-time disk mirroring and hot-standby arrangements such as RAID disk arrays, increasingly cost effective as disk prices fall, through warm standby such as periodic disk-to-disk copies to cold standby or off-line backups that are safely stored off-site.

Full, incremental and differential backups normally take copies of, respectively, all files, files changed since the last full backup, or files changed since the previous backup of any type (although the definitions vary – check your system documentation).

Backup cycles should be designed according to the need to store and retrieve data: longer cycles allow the retrieval of older data, for instance if a problem is not spotted for some months, at the cost of buying more backup media and storage. Short cycles increase wear on the backup media.

“System” and “data” backups are often considered separately since operating system, program and configuration files change less frequently than application data. We generally need both!

Secure storage of backup media

It is important to store backup media such as tapes and DVDs securely for two reasons:

1. To ensure they are readable when required, probably some time later, and are not damaged accidentally by being burnt, crushed, scratched, moldy *etc.*;
2. To prevent them being stolen by someone intent on reading the stored data (confidential data *must* be encrypted on both backups and archives - no excuses - there have been too many incidents involving loss of backup media and public disgrace for the organizations concerned).

Backup media should be stored at a safe distance from the primary storage to minimize the risk that both copies might be lost, damaged or simply inaccessible. In practice, there is usually a balance between ease of access requiring some backups to be held close at hand for routine

operational reasons e.g. in a tape robot and protection which means storing the media off-site well away from the possible zone of destruction that might impact the primary site. The on- and off-site media should be considered as quite separate backup sets with different purposes.

Frequency of backups

The frequency of backups reflects the business need to store and retrieve data, the amount of change and the risks. If the users cannot afford to lose more than a day's worth of data in a serious incident, daily off-site backups are needed. If they would be upset even to lose an hour's data under normal circumstances, real time backups such as disk mirroring or disk-to-disk backups may be needed, in addition to periodic off-site backups.

Archival

Archives are quite different to backups because they are intended for long term storage and the primary data source are often deliberately erased.

It is good practice to take at least two copies of important data to be stored separately, securely.

Always use brand new high quality media, certified for long-term use: don't forget that if the archive is unreadable, you may have no fallback.

Magnetic tapes should be periodically re-verified and re-tensioned according to the manufacturer's recommendations – typically every year or two – and if necessary transferred to new media e.g. to prevent data read-through and avoid gradual degradation of the stored magnetic/optical bits.

It is essential to verify integrity of the archives before overwriting the primary source, ideally by restoring onto a test system, checking and then transferring to the production system.

Do not forget to save operating system and application program files as well as data, or to save the data in a simple format for easy retrieval e.g. tab-delimited flat ASCII files. You may even need to archive special hardware – it is rather hard to find eight inch floppy disk drives these days!

Retrieval from and testing of backups and archives

Speaking from bitter experience, it is necessary occasionally to prove that data can in fact be retrieved successfully from backups and archives. This requires a test system on which to restore and check the data – *never* overwrite the primary production data source until/unless you are absolutely confident that the backups/archives contain valid and complete data!

Other issues

Management of backup and archival media

Keep comprehensive and accessible records of backups and archives, ideally a consolidated library/database system. Don't forget to keep an offsite backup too!

Use of specialist third parties

It is often worth contracting with specialist organizations that provide off-site storage of backups and archives. They normally provide rigorous check-in and check-out procedures and secure environmentally-protected vaults. Estimate the total value of your stored data in order to appreciate the importance of the supplier's credentials and legal liability clauses in the contract.

For more information

Please contact the Information Security Manager or visit the department's intranet website for further advice and information about developing critical systems. We're here to help and the sooner you seek our advice, the more chance we have of helping you meet your goals!

NOTICEBORED

Security Awareness Notelet

Business- and safety-critical systems

Introduction

This briefing outlines important considerations relating to the design, development and testing of computer and telecommunications systems which are critical to the survival of the organization or the users. Such systems are 'special' because the risks of failure are so dramatic.

Think for a moment how you would feel if you knew that the General Ledger system or the controls for the elevator in which you were riding had been specified, developed and tested just like, say, a typical Excel spreadsheet. Would you trust the values in the ledger? Would you even step into the elevator? Unless you were somewhat reckless, you would most likely require greater confidence.

Identifying critical systems

The subject of this awareness briefing begs the question: which systems are critical? There is no absolute definition, rather it is a matter of degree of criticality relative to other systems. The measure used is that of risk, so conventional risk analysis processes (which are the subject of another briefing in this series) can be used to identify critical systems.

Formal methods

Formal methods offer a structured approach to designing, building and proving critical systems. Such methods are built around mathematical models that aim to take into account all possible inputs to a system, such that no combination of events can result in an inappropriate or undefined output *i.e.* they will fail-safe. All the system's possible states can be mathematically proven. The rigor and specialist expertise necessary to develop this level of proof unfortunately incurs significant additional costs, especially in the case of complex systems, hence formal methods are normally reserved for safety-critical systems such as those on the space shuttle, or other extremely critical systems such as (vital parts of) microprocessor designs. However, the techniques make interesting reading for anyone truly concerned to design reliable systems – see [Wikipedia](#) for some useful references.

Conventional development methods

Many critical systems are in fact developed using conventional methods, albeit carefully applied. It is worth pointing out safety and other quality objectives can be compromised by time and/or cost constraints on the development process. Critical systems are necessarily relatively expensive and slow to develop if sufficient attention is paid to them, but as usual this has to be balanced against the value of implementing them and reaping the benefits – in other words, there are risk-based decisions to be made. For example, organizations are well advised to put their best people on the job, and to stick to well tried and trusted methods rather than some of the more extreme types.

Critical systems specification and design processes

Knowing that the cost of changes can increase dramatically during the development process, and that critical systems are likely to be expensive to develop, it makes sense to pay particular attention to the specification and design phases. Requirements should be explored in depth and documented in detail, taking input from all interested parties and, if necessary, competent advisors.

Testing critical systems for high confidence

Independent review by competent professional testers is an especially important control in the case of critical systems. Classic examples include the use of professional testing teams to conduct white-box or black-box tests, penetration testers to confirm whether a networked system is resistant to network-based attacks, and computer auditors to provide assurance on the suitability of business process/application layer controls.

Automated testing is likely to be of benefit on critical system developments, hence investment in testing tools, scripts and processes may be worthwhile. Whereas it may be practically impossible to enter all conceivable inputs to a system, it should be feasible to at least identify and use a full range of values and, especially, to test boundary and fault conditions. Elevator manufacturers specifically test for failures of the mechanical position switches indicating arrival at each floor, and (we hope!) test the elevator control computers for similar conditions.

Other control aspects

General information security controls

The potential impacts of unauthorized changes to or faults within critical systems implies the need for rigorous controls around access, integrity and availability. By this we mean conventional information security controls such as user authentication, encryption and resilience features.

Over-engineering

Critical systems should, as a rule, be 'over-engineered', meaning that corners should not be cut and the design should err on the side of caution. If, for example, it is absolutely vital that a system produces period-end reports on time, it makes sense to ensure that it has sufficient spare processing capacity even when fully loaded in order to avoid delays. It might also be sensible to incorporate integrity functions and other checks that would flag exceptional data or unusual system loading e.g. data files that are more than, say, 25% above the rolling average size, ideally well before the actual period-end processing starts!



Change management

The development process rigor of which we have spoken is impossible to achieve unless the requirements and design are reasonably stable and consistent. Changes are almost inevitable but the processes should include periodic reviews to ensure that key criteria and requirements are not compromised along the way. Competent system architects should be able to review the 'as built' design against the original specifications and verify all the changes – in other words they should be formally documented and traceable to legitimate and authorized change requests e.g. resulting from specific test failures.

Similar change and configuration management considerations apply once the system is in production. Uncontrolled changes are clearly a threat to production systems, all the more so if those systems are business- or safety-critical. How well do our change management processes reflect this aspect, I wonder?

For more information

Please contact the Information Security Manager or visit the department's intranet website for further advice and information about developing critical systems. We're here to help and the sooner you seek our advice, the more chance we have of helping you meet your goals!

NOTICEBORED

Security Awareness Notelet

Change and configuration management

Introduction

Computer and telecommunications systems are normally changed or reconfigured for business or technical reasons. Business-led changes include the implementation of new or modified application systems, typically to satisfy new functional requirements. Technology-led changes include those made to maintain currency with supported versions, and to implement security patches. Other changes may be unauthorized, perhaps malicious, or accidental.

Change management

Whilst some changes 'just happen', this notelet concerns changes that are planned and deliberate, particularly in the context of IT system changes. We start with an outline of the change process.

Phases of change

1. **Unfreeze:** prior to making a deliberate change, careful planning and thorough preparation makes the desired outcome more likely. 'Unfreezing' refers to overcoming the natural resistance to change, removing barriers. In a systems context, it involves preparing, testing and agreeing the change, and granting someone write access to the system;
2. **The actual change:** someone promotes the changed code into production, normally sandwiched between a pair of pre- and post-change backups and, ideally verification tests to check that the change was successful and the system performs as anticipated;
3. **Refreeze:** after the change is done, a period of stabilization occurs during which things gradually settle down and the changed systems becomes business-as-usual. It is good practice to review changes once the dust settles in order to learn from and improve the process.

Information security aspects of change

The pre-release testing in phase 1 should incorporate information security tests, especially where the system is complex or is considered business- or safety-critical. Even minor changes may adversely affect security so, although security testing might be inappropriate for every change, periodic testing may be advisable. Speak to Information Security Management for more on this.

A good proportion of information security controls are associated with preventing unauthorized and unintended system changes such as those caused by hackers or errors. In order to permit authorized and planned changes, these controls must be temporarily relaxed at the start, and re-enabled at the end, of phase 2. While the controls are relaxed, there is an increased risk of unauthorized or unintended changes so it makes sense to limit this period as far as possible.

Changes to system security designs and controls

Information security designs and controls may be changed in isolation but more usually this occurs in the course of other work, for example whilst upgrading software versions or introducing new functionality. Just as with the development of completely new systems, information security requirements should be considered in relation to changes depending on the level of risk. So-called baseline or basic security measures may be entirely sufficient on low-risk systems but clearly changes to high-risk e.g. business- or safety-critical systems are an entirely different situation. Information security Management can help in either case, both assessing the risks and helping to design, develop and test appropriate security controls.



Identifying and responding to external changes

Security threats and vulnerabilities affecting IT systems change frequently. As new security threats emerge such as new forms of hacking or fraud, it becomes necessary to identify and respond to the trends before they become significant. When vulnerabilities are announced in commercial software, there is often something of a race between, on the one hand the vendor and its customers, and on the other those who would exploit the vulnerabilities. We need to maintain vigilance in this area and build the capability to respond quickly when necessary.

By the same token, it is important that our responses to new security threats and vulnerabilities are measured and effective. Knee-jerk reactions to change can cause more serious problems than the threats or vulnerabilities we seek to address. For this reason, there is close cooperation between Information Security, Risk and IT Operations management.

Configuration management

Changes to system configurations (operating parameters) comprise a subset of all system changes, but are considered separately here because they are normally undertaken by IT Operations and related functions whereas application and other system changes are initiated by users, application developers and application support teams.

Configuration changes typically occur in the normal course of managing and using systems. These should be assessed in relation to the risk and criticality of the systems – simple changes to access control and logging rules, for example, could prove disastrous on Internet-facing firewalls, but may be relatively trivial on internal client systems. If possible, systems should be designed with this distinction in mind, for instance by presenting additional warning messages or alarms and saving audit log entries when important system parameters are altered.

Configurations should be reviewed periodically, again in proportion to the risks. It is conventional to develop automated configuration checks for critical systems, for use when they are first implemented and thereafter to confirm that vital parameters remain unchanged.

Changes to business processes

IT system changes seldom occur without altering the corresponding business processes. In conjunction with management and other specialist functions such as Risk Management and Internal Audit, Information Security Management can help assess and react appropriately to the security implications of business changes. Even things such as user guides and training courses may have security implications, especially as many security controls are manual or procedural in nature.

It is worth noting that changes can present opportunities as well as challenges. This notelet is an example of a security awareness artifact that forms part of a managed process of deliberate business and cultural change – it *can* be done (well at least you are reading it anyway)!

For more information

Please contact the Information Security Manager or visit the department's intranet website for further advice and information about IT change and configuration management. There are policies, procedures and guidelines in this area that you ought to be aware of, and best practice approaches that will save effort and minimize risks and costs in the long run.

NOTICEBORED

Security Awareness Notelet

Cryptography

Introduction

For thousands of years, people have been using techniques like secret codes and ciphers to communicate sensitive messages in such a manner that unauthorized parties cannot benefit from the information. Messages have been tattooed on the scalp and thus hidden under the hair of messengers. Every schoolboy and IT geek knows the value of code-words, jargon and slang to conceal meaning from those not in the know. Stored program computers owe their very existence to the need to decode wartime messages composed on mechanical encryption devices. These days, everything from GSM cell phones to online shopping trolleys and point of sale terminals are absolutely reliant on encryption, validation and non-repudiation, all applications of cryptography.

Encryption

Symmetric or shared key cryptography

Modern computers are extremely efficient at encryption using essentially the same “symmetric” or “shared key”, both to encrypt and to decrypt information. Even high speed data streams and multiple video signals can be encrypted and decrypted easily with the right hardware. Up-to-date symmetric algorithms such as AES with keys of at least 64 bits (ideally 128 or more) are highly resistant to cryptanalysis (breaking). The real issue, though, is how to generate and share the keys in the first place, particularly in a typical network scenario with multiple pair-wise links – that is where asymmetric encryption comes into its own.

Asymmetric or public key cryptography and PKI

Public key encryption involves the use of matched pairs of different “asymmetric keys” – one to encrypt, the other to decrypt. It is ‘computationally infeasible’ (*i.e.* practically impossible) to derive either member of the pair from the other, so even if I encrypt something extremely sensitive with my key and publish the key, you still cannot decrypt it unless you hold or obtain the other member of the key pair. Public key encryption is too slow to be used for real-time encryption and decryption of streaming video *etc.* but is ideally suited to the secure distribution of shared keys.

Public Key Infrastructure (PKI) refers to the mechanisms for publishing and exchanging so-called public keys from each key pair on digital certificates that are firmly associated with particular people, systems or organizations.

Choosing suitable algorithms

Faced with a wide choice of encryption algorithms, how should we choose the strong ones whilst avoiding the snake oil salesmen? Here are some tips to help:

- Pick ‘open source’ or published algorithms over proprietary ones. Private keys should be the only items that must remain secret, not the algorithms. Published algorithms are pored over by fellow cryptographers and cryptanalysts, giving confidence in their strength, whereas proprietary algorithms could have fundamental flaws that only appear once in use;
- Choose algorithms that are specified as standards, such as AES and RSA, in preference to those that have not been assessed or, in some cases, were rejected by the standards bodies;
- Good algorithms produce output that is essentially random. Avoid algorithms whose output can be compressed as this necessarily implies repeating patterns, not random values.

Key length and key management

Encryption keys consist of digital words of various lengths. Reduced to the extreme, a one-bit key would have just two possibilities, either one or zero. It would take just one attempt, on average, to guess the key value by a so-called brute-force attack. In all practical systems, encryption keys are much longer, typically 128 bits or more. Every extra bit doubles the number of random guesses needed to crack the code by brute-force.

Longer key lengths are normal for asymmetric encryption. RSA normally uses 1024 or 2048 bit keys, for instance. Part of the reason is that asymmetric keys are often used to encrypt session keys, therefore breaking an AES key will give access to the data exchanged in one session whereas breaking an RSA key could give access to all sessions whose symmetric keys were encrypted with that master key.

Using cryptography for integrity and other purposes

Digital signatures used to authenticate messages and programs

Message Digest 5 (MD5) and Secure Hash Algorithm 1 (SHA-1) are examples of algorithms that create a kind of fingerprint – a fixed length ‘hash’ value - from a variable length packet of data. The mathematics is such that even small changes in the data create significantly different hashed values, making it easy for the recipient’s system to identify that the message has been modified or substituted, provide of course that he has access to the original hash. Public key encryption provides further mechanisms to associate the hash unambiguously with the message. Similar processes are used to authenticate executable programs using the vendor’s published digital certificates to authenticate and validate the programs.

Watermarking and steganography

Digital watermarks, subtle changes in a data file that do not noticeably detract from the original content but can be used to confirm the file’s true origin, are an example of steganography (literally ‘secret writing’). Cryptographic techniques ensure that a digital watermark cannot be modified without the authentication checks failing. Steganography can also be used to hide secret data within, for example, the least significant bits of certain bytes in an image file: these changes make no noticeable impact on the image but, using the appropriate software, the two data files can be separated. This ability to ‘hide in plain sight’ is occasionally used as a means of communicating data out of a heavily access-controlled environment through a so-called covert channel.

Nonrepudiation

Provided the cryptographic system is properly designed and implemented, a user who digitally signs data relating to a transaction cannot later deny (‘repudiate’) it.



For more information

Cryptography is arguably the most complicated area of information security and is a minefield even for experienced professionals. Please contact the Information Security Manager or visit the department’s intranet website for further advice and information about cryptography, encryption, PKI and so forth. Through history, there are many examples proving just how easy it is to introduce serious exploitable vulnerabilities in cryptographic systems. Seek specialist assistance from the outset if encryption is important to your system and to the organization.

NOTICEBORED

Security Awareness Notelet

Database security

Introduction

Many business systems are built around a database of some sort (pun intended). Database software provides the functionality to manipulate and control the data, in an environment that lets us manage the data and programs. This briefing outlines the information security considerations that should be taken into account when designing and building database systems.

Designing secure database systems

Inexperienced database programmers sometimes start developing database systems before they have figured out the requirements and the design, presumably hoping to be inspired during the development process. Whilst this quick-and-dirty approach might suit very simple systems, most databases are complex in nature which is of course the very reason a database is needed in the first place. All development done before the requirements are clarified and the design is documented should therefore be considered disposable – prototyping at best.

Information security requirements for the system are unlikely to be satisfied unless they have been properly analyzed (implying a structured security risk assessment process), thought through and incorporated into the design. The following sections show the kinds of things to take into account during the design.

Database confidentiality controls

Access controls

All databases incorporate access control functions that build on the user authentication and other mechanisms provided by the underlying operating system but they can be considered a set of tools rather than a finished item: it is up to the system architects and developers to make appropriate use of the access controls available in order to build a secure system. There is no shortcut to working out which rôles should be able to create, modify or delete records, which should have the power to change specific data fields within each record, and which will have wide-ranging rights to manage the database.



Encryption

Highly sensitive information, both personal data and corporate secrets, should be encrypted when stored and communicated. Many database systems provide basic functionality to encrypt data as it is stored and decrypt it when it is used, and perhaps to manage the associated cryptographic keys. Please refer to the separate awareness notelet on encryption for further information on this complex topic.

Database integrity controls

Referential integrity

Modern databases have the ability to protect the data against changes that would leave them in an undefined state, for example by linking together records in different tables through common 'key' values. If the system is correctly configured, deleting just one component of a linked series of records will either be blocked automatically, or will result in the entire series being deleted, or will trigger some other relevant activity such as warning the user. The referential links and choices need to be designed and properly configured for the system to behave correctly.

Data validation

Data validation routines automatically monitor inputs, interim values and outputs for conditions that indicate an error. Common examples are missing values; duplicated values; values wrong by a factor of ten or one hundred; letters entered into numeric fields and other data type mistakes; entries submitted in the wrong sequence or put into the wrong field; nonsense values; ... and many more. While the vast majority of these are the result of simple user errors, don't neglect the fact that malicious users may deliberately attempt to subvert the system by entering commands or out of range values.

Database availability controls

Performance and capacity

Sizing the system correctly is theoretically just a simple matter of working out the expected transaction rates, the processing load imposed by those transactions and the normal data volumes, none of which is easy to determine before the system is actually in use. If the system will be performing a critical function, it makes sense both to design a scalable system and to provide plenty of headroom from the outset. Tuning the database is a skilled process best performed during the design phase and tweaked through testing. After the system is in production, there will be far fewer opportunities to improve performance.

Resilience, failover and contingency planning

Depending on the criticality, systems should be designed either to fail gracefully or not to fail at all when placed under extreme stress. Ordinary systems tend to rely on semi-automated or purely manual fallback arrangements whereas highly resilient systems make use of fault tolerance, dual-live configurations and other specialist techniques, all of which incur significant costs. Keeping multiple databases synchronized in near-real-time is no mean feat, especially if they are complex and heavily used systems. In various error scenarios, it is common to roll-back transactions to a checkpoint, correct the errors and reapply the transactions. These checkpoints may provide suitable opportunities to copy data to offline media or to other systems for disaster recovery purposes.

For more information

Please contact the Information Security Manager or visit the department's intranet website for further advice and information about database security controls. Experienced database architects and other specialists are available to help with the specification, design, development and testing of secure databases. Security incident managers and corporate lawyers are standing by in case of security failures in production ...

NOTICEBORED

Security Awareness Notelet

High availability systems

Introduction

No-one is particularly pleased when a computer system they are using suddenly stops working but in some circumstances, the impacts can be disastrous. Imagine if the console screens and indicators on a modern jet aircraft were suddenly to go blank and the controls stopped responding, or if say www.amazon.com seized-up during the shopping frenzy prior to Christmas.

The incessant spread of IT throughout the organization has brought immense benefits but, by the same token, makes us highly dependent on our information processing services and hence on the IT systems and data networks that provide them. If they fail for any reason, a large proportion of the workforce cannot continue working productively and customer service is quickly affected. Availability is therefore one of the core elements of information security.

Availability requirements

Persuading business managers to clarify their departments' IT availability requirements is itself a challenging exercise. Business Impact Analysis (BIA) is a structured process for evaluating and quantifying the effects of system failures that is normally used routinely as part of the organization's Business Continuity Management, but can be used at any time such as when a new IT system is being specified.

BIA involves systematically assessing the likely business impacts and costs if vital IT systems are out of action due to, for example: software bugs; mis-configuration of systems; accidental or deliberate damage or theft of equipment; viruses, worms *etc.*; power failure; fires or floods in the computer room; equipment failure; and many more. Even if the immediate impacts of a systems outage are relatively minor, costs sometimes escalate dramatically at certain points such as when period end reporting cannot be performed, regulatory deadlines are missed, or manual workarounds simply fold under the increasing workload. It is possible to determine availability requirements for individual processes and systems by graphing the project cost profile over time after an incident occurs, although most practical situations are more complex. System and process interdependencies mean that the longest recovery time in effect constrains recovery of the entire chain of linked activities, and shared equipment (especially communications servers) is often critical to numerous business processes.

Performance and capacity planning

Leaving disastrous IT service outages aside for a moment, even under normal circumstances users may experience unacceptable delays and constraints due to limitations of the IT systems. Performance and capacity planning involves measuring and projecting key system parameters to match transaction throughput, batch processing time, storage capacity *etc.* with the requirements – easier said than done if the system is still on the drawing board.

It is worth noting that certain security vulnerabilities may only appear when a system is placed under severe stress, therefore it may be sensible to conduct security testing during routine stress tests. Bear in mind that error messages often give clues to a hacker about the nature of the system: do we really need to disclose the operating system name, version and patching status?!

Failover and redundancy

A common technique to improve availability is to duplicate critical items and provide some form of load balancing arrangement to share the work across them. Clustering of servers and RAID (Redundant Arrays of Inexpensive Disks) are obvious examples and so is having dual-routed communications links. TCP/IP is highly tolerant of routing constraints or failures by design, automatically re-sending or re-routing packets around network trouble spots. Technical measures like these are no panacea, however, especially as there are often common points of failure and there is always the possibility of coincident failures. These issues should be explored during the design process if availability is an important requirement for a new system.

Resilience and fault-tolerance

Resilience involves 'over-engineering' systems such that they can stand a tremendous level of abuse without collapsing – they 'tolerate' a certain level of faults. Ideally, they fail gracefully if pushed beyond their limits, meaning they "fail safe" by default. Fail safe is in fact a generally applicable architectural design principle or rule of thumb that has benefits in many other situations. Journaling and periodic auto-saves are a simple example: if, despite everything, the program crashes while being used, it should be possible to roll-back to the most recent checkpoint and perhaps roll-forward (reapply) journaled changes since the checkpoint. Automating this process can prevent a minor problem turning into a crisis.



Disaster Recovery and contingency arrangements

It is conventional to deploy highly resilient systems in situations where failures are extremely costly or life-threatening. By the same token, it is advisable to consider the possibility of extreme situations (disasters), and to prepare Disaster Recovery (DR) plans that will help the organization resume some semblance of normal processing more efficiently than otherwise might be the case. DR requirements are best considered as part of the general availability issue as it can be extremely expensive and difficult to retro-fit ideal DR arrangements to a production system.

Contingency planning involves preparing for uncertain future situations, paradoxical as that may sound. Simply having DR manuals, system recovery instructions and backup tapes at an off-site location may be all that is needed for someone to rebuild a critical server after a major disaster affecting the primary site. If you are modifying an existing system, don't forget to update any associated DR and contingency plans and, for instance, send an updated set of system installation manuals and tapes to the recovery location (this kind of activity ought to be triggered automatically by the change management system but it doesn't hurt for someone on the project team to make doubly sure).

For more information

Please contact the Information Security Manager or visit the department's intranet website for further advice and information about availability, resilience, DR and related issues. Risk Management may be able to facilitate the BIA process and thus help document the availability requirements.

NOTICEBORED

Security Awareness Notelet

Information security policy framework

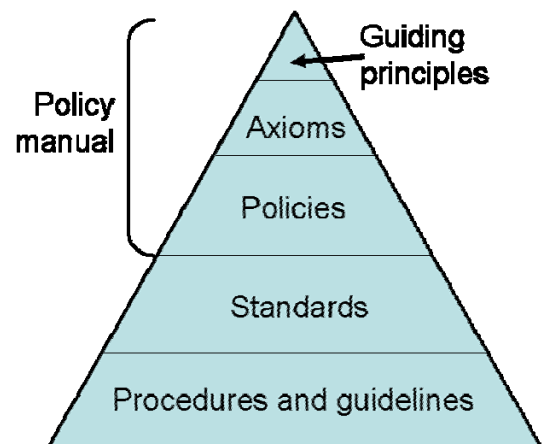
Introduction

Aside from various legal and regulatory obligations imposed upon us by external bodies, a comprehensive set of internal rules relating to information security has been mandated throughout the organization by management. This notelet gives a high level summary of the policy framework, explains the hierarchy and advises you on the need to comply.

The information security policy framework

Information security standards such as ISO/IEC 27002 and the NIST SP 800-series are often used as the basis of an organization's information security policy framework. Standards provide a sound structure, defining all aspects of information security and recommending a broad range of generally-accepted good practice security controls.

The triangle diagram shows how different levels of information security policy-related materials apply to all IT systems and users within the organization. The shape reflects the hierarchical structure, level of detail and hence the volume of materials at each layer.



Principles

At the very top, a handful of broad guiding principles such as “defense in depth” define overarching control objectives for information security giving **strategic direction** to information security as a whole. The principles are commonly mandated by the CEO and/or Board.

Axioms

Axioms are more specific security control objectives. ISO/IEC 27002 defines 39 control objectives that make an excellent suite of axioms. Axioms explain **why** information security is so important and start to define specific **types** of control e.g. “Information security responsibilities must be addressed during pre-employment screening, included in employment contracts and monitored by management during an individual's employment.” Axioms that are formally approved by executives and other senior managers are mandatory throughout the organization.

Policies

Information security policies vary in style. Some mirror the structure of the ISO/IEC 27002 standard. They are normally approved and mandated by senior or middle managers. Policies explain **what** information security controls, specifically, are used to satisfy the control objectives defined in the axioms. An example policy statement reads: “Workers must sign to denote their understanding and explicit acceptance of the terms and conditions of employment prior to being permitted access to the organization's information assets.” A number of other related policy statements supplement this one, collectively satisfying the axiom noted above.

Standards

Security standards provide yet more detail on information security controls and explain **how** they are applied in particular circumstances such as on Windows XP systems. A Windows XP Security Standard explicitly defines Windows XP security parameters relating to passwords, for instance, that are derived from password policy statements stated in the policy manual. Where possible, standards also reflect external best practices from software vendors, standards bodies such as NIST and industry bodies such as the Center for Internet Security (CIS). Unless specifically mandated by management, compliance with external best practice standards is optional but advisable. If you cannot comply, you should be able to justify the reasons.

Procedures and guidelines

The lowest level of the policy pyramid comprises a range of briefing documents, web pages, help text and so on. Procedures are documented processes incorporating manual information security control elements. Guidelines give further information on information security to users of our information assets. Procedures and guidelines contain **good advice**. Whilst these are informal in style, they reference mandatory standards, policies, axioms and/or principles where relevant.

The information security policy manual

The policies, axioms and principles together comprise the Information Security Policy Manual which is a formally controlled document, normally available on the intranet. **You should definitely read and use the information security policy manual.** If you have trouble understanding the manual and how it applies to you, please contact Information Security Management or take a look at ISO/IEC 27002. You should also obtain and become familiar with any standards, procedures or guidelines that apply to the particular system or business processes being addressed by software you are developing.

Compliance and exceptions

Broadly speaking, the policy framework embodies a hierarchy whereby the upper levels are firmly mandated by management, middle levels are mostly mandatory but formal exceptions are permitted, whilst the lower levels are generally recommended but often optional. Policies use the word “must” for mandatory elements and “should” for the remainder. If, for some reason, your system cannot fully comply with a mandatory security policy, axiom or principle, you must obtain a formally approved exception from management. Management will insist that the nominal system owner carries personal accountability for the risks arising from such exceptions, which acts as an incentive to bring systems into line with corporate policies.

A number of control measures are in place to ensure compliance with the security policy framework, including regular and ad-hoc reviews and audits, routine management reporting and, of course, security awareness materials such as this notelet.



For more information

For advice on the information security policy framework and how it applies to your development project, please contact the Information Security Manager or any senior manager.

NOTICEBORED

Security Awareness Notelet

Integrity controls

Introduction

Our organization is utterly reliant on good quality information and hence on the people and systems that produce it. Serious errors in financial and operating data could lead to our downfall but even minor errors may accumulate making information, and the decisions based upon it, unreliable - a principle summed up by the phrase “garbage-in-garbage-out”. Errors in information can propagate quickly from person-to-person and, in an IT context, even quicker from system-to-system. If we are careless with information, it is more likely to get corrupted and become meaningless. Worse still, if someone down the line acts on erroneous information in good faith, who knows what might happen. *People sometimes die because of simple information mistakes.* This notelet concerns controls designed to protect the integrity of information, encompassing concepts such as trust and validation.

Data integrity controls

Section 12.2 of ISO/IEC 27002, the Code of Practice for Information Security Management identifies numerous integrity controls at three key stages of information processing:

1. **Input/data entry controls:** automatic data-entry validation ensures that input values are complete and of the correct type and range (such controls are seldom provided by default – they need to be specified, designed and coded explicitly). Other controls include periodic review of key fields and files; inspection of data entry sheets for unauthorized changes; ‘plausibility testing’ etc.
2. **Controls of internal processing:** validation checks are built-in to application systems to detect any corruption caused by processing errors or deliberate manipulation. Such checks may ensure the correct sequencing of operations and correct handling of error conditions e.g. halt operations rather than continue. Other controls include limits to prevent buffer under/over-run; run-to-run or program-to-program, session or batch control values reconciled with the processing actually performed e.g. 10 values entered = 10 values processed etc.
3. **Output controls:** so-called “management reports” highlight unusual and implausible, inaccurate or incomplete values for review. System outputs need to include sufficient supporting information for the reader or following system to confirm the data presented. Other controls include reconciliation of check totals e.g. if the processed values sum to \$1,200, the output total should have changed by \$1,200; logging and reviewing activities performed etc.



Defining acceptable data ranges and types

A vital consideration in most of the above is being able to specify, in advance, acceptable ranges and types of data expected. At a simple level, an input field for entering, say, a telephone number should not accept and execute SQL commands (badly designed web forms are vulnerable to such “SQL injections”). Calculations that would create enormous interim or final values may exceed the available buffers, allowing user-entered data to write into executable memory space. A household electricity bill of several million dollars is unlikely to be correct and should therefore be sidelined by output integrity checking routines for management review.

Defining acceptable, sensible or plausible values is largely a business matter but people usually need to be led carefully through the definition process. Skillful business analysts can help users figure out the boundary conditions beyond which a value can be deemed unacceptable and passed to an error-handling routine. It is equally important that error handlers are as well designed, coded and tested as the main routines: malicious users like to explore error conditions in the hope that security vulnerabilities lie dormant in these sleepy backwaters of the program.

Manual integrity checks

Processes for using and managing IT systems should incorporate appropriate manual integrity checks e.g. procedures, guidelines, training manuals and help text should emphasize the need for users to check their own work and be alert for obvious and more subtle data errors. Excel, a relatively simple tool, has the capacity to identify unusual or out of range values on the screen automatically but users should be taught to recognize and check the little visual cues.

System integrity controls

New IT systems are tested prior to release to make sure they will work as intended. The extent of testing required reflects the level of confidence desired, which in turn relates to the criticality or importance of the system and hence the risks of integrity failures, and the reliability of the development process, on the theory that a stable, dependable and repeatable process with suitable quality assurance controls is unlikely suddenly to produce a bad product.

On a related point, sufficient resources (skilled people, development tools, elapsed time *etc.*) must be allocated for specifying, developing and testing new IT systems. Integrity of the finished product is highly dependent on the quality of the systems development processes and the team members assigned. It also relates to the engagement and competence of the business people who specify and test the functionality. Don't leave critical IT projects to the office junior!

IT systems incorporate various automated integrity controls such as self-checking routines that run when a system starts up. These should be supplemented with other periodic checks during normal operation. It might be sensible, for example, to validate program hash totals just before processing the end-of-period run, or if a serious error condition is detected [speak to Information Security about how to generate, store and validate cryptographic hash totals].

Personal integrity controls

Personal integrity relates to issues such as trust and ethics. Many employees are placed in positions of great trust by the organization – they have the potential to wreak havoc but are expected to behave sensibly and professionally. Those with privileged access to IT systems, for instance, could potentially add, modify or delete data irrespective of the access controls imposed on normal non-privileged users. Aside from blind trust, other controls include: periodic reviews of work by managers or peers; audit records and secure logs of activities so there is evidence to support or, just as importantly, refute any suspected wrongdoing; pre-employment background checks, perhaps refreshed periodically or when someone moves into a new rôle *etc.*

Managers are well advised to bring trust and commitment issues into the open when appropriate, such as at staff appraisals and team meetings. Accountability and responsibility depend on people knowing clearly what is expected of them, and what are the limits. Policies that sit on the shelf gathering dust are likely to be ignored or forgotten.

For more information

Please contact the Information Security Manager or visit the department's intranet website for further advice and information about integrity controls. Experienced business analysts and application architects have a wealth of knowledge on this subject.

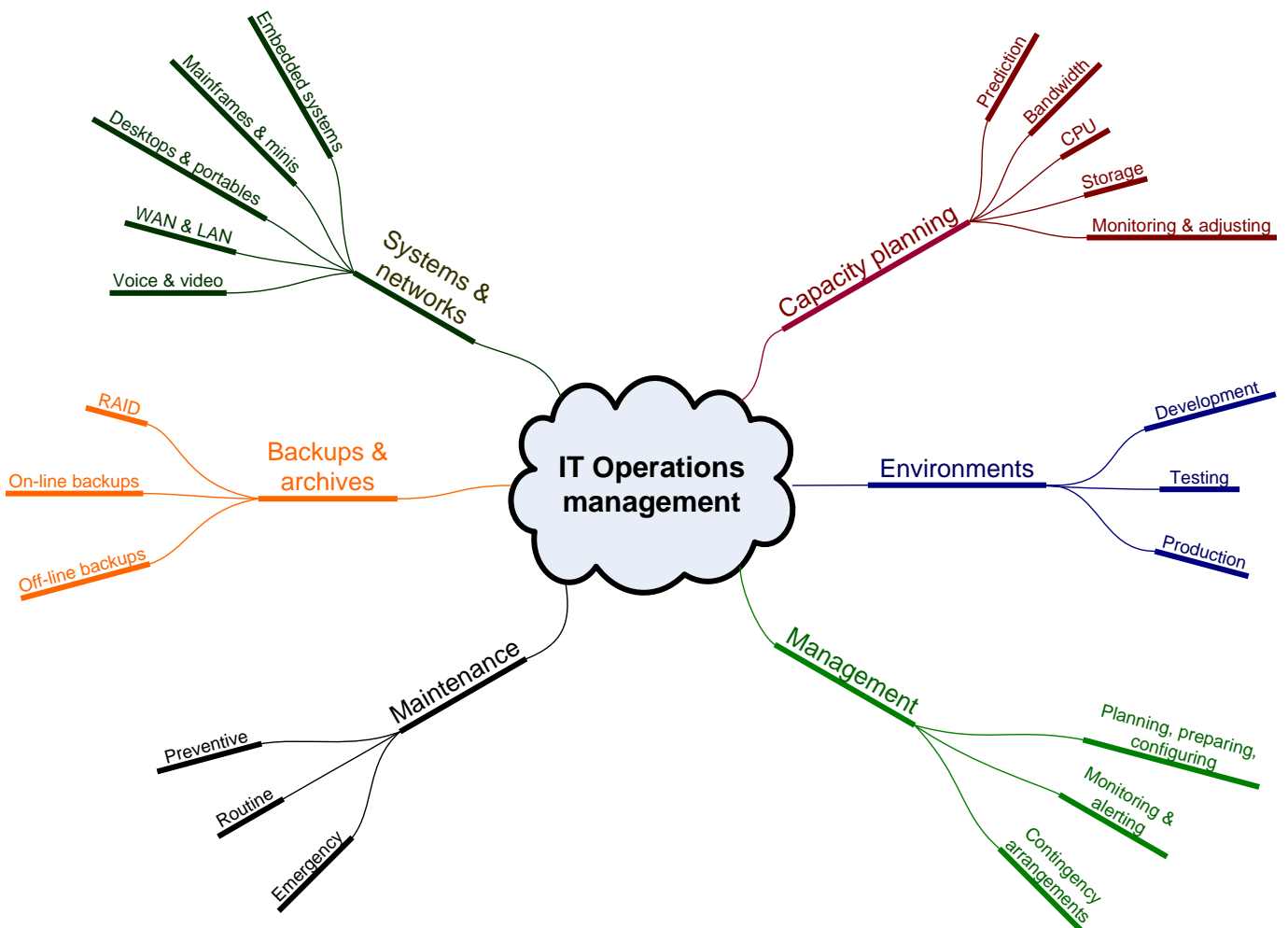
NOTICEBORED

Security Awareness Notelet

IT Operations

Introduction

The mind map below shows the broad range of tasks for which IT Operations (“IT Ops”) is normally responsible. This awareness notelet summarizes the information security aspects that software development project teams should consider.



Security elements of IT Ops’ rôle

IT Ops manage the IT systems and networks, not just those within the corporate data center but embedded systems such as those controlling machine tools on the shop-floor, distributed systems (minis and micros located in user departments), desktop and portables PCs, routers, firewalls, intrusion detection/protection systems and ‘network appliances’.

Routine security-related tasks include: backups and archives; monitoring system performance and capacity; maintaining the IT infrastructure (shared systems, platforms, environments, plans and processes) and supporting facilities, particularly the computer and telecomms rooms, plus the wiring closets and patching racks *etc.*; monitoring logs and responding to alerts and alarms.

Emergency maintenance is what happens when systems or the supporting infrastructure shows signs of imminent failure, or actually fail in service. Typical examples are when fan belts break in air conditioning units or stormy weather breaches the flat roof and rainwater pours through the ceiling of the computer room (don't laugh – buckets, mops and plastic sheeting should be part of the emergency supplies for almost every IT facility!).

Managing IT environments

IT Ops manage a number of IT environments:

- Development systems and networks can be difficult to control since developers often need more-or-less privileged access, meaning that control is shared. IT Ops usually has the responsibility to maintain and support the development platforms, and of course has to develop its own utilities and processes for managing new systems and networks.
- Managing test environments is another challenge since they are subject to intense pressure from the testing and monitoring activities, usually compressed into tight timescales leading up to planned release dates. IT Ops and/or Information Security Management are also responsible for security testing of new platforms and systems.
- Despite the forgoing, Production environments are the main focus of IT Ops. Production outages, whether planned or not, cause impacts on operational activities in the business, yet outages are essential from time to time

Systems/network security management

Managing information security is an integral part of managing IT systems and networks. Planning, preparing and installing new systems necessarily involve configuring the associated security functions. Confidentiality, integrity and availability aspects have to be correctly installed and managed. Examples include: passwords, privileges and access rights to systems, disks, files and data; file system and operating system program integrity controls; backups, resilience measures, capacity and performance.

Monitoring the security alarms and alerts is a vital ongoing activity for IT Ops as these are the signs of possible or actual intrusions, problems with systems capacity and all manner of warnings that Something Is Going Wrong. It is not always possible to respond to warning messages in real time but there has to be a rational process in place to prioritize activities. This includes reviewing system, firewall and intrusion detection logs for subtle signs that hackers are probing the systems or that systems are about to fail. IT Ops procedures therefore link in to the organization's incident management process.

IT contingency planning is yet another important part of IT Ops' rôle. Someone has to match up the capability to recover and restore critical IT applications to the corresponding critical business processes, and make sure they stay in step.

For more information

Please contact the Information Security Manager or visit the department's intranet website for further advice relating to the security aspects of IT Ops. Other notelets in this series expand on related aspects of security such as backups, security administration, logging and alerting, and network security.



NOTICEBORED

Security Awareness Notelet

Logging and alerting

Introduction

Despite all the controls we put in place, incidents still happen on our systems. Examples include data and program/file errors, system events such as 'disk full' and, of course, security events such as 'unauthorized access attempt to payroll.dat' (!). Logs are used to record data about these incidents for subsequent analysis and potentially to use as evidence, perhaps even in court. Alerts are used to raise the alarm, notifying someone that an incident has just taken place and giving them the option to do something about it straight away. This notelet outlines the sorts of things that should be taken into account around logging and alerting during the development lifecycle.

Requirements analysis for logging and alerting

Two complementary approaches to analyzing the requirements for logging and alerting are:

- Looking at the processes in the application and thinking about the possible incidents, errors or other events that ought to be alarmed;
- Figuring out what kinds of information would probably be required to trace incidents, errors and other events after-the-fact, and collating details about the items that would need to be logged.

The analysis involves matching these two lists and, where necessary, adapting the design to fill in the gaps e.g. if reliable timestamps are vital, synchronization of the system clock/s with definitive time servers should be designed-in from the outset. Workshop-style brainstorming sessions are a convenient way to explore the requirements with users, managers and other interested parties.

Logging

Configuring which items to log

Log entries may be generated by any layer of the system: the client and server applications, the middleware e.g. database software, network/operating system and even the hardware. Generally speaking, logging by the lower levels is determined and configured by IT Operations according to their policies and standards and working practices for infrastructure management. Middleware and application layer logging may be configured manually and managed by IT Ops or built-in to the application programs and installation scripts, with the logging configurations in these layers normally being determined by the designers and developers.

Whilst it is easy to just "log everything", this is generally not a good idea. Someone has to analyze the logs to identify issues worth following-up: wading through reams of trivia is more likely to result in serious incidents being overlooked. Capacity is not free, either. Be reasonable about it.



Using system log functions, centralized logging and analytical processes

Where possible, it makes sense for all layers of the system to log to a common logging system and repository. This implies generating log entries in the correct format to merge in with other streams, for example using appropriate APIs to inject records into the system log. If the organization uses centralized logging tools such as Syslog or Kiwilog, or enterprise systems management utilities such as BMC Patrol or Tivoli, hooking into these will align a new system with existing systems including, hopefully, the well-practiced incident analysis and response processes.

Securing the log files

If there is any possibility that log files might be needed for forensic incident analysis such as following security breaches, frauds or errors, it is clearly essential that they are trustworthy, implying that they must be tamper-proofed (access controlled) as far as possible. Consider the use of encryption. Hackers commonly attempt to cover their tracks by manipulating the logs so it is also well worth placing tamper alarms on the log files and logging programs.

Alerting

Configuring and prioritizing alerts and alarms

Alerts and alarms have value only for a limited time after an incident occurs, so it pays to think through the processes governing how and when they will be presented and to whom. System users and managers need different kinds of information to IT Operations. Critical events on a system should clearly be prioritized but criticality is a dynamic concept: the situation is markedly different when all the systems are running smoothly compared to the aftermath of a widespread incident such as a power glitch. IT Ops and others need the ability to modify the alerting levels, for example turning off all but critical alarms during a crisis or a planned software upgrade but, as with logs, the configuration settings should be secured against tampering.

Response procedures

Someone either has to design and document the processes for dealing with alerts and alarms during the development process, or else it will be left to the users, managers and IT Operations staff to decide what to do on-the-fly which is seldom sufficient from a governance point of view. It is sensible therefore for the system architects and developers to at least prepare skeleton procedures and outline flowcharts based on their understanding of the requirements, leaving the users, managers and IT Ops people to complete the procedural documentation. The procedures are likely to evolve with practical experience once the system is in production, so they should be version controlled and periodically reviewed by management to ensure key controls are not lost or diluted over time. If your project is upgrading an existing system, take the current response procedures as a starting point but be prepared to rewrite them.

For more information

Please contact the Information Security Manager or visit the department's intranet website for further advice and information about logging and alerting. Specialists from IT auditors and IT operations can help define logging and alerting requirements based on their experience with current systems. The Security Awareness notelet on audit trails, records and logs is also worth reading if you found this one of interest.

NOTICEBORED

Security Awareness Notelet

Logical access controls

Introduction

Logical access controls provide the technical means of determining what data users may utilize, the programs they can run and the modifications they can make. Access control is clearly of concern on shared/multi-user systems such as application servers and the mainframe, where one user may accidentally or deliberately interfere with resources belonging to another. Even on a single-user system such as a typical desktop or portable PC, logical access controls prevent the user inadvertently damaging the operating system, and help prevent the action of malware such as viruses, Trojans and worms. At the network level, access controls isolate logically distinct network segments and multiple traffic streams sharing a single segment. As well as the users, logical access controls also affect the abilities of computer programs themselves to interact with system resources. In short, logical access controls are a fundamental component of information security.

Access control terms and models

“The term *access* is often confused with *authorization* and *authentication*. *Access* is the *ability* to do something with a computer resource. This usually refers to a technical ability e.g. read, create, modify, or delete a file, execute a program, or use an external connection. *Authorization* is the *permission* to use a computer resource. Permission is granted, directly or indirectly, by the application or system owner. *Authentication* is proving (to some reasonable degree) that users are who they claim to be” [quoted from [chapter 17 of NIST Special Publication SP800-12](#)].

Whilst most of us are broadly familiar with the basic idea that it is possible to restrict read or write access to a file, this is the tip of an iceberg comprising theoretical models for access control. Although beyond the scope of this notelet, Google on [Bell-LaPadula](#) and explore from there.



Access policies and rights

Access policies are the ‘rules’ regarding access to controlled resources, as defined by management. General purpose corporate policies such as “access is only permitted where there is a business need to grant such access” are complemented by more specific policies for individual systems e.g. “access to the Finance system is only permitted where authorized by the Finance Manager”. At the lowest level of detail, policies become logical access rights encoded on the computer systems e.g. “user Fred Flintstone is granted read-only access to the first twelve columns on table Accounts_Paid”. Whereas the topmost level may be documented in the form of corporate information security policies, the middle level

typically has to be clarified by the business managers responsible for specific systems and the lowest logical access level is usually left to the development teams and functions such as IT Operations and Security Administration to figure out for themselves. Therein lies the rub: individual users and programs are granted whatever system access rights are actually encoded on the systems, regardless of what management may or may not have authorized.

Types of logical access control

Systems and applications define access types differently but here are some common labels:

- **Read:** a user or program with this right can view the information;
- **Write:** allows the user or program to modify the information – this usually implies they can also read the information and they may be granted automatic ownership if they create new records. Overwriting information with gibberish would be equivalent to delete access;
- **Execute:** ability to run a program (this label may have different meanings for other types of resource such as directories and devices);
- **Delete:** the user or program can erase data items;
- **Control or owner:** the user or program can alter the access rights, and grant themselves any type of access if they wish;
- **Create or append:** allows a user or program to insert new files, records or field in a database;
- **List or search:** the user or program can see information about other files in a directory;
- **ACLs (Access Control Lists):** give a more granular level of control over rights assigned to individual users/programs under certain circumstances e.g. 'during working hours only'.

Rôle Based Access Control (RBAC)

In all but the simplest of situations, it makes sense to define user access rights not by the specific individuals but by their intended **rôles** on the systems. Three different clerks in Accounts Payable are likely to need similar access rights on the Accounts Payable system, so one access profile would serve for all three of them. If only one of them needs read access to the General Ledger system, a suitable read-only rôle may be defined on the GL system and granted to the corresponding user. If another one also needs access, they can simply be granted access to the read-only rôle without having to define the entire set of access rights from scratch. If one of them leaves the company and is replaced by a new employee, the rôle-based access can remain unchanged while the first person's access is removed and the new person is added.

A subtle advantage of RBAC is that managers, auditors and other interested parties can more readily evaluate suitability of the access rights in effect, even if they have no idea who the individual users are. A "clerk" having unrestricted access rights to the entire GL would hopefully stand out like a sore thumb on a reasonably well controlled system. This implies, though that the functions to print or review access rights were supplied when the system was implemented ...

Practical applications of logical access control

Logical access controls can protect operating systems and application programs from unauthorized modification or manipulation, protecting system integrity and availability. They can also protect the integrity and availability of information e.g. preventing programs conflicting over the same memory space, and restrict the disclosure of confidential information to unauthorized individuals. If most users have limited access to ordinary desktop PCs on the corporate network, for example, they will be unable to install unauthorized, non-standard and possibly unlicensed or undesirable software. The number of Help Desk calls due to the related security issues will fall, albeit at the cost of an slight increase in calls to install the legitimate business applications they need.

For more information

Please contact the Information Security Manager or visit the department's intranet website for further advice and information about logical access controls. If you found this notelet interesting, others in this series might be useful including those on cryptography, operating system security, database security, passwords, user authentication and security administration.

NOTICEBORED

Security Awareness Notelet

Network security

Introduction

The ubiquity of networking protocols such as TCP/IP, UDP, HTTP and FTP is extremely convenient for interconnecting systems but at the same time propagates serious security flaws inherent in these networking protocols and the associated software utilities. System architects and developers need to be well aware of the fundamental constraints in order to develop secure networked systems. This notelet merely gives a brief introduction to the topic.

Data, message and link encryption

It is safe to assume that practically all network traffic is vulnerable to interception by third parties, potentially including internal LAN traffic and all types of WAN. Naturally, the ease and extent of third party access varies according to the communications medium and routing but it is difficult to be certain that both elements are under control e.g. it is not unknown for organizations to route supposedly internal traffic over public Internet links due to configuration errors. Good practice suggests therefore that sensitive data should be encrypted when communicated over any network.

Encryption can be applied at various stages of the process. SSL, for example, encrypts data flowing between the client browser and the web server, whereas VPNs can provide 'encrypted pipes' with unencrypted data flowing at either end of the VPN links. End-to-end encryption guards against interception even in supposedly trusted parts of the link but is not always appropriate – for instance, antivirus software on a proxy server cannot normally scan SSL traffic unless special arrangements are made to proxy the SSL link (which introduce security concerns of their own).

Switches, routers and firewalls

Network switches are more-or-less dumb junction boxes. In recent years, switches have begun to incorporate security functions, blurring the line between switches and routers. Routers are like intelligent junction boxes that route traffic to appropriate network segments depending on routing rules and the content of network packets. Firewalls are sophisticated routers with a particular focus on security. They inspect the data content of packets as well as the routing information and apply more complex rules, often tracking the state of the network connections at higher layers of the OSI stack. A basic security rule states that firewalls should be dedicated single-purpose devices, the reason being that other applications such as web servers make the configuration more complex and are likely to introduce security vulnerabilities that could be used to bypass or undermine the firewall security rules.



Internet-facing systems and wireless networks

Whilst private networks and internal LANs are not immune to security problems, security is a particular concern for systems directly connected to the Internet or located in a De-Militarized Zone (DMZ) between the Internet and internal networks. The Internet is home to millions of potential threats, both automated and manual, so vulnerabilities in Internet-facing systems are likely to be ruthlessly exploited unless they are 'hardened'

i.e. tightly secured having very few applications loaded and ports open. An unpatched system with known security vulnerabilities is likely to be compromised literally within minutes of being connected to the Internet – it's the lion's den of networks.

Wireless networks such as WiFi are basically untrusted and should be treated just like the Internet. The original WEP encryption standard for WiFi is barely worth using while the later WPA also has significant flaws, so WPA2 and/or VPNs are preferred. All wireless networks are potentially vulnerable to Denial of Service attacks through strong noise signals on the radio frequencies being used, including interference from other devices sharing the waveband. Beware that wireless networks often extend beyond the rated range if users or hackers have high-gain antennas.

Network Intrusion Detection Systems (NIDS)

NIDS are specialized network applications that monitor network connections and traffic for the symptoms of hacking and worm attacks. If a pattern of abuse is detected, NIDS can simply log events and raise alarms, or they may initiate automated responses such as shutting-off ports. More sophisticated responses are the realm of Intrusion Prevention Systems. Like antivirus software, NIDS are only as good as their pattern recognition capabilities and new attack methods are constantly being invented. NIDS should be treated as a valuable supplement to, but not a replacement for, other forms of network, server and application security.

Network integrity issues

Basic integrity features are built-in to all networking protocols to overcome fundamental constraints of data communications such as 'noise on the line'. Examples include Cyclic Redundancy Checking (CRC – a method of identifying bit-level changes), 'sequencing' (so messages can be reconstructed from packets that arrive out of order due to taking different routes *etc.*) and 'handshaking' (exchanges where, unless the recipient confirms receipt, the sender re-sends information). Additional stronger integrity measures are available through IPsec and other layered protocols using encryption, including authentication of message contents and end-points.

Network availability issues

Today's networks are highly resilient, on the whole, thanks to packet-switching, multiple redundant links *etc.* This can lead to a degree of complacency, however. Anyone who has suffered an extended network outage due to a broken line (typically through someone digging-up the network cables) or equipment failure will appreciate the value of having suitable fallback arrangements. The spread of IP telephony is making this issue even more important: an organization whose network connection fails may be unable to use both email/file transfer and the telephone without some alternative connection. Under Disaster Recovery conditions, network bandwidth is commonly very limited so it becomes essential to prioritize urgent traffic, perhaps using Quality of Service (QoS) settings and, where possible, offloading large file transfers *etc.* to tape or other physical media ('courier net' or 'sneaker net').

For more information

Please contact the Information Security Manager or visit the department's intranet website for further advice and information relating to network security. Other notelets in this series cover related issues such as encryption and operating system security, and various aspects of application security. The Network Operations team would also be pleased to assist with the evaluation of security requirements and design of secure network facilities.

NOTICEBORED

Security Awareness Notelet

Operating System (O/S) security

Introduction

Operating systems (O/Ss) mediate access to system resources such as disks, memory, keyboards, screens and networks by application programs, providing important security services such as user authentication and access controls. Aside from BIOS-level embedded systems, SCADA and other special purpose devices, most computers are multi-function devices and therefore have a wide range of capabilities and options. In many ways, the O/S can be considered the foundations of a system, making it important to (a) choose a suitable platform for the application you are building, and (b) install and configure it properly. This notelet gives some hints about the choices and tasks ahead of you in building secure systems.



Choosing a suitable hardware platform and O/S

Individual O/Ss have their strengths and weaknesses. Portable/open source O/Ss like UNIX and Java are relatively cheap but tend to be 'fragmented', composed of separate programs and utilities developed small army of independent developers. Proprietary O/Ss such as MS Windows and IBM z/OS are more coherent and standardized but generally have a more restricted selection of utilities, applications, hardware ... and developers. If you need the enormous number-crunching power of a supercomputer, are developing for a particular model of PDA or mobile phone, or will be processing classified military information, your choice of operating system and application software may be very limited. Midrange servers present far more diverse options and security is one of the factors to take into account. Think "horses for courses".

Securing the O/S

Installing and patching the O/S

O/S vendors are constantly working to fix security vulnerabilities reported to them by customers, hackers and their own staff. When a serious vulnerability is identified, the fix is generally supplied first as a patch for the latest O/S version and, where applicable, for a number of previous versions shortly afterwards. At some point, though, the vendor will refuse to support too many versions and will 'retire' old software, eventually withdrawing all maintenance and support. This approach places the burden on customers to upgrade their O/S and, where necessary, the hardware and layered application software too since they are usually interdependent.

From a security perspective, therefore, it makes sense for us to stay at or very near the latest version of any O/S in order to ensure that security patches are available promptly, and to obtain newer software that incorporates built-in security rather than having to be patched. There are usually commercial pressures as well – legacy systems support gets expensive, especially when you factor in the cost of finding and retaining developers and specialists with rare legacy skills. Ideally, lifetime system upgrades should be factored into the cost benefit case from the outset.

Configuring the O/S

Straight out of the box, systems vary widely in their default security status. Vendors of mid-range and desktop systems have traditionally supplied their software relatively open, making it easy for customers to get it running but at the risk that security holes are neglected. There are vague signs of a shift in emphasis to supplying software 'secure-by-default', such that customers have to open up network and user access explicitly. Either way, it is all too easy to mis-configure systems, leaving gaps in the defenses. There is a strong argument for scripting post-installation verification checks to ensure that key security parameters are correctly set before new systems are put on the network or released to users. Independent competent pre-release security testing is strongly advised for any system whose security is important *i.e.* business- or safety-critical systems.

Managing and operating the O/S

The IT Operations people who configure, manage, support and operate production systems have numerous systems to manage and juggle priorities as best they can. System architects and developers can help immensely by providing them with systems that are inherently well designed, well-behaved and well-documented. Examples include:

- Where possible and appropriate, choosing O/Ss that are already in use in the organization and hence the associated skills already exist;
- Applying O/S security standards;
- Plug-in-and-go installation routines that install from CD-ROM/DVD, pre-configure and verify the software to a known starting point (useful also in Disaster Recovery situations, for example to rebuild the system following a virus infection);
- Scripted system and application startup and shutdown routines;
- Use of parameters for configuration options such installation directory, logging directory *etc.*
- Resilient 'fail-safe' systems that behave predictably and fail gracefully if at all *e.g.* urgent systems management functions are run at a higher-than-normal priority; routine background operations run at normal or below-normal priority; long batch-type operations have checkpoints and restart functions, plus self-monitoring routines;
- Backup and recovery routines with simple configuration options;
- Management and monitoring/reporting interfaces that clearly identify key parameters and trends *e.g.* performance and capacity monitoring; configurable alarms;
- Security utilities and tools for log analysis, access rights configuration and reporting, validating configurations and integrity-checking data files and programs *etc.*;
- Operations manuals that explain the architecture and various system components and describe the correct operation of systems/network management functions.

Investing effort into these aspects during the development phase pays off in the long run in terms of the lifecycle support costs and, just as importantly, stability and security of the system.

For more information

Please contact the Information Security Manager or visit the department's intranet website for further advice and information relating to O/S security. Other notelets in this series cover related issues such as encryption and network security, backups, logs, security management and various aspects of application security. The IT Operations and Architecture teams can assist with the evaluation of security requirements and design of secure systems, and will gladly share with you their preferred/production acceptance standards for configuration and operations aspects.

NOTICEBORED

Security Awareness Notelet

Passwords

Introduction

Passwords, pass phrases, PINs (Personal Identification Numbers), private cryptographic keys *etc.*, collectively called “passwords” in this notelet, are classic examples of secret credentials used to authenticate someone or something. Since anyone who presents the correct credentials will appear authentic, passwords must be kept confidential to have any meaningful value. This notelet gives an overview of the security control requirements relating to passwords and describes the factors to be taken into account when designing and developing systems that use them.

Password controls

Password storage, display and communication

Passwords must never be stored unencrypted on disk or paper. Ideally, they should not be stored at all. A common technique is to create and store hashes of user passwords, using a suitable cryptographic hashing algorithm and randomly selected seed value – then, when a user presents a password for authentication, it is hashed and compared with the stored hash value, yet anyone who obtains access to the hash cannot readily determine the password (hashes should still be protected against unauthorized access to prevent automated brute force attacks e.g. using “rainbow tables”).



Passwords must never be displayed in clear text on the screen. Use standard system functions to replace password characters with asterisks *etc.* Do not provide help desk staff, system administrators *etc.* the capability to decrypt and store or display actual passwords.

Passwords must never be passed in clear over the network, particularly not an insecure/untrusted network such as the Internet. Challenge-response protocols are preferable.

Password parameters

Like short encryption keys, short, easy-to-guess passwords are highly vulnerable to brute-force attacks. A hacker can try numerous passwords in the hope of finding a match. Consult the corporate information security standards for details on password parameters such as minimum length and complexity, designed to make it harder to guess the password.

Password verification systems should be designed to prevent people making multiple guesses in quick succession, although it is conventional to allow up to two wrong passwords in sequence before locking out the user after a third in order to cater for users who simply forget their passwords. To reduce help desk calls, it is preferable to reset the lock-outs automatically after a configurable cooling-off period, with the lock-out period reflecting the system risk level e.g. low risk systems might reset after 10 to 30 minutes; high risk systems might not reset for hours or days, if at all. Alerts/alarms and additional countermeasures may be required to respond to automated password guessing attacks, for example to identify sequential failed guesses against different user IDs from a single terminal or IP address, even if each user ID is only guessed once.

Password set and reset functions

Resetting forgotten passwords creates a significant drain on IT help desk resources everywhere so anything your system can do to reduce the problem will help, provided you do not compromise the basic principles of confidentiality of course. Some low-security systems accept password hints, for example, that are chosen by users when they change their passwords – be sure to check that users do not simply enter the actual password as a hint! Password hints are unlikely to be acceptable for medium- to high-security systems since they give too much help to hackers. The same goes for simple questions such as “What is your favorite color?”, even if you use several. Question and answer pairs supplied by users are slightly better in that users can pick more obscure things to ask but, naturally, the more obscure they are, the greater the chance that users will not remember!

For medium- to high-risk systems, it is preferable to authenticate the user in person if they forget their password. Have a trusted intermediary check their photo ID *etc.* and vouch for them.

When setting or resetting passwords, help desk and operations staff must be encouraged to choose and dispense high quality passwords for two key reasons:

1. Simple/weak passwords may be easily guessed. If I overhear you at your desk calling to reset your password, I might quickly try guessing your new password and gain access to your system. When a new system is released, I might simply guess the default passwords.
2. Someone in authority giving the user a weak password implies that such bad practice is acceptable. Conversely, complex passwords (ideally random strings of upper and lower case letters, digits and punctuation) make the point that passwords should always be strong.

Cryptographic keys

Cryptographic keys *must* be stored securely, if at all. Keys that are only temporarily resident in RAM are less of a concern but are still potentially vulnerable to sophisticated attacks *e.g.* using rootkits or physical access to the DRAM devices. Ideally, keys should only ever be stored in certified hardware cryptographic modules that incorporate strong physical and logical access controls *e.g.* tamper-proof and tamper-evident enclosures; mutual authentication of devices requesting and providing keys.

Key management is a significant concern for cryptographic systems. Obviously enough, encrypted information should not be accessible without the correct key, therefore loss of the key could be a very serious problem. Systems to store and distribute crypto keys, and indeed safe keys, must be extremely well secured to prevent them being disclosed, damaged or lost.

In summary, cryptography is a specialist area of system security: if data confidentiality or integrity is important enough to require the use of encryption, consult Information Security for advice.

Multifactor authentication

Despite all the above, passwords may be insufficient to authenticate users on some systems, especially privileged users. Multifactor authentication using tokens such as magstripe or smart cards, SecureID *etc.* and biometrics such as fingerprints or iris scans, in addition to passwords or PINs, should certainly be considered during the design phase where the risk of unauthorized access is unacceptable.

For more information

Please contact the Information Security Manager or visit the department’s intranet website for further advice and information relating to passwords, PIN codes, crypto keys *etc.* Other notelets in this series cover related issues such as encryption, security management and authentication. The issues in this notelet are best considered early in the design phase so please seek help at the earliest opportunity to avoid the costs, delays and constraints of retro-fitting security.

NOTICEBORED

Security Awareness Notelet

Physical IT security

Introduction

Computer systems cannot be secured through logical controls alone. Physical access to the systems could permit a knowledgeable person to bypass many forms of logical control such as BIOS passwords and, of course, even a complete idiot could steal or physically damage computers rendering them unusable. Power failures/surges, overheating, fire/smoke and floods are further physical threats to consider. The security controls outlined in this notelet are necessary to minimize the physical security risks affecting IT systems hardware.

Physical access controls

PCs

Users are generally entrusted with personal computers (desktops, portables and hand-held devices) and peripherals since it is in their interests to look after the equipment and the impacts of breaches are limited. Even so, the organization should make sure basic controls are in place. Equipment intended for portable use should be physically robust by design. Tamper-proof asset tags should be fitted to valuable IT assets and procedures documented for periodically checking IT assets against inventory records. Policies and guidelines to users should point out that they are responsible for protecting the equipment against harm or loss, and for reporting breaches promptly e.g. to enable remote access to be suspended. They should ideally be held personally accountable for equipment issued to them.

Mainframes

It is clearly inappropriate for users to have corporate mainframes in their offices or homes, not least because of the size, power/heat load and noise. These obviously belong in properly serviced computer facilities where physical access is normally restricted to authorized people using physical locks, pass-cards, CCTV *etc.* Computer centre design is well beyond the scope of this notelet.

Servers

Physical security for other shared IT equipment such as departmental/mid-range servers, LAN switches *etc.* is often neglected. Thought should be given to the need to protect all equipment according to the risks: do not forget that the physical asset value may be only a fraction of the true worth, especially if you consider the amount of corporate data routinely passing through or stored on shared IT equipment. Shared servers, LAN equipment *etc.* should ideally be located in secure computer rooms, or at least in locked cabinets that restrict unauthorized physical access. Tamper alarms, remote-reading temperature monitors and internal Uninterruptible Power Supplies *etc.* can reduce the impact of incidents.



Communications cables and media

Physical cabling is the Achilles heel of many IT installations, with cables often draped loosely around offices and termination boxes left unlocked. Fiber-optic cables are less vulnerable to tapping than copper or wireless connections but are more vulnerable to physical damage. Cable conduits and ducts should be used where possible, with tamper-proof markings and secure fittings to limit access where risk-justified. It is difficult to restrict physical coverage of wireless LANs *etc.* and logical security measures are imperfect - avoid Wi-Fi *etc.* unless your security requirements are minimal and you know what you are doing. Protect portable storage media disks, diskettes, CDs *etc.*) against damage, theft and unauthorized access using secure cabinets, fire safes *etc.*

Environmental services

Power control

A failure of mains power is obviously bad news for IT equipment, even those with internal batteries or Uninterruptible Power Supplies (UPSs) *unless* power is restored before the batteries are exhausted. Standby generators are essential in parts of the world that suffer frequent power outages, or where failures are less common but the impacts of service outages warrant the cost – most dedicated computer rooms fall into this category. UPSs are generally advisable to avoid even temporary power interruptions whilst standby generators start-up following a blackout, and to smooth out common problems with the electrical supply (so-called brown-outs/dips, spikes and surges). Power engineering is a specialist discipline – consult an expert to avoid causing more problems than you solve. Even if you will be installing a new system in a well-appointed computer room, don't forget to inform Facilities so they can track the power usage and heat load. Lightning protection and earthing should also not be neglected on security, health and safety grounds

Temperature control

Most IT equipment works best at “office temperature”, around 21°C. Virtually all the electrical power consumed by equipment is emitted as heat, therefore a heavy concentration of powerful IT equipment is likely to cause localized heating of the room. Unchecked, this can cause overheating and sudden equipment failure or shutdowns, if not fire. Reliable air conditioning is therefore a vital control in most computer rooms, along with over-temperature alarms ideally fitted with remote alarms at permanently-manned security desks for times when there is nobody working inside.

Fire, smoke and flood control

Detective controls such as fire/smoke alarms, water detection *etc.* for computer rooms should be supplemented by preventive controls such as limiting flammable materials and corrective controls such as hand-held and perhaps automatic extinguishers to minimize the associated risks.

Support and maintenance

Even the most secure computers and computer facilities need to be professionally serviced. Regular maintenance allows adverse trends to be identified and faults corrected before incidents occur, while support or call-out contracts improve the chances of equipment issues being resolved before IT services are affected. The same principle applies to software and systems support.

For more information

Please contact the Information Security Manager or visit the department's intranet website for further advice relating to this notelet. The Physical Security and Facilities Managers can also assist in this area and should be consulted if physical security is a serious concern for IT equipment being delivered and configured by your development project.

NOTICEBORED

Security Awareness Notelet

Security administration

Introduction

In an ideal world, IT systems would look after themselves with minimal need for routine administration. In practice, though, systems need to be tended, security aspects included. Security administration involves a blend of configuring, monitoring and actively managing security elements.

Administering systems security



If your development project will deliver a system for installation and management by another function, group or person, it is *your* responsibility to provide the procedures, guidelines and tools to enable them to install, configure and generally administer the production system. IT Operations staff are normally responsible for hardware and software installation and configuration, installing equipment in racks and hooking up power and comms cables, for instance. Software installation routines typically set up new user IDs for system administrators, the sysadmins then have to set or reset default passwords, configure security-related parameters such as access rights, backups *etc.*

IT Ops normally manage, monitor and support the operating system and communications layers for all production systems, ensuring that systems interface correctly with the remaining IT infrastructure. They have generic responsibilities to protect the IT infrastructure as a whole and are therefore entitled to ensure that new systems and changes will not cause adverse impacts. Whereas IT Services Management may negotiate Service Level Agreements on behalf of IT, IT Ops are in the front line for delivering reliable IT systems and services. Liaise with IT Ops to avoid landing them with an impossible job.

Things that will make IT Ops' job easier include:

- Heavily-automated system installation routines/scripts with well-documented procedures and installation self-checks to validate complete and correct configuration (validation routines that can be run at any time may be useful if a system is not performing correctly or perhaps to confirm that a system has not been hacked);
- Parameterized/configurable installation processes such that systems can be installed in suitable directories with appropriate disk space limits *etc.*
- Automated functions for routine administrative tasks such as backups and archives;
- Management and monitoring functions to track and control aspects such as disk capacity, performance, logs, alarms/alerts *etc.*;
- Consistent user interfaces and, where applicable, hooks for enterprise management tools;
- Limiting use of system privileges by administrators and users;
- Complete and accurate systems documentation.

Administering middleware and application security

“Middleware” sitting between the operating system and layered applications, and applications themselves, are usually administered by experts in the respective programs. Oracle database administrators, for example, have specialist skills associated with installing, configuring and managing all Oracle applications installed on a system. The same applies to those responsible for complex enterprise software such as SAP R/3. These experts each have their own standards, procedures and guidelines, and naturally prefer new systems to fit-in with existing processes wherever possible – the best way to ensure that is to draw them into the development process from requirements specification and design through development and testing to implementation.

Many information security controls are found in the middleware and application software layers. Examples include:

- Specific application and data access rights assigned to user rôles;
- Software routines for validating programs and data;
- Software resilience measures such as keep-alive timers, real-time synchronization and automated failover;
- Business logic controls, work flow sequencing *etc.*

Once a system is running in production, middleware and application support teams administer these controls either by configuring dynamic parameters or by changing static values and making software changes through conventional change control processes.

Software development teams can help by automating routine tasks and providing ‘dashboards’ for monitoring and controlling complex systems, along with comprehensive documentation.

Administering user security

User administration is another important factor in providing secure and effective IT systems and services. User admin is largely a matter of configuring and managing user access rights such as:

- Setting up userIDs and passwords for everyone who will need to use the system when installed;
- Allocating users to appropriate rôles predefined on the system;
- Applying access rules defined by management *e.g.* to enforce divisions of responsibility;
- Re-authenticating users and resetting passwords when they forget them;
- Periodically reporting access rights for management review and altering them when authorized change requests are received;
- Suspending or altering access rights when users move departments, leave the organization, or when security incidents occur.

Speak to the manager responsible for administering user security to find out which tools, systems and processes they use. Ask them for applicable policies and standards. They can also advise you on issues such as correct formatting of userIDs and passwords, password change mechanisms, access rights reporting *etc.* As always, automated functions and documentation of the user admin tasks helps enormously.

For more information

Please contact the Information Security Manager or visit the department’s intranet website for further advice relating to security administration. Other notelets in this series expand on related aspects of security such as backups, database security and IT operations. IT Operations, DBAs, Security Admin *etc.* will gladly explain their standards, procedures and guidelines for administering systems, preferably well before you anticipate delivering yours into production.

NOTICEBORED

Security Awareness Notelet

Security-relevant laws and regulations

Introduction

The continuing trends towards e-business and globalization have accelerated the adoption of and reliance on IT which has led in turn to an increased level of legal and regulatory controls in the IT arena. This notelet outlines the relevance of various laws and regulations to information security aspects of systems development but please bear in mind that *this is a generic security awareness briefing not legal advice!*

Laws and regulations

Corporate governance and financial controls

Despite applying specifically to certain US-stockmarket-listed corporations, the Sarbanes-Oxley Act (SOX) has had global ramifications. Similar regulations are often imposed on companies by the national stock exchanges and legislatures where they trade or are listed e.g. the 8th Directive on Company Law in the European Community; tax laws everywhere; companies or corporation acts in many countries. In general, their aim is to prevent major frauds and scandals such as Enron by tightening up the requirement for senior management to attest to *i.e.* make formal, legally-binding statements about the accuracy and completeness of the reported financial accounts. Information security comprises a very significant element of the governance controls providing management, stakeholders and regulators with confidence in an organization's governance.

Basel II requires banks to implement enterprise risk management including their IT risks and maintain adequate capital to protect their financial stability.

Confidentiality, privacy and data protection

About half of the world has enacted legislation to protect data privacy. The Information Privacy Act [Australia], Personal Information Protection and Electronic Documents Act (PIPEDA) [Canada] and Data Protection Act [most European countries] aim to protect personal information, generally defined as information relating to identifiable, living individuals that they would consider private and personal. Information security controls are necessary to ensure personal privacy and data integrity, including situations in which personal data are transferred abroad. In Europe, at least, the definition of personal information encompasses computer data such as personnel databases, video/CCTV and audio data, and other/hardcopy formats such as card index systems.

The Gramm-Leach-Bliley Act [USA] concerns the privacy and security of financial information. It effectively mandates the use of risk assessment by banks to derive their specific control requirements. The Security Breach Information Act (Californian SB 1386) and similar disclosure laws in other US states force companies whose credit card processing systems are breached to inform all the credit card holders of that fact. They support the principle that organizations which are responsible for maintaining the security of financial (and other) valuable information entrusted to them by third parties can be held accountable for that.

Some personal data are classed as especially sensitive. Laws such as the Health Insurance Portability and Accountability Act (HIPAA) [USA] broadly define the need for administrative, physical and technical controls to secure individuals' medical data. Other examples of sensitive personal data are information on a person's sexual orientation and their criminal history.

Intellectual property rights

Laws in most countries provide legal protection for the owners of intellectual property. Terms such as 'ownership' and 'intellectual property' are quite tricky to define in law, thus contributing to extensive differences in the small print and legal status of software licenses, patents, trademarks and designs around the world. This is a minefield for the unwary. Tame your lawyer.

Industry-specific laws and regulations

The financial services, medical and defense industries, in particular, must comply with numerous specific rules, as must professionals such as lawyers, accountants, auditors, doctors and dentists. It is not appropriate to reference them in this generic notelet – speak to the relevant legal, compliance and regulatory function/s in the organization for further information.

Laws and regulations against hacking, spam, wiretapping, spyware *etc.*

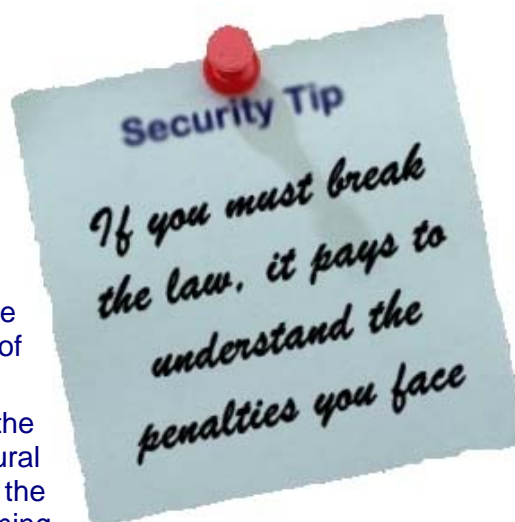
Additional legislation has been enacted or is being introduced in many countries to cover hacking, spam, wiretapping, spyware and so forth. Examples include the "CAN-SPAM" Act [USA], the US Economic Espionage Act [USA] and the Computer Misuse Act [UK]. Even health and safety laws may apply where IT systems are controlling safety-critical equipment, or where users may be put at personal risk through their management or use of systems.

Compliance aspects

Managers and staff must proactively support the framework of information security and related controls. More than just tolerating the inconvenience of login controls, divisions of responsibility and so on, proactive support helps avoid the threat of legal sanctions such as fines and jail terms.

Compliance activities include:

- Awareness, training and educational activities to ensure that everyone understands their obligations in respect of information security and other controls;
- Periodic/regular reviews, checks and audits on the operation of the controls, including technical, procedural and managerial controls, both internally (within the organization) and in some cases externally e.g. confirming that third parties are complying with their contract or license terms;
- Ad-hoc/unannounced checks, not to 'catch people out' but to fill in the gaps between scheduled reviews and to focus on specific problem areas or perceived high risks;
- Structured management processes for dealing formally and fairly with any noncompliance, plus informal processes such as informal/off-the-record reminders to staff about protecting sensitive or valuable information;
- Legal action, generally as a last resort. This is an expensive option that cannot easily be dropped or reversed once started. Professional legal advice is essential. This is not it.



For more information

Please contact the Information Security Manager, Legal and/or Compliance Departments for further advice relating to the legal and regulatory aspects of information security. Compliance obligations should be identified as mandatory requirements early in the development lifecycle and a close eye kept on any that arise thereafter.

NOTICEBORED

Security Awareness Notelet

User authentication

Introduction

We authenticate individual people (users) to be sure they really are who they claim to be, prior to deciding whether they are allowed to access our information assets. We hope to spot imposters and fraudsters, and differentiate them from authentic individuals.

User authentication mechanisms

It is comparatively straightforward for people to authenticate other people face-to-face. We may simply recognize them and can quite easily check someone's credentials such as a company photo pass, drivers' license, credit card or passport - not just checking the printed details but checking the authenticity of the credentials. However, it is much harder for computer systems to authenticate people, especially those at the far end of a network connection or telephone line where authenticating physical appearance and credentials may be difficult.

Authentication is not just about preventing unauthorized access – it is also used to keep tabs on legitimate users, for instance by keeping secure system records to prove exactly what someone did on the system. If they later claim to have done something different, the audit trail may be checked to verify or refute their claim (non-repudiation).

Single- and multi-factor authentication

Passwords associated with userIDs are the main mechanism for authenticating users. The network or system login process is an obvious example. If we do not require users to enter personal userIDs at logon time, we have no obvious way within the system of associating their online activities with them, specifically. If we request userIDs but not passwords, someone can easily enter a false userID and, as far as the system is concerned, assume the identity of another person.

The crux is that passwords must remain secret, only known to the person we are authenticating and difficult to guess. As far as the computer is concerned, if someone claiming to be John Doe logs in with John Doe's password, then they *must be* John Doe. Passwords are **something the user knows**.

Security tokens such as credit cards and smartcards add another level of checking. Tokens are **something the user has**. To have any value in authentication, tokens must be kept secure by the person to whom they are issued, and must be difficult to forge. If I steal your token or make a copy, I have a better chance of masquerading as you.

Biometric checks such as retina or fingerprint scans are more difficult to fool than the others because they relate to **something the user is**. There are two important drawbacks with biometrics: (1) the equipment to test biometrics in a secure manner is not widely available and generally has to be purchased and installed for each user at some cost (and remember that if I can manipulate the data stream passing between your hi-tech biometric device and computer, there's a



chance I can inject false data replicating the response from an authorized fingerprint, iris pattern or whatever); (2) biological features are inherently difficult to measure due to natural variations including day-to-day physiological changes in the person and repeatability issues with the measurement process, which limit the ability to authenticate people reliably. More arduous enrollment and checking procedures can help provided you can persuade people to participate.

Multi-factor authentication combines authentication techniques from more than one of the above categories - not just multiple passwords, for instance, but passwords plus tokens and/or biometrics. Chip-and-PIN credit cards are a good example: purchasers must present difficult-to-forge smartcards that are checked by the card reader (a form of token), and enter a secret PIN code (the second factor) to authorize the sales transaction.

The rôle of cryptography in authentication

Cryptography is often used for authentication and validation purposes (integrity), as well as for access control (confidentiality). Examples include:

- Use of encryption and random-numbers-used-once (“nonces”) in challenge-response processes to both authenticate the person or system on the other end and avoid disclosing secret passwords or keys to hackers monitoring the wire;
- Cryptographic verification of digital signatures on emails and other electronic messages, especially those containing important financial or other transactions, to ensure that the messages originate from the supposed originator and to check for unauthorized modifications;
- Cryptographic authentication of smart cards and other security tokens to check for fake credentials.

Cryptography has an important rôle to play in maintaining the confidentiality of secrets used as credentials, namely passwords, PIN codes, private keys *etc.*

Identity theft

Criminals are actively exploiting security weaknesses to steal passwords and other credentials from users, and using the illicit information to masquerade as those users. In the financial services industry, social engineering, phishing and Trojan horse programs are the mechanisms most commonly used to steal identities. Controls against these attacks include, at the user end:

- Procedures and educational materials to make users more resistant to social engineering;
- Anti-phishing browser toolbars;
- Antivirus, anti-spyware and anti-rootkit software and other malware controls.

At the financial services end, identity theft controls include:

- Multi-factor user authentication (see above);
- Avoiding the use of emails and web pages that could easily be spoofed to mislead users;
- Dynamic authentication, sophisticated fraud detection and transaction validation controls that pick up on potentially fraudulent patterns of transactions and either block them or request additional identity verification;
- Manual response procedures *e.g.* to take phishing sites offline as soon as practicable.

For more information

Contact the Information Security Manager about user authentication or visit the department’s intranet website. Further notelets in this series cover passwords, cryptography, security administration and other related topics.

NOTICEBORED

Security Awareness Notelet

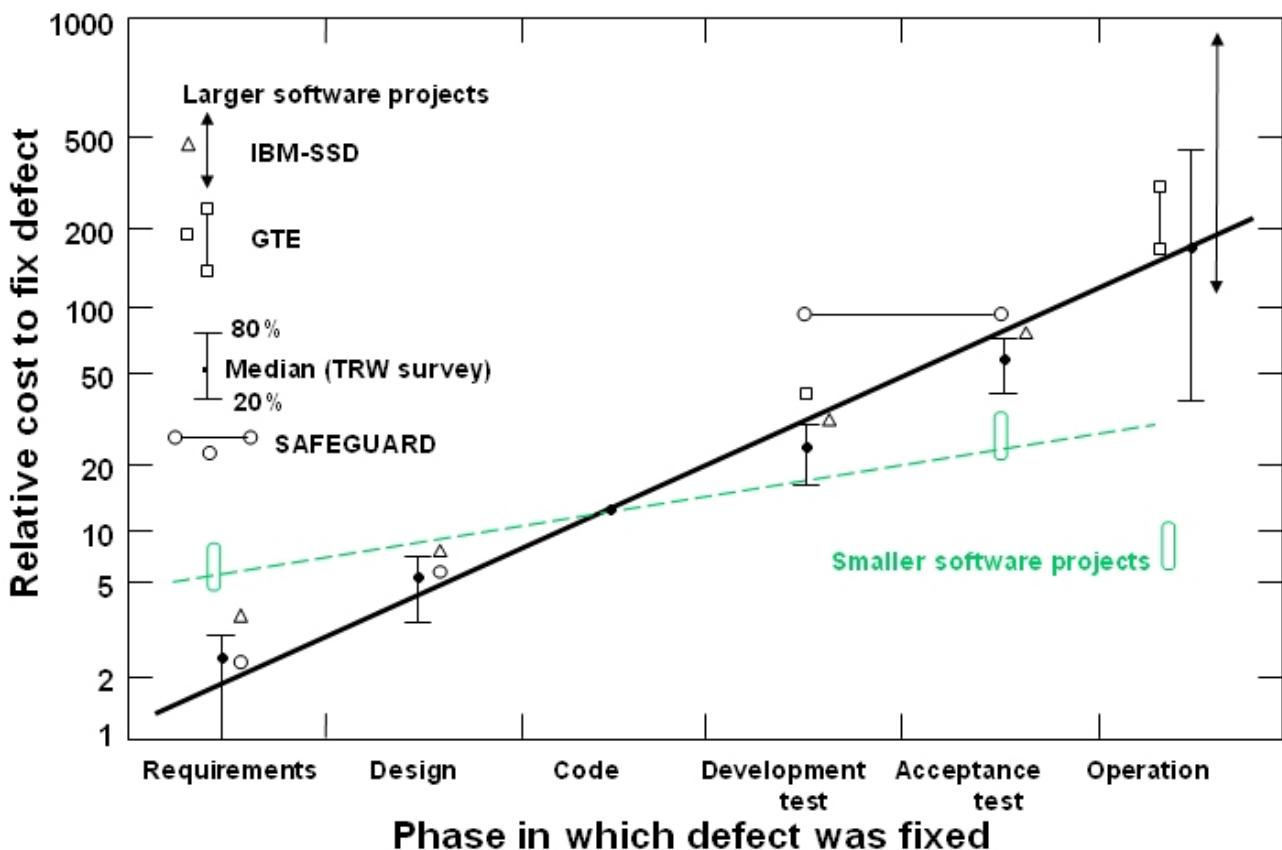
Security integration in software development - overview

Introduction

Information security controls follow the same project/lifecycle phases as software: the security requirements are analyzed, controls are designed, developed and tested, then they are implemented, used, managed and maintained. Eventually the controls are retired, often to be replaced by the next shiny new version. This means that information security activities can be directly aligned with development projects, and form the basis for this series of awareness notelets.

Cost of change

The graph shows research data* supporting the commonly held view that the costs of making software changes escalate markedly as a development project nears implementation, especially for larger projects (*note the exponential costs axis*):



The implication is plain: it pays to fix defects as early in the lifecycle as possible. Defects in this context are bugs and design flaws - discrepancies between what the software actually does and what is truly required and/or expected by the users. Clarifying user needs is the first challenge.

* From [University of Southern California](#)'s Center for Software Engineering

What happens if we don't get security right from the start?

The Windows Meta File bug from 2006 serves as a case study on what can happen if security-related bugs are discovered once a program is in production:

- Hackers discovered and started exploiting the .WMF bug (a “zero” or “oh day exploit”). Reports of websites that downloaded maliciously crafted .WMF files began to surface. Microsoft Windows customers worldwide were hacked. Soon the story hit the mainline news media;
- Microsoft started analyzing the bug, coming under extreme media pressure to release a bug fix quickly. Internal incident management and damage limitation activities started in earnest;
- Knowing that the bug was being actively exploited and that the Microsoft fix would not be ready for days or weeks, Information Security Managers everywhere were faced with difficult risk-based decisions: should they disconnect their organizations from the Internet and cause business disruption or should they stay connected and risk being hacked?;
- Microsoft discovered that the bug had lain dormant for years in code that was reused many times, meaning that it would be tricky to fix without breaking lots of applications. Bug fixing and testing continued apace;
- Before Microsoft's bug fix was released and whilst the bug was being exploited, Ilfak Guilfanov released a bug fixing patch. Microsoft suffered more adverse publicity having been beaten to the fix by an independent software developer. Information Security Managers faced more tough decisions about whether to apply the “unofficial” patch or wait for Microsoft's;
- Microsoft released the official bug fix slightly ahead of the regular monthly patch schedule, but only for recent versions of Windows. Customers with recent versions of Windows started testing and installing the patch. Unpatched customers continued being hacked;
- Eventually, patches were released for older versions of Windows to those with legacy support contracts;
- Comments about the incident continued for months, with some wags raising conspiracy theories that Microsoft ‘must have known about the bug’ and in fact ‘might have deliberately created it as a back door into Windows systems’. The theories were not widely believed but nevertheless the suspicion further challenged Microsoft's public image, already tarnished by their public admission that another critical security bug had escaped into commercial software;
- Even today, years later, there are *still* reports of unpatched systems being hacked as a result of the .WMF bug.



The customer impacts, development, testing and implementation costs, and adverse publicity associated with this one incident no doubt amount to several millions of dollars. We can only guess at how much would it have cost to analyze the security requirement and code the software securely in the first place – perhaps a few tens or at most hundreds of thousands of dollars. **That is why security needs to be built-in to systems not bolted-on.**

For more information

Contact the Information Security Manager or visit the department's intranet website for more information about integrating information security into software development processes. Further notelets in this series expand on security elements of system development lifecycle phases.



Planning and managing secure developments

Introduction

Information security controls don't usually 'just happen' by chance – they have to be actively developed. This notelet discusses the management processes necessary to ensure that information security is properly integrated into software development and procurement projects.

The business case for information security

Security functionality and controls can be costly to develop and therefore it is perfectly reasonable for management to insist that the expense is justified in relation to the return it will generate. This is easier said than done, however, because of a central security paradox: a highly secure system will suffer few if any security incidents but it is seldom possible to demonstrate that expensive incidents would have happened if the security controls had not been effective. The value assessment therefore tends to be largely based on conjecture around the risks.

ISO/IEC 27002's advice

Section 12 of ISO/IEC 27002, the international standard Code of Practice for Information Security Management, covers "Information systems acquisition, development and maintenance". The first control objective in section 12 is particularly relevant:

12.1 Objective: To ensure that security is an integral part of information systems. Information systems include operating systems, infrastructure, business applications, off-the-shelf products, services, and user-developed applications. The design and implementation of the information system supporting the business process can be crucial for security. Security requirements should be identified and agreed prior to the development and/or implementation of information systems. All security requirements should be identified at the requirements phase of a project and justified, agreed, and documented as part of the overall business case for an information system.

Building business cases that hold water

One characteristic of information security is that the impacts of control failures can be devastatingly expensive, especially if one takes the broad view that security controls protect confidentiality, integrity *and* availability. Working from the precept that we are simply trying to cost-benefit justify the security controls, we can use the rather open-ended value of potential security incidents to our advantage. Essentially, we just need to provide a sound basis for determining that the costs of developing secure software are heavily outweighed by the financial savings accrued by reducing the number and/or severity of security incidents. Provided the 'profit margin' is sufficiently large, the precise values in our value projections are relatively insignificant.

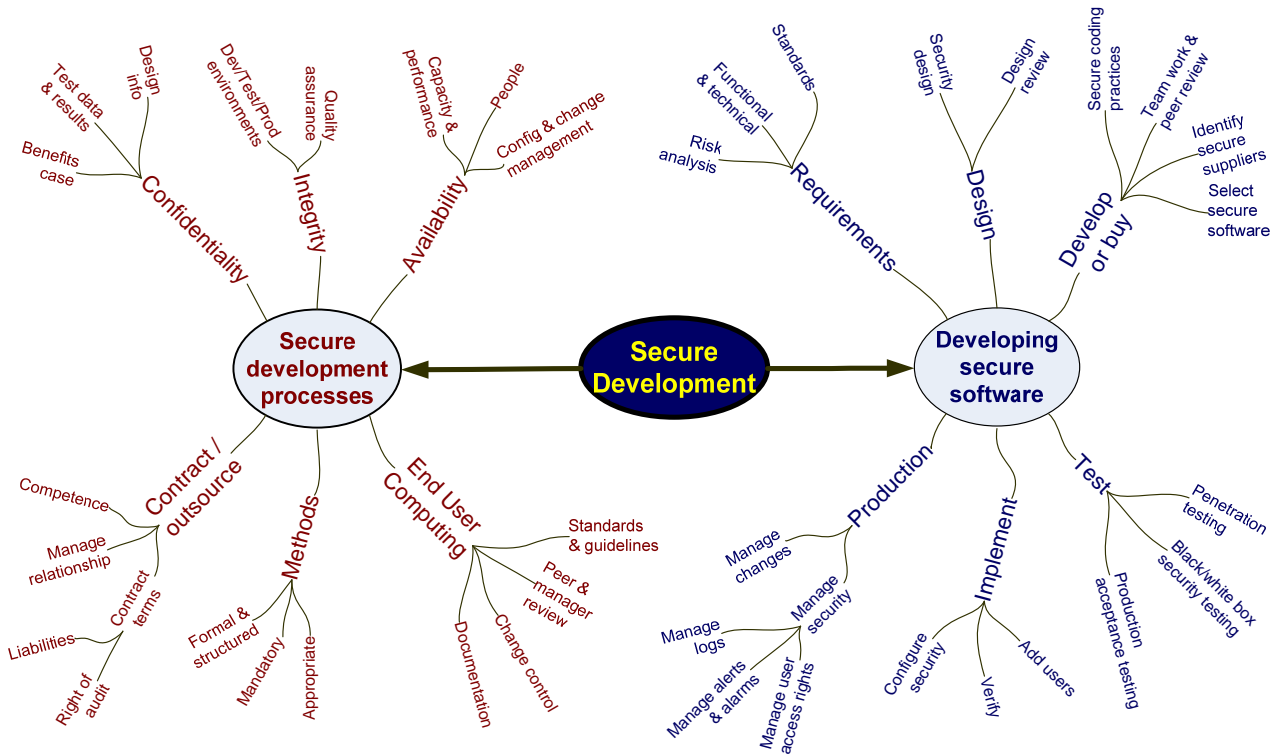
Take for example the case for conducting an independent security design review. We can estimate the costs of such a review fairly easily by contacting a few potential bidders for their indicative charges. The potential benefits include reduced re-work during the development and fewer security bugs that emerge in production. If, say, this is a financial management system, the security impacts could include errors in the accounts, unexpected losses and perhaps even legal or regulatory action under financial control and governance rules, plus of course the direct costs to fix the security problems. It shouldn't be too difficult to persuade management that it is in their interest not to be in a position of having to explain inadequate investment in security controls.

Planning security activities through the project

ISO/IEC 27002 section 12 is recommended reading for all development project managers. In addition to the control objective noted earlier, it covers: security requirements analysis and specification; correct processing; validation of input, processing and output plus message integrity; cryptographic controls; security of system files, control of operational software, protection of system test data and access control to program source code; security in development and support processes including change controls, technical review of applications after operating system changes, restrictions on changes to software packages, information leakage and outsourced software development; and technical vulnerability management.



Other security awareness notelets in this series discuss the various information security activities that should be integrated into each of the main phases of software development and procurement. Key activities that should be planned are shown on the right-hand mind map below:



Planning security for the development project environment

The left-hand mind map shows security aspects to take into account when planning a development project. There may well be confidentiality, integrity and availability considerations for the development project materials and systems, for example is it wise to publicize details of the security design beyond those who have a need to know?

For more information

Contact the Information Security Manager or visit the department's intranet website regarding the planning and management of information security in software development projects.

NOTICEBORED

Security Awareness Notelet

Information security risk analysis

Introduction

The risk analysis/risk assessment processes outlined in this security awareness notelet provide the means to evaluate and define information security requirements rationally and systematically for a new application or for changes to an existing one.

Assessing the key components of information security risk

In the context of information security, it could be argued that the only risks worth considering involve the coincidence of threats, vulnerabilities and impacts. If any one of these three elements is zero or negligible, the risk will also be zero or negligible. This implies a rational way to assess information security risks by considering each element in parallel, then bringing them together.

Information security threat assessment

Threats are the external factors that threaten confidentiality, integrity or availability of the system and/or the data it processes. These include deliberate or directed threats such as hackers and fraudsters plus accidental and nonspecific threats such as accidents, bad weather *etc.* Threats can be determined by looking at those that have affected existing systems and by brainstorming (especially good for considering completely novel threats).

Information security vulnerability assessment

Vulnerabilities are the inherent security weaknesses in a system or process. Whilst some information security controls reduce threats and impacts, most are intended to minimize technical, procedural or physical vulnerabilities. Creative techniques such as brainstorming, tempered with experience of vulnerabilities in existing systems and analysis of the likely technical environment for the system, can quickly draw up a set of vulnerabilities to consider.

Business impact assessment

This is generally the most business-focused element of the risk assessment process. It looks into the potential adverse effects on business processes, information value *etc.* if confidentiality, integrity or availability were to be compromised, regardless of the precise mechanism. A common approach is to explore some 'worst case scenarios' based on the kinds of security incidents that would give conscientious managers sleepless nights.

Information security risk assessment

To complete the risk assessment, we need to pair up threats with vulnerabilities to identify situations or scenarios called "use cases" and "abuse cases" that would cause significant business impacts. For example, a fire threat acting on a vulnerable lack of fire protection could cause substantial damage to the computer system; an internal fraudster exploiting his/her access to the system could make unauthorized funds transfers. If the threats, vulnerabilities and impacts are ranked even crudely into high/medium/low, the risks can easily be prioritized. Risk prioritization gives a natural sequence for designing security controls.



Determining information security control requirements

Having undertaken a structured risk analysis, the next step is to determine and document the requirements for controls to address the identified risks, ideally starting with the most significant risks. Security requirements may be classified in several ways such as the examples shown below. If security is a critical success factor for a system, there is value in comparing and cross referencing the different classifications to develop a comprehensive suite of control requirements.

Information security needs and features (example)

Needs: compliance with corporate policies for passwords; SOX-compliant integrity controls ...

Features: automatic response to fraud alerts by dynamically altering transaction limits ...

Preventive, detective and corrective control requirements (example)

	Preventive	Detective	Corrective
Confidentiality	Username & password required for normal user access to system Smartcard authentication required for access to sensitive transactions Biometric authentication required for access to privileged application or systems management functions	System routinely logs valid and invalid login attempts, access to sensitive functions and data, using secure log files Logs are analyzed weekly using automated routines, reporting to Security Admin	Appropriate analysis and follow up action is instigated if serious issues are found in the logs
Integrity	Input fields enforce mandatory formatting and other data validity rules Input tests apply to manual data entry and automated interfaces	Referential integrity rules applied in DBMS Periodic database integrity routines test for internal and external inconsistencies	Invalid data held in suspense and noted in management reports Suspense items trigger automatic escalation if not resolved within 48 hours
Availability	Server and key communications systems supplied from generator-backed no-break UPS System implementation/ changes cannot be implemented until all category 1 test failures are fully resolved	System facilities to monitor power status are used to inform Operations staff of power supply failures Performance and capacity monitoring targets defined in SLA, including response trigger levels	Operations staff contingency procedures incorporate appropriate responses to power failures Recovery priority defined in SLA

Functional vs. technical control requirements

Functional controls: user rights administration; system security administration; backup administration; dynamic alerts prioritization and response; log analysis and reporting; ...

Technical controls: regularly updated antivirus software on all systems; 256-bit AES encryption for personal details; intrusion detection system on WAN links; ...

For more information

Contact the Information Security Manager and/or Risk Manager for help to determine your information security control requirements. Their experience and understanding of information security threats, vulnerabilities and impacts, and knowledge of suitable controls, will prove invaluable.

NOTICEBORED

Security Awareness Notelet

Information security architecture and design

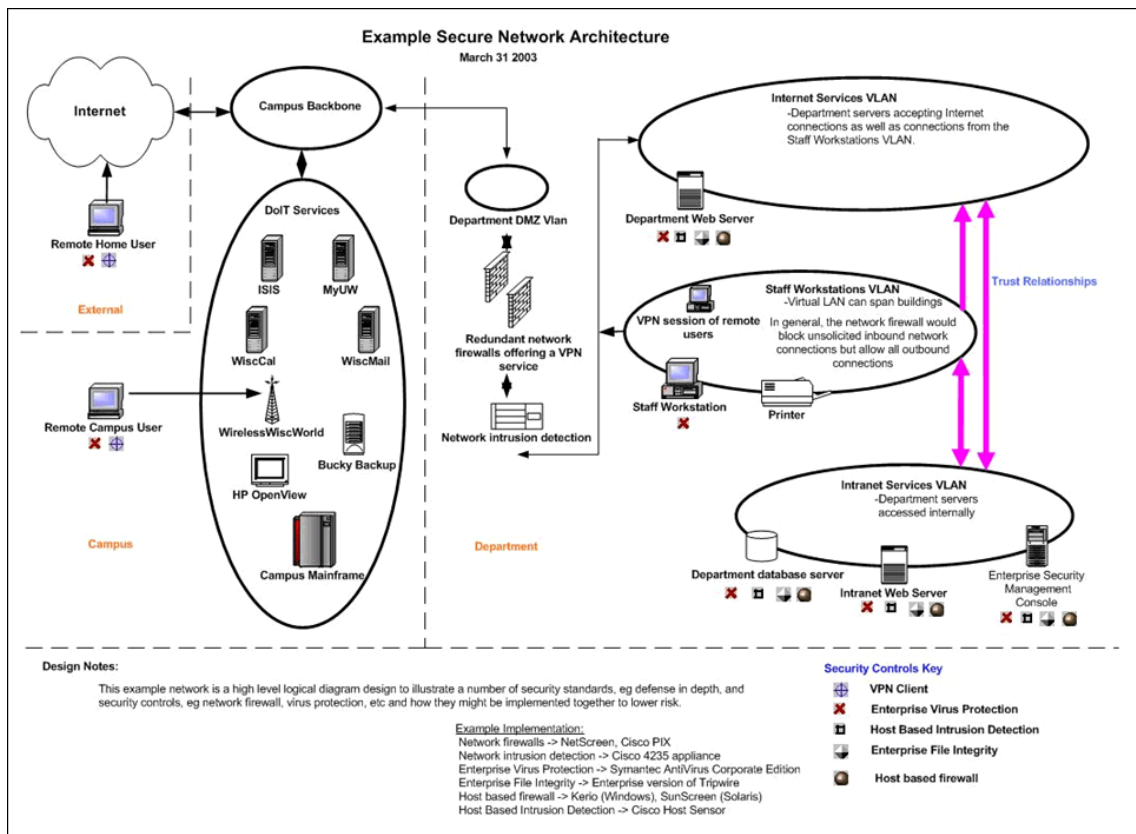
Introduction

Having determined the information security control requirements by risk analysis, the next step is to design the security architecture, assuming that we are going to be developing a system from scratch [if instead we intend to procure a ready-built system, check the next notelet in the series]. This notelet implicitly assumes we are preparing a discrete **security design specification** although in practice information security controls will often be specified as an integral part of other system design documents unless security is a critical success factor.

Developing a coherent security architecture

Rôle of the Security Architect

Whilst some organizations employ Security Architects, this is not the only option. Many that have permanent in-house software development functions have software architects with an interest in information security or else have access to external expertise from specialist consultancies and experienced contractors. Even small organizations are well advised to seek expert advice when designing systems security where the impacts of security incidents would be disastrous. The architect's rôle is to consolidate and consider the information security requirements determined in the previous phase, and prepare specifications for the corresponding controls. This typically involves starting with a high level design, then developing more and more details until the developers have sufficient guidance to be able to code the system, write the user guide etc.





Integrating disparate requirements

A central purpose of the architectural design phase is to ensure that all requirements are satisfied in the most efficient manner possible, and that any conflicts are identified and resolved rationally, with management approval where necessary. Performance/capacity and backup requirements are often developed independently of other information security elements, for example, and disaster recovery is sometimes the responsibility of a separate function distinct from Information Security Management. Access controls and user authentication requirements are often at odds with usability and performance goals – resolving discrepancies like this is where the investment in information security risk analysis starts to really pay off as there should be information to hand to justify and support the business needs for information security controls.

Enterprise security architecture

Some organizations define ‘enterprise security architectures’, grandiose plans for information security as a whole, or more narrowly-focused design guidelines for specific elements of security such as intrusion detection or identity management. Such broad-brush patterns or objectives for security can help align the designs for individual systems, for instance by hooking them into enterprise-wide security structures, functions or systems where appropriate. They can also constrain design choices, implying the need for compromises and management decisions especially if specific security requirements cannot be adequately satisfied by the enterprise directives, in which case it may perhaps be appropriate to update, modify or extend the enterprise architecture accordingly.

Information security policies, standards, laws, regulations and conventions

Even if there is no overarching enterprise security architecture with which to comply, organizations typically have policies, standards and guidelines that cover certain aspects of information security. New systems and changes to existing systems need to comply, as far as possible, with these requirements otherwise exceptions are required that will need to be considered and explicitly approved by management. Similarly, requirements in laws and regulations may have to be factored into the design although ideally these should have been identified explicitly in the requirements phase.

Reviewing and approving the security design

The completed security design should be reviewed against the documented requirements, using best practice standards such as ISO/IEC 27002 to ensure that all the information security risks have been properly considered. An independent security design review by someone with information security experience and/or qualifications such as CISSP may prove beneficial in the long run. If the organization lacks the skills, this would be a suitable job for a qualified consultant or IT auditor. Management should approve the final output prior to the development phase.

For more information

Contact the Information Security Manager about security architecture and design or visit the department’s intranet website. Other notelets in this series describe the upstream and downstream processes that respectively feed into, and follow on from, the design phase.

NOTICEBORED

Security Awareness Notelet

Selecting and procuring secure IT systems

Introduction

Bespoke system development is only strictly necessary in specialist areas and/or where competitive advantage cannot be obtained from commercial-off-the-shelf (COTS) products. Frequently, therefore, “system development projects” are in fact mostly about selecting and procuring suitable COTS products, then configuring and if necessary customizing them to suit the specific organizational requirements. The advice in this notelet is intended to help with the process of selecting and procuring IT systems that meet the organization’s security requirements.

Selection criteria

Considering *all* the requirements

Although this notelet concerns information security, in practice information security requirements never comprise the entire specification, in other words they supplement or are supplemented by other non-security requirements. Potential solutions may thus be assessed on the basis of their fit with the security requirements and other requirements, either separately or together. The final decision, though, is bound to take all factors into account so it makes sense for the whole team to develop a comprehensive assessment, slotting security requirements in with the other criteria.



Weighting the requirements

The example spreadsheet below shows a simple way to rank and weight the information security requirements on a percentage scale, and then score the products against each criterion on an arbitrary scale (0=*no fit* to 10=*perfect fit* in this example). The raw scores in this fictitious example would have put product A in the lead but, having multiplied the raw scores by the weightings, product B wins the contest. A practical application of this process would normally include other non-security criteria, more details expanding on each of the requirements (perhaps in a separate document) and of course scoring columns for all the products being compared.

Requirement / criterion	Weighting	Score			
		Product A		Product B	
		Raw	Weighted	Raw	Weighted
Confidentiality protection	35%	8	2.8	7	2.45
Legal compliance	30%	3	1	7	2.1
User administration functions	15%	5	0.75	6	0.9
System administration functions	10%	7	0.7	2	0.2
Supportability and price	10%	9	0.9	8	0.8
TOTAL	100%	32	6.15	30	6.45

Using independent/formal security assessment standards

The high cost of assessing and procuring secure systems, particularly in the defense industry, has led to the development of Common Criteria (CC – ISO 15408) with pre-defined Evaluation Assurance Levels (EAL) – formal standards for security against which vendors may submit their products for independent testing. The testing process is thorough and expensive for the vendors but the resulting assessment, effectively a quality mark, is anticipated to lead to more sales. Smaller vendors that struggle to invest in certification may be disadvantaged but in theory at least the procurement process is simpler for buyers, provided their requirements can be matched to an appropriate EAL. Even if they have unusual requirements, CC can be used to pre-qualify bidders.

The IT system procurement process

Competitive tendering vs. sole sourcing

Procurement professionals have nearly as many Three Letter Acronyms (TLAs) as IT people e.g. Invitation To Treat (ITT), Request For Proposal (RFP), Request for Quotation (RFQ) etc. Individual organizations differ in how they interpret the TLA's, therefore is definitely advisable to consult with Procurement to avoid using the wrong term, especially as there may be legal implications e.g. proposed purchases over a certain value may *have* to be announced formally in regional or industry media, giving a broader range of bidders the option to bid. Sole sourcing, contracting with a specific vendor, may sometimes be appropriate for example where the market is small, when a particular product is known to be ideal match to the requirements, or where other considerations mean a competitive tender is not practicable e.g. due to lack of time to complete the full tendering and assessment process. Don't forget to incorporate confidentiality undertakings into the documentation if the requirements are sensitive.

Assessing proposals and bids

The weighted scoring noted above is seldom as clear cut as it might appear. There are often subjective considerations to the scores and sometimes significant disagreements between individuals assessing the bids. A roundtable meeting where people have the opportunity to report and discuss their findings will usually allow a compromise to be met, if not further assessment may be needed or an arbitrator may be called on to help reach a decision. Every effort should be made to avoid 'tweaking' the weightings after the products are scored as this discredits the process.

Negotiating a mutually acceptable deal

Security requirements are sometimes used as bargaining chips in the sales negotiation process. If, say, price becomes a sticking point, it may be possible to negotiate better product support and faster turnaround on security patches. At the end of the day, though, the deal must be equitable or the 'winner' ultimately risks forcing the 'loser' out of business.

Closing the deal

It is occasionally necessary to incorporate clauses relating to information security into procurement contracts, such as the right to gain access to the vendor's information security test reports or even to conduct independent security testing or audits of a system being developed or customized externally. Liabilities and penalties for inherent security vulnerabilities would be nice to have in theory – good luck if you intend to take this approach!

For more information

Contact the Information Security Manager or Procurement for advice and assistance to select and procure secure IT systems.

NOTICEBORED

Security Awareness Notelet

Securing development and test systems

Introduction

Unless an organization has already suffered a serious security incident involving its development and test systems such as theft of source code or release of virus-infected programs, it is rare to find information security controls in these environments. The approach outlined in this notelet treats dev and test systems as if they are valuable information assets worthy of protection – hardly a revolutionary concept yet strangely alien to some organizations.

Assessing the risks associated with development and testing

Information security controls for the systems on which applications are developed, tested and supported can and indeed should be specified, designed and implemented on the basis of a risk assessment. Their security requirements can be derived and documented just like those on production systems:

- **Confidentiality:** if your project is writing software for commercial release or proprietary software that is anticipated to give your organization competitive advantage, think about the implications if specification and design documentation, source code or pre-release object code were to be stolen or distributed without authorization. Even if the bulk of the software is not sensitive, be careful with parts relating to security for the finished application such as encryption routines, security administration functions and logging and alerting;
- **Integrity:** dev and test systems should be protected against viruses, worms and other malware in the same manner as all other computer systems. Furthermore, how can you be sure of the quality and integrity of application code if your development and test systems are subject to uncontrolled changes? If there is a realistic threat that someone might install a Trojan or backdoor in the system, you need to control the development environment including the operating system, compilers and source code. By the way, software professionals are perhaps the most capable hackers of all, with daily hands-on access to the systems ...;
- **Availability:** the performance and capacity of dev and test systems can be limiting factors in getting new code ready for production under normal circumstances, let alone following a disaster. In the case of development projects with fixed delivery schedules such as those delivering systems for legal or regulatory compliance or where Marketing have publicly announced the launch date well in advance (!), availability can be a critical issue. Resilience and Disaster Recovery arrangements should be considered.

Controlling access to development and test systems

Access controls start with physical access to the boxes and proceed through logical access controls in the underlying operating system and middleware right up to controls over development code and data, especially if production data sets are used e.g. for analysis or testing. Just as with production systems, rights and privileges on the systems should be restricted on the basis of legitimate business needs and trustworthiness of the individuals concerned, with clear accountabilities. It is seldom a good idea to let developers manage their own systems and management of test systems by the same IT Operations personnel who will manage the production systems gives them a low-risk opportunity to practice and hone their skills. Sensitive documentation (physical and electronic) should also be controlled under (virtual) lock and key.

The issue of developers 'needing' system privileges is an old chestnut. The very word "privilege" implies that enhanced access rights are somehow desirable and most developers are highly adept at justifying their 'needs' for privileges in a most convincing manner. There are strong arguments, in fact, to restrict their use of privileges for example to ensure that end users will not need privileged to use the production applications. Restricting the privileges and access rights needed to promote code into test or production environments is an important element in the next section.



Change control and configuration management

Maintaining management control over requirements specifications, designs and source code is a prerequisite to delivering high quality, reliable and supportable production systems. There really is no excuse for sloppy practices in a professional software development outfit. Numerous tools are available to help manage versions of software and documentation – the problem is not so much with the tools but rather persuading developers and testers to follow proper procedures consistently. It should be possible for someone – a manager, an auditor, a peer - to trace the entire sequence of changes to any controlled item, and ideally they should be able to figure out *why* each of the changes was necessary.

The 'promotion' or 'release' of code from development into test or into production environments is a classic process control point. The processes must be such that code cannot get into production until it has successfully completed the complete sequence of planned unit tests, system tests, user tests *etc.* and been approved for production use after the final stage of testing, sometimes known as "production acceptance testing". Failing to apply the necessary rigor to this process significantly increases the risk of inappropriate code releases containing bugs that impact production services. Even if testing is thorough, the final release step carries a degree of risk which implies the need for contingency measures just in case things go wrong – examples being pre- and post-release full system backups, enhanced support and heightened awareness by management, technologists and end users.

Performance and capacity management, resilience and DR

Development and testing activities can heavily load systems even if a good proportion of the code and documentation is written on the developers' desktops. Compiling code, running automated tests and especially stress testing, of course, all load up the servers. Disk capacity is another finite resource that may be in short supply, given the need to retain multiple versions of source code, object code, libraries, test data, documentation *etc.* It makes sense therefore to provide and maintain suitable server performance and capacity for development project teams.

As noted earlier, resilience and disaster recovery arrangements for the development and test servers should not be neglected, especially on development projects with tight timescales and fixed delivery dates. Preparing resilience and DR arrangements for the production system is a hidden benefit of this approach.

For more information

Contact the Information Security Manager for help to secure your development and test environments, visit the department's intranet website, or take a look at other notelets on topics such as operating system security, database security and logical access controls for more advice.

NOTICEBORED

Security Awareness Notelet

Secure coding practices

Introduction

Unlike human beings, computers do exactly what they're told to do, right or wrong, without complaint. This is simultaneously both a benefit and a drawback: they slavishly follow programs regardless of whether the instructions are correct or complete nonsense. They are purely mechanistic in nature. If their instructions include security rules, they will apply them consistently. If security rules are missing, they have no way of 'knowing'. Computers are a long way from true artificial intelligence, at least in any regular business setting, but if properly programmed, they do have the capacity to emulate intelligent behavior. The point of this notelet is to guide programmers towards sensible, secure coding practices, steering systems away from 'artificial stupidity'.

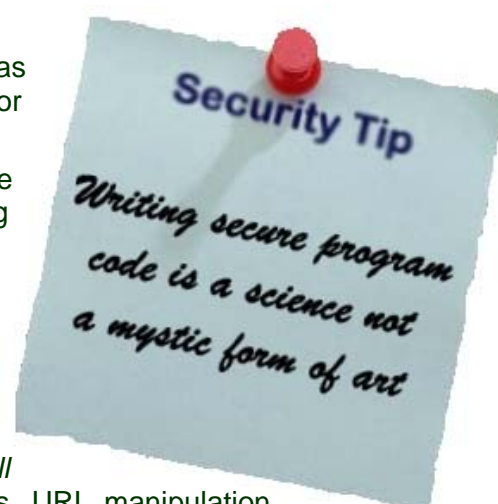
Coding to requirements and design

Well-written program specifications incorporate all manner of security requirements and, of course, professional developers do their utmost to code to those specifications and thus deliver secure systems. By implication, badly-written program specifications unfortunately do not, in which case professional developers face a quandary: code to the spec or code to best practice? Things are never quite so clear-cut in practice but, on the whole, it is better to recognize and respond to errors and omissions in the specs than to develop insecure systems. If it helps, the Information Security Manager will gladly review security specifications and designs and, if necessary, revisit the risk analysis and maybe the architectural decisions that preceded them to ensure that information security control requirements are properly considered in all systems. Banish insecure designs!

Secure coding techniques

Developing secure code is, in large measure, the same as developing high-quality bug-free code. Beware shortcuts! For examples:

- Define all constants and variables consistently, being sure to use the correct data types and, ideally, agreed naming standards;
- Pass data values to functions and subroutines through explicitly-defined parameters wherever possible;
- Assume that *all* user supplied data are potentially malicious, or inaccurate and incomplete at least. Check the validity of *all* user data, including range- and type-checks. Assume that client-side applications are *all* inherently untrustworthy. Anticipate SQL-injection attacks, URL manipulation and buffer overruns. It pays to be paranoid;
- Be especially wary of user data supplied over the Internet or other situations involving remote users that cannot be absolutely positively authenticated;
- Trap and handle even the most obscure of error conditions. If you ever find yourself writing "An undefined error has occurred", seek advice from your peers – there are probably still several more error types worth trapping. Make it a point of honor if you will;
- Document your code. Document your code. Document your code ...



Using library modules, middleware and operating system security functions

This one is easy: if you find yourself writing a security function or module that is *bound* to have been written before, search for a pre-written routine to do what you are doing, find something good and, er, take the rest of the afternoon off. If it already exists just reuse it or, at the very least, use it as a starting point. If it does not exist, keep searching and researching. If your code is genuinely novel, then you have a tremendous opportunity to contribute something truly worthwhile to the profession – you get to write the pre-written routine that others will seek out and use, even if only within the organization. This great honor carries a significant responsibility, however: you need to write code that is beyond reproach, for example having tightly defined parameters and good quality documentation. Remember that those around you are the hardest people to please. They will spot the shortcuts and logical flaws ... and quickly, if you are lucky.

By the way, do not lock yourself totally into the particular narrow environment in which you find yourself working right now. Applications can and indeed should use middleware and operating system functions where possible. BIOS calls are cool. APIs were written to be used.

Learning from the testers

Peer reviews, unit tests, system tests, security tests, stress tests – so many learning opportunities! Seriously though, the real value of testing is having someone who lacks your tremendous knowledge, understanding and experience, try to use and perhaps break the system you have so painstakingly coded. With the best will in the world, you can only test your own work to a point. The trouble is that we all have blind spots, things we simply don't see for looking too hard. We make assumptions about the way systems are used. We take it for granted that other functions provide complete and reliable data. We place far too much faith in users and IT systems. Having Someone Else review and test Our systems is a priceless gift – a different perspective.

I bet there are things in this notelet with which you disagree, things you would have changed or added or just put differently. This notelet is the product of conscientious research, production and quality assurance processes yet it is imperfect. Your feedback could make it better.

Experienced security testers have the knack of finding situations that consistently expose glaring security bugs. Sure, it smarts to admit that the system you have so conscientiously and painstakingly coded is perhaps not quite right but spare a thought for the testers. They have done something that seemed perfectly legitimate to them. Simply by using the system, they have revealed unexpected or unwelcome behavior. Whilst it's natural for the designers and developers to get all defensive, the real point is that there is a mismatch between actuality and desire. Every bug report is an opportunity not only to improve the code but to extend that ever-expanding list of mistakes-to-avoid-in-future and conditions-to-check-for.

In case the message still hasn't quite hit home, consider this: bugs identified prior to release are tens or hundreds of times cheaper to resolve than those that materialize in Production. Bugs that are stifled before they even reach unit test stage are practically free. Bugs that escape the process can be career- and perhaps life-threatening.

For more information

In addition to other notelets in this series covering operating system security, database security, encryption, logical access controls *etc.*, feel free to contact the Information Security Manager or discuss secure coding practices with your peers and managers. There are numerous books and quite a few courses on this topic ...

NOTICEBORED

Security Awareness Notelet

Security testing

Introduction

The previous notelet in this series discussed the value of testing from the developer's perspective. This one takes the testers' point of view, with a focus on security testing.

The purpose of (security) testing

Some might claim that security testing is meant to demonstrate that a system meets its specifications and is secure. Others would say, conversely, that it is intended to identify security vulnerabilities, proving that a system is insecure. We contend that the true purpose of security testing is to simulate *both* legitimate *and* illegitimate usage, identify conditions that evoke bugs, and characterize those conditions in such a way that they can be repeatedly demonstrated until solved. Finding lots of bugs is not really the aim – the designers and/or developers need to be convinced that (a) identified security issues are genuinely the result of bugs, and (b) they are unacceptable flaws in the system that need to be fixed before the system goes into production in order to avoid significant security incidents and impacts on the organization.

Planning and resourcing security tests

In any real-world situation, there are literally an infinite number of possible tests that could be performed on a system but a finite time in which to perform testing. The key to effective test planning, then, is to prioritize. In relation to security tests, risk forms a rational basis on which to select the tests to perform first, in other words pick off high-risk aspects or situations that *should* concern the business users.

Allocating suitable resources to security testing is essentially the same as for other types of tests. If information security is a key success factor for your system, put experienced security testers on the job and give them the time to plan and conduct meaningful tests. Allow for a mix of structured and unstructured “guerilla” testing, giving testers the latitude to ‘follow their noses’ and explore potentially dubious aspects of the system for bugs.

Functional security testing

Security functionality requires functional testing just like any other system functions – this includes functions for data entry validation, security administration, logging/alerting, backup/recovery, Disaster Recovery, fraud prevention *etc.* The corresponding functional specifications should explain how these functions were anticipated to work and therefore suggest high level test plans. Users familiar with these functions on other systems will probably approach testing quite differently to someone with no prior experience, although it can be argued that both approaches have value.

Technical security testing

Technical testers approach security testing with knowledge of the internal design of the system. They create test conditions specifically to exercise parts of the program most likely to contain vulnerabilities (*i.e.* security bugs) such as incomplete data entry validation, race conditions, error handling and interfaces in general. They may use testing tools to automate certain tests.

Performance, capacity and stress testing

The way systems handle being pushed beyond their design limits can be quite revealing. They often exhibit unexpected behaviors when heavily loaded or pushed almost to the point of collapse. Little timing glitches can open up into significant opportunities for hackers to exploit. Checking routines may be cut short or bypassed and logging functions may fail completely if there is no disk space. On top of that, built-in capabilities to load-share, rate-limit/throttle transactions and, in the end, gracefully withdraw service all need to be tested. You really don't want to find out that an overloaded database corrupts its data tables in production – believe me.



Penetration testing

Naïve people sometimes think of 'pen testing' as synonymous with security testing, whereas it is just one type of security test. Pen testers try to break into your application from the network, system and/or user interfaces, using the kinds of techniques and tools favored by real world hackers. Pen testers often succeed, given enough time and access. You can learn a lot from pen test reports, even more from watching them go about their work and talking to them about the issues they uncovered. Be sure to engage competent and trustworthy pen testers as they will gain sensitive inside knowledge about your systems and networks and their security vulnerabilities. "Keep your friends close, your enemies closer".

Testing security documentation *etc.*

Don't forget to test or review system security configuration and operations guides, installation scripts, training manuals *etc.* too. Review the manual control steps within procedures e.g. instructions to users on how to check exception reports, logs *etc.* regularly for security issues and escalate any they find.

Production Acceptance Testing

The final stage of testing before implementation is doubly important. It is the last chance to confirm that security and other issues raised in previous tests have been fully resolved or at least partially mitigated in some fashion (workarounds). It is also the point at which Operations decide whether they are prepared to accept responsibility for managing the system as part of the production IT infrastructure, and when the nominal system owner/s accept accountability for system security. Once the system is accepted, there is a formal handover from the development and testing teams

Providing useful bug reports

To finish, a quick note relating to the purpose of testing: remember that you are aiming to provide information that will help demonstrate security issues sufficiently well to enable someone else to reproduce, assess and hopefully fix them. The more explicit you are, the easier their job.

For more information

Other security notelets in the series hint at numerous information security aspects worth testing. Information Security Management can usually provide guidance around test planning and may be able to participate in hands-on testing, given sufficient prior notice.

NOTICEBORED

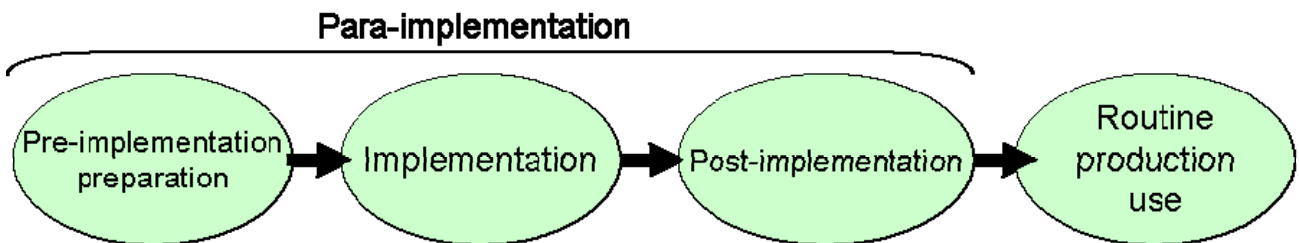
Security Awareness Notelet

Security implementation

Introduction

In relation to IT systems, the term 'implementation' is commonplace but seldom defined. Here, we look at the range of activities associated with putting a developed and tested application system into production, focusing naturally enough in this security awareness notelet on the information security aspects. We'll start with a simple process model showing the key parts of this phase.

Implementation process



Pre-op preparation	Operating theatre	Intensive care	Recovery room
Confirm sign-off; full backup to secure the previous configuration in case of serious problems ahead	Object code migrated into Production under change control and executed/launched; technical installation testing; initial configuration; functional testing; first full backup (clean install); release to users	"Heightened awareness" support with Help Desk and technical support team on full alert; 'life support' monitoring in place	Tuning; transition to routine monitoring and support; becomes business as usual

Pre-implementation preparatory activities (pre-op)

The introduction mentioned putting a *tested* system into production. Prior to IT Operations installing the software, they need to confirm that the software has been "signed-off" (formally approved) as having successfully completed production acceptance testing in the previous phase. The pass criteria should have included preparation, delivery and testing of installation instructions and scripts, plus ideally post-installation verification and configuration scripts that help automate and speed the implementation process.

If the implementation is a change or upgrade of an exiting system, it is good practice to take a backup of the original system immediately prior to installing a major change as a last resort in case the change needs to be backed-out. This normally implies blocking users from the system while a full system backup is taken, either using disk-to-disk-to-tape copies or simply un-mounting one member of a fully synchronized volume shadow/RAID disk set to save time.



Promotion of code into the production environment (the op)

In this phase, the object/executable (not source) code is migrated into the Production system under authority of a change control permit, and then executed. Initial configuration of the new system may occur in parallel with or following technical installation testing to confirm that the installation was complete – see below regarding the configuration of security elements. Basic functional testing may be undertaken to check whether the system will operate as expected for users, perhaps as little as confirming that there is a login screen, sometimes much more. The first full backup of the new system should be taken at about this point to provide a “clean install” copy – this may be needed much later if the system is corrupted e.g. by a virus, and needs to be re-installed. Finally, the new system is released to users and the real fun begins.

Security configuration

Access rights and userIDs

In the headlong dash towards production acceptance testing and implementation, project teams often “forget” (neglect) to arrange new userIDs and configuration of or changes to access rights and privileges. Whilst some may be included within installation scripts, Security Administration is normally tasked with a bunch of configuration activities immediately before releasing the system for production use. Please give them sufficient advance warning to reduce the number of ulcers all round. Pre-defining typical user rôles helps cut down on the antacids.

Logging and alerting, backups and other security-related parameters

The configuration of security parameters includes logs, audit trails, alerts/alarms and so forth. Ideally these are also covered by the system installation scripts but if IT Ops are expected to press the buttons, it helps to tell them what buttons to press - they are not entirely clairvoyant.

Para-implementation support activities

Systems support should be placed on high alert during the change process to deal with any unanticipated technical issues associated with the implementation. Despite all that pre-release testing, this is *the* most critical period for most systems, the time when things are most likely to go wrong. Security problems such as mis-configuration of access rights for example can prevent programs from loading, executing and/or interacting correctly with the rest of the system, the network, IT Ops or the users. Business critical systems deserve intensive support.

Contingency arrangements should be explicitly planned and ideally tested for significant changes to business-critical systems. An option worth considering is to take offline, upgrade and test one member of a redundant pair of fully resilient systems: then, when it is ready to release, take the other member offline during the release and retain it as an emergency fallback. Similar processes may be feasible with Disaster Recovery systems.

Transition to Business As Usual

Soon we hope, new systems bed-in and the special technical support arrangements can be stood-down. Now is a good time to review the go-live process and learn any lessons for next time.

For more information

To avoid changes being rejected outright, it pays to speak to the Information Security Manager and IT Operations for advice on the security aspects of implementing new systems or significant changes to existing systems. To avoid delays, contact Security Admin in advance regarding configuration of new userIDs and access rights.

NOTICEBORED

Security Awareness Notelet

Security maintenance and support

Introduction

This notelet essentially addresses the security issues relating to mature IT systems. Long after systems are first implemented, they become 'part of the furniture'. Stable, reliable systems tend to be forgotten during their middle years, familiarity breeding contempt – they are simply taken for granted. Unless someone maintains an interest in the information security controls, risks can build up until incidents occur. Professional security maintenance and support activities however can significantly extend those middle years, milking more value from IT and avoiding serious incidents.

Maintaining information security

Through a form of entropy, most things gradually decay unless they are actively tended and maintained: information security is no different. Here are some common examples:



- As users come and go, dormant userIDs accumulate and access rights/user rôles become mixed-up and confused through gradual evolutionary changes in the organization, compounded by 'downsizing' and similar dramatic restructures;
- Some users gain excessive rights and privileges that are never properly revoked;
- Logging and alerting systems become somewhat irrelevant and neglected, often because they are not suitably updated when functional or system changes are made;
- Technological advances render the original technical security controls obsolete, and likewise for process changes and manual/procedural controls;
- Some changes on production systems may not be fully replicated to Disaster Recovery systems. Others including business process changes may unknowingly compromise security controls;
- Numerous vulnerabilities are exposed and hopefully (!) patched, leading to a mish-mash of fixes to fixes, workarounds and mitigating activities. Years of partially documented changes accumulate. The technical support people who know what has occurred become older and eventually leave or retire. In the end, there is no-one left who truly understands the system.

Regular system maintenance activities should take account of these and similar issues, the idea being to avoid system security getting too far out of step with the ever-changing technical and business environments. System security documentation is one of the most neglected areas.

Another obvious way to halt the decline towards insecurity is to take stock every so often. A major system upgrade, for instance, presents a golden opportunity to review and update the original system security design and documentation, perhaps even to back to basics and refresh the original risk assessment. Security checks slightly in advance of the annual audits of financial systems can forestall problems with Sarbanes-Oxley Act compliance etc. Computer audits often reveal security problems that are glaringly obvious when singled out by a fresh pair of eyes, yet were ignored or unrecognized by those who work with the systems every day.

Supporting information security

One advantage of middle-aged IT systems over their younger counterparts is that the organization has built up experience of actual security incidents. While nobody especially likes to rake over the coals, reviewing incident reports can reveal patterns indicating inherent system or process issues and can provide the commercial justification for security improvements. Insufficient data accuracy may be a symptom of inadequate data integrity controls. Poor system performance and frequent service outages probably reflect a lack of focus on availability controls such as capacity, resilience and redundancy. The good news is that many of these issues may be solved given enough management support and technical attention, and can extend the useful life of systems.

Records of support calls can also provide a wealth of useful knowledge about systems security. If, for instance, a particular system causes a disproportionate number of password-reset requests, perhaps it is as well to review the password parameters and consider using certificates or other modern multi-factor authentication methods on that system. Similarly, numerous Help Desk queries relating to access rights might reflect badly defined rôles and responsibilities, and a peak load of performance-related calls may mean it is time to review system capacity and perhaps call on the database and application support teams to give the system a tune-up [by the way, it could also indicate a system compromise or a significant change in use, all of which perhaps deserve further investigation].

Given the above, it is clearly important that support staff record incidents and support calls, and that managers periodically review the data for adverse trends. Security concerns experienced by multiple support functions such as Help Desk, Data Security and IT Operations should be correlated in order to identify related or common issues.

It is not a bad idea for someone to keep a 'little black book' of incidents, issues, wish-list items and notes on each of the systems and invite input from the various support functions. Reviewing the book when system changes are being planned may identify opportunities to cover-off related problem areas at little extra cost.

A note on security metrics

Business cases used to justify investment in IT systems generally make rather bold predictions about the benefits the organization expects to offset the development costs. Security benefits can play a significant part in the benefits cases for business-critical systems or those handling sensitive or vital data. It makes sense therefore to measure and track the claimed benefits and, if this is done consistently, the benefits can be used to drive out even more value from systems. By the same token, however, it is necessary to track the maintenance and support costs too. Plotting the curves should show the negative effects of incidents and the positive value of corrective and preventive maintenance activities. All of this analysis will contribute to the quality of future investment proposals.

For more information

Contact Information Security, Risk Management or Internal Audit for help to evaluate systems security and recommend improvements. Help Desk, IT Operations, Finance and other support functions can provide information to help management make informed investment decisions.

NOTICEBORED

Security Awareness Notelet

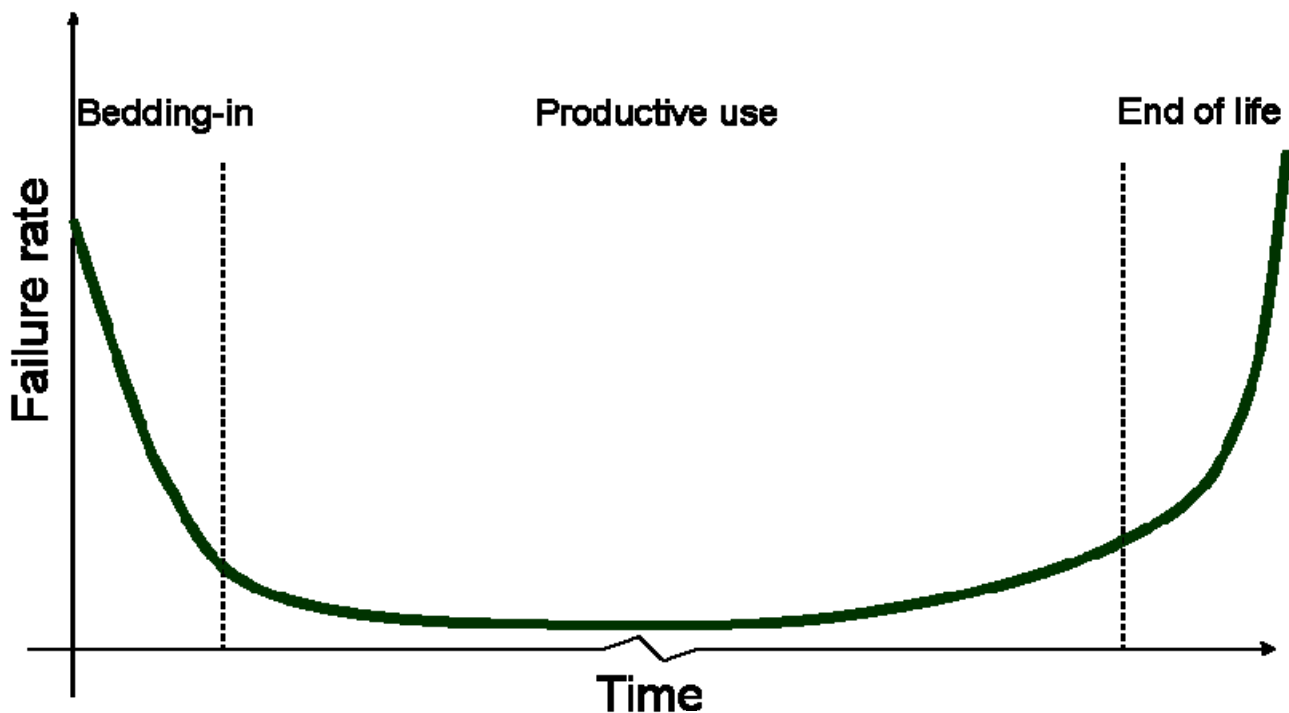
Retirement/replacement of legacy IT systems

Introduction

Following the system development lifecycle to its logical conclusion, this notelet addresses the information security issues relating to legacy IT systems nearing the ends of their lives.

Recognizing the need for change

IT systems rarely become “legacies” overnight, although it does sometimes happen e.g. when new laws or regulations come into effect and non-compliant systems are retired. Old age creeps up on us all. The ‘bathtub curve’ show below applies to hardware and software systems:



After the initial settling period when teething problems on a new system are resolved, failure rates remain low and relatively stable for the bulk of the system lifecycle but eventually start to rise inexorably as hardware literally ‘wears out’ and software support fades away. The number and severity of failures increases to the point where the system becomes very expensive to keep in service. Failures in this context include security incidents such as an accumulation of inappropriate access rights, compromise of encryption algorithms, data integrity errors *etc.* It is really up to the system owner to realize that a system is no longer economic although the users and system administrators may have to point it out, perhaps using data from the incident, problem and change management processes to support their case.

Picking-over the bones of legacy IT systems

Legacy hardware, software and processes are all candidates for recycling, to some extent. Worn out mechanical items will become junk but software and processes don’t exactly wear out.

Software modules/functions/subroutines are often stored in libraries. There may be mileage in checking over the libraries associated with legacy systems for items that can be re-used on other systems, perhaps even to speed up the development of replacement systems.

Business processes may also be worth re-using, implying the need for someone to identify things the organization wants to stop doing, continue doing and change. Business Analysts are skilled at preparing requirements specifications for new systems, especially where users and system administrators can articulate their needs eloquently. Statements like “The old system was rubbish at X” or “It was great at Y” may help tease out true requirements.

As to actually designing and building replacement IT systems, please refer to the earlier notelets.

Removing redundant IT systems from service

“Easy,” you may think. “Just switch them off!” Technically that might be true but there is usually more to it than that. To avoid disruption to the supported business processes, it is often necessary to develop and implement replacement systems *before* the legacies are shut down, maybe with a period of overlap and comparison known as parallel running. Bear this in mind when considering the bathtub curve – it pays to plan for implementing the replacement *before* the legacy system hits the steep cost escalation at its end-of-life. The lag can be substantial for large systems and slow investment decision-making processes.

Before an old system is actually powered-off and removed from the computer rack, think also about whether there are any archival requirements for both data and hardware. How will you be able to read those old 8-inch floppy diskettes in the fire safe when the system is long gone? Now may be your last opportunity to transfer data via the network or serial connections onto more modern hardware and storage media, and generic file formats such as flat ASCII.

Secure disposal of redundant IT systems and media

What should we do with redundant IT equipment, disks, tapes *etc.* if they are no longer in use? Before you throw items in the trash or drop them in the recycling bin, consider the value of the data that might still be stored on them. Old computer disks regularly turn up at auctions and secondhand sales with sensitive data completely intact. Researchers from various security companies are not alone in rifling through junk yards looking for interesting finds: hackers and reporters just love this stuff. Unless your CEO looks forward to appearing on the TV news explaining how confidential personal data ended up on a PC sold to an unsuspecting member of the public, either use secure logical data disposal methods or physically destroy the media, perhaps both. Comparing the cost of professional disposal against the adverse publicity, let alone prosecution and fines, should be an easy matter in most cases.



For more information

Information Security Management can help review legacy IT systems from the security point of view, identify the plus points and ‘opportunities for improvement’ and work with you to specify replacement systems.

NOTICEBORED

Security Awareness Notelet

Security aspects of 'End User Computing'

Introduction

Whereas the other security awareness notelets in this series are focused on mainstream professional IT systems development projects, this one offers security advice to 'end users' developing programs on their desktop computers, laptops and Personal Digital Assistants.

"I only use Excel – does that make me a developer?"

The answer is probably yes, that is unless you purely enter data into pre-written spreadsheets and run standard reports. The people who created those spreadsheets are developers, even if they did little more than enter a few simple formulae and a bit of formatting. The same thing applies, by the way, to people who write macros or scripts in word processing and other desktop applications, or who develop databases in MS Access *etc.* – they are all forms of programming.

The point is that many employees are certainly capable of developing software using today's desktop PCs, laptops and even PDAs, and in fact many do. Software development is no longer something confined to the ranks of IT geeks in white lab coats, lovingly tending to the mainframe.

Typical information security considerations

Some untrained/amateur programmers are every bit as highly motivated and skilled as their professional colleagues who have been through IT college and have all the right qualifications after their names. Some of them work in the IT department, others are acknowledged as "power users" throughout the organization. A few do it at home in their spare time. Most general IT users however are simply not in the same league. They may have learnt a few programming tricks by trial and error, maybe even read a book or two, but they do not have the same comprehensive grounding as trained and experienced IT professionals who make a career of IT.

Information security is a tough subject for many geeks and is seldom a core topic on college IT courses. Few amateur developers fully appreciate the importance of information security controls in the software they develop – generally speaking, information security doesn't even get a second thought. This is naturally a sweeping generalization and, sure, there are exceptions. The very fact that you are interested enough in the subject to be reading this notelet puts you in a special category for a start!

The complete absence of structured software development processes is a common symptom of amateur programming. There are no documented requirements or designs, no flow-charts or process diagrams. Version control is primitive at best. Desktop developers offer precious little support for software released to their colleagues, often because they are too busy with the 'day job' or they lose interest in systems they have developed. These symptoms all militate against creating secure systems. With no security risk analysis or specifications up front, key security controls are quite likely to have been completely neglected. Classic examples are access controls, routine backups and audit trails. Data entry validation, a vital control for data integrity, is a rarity in home-grown spreadsheets *etc.*

With no written specifications as such, testing an application depends entirely on the knowledge and diligence of the developer and, perhaps, a few tame users willing to 'try out the program'. Again, unless someone has an interest in this area, thorough security testing is rather unlikely to take place.

“OK then, what *should* I be doing?”

Think about the information security requirements and design the controls

The sorts of information security questions to ask about a new system are:

- **Confidentiality:** are any of the data sensitive e.g. personal information or commercial secrets? Is the program itself commercially sensitive? If it fell into the wrong hands, would the organization suffer? Are there any legal requirements for privacy?
- **Integrity:** what might happen if incomplete data are entered, or if the data or calculations are wrong? If someone entered a nonsense value, could it corrupt the whole system?
- **Availability:** who will need to use this system? Will it be ready in time? What would be the consequences if the system went slow, ran out of space or stopped working altogether?

If there are significant security requirements for the system, you need to take more care over the design. Confidential data may need to be encrypted to control access. Critically important values should be checked as they are entered/calculated to pick up typing errors, out-of-range values *etc.* Please seek help in these situations - if are unsure what to do about security, don't just struggle along by yourself but call the IT Help Desk or speak to your manager.

Building-in security

Developing security controls should, in theory at least, flow naturally from the specifications phase. They should be built-in, typically incorporating the following types of control:

- **Preventive controls:** designed to stop bad things happening e.g. preventing unauthorized access to data, preventing incomplete or out-of-range data entry;
- **Detective controls:** raise warning messages or alerts if bad things happen e.g. “Telephone numbers should not contain letters!”;
- **Corrective controls:** if other controls fail, these help limit the damage when bad things happen e.g. data backup and recovery processes.

Securing a system is not purely a matter of incorporating technical controls into the code. Procedures for those using and managing the system are just as important. Don't forget to write suitable procedural controls into the process descriptions, user guides *etc.*

Testing security

Take care to test any critical controls carefully, referring back to the design specifications if necessary. Test things such as range limits, type checks and control totals e.g. the sums of column and row totals on a simple financial spreadsheet should normally work out to be the same value. If the system has access controls, see whether users are in fact prevented from accessing unauthorized areas. Don't forget to check that procedural controls are properly described in the accompanying user or management documentation.



For more information

The IT Help Desk is used to fielding calls about IT development issues from end users and can assist with basic information security aspects. They will assign calls to other IT support staff including Information Security Management where appropriate.

You are welcome to approach Information Security directly for advice, especially if you are concerned about security for a desktop development that supports a critical business process, feeding data into the financial systems for example. Speak to an expert before the auditors come a-knocking at your door.

About the author

This briefing pack was written by Dr Gary Hinson CISSP, CISM, CISA, MBA, [IsecT Ltd.](#)'s Chief Executive Officer. Gary is an IT governance specialist with more than two decades experience in information security, risk management and IT audit, including numerous IT development project assignments. Gary is passionate about [information security awareness](#) (more below) and the [ISO/IEC 27000-series information security management standards](#). He contributes to the continued development of the standards through JTC1/SC27, the ISO/IEC committee responsible for them, and run the [ISO27k Implementers' Forum](#).



IsecT Ltd. is an independent IT governance consultancy focused on information security and IT audit. IsecT ("security in IT") is based in New Zealand. Our corporate website is www.isect.com.

NOTICEBORED

is IsecT's innovative security awareness product. NoticeBored subscribers receive a fresh 'module' of high quality security awareness materials covering a different information security topic every month. The materials comprise seminar presentations, briefing and discussion papers, newsletters, posters, mind-maps, case studies, sample policies, white papers, screensavers, awareness tests and surveys *etc.* All the materials including this briefing pack are provided to customers in industry-standard editable file formats e.g. Rich Text Format/Word, PowerPoint, Visio & JPGs. Please visit www.NoticeBored.com for more information.



Isect Ltd.
Castle Peak, 1262 Taihape Road,
Hastings RD9, NEW ZEALAND
www.isect.com
+64 6874 3344

Copyright and disclaimer

This briefing pack is provided for security awareness purposes. It offers generic, high level guidance on a selection of commonplace information security controls for software development. Because it is generic, it cannot fully reflect every user's requirements. We are not familiar with your specific circumstances and cannot offer tailored guidance to suit your particular needs. It is not legal advice.



The PDF version of this work is copyright © 2008, [Isect Ltd.](http://www.isect.com), some rights reserved. It is licensed under the [Creative Commons Attribution-NonCommercial-Share Alike 3.0 License](http://creativecommons.org/licenses/by-nc-sa/3.0/). You are welcome to reproduce, circulate, use and create derivative works from this provided that (a) it is not sold or incorporated into a commercial product, (b) it is properly attributed to Isect Ltd., and (c) derivative works are shared under the same terms as this.

The MS Word version is governed by the NoticeBored license agreement.