

NOTICEBORED

Building information security in through awareness

Issue 80

Developing *more secure* software

January 2010

Editorial

“Development By Denial (DBD) - Everybody pretends there is a method for what’s being done, and that things are going ok, when in reality, things are a mess and the process is on the floor. The worse things get, the more people depend on their denial of what’s really happening, or their isolation in their own small part of the project, to survive.”

Scott Berkun

If computers do exactly what they’re told, then making them behave securely is simply a matter of specifying the security requirements and developing the appropriate software, isn’t it ...? Unfortunately it’s not nearly so simple in practice. “Behave securely” is rather vague for a start. What does that actually mean?

Furthermore we humans sometimes create flawed designs, write bugs into the code, test inadequately and fail to implement, configure and maintain systems securely. Is it any wonder that patch-and-pray remains a common approach to maintaining system security, despite its obvious inadequacies?

January’s awareness module examines the causes of insecure software and suggests possible solutions, particularly approaches that help avoid security vulnerabilities being introduced in the first place rather than having to be repaired later. “Build security in” has got to be better than “add security on”, surely! We don’t have all the answers, admittedly, but awareness and education should take things in the right direction at least.

Happy new year everyone. May 2010 be the year of built-in-security. ■

Gary Hinson, NoticeBored Editor

Background

Prior to the rise of the IT industry, early adopters of computing had little choice but to develop their own application software, sometimes even their own operating systems and hardware. In the fifties, sixties and seventies, software development was almost entirely the realm of a select band of white-coated computer professionals, locked away with their toys in some dimly-lit back room. As the release of more user-friendly development tools by companies such as IBM, Digital, Oracle, Apple and Microsoft, coupled with much cheaper computer hardware and the spread of PCs, led to many others taking up the challenge, so software development has become something almost anyone can do. Whether that is a good or a bad thing is not entirely obvious.

Developing *secure* software is no easy task. Despite substantial investment in its Secure Computing Initiative since 2003, Microsoft still releases security patches most months. It’s small comfort to think that the situation would probably be *even worse* without the pressure from Bill Gates to smarten up their act on security, or to discover that many of the security vulnerabilities being patched address bugs and flaws in legacy code written before the big security push. We are still patching.

Software development is something most of today’s IT professionals study at college but lots of programs in typical corporations are written by self-taught amateur programmers, many of whom don’t even appreciate that they are developers. It’s an unfortunate fact that information security risks and controls are barely covered either way. Few college IT courses pay much attention to information security and even ardent supporters of the concept of “citizen programmers” would be hard-pressed to find genuine examples of secure spreadsheets *etc.* written by unqualified employees without IT assistance, guidance or involvement.

Information security risks relating or leading to the development and implementation of insecure software

The wide range of information security risks that affect IT systems once they are implemented is amply covered by other NoticeBored newsletters and awareness materials. This section purely examines the information security risks arising from the development or acquisition and implementation of insecure software, meaning *software that does not fully satisfy the functional and technical requirements for security*.

Threats

Thinking about the development/acquisition process, it is conceivable that fraudsters or hackers working within the organization or at third party suppliers might want to insert backdoors, logic bombs or other hidden functions into software with the intention of exploiting them later ... but situations of this nature are quite rare as far as we know: they are hardly likely to flag the fact that they are deliberately compromising our systems! The possibility is real however – ‘Easter egg’ functions hidden in some commercial software demonstrate the threat albeit with benign effects, as far as we know.

The bigger threat by far is errors and omissions, in other words incompetent development leading to the release of insecure software.

“Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations. (They are also large, expensive to maintain, difficult to manage, and they pollute the environment. It is astonishing that these devices continue to be manufactured and deployed. But they are sufficiently pervasive that we must design our protocols around their limitations.)”

C. Kaufman, R. Perlman, and M. Speciner (2002) Network Security: PRIVATE Communication in a PUBLIC World. Prentice Hall, page 237

Limited budgets, tight deadlines and, most of all, insufficient appreciation by management of the

true value of information security collectively threaten the design, development and implementation of secure IT systems. Without management support, security budgets may be inadequate and the work may not necessarily be given sufficient emphasis in relation to conventional functions. This begs the question: how can we increase management support? A significant part of the answer involves security awareness e.g. an understanding of the costs of failing to take due account of security, or better still the benefits of addressing security proactively from the very earliest phase of a development project.

“Finding and fixing a software problem after delivery is often 100 times more expensive than finding and fixing it during the requirements and design phase ... This insight has been a major driver in focusing industrial software practice on thorough requirements analysis and design, on early verification and validation, and on up-front prototyping and simulation to avoid costly downstream fixes. The only thing we have changed since 1987 is to add the word "often," to reflect additional insights on the relationship. For one, the cost-escalation factor for small, noncritical software systems is more like 5:1 than 100:1, enabling such systems to be developed most efficiently in a less formal, "continuous prototype" mode -- but still with emphasis on getting things right early rather than late. Another is that the cost-escalation factor can be reduced significantly even for large critical systems via good architectural practices. These reduce the cost of most fixes by confining them to small, well-encapsulated modules.”

*[Software Defect Reduction Top-10 List](#)
by Barry Boehm and Victor Basili*

Vulnerabilities

A few vulnerabilities were mentioned already – inadequately trained and experienced developers are more likely to produce poor quality and often insecure programs. Poorly defined or inconsistently applied development processes or methods are another potential source of problems. Take a good look at your organization’s software development methods when you next get the chance: do they incorporate appropriate information security activities at every stage of the lifecycle from business case to retirement? Are they sufficiently well described and mandated by management to discourage developers from

taking potentially dangerous short-cuts? Home-grown development methods are seldom as hot on security as those that are based on sound research and significant experience in the field: mature methods really are worth the effort.

Most organizations these days rely heavily on third parties for their software. While buyers *can* specify their security requirements in detail and test products extensively prior to acceptance, still they must place trust in the competence and integrity of the suppliers of software, firmware and hardware. Buying security-approved products gives additional assurance provided buyers have confidence in the approvals process, and provided suitable products are available through this route at a reasonable price. In most cases, however, buyers have little option but to use off-the-shelf operating system, middleware and application software, development tools, system platforms *etc.*, often from a number of different vendors.

[Reflections on trusting trust](#), a classic article by Ken Thompson published in the August 1984 issue of *Communication of the ACM*, demonstrates the feasibility of deliberately compromising software using a function hidden in, say, the compiler, in such a manner that even a detailed line-by-line examination of the source code would not reveal any flaw. The point is that there are hard limits on the value of checking code and hence a lot of software security is inevitably down to blind faith.

“Priority one is aligning software development and operational processes with strategic business objectives. Sometimes technologists forget why they are doing what they are doing. Yet most software today is created to service business. Software security practices and mechanisms will succeed only to the extent that they have clear and explicit connections to the business mission.”

Gary McGraw (2006), Software security – building security in. Addison Wesley, page 241

Impacts

The impacts of failed software security controls are many and varied, of course. They can be loosely categorized as failures of confidentiality, integrity and/or availability.

In a more profound sense, the possibility of software security failures can have an adverse

effect on the trust placed in computer systems by individuals and organizations. To put that the other way around, an organization’s demonstrable ability to develop secure software can increase the degree of trust placed in the systems and can therefore facilitate greater use of IT. NASA managers, for instance, would not have the confidence to fly space missions if they did not believe the software systems designing, controlling and monitoring the flights were secure. Their confidence stems partly from the quality assurance processes, training and experience of their software architects, developers, testers and implementers. Seen in this way, strong software security is truly a business enabler, in the same way that high quality software is A Good Thing.

Professional software development methods are not only relevant for professional software developers. Integrating information security with the software development lifecycle has benefits for anyone involved in specifying, designing, developing, testing, using and maintaining software. This includes non-IT employees who develop spreadsheets, macros and custom reports. They too are software developers and they too should satisfy information security requirements for whatever they produce.

Limited integration of information security into the development process inevitably leads to the release of insecure systems.

Starting with users who are unable or unwilling to consider, determine and/or document and clarify their security requirements or participate in risk assessment workshops, business analysts have a tough time to prepare comprehensive security requirements specifications. If the security requirements are inadequately defined, necessary security controls are likely to be neglected during the development phase and security testing will be difficult. Even if the risk assessment and requirements definition activities are strong, there is still a chance that security will be under-represented in the system designs (especially if there are particular functional requirements that dominate everyone’s attention). They may not be adequately coded unless the developers are competent and have the necessary time to focus on security – the dearth of information security knowledge, competence and interest among IT professionals is therefore another vulnerability.

The potential issues don't stop there either. Some organizations place undue emphasis on penetration testing *in lieu* of comprehensive security tests. Pen-testing is not bad at finding the kinds of vulnerabilities that hackers might exploit but not so strong when it comes to reviewing security measures other than access controls. The day-to-day realities of typing errors, performance problems and awkward security procedures that encourage users and administrators to take short cuts remain the primary cause of security incidents, and are unlikely to be exercised (or is that exorcised?) by pen-tests.

Software testing is almost always performed under extreme pressure because, generally speaking, there is no contingency left in development project plans when release deadlines loom. Security testers have to compete hard for their share of the rapidly diminishing cake.

System implementation is yet another problem area. Systems are not always delivered to IT Operations in the form of fully scripted installation kits with post installation configuration and verification routines. It is not uncommon for Security Administrators to be sent hastily written notes with lists of users who 'need their userIDs setting up immediately' on the eve of the planned go-live date, yet there has been no effort to define and configure the corresponding system rôles or access rights.

Incomplete change management processes and divisions of responsibility can allow developers unrestricted access directly into Production, leading to incidents through unplanned and unauthorized changes. Maintenance and support activities may bypass change controls and, if the security implications are not carefully monitored, can lead to a gradual reduction in system security. Delays in releasing and applying security patches are a particular concern on legacy systems.

When IT systems are worn out and are finally decommissioned, how many organizations take due care of the residual value of the data, such as taking data extracts for archival purposes and securely disposing of the media? Nobody pays much attention to legacy systems – the name itself gives a clue about most managers' attitudes towards them. They are seen as a burden.

Conclusion

“It is conceivable that secure software might be developed using insecure processes but frankly I doubt it. When it comes to developing business-, mission- or safety-critical software, I personally wouldn't be comfortable with anything other than the rigorous application of a structured development method. That comment is borne of years of advising and auditing development projects, and an active interest in the quality assurance aspects of software development methods. Be very thankful that your vehicle's anti-skid braking system wasn't produced by the incompetent application of an extreme programming technique to an unrealistic deadline!”

Gary Hinson, NoticeBored newsletter #30, Nov 2005

Key to the development of more secure software is maturity of the software development processes and methods in use, particularly the integration of quality assurance within software development. Without this, the chances of identifying and reliably satisfying all manner of functional and technical requirements (including those for information security) are severely constrained. This month's batch of NoticeBored security awareness materials provide information and encouragement for your staff, managers and IT professionals but it will take your support and a concerted effort to mature your organization's approach to software development. Step 1 is to recognize that it needs to be done. What do you think? ■















“Good design can reduce the chance of mistakes, lapses, and slips. Designers should develop clear communications that convey specific instructions so as to reduce the chance that users will make mistakes while completing security-critical tasks. They should minimize the number of steps necessary to complete the task and, whenever possible, provide cues to guide users through the sequence of steps and prevent lapses. To prevent slips, designers should locate the necessary controls where they are accessible and arrange and label them so that they will not be mistaken for one another.”

Lorrie Faith Cranor (2008), “A Framework for Reasoning About the Human in the Loop”, Cylab, Carnegie Mellon




January's awareness module


This month's NoticeBored module contains the following items:


Awareness materials for all employees

1. **Seminar: developing secure software**  12 sl
Focuses on 'citizen programming' with guidance for staff around securing spreadsheets *etc.*
2. **Poster images: secure software**  x6 JPGs
Thought-provoking images. What *do* we mean?
3. **Screensavers: secure software dev't**  x4
Play the slides and posters over and over ...
4. **Awareness stickers: secure software**  x2
Sticky reminders – but not tacky, oh no.
5. **Bookmarks: secure software dev't**  x4
Pithy quotes plus memory joggers on this topic.
6. **Guideline: citizen programming**  1 page
Makes information security an explicit concern for desktop developers.
7. **Case studies: secure software**  2x2 p. ea
Realistic examples stimulate discussion.
8. **Top tips: secure software**  1 page
8 top tips – just the key points for employees.
9. **Take home message: secure software**  1 p.
10. **Crossword puzzle: secure software**  2 p.
11. **Awareness survey: secure software**  1 p.
12. **Awareness test: secure software dev't**  1 p
13. **Glossary: secure software dev't**  6 pages
14. **Hyperlinks: secure software dev't**  online
Useful web resources for further study.


Awareness materials for managers


15. **Mind-maps: secure software**  4 Visio diags
Gets your brain in gear.
16. **Board agenda: secure s/w dev't**  1 page
Does the Board approve of how we do it?
17. **Mgmt seminar: secure s/w dev't**  11 slides
Highlights for managers to discuss, with the speaker notes providing additional guidance.

18. **Model policy: secure software dev't**  3 p.
2 axioms and 8 policy statements this month.


19. **Elevator pitch: secure s/w dev't**  1 page
More short than sweet. No-bull guidance.


20. **Exec briefing: secure s/w dev't**  1 page
One whole page!


21. **Mgmt briefing: secure s/w dev't**  4 pages
Guidance for software development projects.


22. **Metrics for secure s/w dev't**  3½ pages
Suggestions on how to measure and improve development for more secure software.


Awareness materials for IT pro's


23. **This newsletter: secure s/w dev't**  6 pages
Risk analysis and introduction to the topic.


24. **Awareness activities for January**  6 pages
Answers the rhetorical question: "What do we do with all this awareness material?"


25. **Tech seminar: secure s/w dev't**  14 slides
For project team members and their managers.


26. **Dev't project security briefing pack**  68 p!
Comprehensive but accessible suite of security advisories for software development project managers and team members. Covers typical system security controls plus information security aspects of the development process.

27. **Tech brief: secure the dev't process**  6 p.
Protect the valuable information assets created and used by software development projects.

28. **Tech brief: secure s/w dev't methods**  5 p.
Different methods have different security values.

29. **Tech brief: secure software dev't**  6 pages
Phase-wise guide to security in development.

30. **Development proj. security checklist**  6 p.
A checklist of security activities and deliverables expected of most software development projects, provided as a prompt to action.

31. **Controls checklist: secure s/w dev't**  5 p.
IT auditors use checklists like this to compare the actual process and security controls against their anticipations. Get ahead of the game by self-checking and improving *before* an audit.

NoticeBored topic diary

These are the next three awareness topics we'll be covering. Please remember that NoticeBored is explicitly designed to support continuous, rolling awareness programs. A month is long enough to cover each topic in some depth, but hopefully short enough not to bore the pants off everyone who doesn't live and breathe information security like we do. ☺

February – cryptography, the science of secret writing

Cryptography is an intensely mathematical science and mostly far too deep for us, let alone the mere mortals who are in the sights of most security awareness programs. So, we'll be taking a gentle peek under the covers with as little mathematics as we can safely get away with – more Boy's Own secret decoder ring than Rindael and S-boxes, thanks. What do staff, managers and IT pro's *really* need to know about crypto? Find out with us next month.

March – malware

What's hot in the fields of malware development, exploitation and research? We're looking forward to March's awareness module with an update on the state of the art from a renowned authority on malware.

April – identity theft, an authentication failure

With authentication as the underlying information security issue, we will be exploring that kind of fraud known as identity theft, perpetrated using dark-side techniques such as phishing, spyware and Trojans. Since the security defenses involve staff, management and IT working together, this is a broad-based awareness topic. ■

NoticeBored & IsecT news

Earlier this year, we were given the opportunity to review and comment on the draft of a forthcoming information security management book for the latest version 3 of ITIL (IT Infrastructure Library). The authors and publisher graciously adopted our long list of suggested changes, including numerous updates regarding the ISO/IEC 27000-series standards and (surprise surprise!) additional emphasis on the value of raising employees' awareness of their information security obligations, thus supplementing and leveraging various technical security controls. ITIL and the derivative standard ISO/IEC 20000 lay out a comprehensive structure for managing the provision of IT services. When released next year, the new ITIL security management book should help integrate good information security practices with good IT management practices.

As another year draws to a close, we're reflecting on what has changed in information security terms since 2008. Aside from the increasing interest in the security issues arising from cloud computing and virtualization, information security has been largely overshadowed this year by the global economic downturn resulting from the banking crisis. As to what effect this may have had on information security, time will tell. We suspect desperation may have led some former employees to exploit their former employers but *so far* they mostly seem to have slipped under the radar. ■



Newsletter published by:

[IsecT Ltd.](#)
Castle Peak
1262 Taihape Road
RD9 Hastings 4179
New Zealand

Tel: +64 6874 3344

Copyright and disclaimer

All NoticeBored materials including this newsletter are protected by international copyright law. For more information, please contact the copyright holder, IsecT Limited (visit www.isect.com or see our contact details above) or speak to a smart lawyer. You are encouraged to circulate the Adobe Acrobat version of this newsletter to anyone *provided* that it remains unchanged and intact (including this copyright notice), is not embedded in any other product or service and is provided free of charge. The information in this newsletter is provided free, for information only and 'as is'. Whilst believed to be at least partially correct, it is in no way comprehensive and parts may not be entirely true. It is provided for interest only and is not intended to be relied upon as formal advice, particularly absent the remaining NoticeBored awareness materials which provide supporting details. It is neither legal advice nor a horoscope. Seek your own legal advice from a qualified legal practitioner, and check your own crystal balls. No liability is accepted for any errors or for any losses that may be incurred if any such information is relied upon. Like the global economy, the value of this newsletter may go down as well as up.