Information Security Now - 22

John Mitchell

Software integrity is a security issue and as such should fall within the CIAC domains of confidentiality, Integrity availability and compliance. Although we are primarily interested in the capability of software to do exactly and only what it is specified to do, we ensure its integrity by having excellent confidentiality, availability and compliance processes in place – or at least we should have. I suspect that most software integrity issues are not caused maliciously, but are as the result of poor change management processes. However, poor change management does provide an opening for malicious manipulation, so, with my audit hat on I will attempt to explain where I believe the main issues to be.

Firstly and most importantly the majority of change management processes are based on trust. Trust in the programmer to correctly make the requested change; trust in the systems people to adequately test the change and trust in the user to accept the change after suitable testing. Unfortunately, trust is not a control mechanism, but an act of faith. If you don't believe me, the please send me a signed cheque made out to me with a blank amount. I promise, not to insert a large amount, nor to submit the cheque for clearing. It will be interesting to see how trusting you are by the number of such cheques I receive. The problem with trust is that you only find out that is misplaced after the event, so it is a pretty useless prevention mechanism. Testing is actually a trust substitute. You don't really trust the programmer to get it right so you go into detection mode via a test mechanism. Now this is all well and good, but unfortunately most test mechanisms centre on the authorised change. If the programmer inserts some other code at the same time as the authorised change, then the chances of it being detected during the testing process are negligible. So let's not be naïve. Let us remove trust from the equation. My (untrusting) change management process goes like this.

1) receive authorised change request;
2) security officer (or equivalent) retrieves digitally signed source code from once write-only media held in an off-line store;
3) source code is made available to the programmer;
4) programmer makes change, produces executable & tests it;
5) programmer returns amended source to the security officer;
6) security officer does an electronic compare between original & amended sources;
7) another programmer compares the code changes against the change request;
8) Assuming that no illicit code is detected by (7), then the security officer produces a digitally signed executable and promotes it to production;
9) The security officer stores a copy of the amended source and executable (both digitally signed) onto once write-only media into the off-line store mentioned in (2) above;
10) Every time the program is now executed its signature is checked.
11) On a regular schedule the production executable is automatically compared with the copy from the off-line store to detect any really sneaky

manipulation of the production code & signature.

For clarity I have abbreviated the process by removing system and user testing, but you can add those where you like. The points being that:

1) the programmers know that if they insert illicit code it will detected;
2) the security officer does not have access to code editing tools and therefore cannot amend the code;
3) any back-door changes to the production code will invalidate the signature and so will be detected when the code is executed;
4) as copies of the source and object are held off-line they cannot be remotely amended;
5) the regular comparison between the stored object and the production code will detect any change to the latter which has somehow been done without invalidating the signature (not that I believe that this could be achieved anyway).

Of course, if you had collusion between the first programmer and the checking programmer then you are in trouble, so it would be sensible to rotate the checking and to have a bonus system which will reward diligent checking. Most of what I propose can be easily automated which puts into level four/five on the CMMi scale and thus makes it Sarbanes-Oxley compliant too. Which brings me to my last point. Many so called controls I examine are just processes. There is no real control in the sense of prevention, or detection that can be effectively measured and I am constantly amazed that many security officers (and auditors for that matter) cannot define what a control is, or how it works, which is why I am forever telling them that the nice move from inherent red to residual green in their risk register is a figment of their imagination. Non more so than with their change management processes which are usually beautifully documented with wonderful flow charts, but a primarily trust based.

John is Managing Director of LHS Business Control, a corporate governance consultancy. He is a member of Council and a former chair of the Information Risk Management and Assurance (IRMA) specialist group. He can be contacted at: john@lhscontrol.com, www.lhscontrol.com, or on +44 (0)1707 851454