Intelligent Buildings International
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tibi20

Cyber security, building automation, and the intelligent building

David Fisk [a]

[a] Laing O'Rourke Centre for Systems Engineering and Innovation, Imperial College London, London SW7 2AZ, UK

Version of record first published: 14 Jun 2012

To cite this article: David Fisk (2012): Cyber security, building automation, and the intelligent building, Intelligent Buildings International, DOI:10.1080/17508975.2012.695277

To link to this article: http://dx.doi.org/10.1080/17508975.2012.695277

iFirst

Intelligent Buildings International
iFirst article 2012, 1–13

RESEARCH ARTICLE

Cyber security, building automation, and the intelligent building

David Fisk*

Laing O'Rourke Centre for Systems Engineering and Innovation, Imperial College London, London SW7 2AZ, UK

(*Received 20 March 2012; final version received 26 April 2012*)

Extending the narrow scope of building management systems to form the 'intelligent building' has led to the widespread use of proprietary 'enterprise' software platforms and networks in both monitoring and control. The PC user is only too familiar with the vulnerability of networked computers to malicious software. But it is only recently that attention has been paid to the possibility of virus damage to process controllers. The hazard for 'building management systems' functionality is real but the risk is difficult to assess. Pressures on system procurement and upgrades aimed at improving functionality bring with them increased exposure to the probability of a successful attack. Most published security protocols may engender a false sense of security. The key defence is to ensure a 'fall-back', 'black start', 'dumb capability' within the intelligent building.

Keywords: building automation system; integrated building management system; intelligent building; reliability; cyber terrorism; cyber security

Introduction

Key physical components in the intelligent building, such as wi-fi transmitters or routers, are invariably imported from information and computing technology (ICT). But users of this technology elsewhere are only too familiar with computer viruses that infect network software. The market for virus protection has been estimated at $2B each year (e.g. Savvas 2007). But it is hard to find any discussion on cyber-security issues in intelligent building design. Although overt security in the form of 'intelligent' entry control and fire evacuation systems have been an intrinsic part of modern services design, the integrity of the control system itself has not until recently been discussed. An intelligent building while offering greater service also poses a cyber threat to its occupants. This article assesses the nature of the hazards the risks and their influence on intelligent building technology.

The intelligent buildings community might rightly protest that there have been no significant reported successful attacks on building management system (BMS) technology over the 40 years that technology has been deployed. But this is not a case of a perfect safety record being able to engender confidence that no such attack will happen. The risk landscape for this technology has been changing by stealth from almost non-existent in the 1960s to the rewriting of major procurement guidance and security alerts about BMS software in this decade. The first part of this article traces the background technology developments that have shaped the new landscape. It then

*Email: d.fisk@imperial.ac.uk

ISSN 1750-8975 print/ISSN 1756-6932 online
© 2012 Taylor & Francis
http://dx.doi.org/10.1080/17508975.2012.695277
http://www.tandfonline.com

reviews the relevance of activity in the wider Supervisory Control and Data Acquisition (SCADA) industry. Finally, it looks realistically at the likelihood that ICT technology would be sufficient to protect an intelligent building over its lifetime.

Cyber security and process controller technology

The 'safe period'

The original BMS with direct digital control (DDC) were dedicated hard-wired machines (e.g. Honeywell 1965). If there was an aggressor hazard it was presented by the rogue operative with a hammer! In the next stage of technology development in the 1980s, communication protocols were developed to enable devices from different manufacturers to be connected to the same *dedicated* communications bus. That enabled satellite outstations with (what was then) real computing power and the capability for outstations to receive instructions that changed or loaded programmes as well as executing them. Given that the controls might be expected to last at least as long as the conditioning system – 10–15 years – the ability to update software was essential. At this stage of technology development, the risk comes from both the rogue operative and the rogue maintenance engineer. The risk with the latter was low because the machine code in the device was bespoke and so hard to tamper with and the point of entry of the attack likely to be evident. At the beginning of this stage of development, commercial offices, if they used ICT at all, did so in batch main frames or stand-alone PCs. But the Microsoft Windows platform soon expanded enterprise ICT to server and network technology. Those networks expanded beyond the local area into a world-wide web. In a parallel unwanted development, malicious software and hacking was also beginning to emerge on ICT networks in the early 1980's.

The effectiveness of malicious software is a reflection of the intrinsic weakness of the host system. As an example, one particular problem, symptomatic of this area, was the existence of 'back doors' in host systems. The engineering researcher seeking to demonstrate the feasibility of intelligent building software is not likely to address maintainability in drafting prototype code. But in commercial applications, the vendor of software may well wish to 'enter' a system, for example to debug a specific problem that arose during development. 'Back doors' are then left open which are supposedly known only to the vendor or through oversight never get removed when the product is rolled out. One of the most infamous back doors was inserted in the ubiquitous C language. The C compiler hid in 'Login' a facility that would enable those with the knowledge to login as any user on any system with a special password. Trying clean source code to recompile the compiler simply reinstated the back door (Thompson 1984). The code promulgated itself over networks.

The rapid increase in computing power and the opportunities that came with it meant that codes became ever larger and more complex. Once data communication protocols were established, it became plausible to use the same software proprietary platforms (especially later generations of Microsoft Windows) that were being used by the enterprise's own network. Indeed the software overhead of writing a bespoke platform made it increasingly unlikely that any manufacturer would write machine code. However, mainstream software engineering seemed so concerned with malicious software getting access to corporate and personal data that the idea that the same aggressive techniques could be used to cause hardware to malfunction was hardly every discussed. This might have then been justified. Computer hacking and virus promulgation were dominated by criminal greed rather than hacker hubris. Accessing bank accounts and other personal details was much more profitable than frustrating controller set points. The idea of inserting malware to frustrate network operation was hardly considered.

Post 9/11 assessments

The terrorist attack on New York in 2001 spawned wide ranging assessments of risks from 'innovative' terrorists. Tucked within this large material was the recognition of ICT risks to infrastructure from cyber terrorism. Much of this material drew on earlier work that shadowed conventional virus technology (Rathmell 1997). Industrial sites or email service could be attacked by denial of service or just hacked. But it became evident that SCADA could also be compromised. BMSs are a subset of this technology family. 'Programmable logic controllers' are relatively more common in process control technology but other elements are similar. Incidents are recorded in the INL database (Turk 2005). A typical symptom would be the sudden invisibility of an outstation, or the outstation resetting to default set points. In many cases the problem was first interpreted as a hardware problem. It would take typically a week to eradicate. Characteristically, the means of introducing the malware usually cannot be identified, although all the cases study post mortems in the INL database exposed vulnerabilities such as 'back doors' or operations protected by inadequate strength passwords.

The attack takes place in three stages:

- The network integrity has to be breached usually in a low-profile area. For example, knowledge of file-share facilities with external enterprises can represent weak points through which the aggressor gains entry. Or sometimes remote controllers use the internet for part of their data pathway. This stage is 'human' in the sense that the attacker needs knowledge of the broad properties of the network.
- Once the aggressor is in the network malign computing power can take over. At one extreme this allows brute force attacks throwing millions of possible passwords at devices with no 'number of attempts' check. IP addresses and so weak passwords on the whole system can be identified by the invader.
- The invasion secured it delivers its intent.

This situation was a change in risk. The rogue employee or maintenance engineer, or indeed rogue programmer, were all agents whose identity was likely to be discovered after the event. But the anonymity of cyber aggression weighs heavily in favour of its use. A small group could risk a large attack because their identity would remain undisclosed and so be able to attack again (albeit by a different strategy). More than half of the events in the INL database were initiated anonymously. The number of cases was nevertheless small compared with 'conventional' cyber threats which for key US institutions might total in the thousands. Events in 2010 changed the picture and risk assessment.

Stuxnet attack

In 2010, a PC in Iran began to repeatedly reboot itself. That would sound familiar to PC owners who have suffered a virus attack. The virus, now labelled Stuxnet, represented a large coding commitment by an unknown agent (Weinberg 2011). It had around 15,000 lines of code. But what would have been at once conspicuous if inserted into an early DDC BMS (with something like 64-k RAM) was easily lost at modern download speeds and disk storage space. Once it had infected a host, it sought to communicate on a Windows platform with other devices that were running Step 7 the Siemens systems used in their programmable logic controllers. Siemens are of course one of the world's largest manufacturers of controls and control systems. Their devices are everywhere. They dominate much of the Smart Grid market. Industrial controllers are not themselves usually connected to the internet (or so their operators think!), just to keep them quarantined.

How did Stuxnet achieve the first step? It installed itself on any USB drive inserted in the infected system and then went wherever the drive went next. Inserting the drive inserted the virus as the drive was installed. Such drives are routinely used to transfer data between standalone networks. Stuxnet transfer was activated simply by inserting the drive. It then was ready to insert itself in any clean USB stick inserted later. A flavour of the power of engineered malware is given by how Stuxnet hid from site operators that programmable logic controllers were under attack. Siemens had designed the input process image to the controller as read–write instead of read. Usually a harmless extended flexibility it enabled Stuxnet to play back to the main system recorded process data as if the device was working normally and not under attack. In once sense Stuxnet is exceptional because it was a bespoke virus carefully designed to frustrate a specific plant in Iran. The strong suspicion is that it stopped the Iran uranium enrichment programme for a while in 2009. But it signified the wider potential of malicious software to those who write it and the vulnerabilities of SCADA systems.

Stuxnet is now patched (Siemens 2011) but unfortunately the idea is out that malicious software can infect plant controllers at the very time that SCADA engineering is tending to move away from physically quarantined control systems to fully integrated information systems embedded in enterprise software. Around 140 SCADA events (as opposed to the legions of vulnerabilities) were recorded in 2006 (ISID 2006). The US now has a strategy to secure SCADA systems (DHS 2009).

Smart grid vulnerability

One area where this threat is fully recognized, that is especially relevant to the intelligent building, is the 'smart grid'. Whereas the intelligent buildings research community might require some convincing that they face a cyber-threat, power engineers are much more aware because they have already experienced on a massive scale the impact of malfunctioning control software. The vast regional-scale North American power failure in 2003 was initiated a quite commonplace HV grid line fault. But the failure to contain its effect to one US State was due to a failure in the IT system that supported the inter-regional system of control. A *running* bug overloaded data terminals until their screens were refreshing slower than lines were tripping out. Power engineers at least are aware that software can wreck hardware. There is another lesson from that event that reflects a characteristic of interconnected engineering systems (Fisk 2004). Failure of a ubiquitous support system can undermine other stand by systems. The NY telephone back-up system remained operational in the 4-day blackout – unfortunately the telephone key pads in payphones were run by mains power so the systems were useless. Cell phone systems worked but their effective life within a regional power blackout was the life of the weakest handset battery not the standby power at the local transmitter.

The promise that links the intelligent building to the wider smart grid is a new form of demand side management with devices inside the building such as smart meters and the BMS, dealing directly with the power network's own software network posting prices. The smart grid is clearly on the intelligent building agenda if not the risks it poses (Chen 2004). ASHRAE is already involved in some of the standards (Bushby 2011). That represents the upside, but the downside is that 'successful' malicious software could shut the whole system down for weeks. The US Government estimates that the business for cyber security on the smart grid is around $21B over the next five years. So the BMS could be linked to a 'trusted' network itself under attack, or the BMS could be (in theory) a hole or backdoor by which malicious software reaches a larger system.

In summary, as the arrival of DDC at the end of the 1960s, while the individual hazard of services failure is largely the same, the risk landscape has changed for the worst. Parallel activities unrelated to building services, whether criminal or military, have provided the considerable programming resources to develop malicious software. Its attraction to malicious agents over physical attack is its anonymity. Keeping the intelligent building control system tightly quarantined is obviously one option. But the opportunity cost mounts and mounts as maintainability, access to enterprise data and products such as time of day power tariffs profiled in minutes become available to those openly connected. Although the research prototype might be isolated, it seems very probable that commercial realizations that promise the builder owner 10–15 years of service would have entry points.

Open systems are always vulnerable

Stuxnet got through vulnerability in a modern Windows platform and has been patched. But as a matter of formal logic, an *open* software platform can only be protected from malicious software that has been identified, but not against malicious software yet to be deployed. This is because logically it is the operator who defines 'malicious' not the machine. Much of the literature on SCADA explains elaborate protection systems against aggressors. But the building owner needs to be realistic about their delivery given their failure even at highly protected systems under focussed attack (e.g. COS 2012).

The argument is commonsense although it can be formalized. Conceptualize intelligent building software as something that transforms a building in state A to state A1 at the next time interval. Because it is an open system, the software could be reprogrammed to transform from state A to state Y. But in strict state space terminology that means its original state was really [A, a, c], where '*a*' designates the current software and 'c' the new software in the system buffer, because we need knowledge of all three to determine the next state. To be consistent, the final state is [Y, y, $< >$] where '*y*' is the new programme, 'a' upgraded by 'c' and the buffer is now empty designated by $< >$. If 'c' is a 'known threat' then it is easy to conceive virus protection software in '*a*' that results in [A, a, c] $\rightarrow$ [A1, a, $< >$]. Trivially 'c' might not have the right administrator password. But it was the external agent who programmed '*a*' to check for passwords. But the programme *y* makes the building state transformation and as a consequence it cannot be used to discover the outcome of loading *c* because *y* is not in place before loading. In response suppose *a* is somewhat more sophisticated programme only a part of which, *a*1, would actually transform the building state. If *c* is read and used to upgrade a replica of *a*1 in a simulation programme in *a* and so checks against *Y* as an outcome before *c* is allowed to modify the actual *a*1, it requires only a code that switches after being checked to thwart the system. The programmer of '*a*' would not only have had to anticipate *Y* but an outcome that led to *Y* after testing. Put another way a code '*a*' cannot authenticate itself. This kind of undecidability problem is not unique to computer code, and is commonsense. It is simply a modern version of *qui custodiet ipsos custodes*? Most computer cyber defences are based on a more prosaic strategy: the open system can be defeated but its security lies in the implausibly long time required for a sustained successful attack.

Most virus protection services guard against *known*, that is, characterized, risks with identifiable signatures. Thus, for at least one user this is closing the stable after the horse has bolted. The cards are not entirely in the hands of the aggressor. The compiled code in a programmable controller is usually not easily interpretable, and efforts can be made to reduce the ability to tamper with it. So aggression is likely to be focussed on the system characteristics of the objects into which the code is bundled (e.g. buffer overflow), and these are often provided in sources like manuals or exposed on the web. The programmer can of course password protect against reprogramming. But the password is then the focus of vulnerability. A conventional safe combination lock presents the same formal problem. What is unique is that the intruder does not need to break in physically to challenge it.

The problem is especially acute for old software platforms in newer environments. Software platforms become either unacceptably slow from the burden of accumulated patches or technically obsolete in meeting new demands. In either case support is withdrawn by the vendor, usually Microsoft. Support for Windows XP SP2 ended July 2011. Other platforms like Vista Release to Manufacturing have had a shorter life. Indeed even older platforms like Windows 98 are, significantly, thought only to be left in legacy SCADA systems such as utilities (Gold 2009). Here is the issue. If the intelligent HVAC system is to have a life of more than a decade it needs to have outside connectivity. But that implies connectivity to the threat. Because high-profile hardware hacking has gone for high-profile targets more low-profile building services have hardly had a mention. As a consequence discussions on new BMS capability make little or no reference to cyber security. There is even an acronym CAFM – computer-aided facilities management – that links the BMS to the enterprise's system (Teicholz 1991). Of course data can be exchanged between systems with a high level of security. A dedicated server placed between the enterprise system and the BMS is one solution. But that requires investment that those believing the BMS poses few risks might be reluctant to make.

It is in principle easier to detect that an attack *has* penetrated into a SCADA system than it is for other components of an enterprise system. This is because there is a better defined 'normal' pattern of traffic when controlling and monitoring a process than transferring more general information (Cheung *et al.* 2006). 'Malicious behaviour' is then interpreted as uncharacteristic deviations. Intrusion detection systems are then added with a formal model to create alert conditions (Valdes and Skinner 2000). Maintenance upgrades could create an alert, but as the operator knows maintenance is underway this alert can be ignored. In industrial applications this form of intrusion detection can be a valuable service, especially in batch processing. A warning in the early stages of a dangerous process could avert disaster. In a continuous process like a building service the gains are less obvious. The building service will need to be withdrawn while it is cleaned and that can take some while. Indeed if system withdrawal was the aggressive intent, an intrusion that triggers an alarm is then enough and much simpler software to write.

Assessing the risk

In an unexceptional intelligent building a bespoke virus like Stuxnet should be viewed as exceptional hazard simply because of the work involved in creating it and the need for specific intent. Consideration of existing SCADA case studies offers a few pointers to the kind of hazards and consequences posed by networked BMS technology (Table 1). It might be concluded that

Table 1. Programmable Logic Controller (PLC) infected on networks in SCADA database (Turk 2005)

Symptom	Diagnosis
Controller reset to zero	Fault in TCP/IP stack
PLC's supplied with Web access	Servers unconfigured with factory setting default password
Safety system disabled by 'worm'	Contractor's infected computer connected directly by telephone dial up
PLC shutdown	Remote location meant controller shared network with internet traffic
Valve cease to operate	Wi-fi link overwhelmed by interference
Substations go invisible	SCADA connected to network via old routers with old software
Valves pass control to external agent	Wi-fi ingression followed by maintenance software on rogue laptop

- *First the BMS risk assessment must assume that the wider network can always be invaded.* Some very high-profile networks with very sophisticated protection have proved to have fatal vulnerabilities. Larger network within which the SCADA is embedded provides the location of backdoors or holes. Large heterogeneous networks are hard to update and maintain. Some parts might offer wi-fi access or file share or modem dial-up access. These unmonitored entries to the larger network seem inevitable as a network ages.
- *Incidental infection is then always a risk.* Malware seems as effective at disabling the SCADA part of the system as it is at infecting other enterprise units.
- *Targeted infection differs in character.*

Legacy risks

There are two factors which seldom earn discussion which affect legacy systems. The older the system, the more vulnerabilities become known in the aggressor community. The greater its vintage the greater discrepancy between the power of computing technology deployed by the system compared with that available to the assailant. This section attempts to quantify the situation.

The increase in computing power available to aggressors is presumably related to Moore's Law'. This is usually stated as a doubling of chip density every two years. To be conservative let this represent a 35–40% increase year on year in computing power (i.e. as the square root of transistor density). Guessing the time properties of vulnerabilities is more problematic. The very worst case is that the initial vulnerabilities were not repaired at all but become known over time. Indicative numbers are given by an influential article (Ozment 2006) that studied vulnerabilities in UNIX code patched by an 'open source' community. In a 10-million line code (not untypical for a platform) vulnerabilities in most releases patched 10 or so vulnerabilities but these included vulnerabilities introduced by previous patches. Figure 1 is adapted from this work. Vulnerabilities eventually settle down to a low level at which a patch removes as many vulnerabilities as it introduces. This section's model assumes this unhealthy equilibrium heralds the arrival of a new upgrade system. A new system is riddled with (initially undiscovered) vulnerabilities (about 60% of the vulnerabilities in the source article were present in the foundation code) but the aggressors search is reset to zero with no patches as yet to give clues where the vulnerability might lie. Multiplying computing power by vulnerabilities gives Figure 2. Initially patches keep up with external computing power growth. But once patching becomes self-defeating computing power takes over and potential vulnerability discovery increases.

For incidental infection, it is less likely that aggressors will devote time to finding general vulnerabilities on obsolete platforms when there are new systems to exploit. The legacy threat of Figure 1 will not be realized. But a targeted attack reverts to Figure 1. Once the aggressor has identified a sub-system with obsolete code, it becomes a focus for attack. Somewhat unhelpfully for delivering a clear cut strategy, keeping obsolete code on a networked SCADA enhances its resistance to infection from a new virus, but raises its risk from a targeted attack.

The central conclusion of this section is that large systems will always have entry points, so that the rate of aggression is limited only by the number of aggressors, much as the number of burglaries is limited only by the number of burglars not the number of burglar alarms. Given this perspective on steps 1–3 of an incidental infection ought to be treated very much as any other component failure. As a consequence, the wise intelligent building will have a back-up plan if it were to become infected. The only difference from a hardware malfunction would be its extent. If one controller is infected then very likely all similar controllers in the intelligent building network will be infected. A characteristic of malware infection is that it takes several days to clean, as in the C program example earlier, because the code may need to be rebuilt

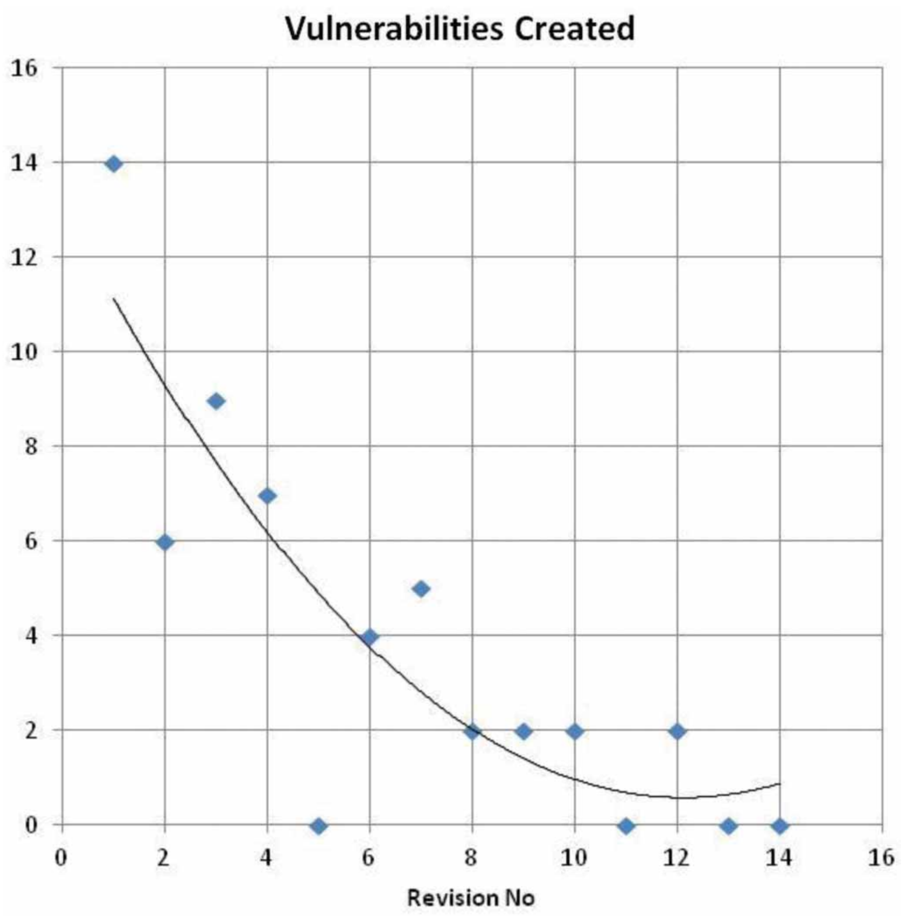

Figure 1. Software vulnerabilities per revision.

from scratch not just rebooted. The backup plan presumably implies some hands-on operated basic control. Reactions to a recent working paper (Fisk 2011) suggest this option is often missing.

Directed attacks

A directed cyber attack is very different in character from an incidental infection. The threat is now a dynamic enemy and inherits many of the aspects of game theory rather than risk analysis. ‘The greatest threat is the threat ranked second’ etc. The usual strategy is ‘hope for the best’ (in this case threats are directed elsewhere) but ‘plan for the worst’. The point is that if the Red Team detect that Blue have a fall back plan, the gains for a Red victory might not even merit the attack. But if Red detect that the Blues have ‘burnt their boats’ then Red is justified in dispatching its entire force with the promise of total victory. Current risk assessments are undeveloped in this sense. Because, as frequently observed, there is no immediate gain from accessing the BMS network, which is the very reason not to leave it wide open. Indeed it is because risk assessments fail to find an answer to ‘why would they?’ that actual system commissioning or maintenance may even fail to deliver elementary security (like avoiding trivial passwords), that was assumed in the risk assessment. A hypothetical example makes the point.

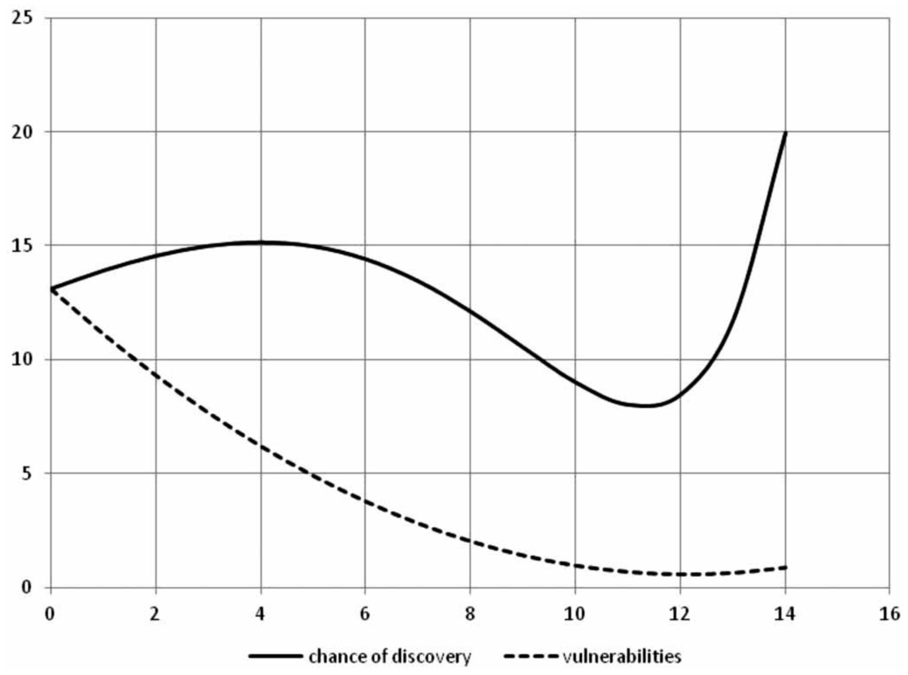

Figure 2. Vulnerabilities and Moore's Law Rate of Search.

A hypothetical cyber threat analysis

Throughout the world, HVAC systems service the vast gaming floors of super casinos. These buildings face threats both from some of their clients, from some external agents and may be even from their shadier competitors. Consequently these buildings employ specialist services that include advanced access control ICT to protect cash, scan employees, and record fast-moving hands on gaming tables. These could be integrated into an intelligent super casino system, using the data to anticipate conditioning loads or lighting levels. But in practice they would be kept distinct to retain their integrity. The BMS would provide its own monitoring data.

Of the services themselves, lighting is critical. If the lights were to fail even for a moment the casino would have lost sight of thousands of chips in play. Consequently super casinos employ large standby generation sets. But to cover start-up they also have large banks of standby batteries. If start-up fails for some generator sets, the BMS sheds non-lighting load. But under the 'plan for the worst' the battery back-up provision is enhanced to allow time for gaming to be halted under lit conditions if all sets fail, before emergency lighting comes on and the casino is evacuated. But like a vulnerability patch, this extra investment highlights the point of weakness.

Now an 'intelligent super casino' might automate the entire process so avoiding human error or oversight (like an operator tea break). But this would be a very dumb thing to do. The supervisory system now only needs to be infected to jeopardize the entire security investment.

The wise 'plan for the worst' would be to hardwire the entire standby power system independent of other systems. This, of course, is the strategy usually deployed in fire control. But a super casino is a substantial local load (say 3–5 MW) and it may never be quite free of links to an HV smart grid that is desperate to find interruptible load. Indeed providing such a link would facilitate 'hot start' of the standby generators as grid failure approached and so apparently improving system reliability. The security analyst should also ask what the SCADA system is doing where there is a hardwired backup system. If the data acquisition is for 'peace of mind' one might arguably ask how it survived the value engineer. If it is to permit operator intervention on hardwired failure then the analyst is faced with the infamous ‘false red’ problem. Is the ‘red’ a failure of the system or a failure of the data acquisition induced by an infection? Having expanded this analytic trail the analyst might usefully back track to the beginning. If the HVAC system just locked down and refused to black start, an internal temperature of 40°C and 80% relative humidity would be enough to clear the complex.

BACnet security standards

The argument this far has shown how the risk to SCADA systems has crept up as ICT technology advanced. It has looked at both incidental and targeted infection at a high level. It is now time to probe further into the state of real systems. There often seem more enthusiasts uncovering ‘vulnerabilities’ than malicious forces exploiting them. But Stuxnet has focussed attention on SCADA systems, thus increasing the attention of hackers, and vulnerabilities are frequently reported ahead of manufacturers’ deciding what if anything they wish to do about them.

The BACnet standard was devised to enable the full power of networked intelligence that included building management systems. To ASHRAE’s credit, a special Network Security Working Group was formed actually ahead of 9/11 because of mounting concerns about the vulnerabilities in the standard (Figure 3). BACnet allows transmission across networks only part of which are ‘trusted’. The idea is to gain the corporate advantages of wide data dispersal. Progress has been reviewed by Holmberg (2003). The review is somewhat predicated by the belief that BMS data are of little value and predate realization of the threat to Programmable Logic Controllers. Nevertheless, some alarming features emerge. First many of the conventional protections exist but would have needed to have been commissioned (e.g. setting limits to login failures, removing default passwords) at site installation. There is no requirement as yet to require the machine to assess its own vulnerability once ‘set to work’. The encryption standard used is the 1976 Data Encryption Standard. However, Moore’s law means that 23 years later the password key could be deciphered in less than 22 hours. BACnet was also issued without provision for buffer overflow. This is vulnerability where data sent to the device is so large that it exceeds the memory space allocated to the data buffer, and so maliciously starts to overwrite adjacent memory storage with destructive effect. (The same effect is a known weakness in programs written in C++.) Indeed as late as 2010 Symantec issued an urgent alert of the danger from BACnet to the servers via the Open Process Control interface because a ‘.csv’ (i.e. a standard spreadsheet database format) could overload buffers and create denial of service.

The inherent problem seems to be that while operators of BMS believe their data have no value they have been less than motivated to install even elementary security and so increasing the probability of being the focus of a successful malicious attack.

Discussion

This article has brought together findings from a wide range of studies that could have a bearing on the integrity of an intelligent building in the presence of cyber attacks. There appears to have been a disconnect between the frantic backroom activity on improving the security of SCADA systems in general and the parallel activity of improving the intelligence of building services. The natural way of putting all this together is to deploy a whole-system design approach. Intelligent buildings are buildings serving functions first and intelligence second (Clements Croome 2004). What needs to be added, in an uncertain world are whole system requirements for security and system integrity.

In a whole-system perspective the argument begins with what is required of the buildings systems, not whether they are intelligent or dumb. In that specification it becomes evident

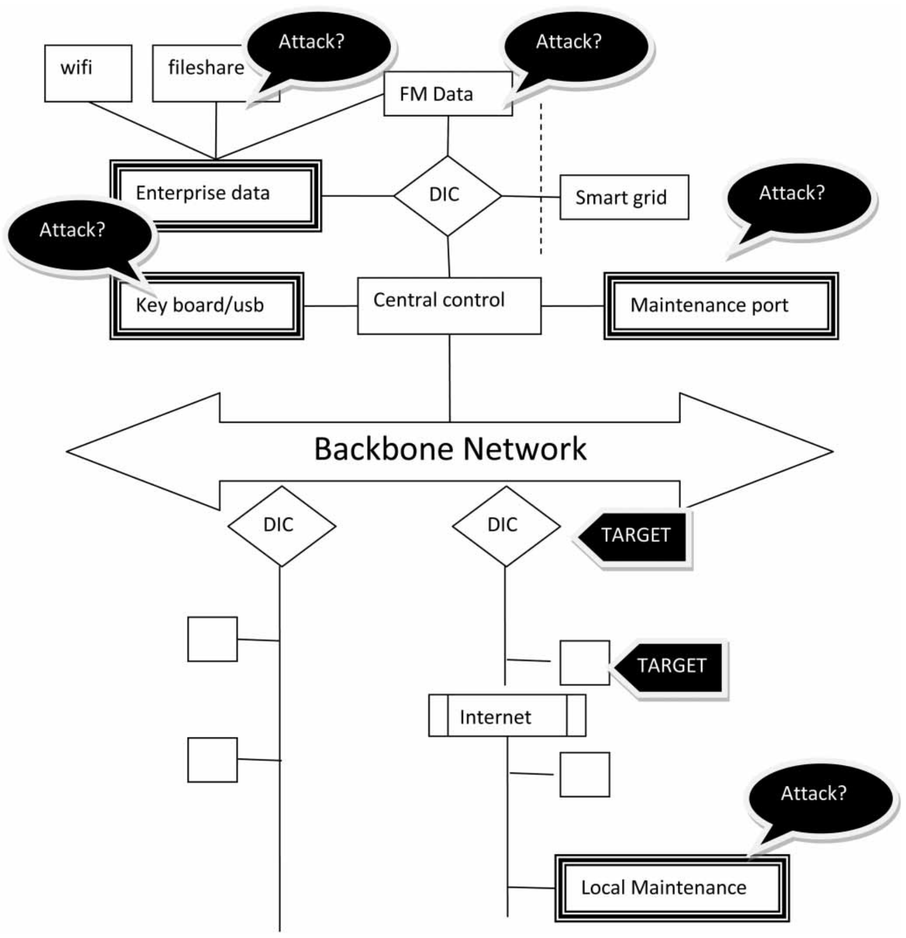

Figure 3. Vulnerabilities in BACnet.

from the beginning to identify what kind of loss of service the building occupants wish to survive. If a full loss of service can be tolerated then no subsystem requirement is generated. But if there is a minimum service (20 min lighting in the example of the casino) then a subsystem requirement is passed down. Similarly if the building is expected to be a target for malign intentions as a consequent of its functions, the need to offer resistance feeds down to the function of several subsystems, not just access systems. Finally the longevity of the structure and its systems is a top-level whole-system requirement which turns out to have consequences for security.

On the arguments advanced above the whole system approach should always assume that the networked system can be compromised. The design needs only consider the probability when this assumption needs to divert resources. Thus, if there is no subsystem requirement for basic performance there is no requirement to address system vulnerability. If there is a basic performance requirement then this also applies to the control and data acquisition network performance. From the intrinsic vulnerability of software systems, it follows that the basic system (e.g. the bare minimum standby generators) should normally be independent of the intelligent-building software (much as a warship still carries a sextant should the GPS be jammed). This is not current practice as far as can be discerned from existing ASHRAE and CIBSE standards and can be viewed as the central conclusion of this study of the current state of security. The absence of basic provision both increases the hazard and the incentive to attack and so the probability of aggression.

The longevity system requirement which permeates many of the subsystem choices also influences the cyber risk strategy. A relatively short life building is unlikely to acquire legacy systems. But a long life building requirement imposes new monitoring and control subsystem requirements. This article has shown that the requirements bifurcate depending on whether the building is likely or not to be subject to a targeted, as opposed to coincidental attack. This situation is not that different from conventional security devices. Thieves may not know how to pick an old lock unless access was worth the special research.

Ambitious whole-system design extends beyond subsystem component choice to component commissioning and operation. This has traditionally been a weakness (e.g. default passwords) in subsystems not thought as high-profile risks. Consequently, the probabilities of attack have to be ramped up twice. First to attain a realistic level, and secondly to reflect an assumption by an aggressor that the commissioning may not have been all it should have been. At the time of writing (May 2012) proprietory software is not available for checking networks that include SCADA systems.

However, many buildings already have made substantial investment in standby equipment and so have something to protect. The commonly quoted figure for the US is 100,000 standby units. So there is arguably a substantial collection of buildings that need to take loss of function seriously. Unfortunately, the discipline of a whole-system approach is rare in construction and it is therefore plausible that buildings are vulnerable because of overlooked BMS security.

Stuxnet is disabled, but the war is not over. In 2008, a virus now called *Conficker* began to infect computer systems. Within 18 months it had spread to 8 million computers worldwide (Bowden 2010). Maybe that includes the one on which this article is written, because to date (May 2012) no one knows what it will do, and PC owners are unlikely to know they have it. Like Windows Updates but in reverse, the virus (strictly a 'worm') receives patches from 'out there' that close vulnerabilities to the anti-malware programmes the user has bought, all the while it sits unnoticed among the several thousand files on the user's hard drive. Not that this neat feature matters all that much on those machines where it has already disabled Windows Automatic Update without the owner noticing! Tomorrow *Conficker* might wake up. Will the building's lights go out, or just flash on and off, a Xmas tree for all to see? Where, come to mention it, was the CEO when it happened? In the lift?

Conclusions

If intelligent buildings are the future, then so too are cyber threats to building services. Although the hazard is much the same as a physical attack on key components like boilers or substations the likelihood will in some cases be higher because the source of the attack will remain at large. The characteristic would be a shutdown of intelligent functionality for several days. Although grateful for the ICT industry's attempts to contain this threat, the wise building owner should assume that on a network of any size human oversight will have left open entry points to be discovered. Unless the threat is so critical that the loss of functionality resulting from complete isolation is tolerable (really? Even for upgrades?). The owner should plan for periods during which 'intelligence' is not available. The model developed here suggests that for incidental infection the most sensitive time is around five years after the software's creation date. For targeted attack the risk simply increases from that point onwards.

The plan is relatively simple. The correct strategy is to draw up a 'plan for the worst' rather than rely on assertions by software and hardware providers. They will no doubt do their best (e.g. Novak *et al.* 2007) but cannot offer comfort on 'unknown unknowns'. What functionality is retained if there was a failure in the BMS? ‘Intelligence’ in buildings often addresses finding the ‘last 20%’ of performance. Although a shame to temporarily lose this gain from new technology, it might be better to have some basic hardwired functionality than no functionality at all. Determining current functionality may need detailed site inspection. For example, so much attention may have been paid to the physical integrity of a large data centre that the cooling unit, as an actual onsite installation, details could have been entirely overlooked and assumed to be as on the drawing!

An identified minimum level of service and hardware hardwired that can provide it is thus essential. The very existence of such a plan may not make the reward of a targeted attack worthwhile.

References

Bowden, M., 2010. The enemy within. *The Atlantic*, 23–24 June.

Bushby, S.T., 2011. Information model standard for integrating facilities with smart grid. *ASHRAE Journal*, 53 (11), B18–B22.

Chen, A., 2004. Multi-building internet demand-response control system. *Environmental Energy Technologies Division News,* 5 (1). California: Lawrence Berkeley National Laboratory.

Cheung, S., *et al.*, 2006. Using model based intrusion detection for SCADA networks. *Science & Technology*, 329 (7461), 1–12.

Clements-Croome, D., 2004. *Intelligent buildings design mangement and operation*. London: Thomas Telford.

COS, 2012. NASA cyber security. *Committee on Science House of Representatives*, February 29.

DHS, 2009. *Strategy for securing control systems*. Washington, DC: Department of Homeland Security.

Fisk, D.J., 2004. Engineering complexity. *Interdisciplinary Science Reviews*, 29 (2), 2–10.

Fisk, D.J., 2011. Bug in the BMS? http://www3.imperial.ac.uk/lorsystemscentre/workingpapers [Accessed 19 March 2012].

Gold, S., 2009. The Scada Challenge. *Network Security*, (8), 18–20. IEEE.

Holmberg, D., 2003. BACnet Wide Area Network Security Threat Assessment. Technical report, National Institute of Standards and Technology, NISTIR 7009, July.

Honeywell, 1965. *Introducing the series 16 DDC system*. Minneapolis: Honeywell Inc.

ISID, 2006. *Industrial security incident database*. Burnaby: AITG British Columbia Institute of Technology.

Ozment, A. and Schecter, S.E., 2006. Milk or wine: does software security improve with age? *In: Proceedings of the Fifteenth USENIX Security Symposium*, 31 July–4 August 2006, Vancouver, BC, 93–104.

Novak, T., Treytl, A., and Palensky, P., 2007. Common approach to functional safety and system security in building automation and control systems. *In: Proceedings of 12th IEEE International Conference on Emerging Technology. Factory Autom. (ETFA’07)*, Patras, Greece, 1141–1148.

Ralstona, P.A.S., Grahamb, J.H., and Hiebb, J.L., 2007. Cyber security risk assessment for SCADA and DCS networks. *ISA Transactions*, 46, 583–594.

Rathmell, A., 1997. Cyber-terrorism: the shape of future conflict? *RUSI Journal*, 142 (5), 40–45.

Savvas, A., 2007. Anti-virus software market grows. *Computer Weekly*, 13 July.

Siemens, 2011. http://support.automation.siemens.com [Accessed 1 December 2011].

Teicholz, E., 1991. *Computer aided facilities management*. New York: McGraw Hill.

Thompson, K., 1984. Reflections on trusting trust. *Communications of the ACM*, 27 (8), 7614.

Turk, R., 2005. Cyber Incidents Involving Control Systems. Technical Report. Idao National Laboratory, US DOE, INL/EXT-05-00671, October.

Weinberg, S., 2011. Is this the start of cyber warfare? *Nature*, 474, 142–145.

Valdes, A. and Skinner, K., 2000. Adaptive, model-based monitoring for cyber attack detention. *In:* H. Debar, L. Mé and S. Felix Wu, eds. *Recent Advances in Intrusion Detection (RAID 2000), Toulouse, France,* Lecture Notes in Computer Science, Vol. 1907, Springer-Verlag, Berlin, 80–92.