HAL Software

ENTERPRISE IOT SECURITY CHALLENGES FACING INDUSTRIAL AUTOMATION.

Date: August 2014
Version: 1.3
Author: Cormac Garvey CISSP (28432).

HAL SOFTWARE LIMITED
ASHGROVE HOUSE
KILL AVENUE
DUN LAOGHAIRE
COUNTY DUBLIN
(T): 00353-1-2896262 / (M): 00353-87-7710560
INFO@HALSOFTWARE.IE

TABLE OF CONTENTS

1 STUXNET FALLOUT

1.1 ITS BEEN 4 YEARS SINCE STUXNET..WHATS CHANGED?

If all the IoT Cybercrime articles were to be believed, then impending IoT hackers are going to destroy factories and make cars crash..according to the Economists special report on Cyber-Security (July 2014)

Stuxnet & Saudi Aramco Shamoon happened in 2010 and not much else had happened ..until June 2014 and HAVEX RAT (remote access Trojan) . HAVEX is targeted at industrial control systems and manifested itself as trojanised software available for download from control system vendors. (http://www.f-secure.com/weblog/archives/00002718.html)

Happily, outside of Stuxnet ,Shamoon and HAVEX, there has been nothing further to report on the industrial automation cyber security virus front ,other than the fact that there are a lot of IoT industrial cyber security experts peddling their rehashed wares, trying to cash in on clients fears, 'Y2K' style. The scare mongering needs to stop. The real solutions need to start. Stuxnet was most likely a government backed virus that found its way onto a physically isolated network by some clandestine action involving a usb stick and Siemens step 7 software.. The lessons here are .. Access control & cyber security as per usual (implement the SANS institute top 20) to prevent unauthorised devices, and a re-appraisal of the security risks associated with each and every software control application (This paper concentrates on Microsoft technology) . And a removal / replacement of those applications found not to be secure.

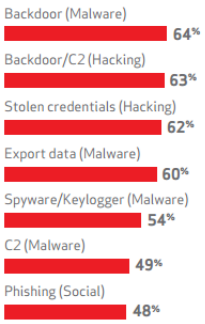
The Verizon data breach (2014) report outlines the following nine main mechanisms of general cyber attack, five of which are of particular relevance to manufacturing:

- Cyber Espionage
- DOS attacks
- Web App attacks
- Insider misuse
- Physical theft & loss

The other 4 modes of attack are less relevant to Enterprise IoT; Payment card skimmers, Crimeware, Point of Sale intrusions, miscellaneous errors.

The report does not mention any specific manufacturing hacks, or any hacking methodologies unique to control systems. As can be seen from the table below, according to the report cyber espionage appears to be the main objective of the attacks. There is not yet a category entitled 'remote equipment control / destruction' for the likes of HAVEX Trojan. Either Industry is doing a good job of preventing remote takeover of their equipment, or hackers are not yet interested, or the systems are presently unhackable.

Hacking mechanisms



INDUSTRY	POS INTRUS-ION	WEB APP ATTACK	INSIDER MISUSE	THEFT/ LOSS	MISC. ERROR	CRIME-WARE	PAYMENT CARD SKIMMER	DENIAL OF SERVICE	CYBER ESPION-AGE	EVERY-THING ELSE
Manufacturing [31,32,33]		14%	8%	4%	2%	9%		24%	30%	9%

(From: http://www.verizonenterprise.com/resources/factsheets/fs_dbir-industries-manufacturing-services-technology-threat-landscape_en_xg.pdf) & from Verizon DBIR 2014 report: rp_Verizon-DBIR-2014_en_xg.pdf

Note C2: command and control servers. For eg authentication servers in a domain.

The basics of securing systems: Here are the top 5 security controls as outlined by SANS (http://www.sans.org)

SANS - Critical Security Control	SANS - Critical Security Control Description	HAL Software Comment
Inventory of Authorised & unauthorised devices	Reduce the ability of attackers to find and exploit unauthorised and unprotected systems.	Minimise Auto discovery of devices
Inventory of authorised and unauthorised software	Identify vulnerable or malicious software to mitigate or root out attacks	No software that does not meet MILS equivalent security specifications.
Secure Configurations for Hardware & Software on Laptops, Workstations, and Servers	Prevent attackers from exploiting services and settings that allow easy access through networks and browsers	Ensure these images meet MILS equivalent security specifications.
Continuous Vulnerability Assessment and Remediation	Proactively identify and repair software vulnerabilities reported by security researchers or vendors:	Automation software has not been vulnerability tested to anything like the same extent as web facing software.
Malware Defenses	Block malicious code from tampering with system settings or contents, capturing sensitive data,	Manufacturing Enterprise implements antivirus security. Not all software is tamperproof however. .Dlls & .Exes should be digitally signed and verified.

1.2 EXECUTIVE SUMMARY

Hi risk Control systems technology should not be exposed to Public Internet level threats. Low risk control related systems (for e.g. a Historian, or SCADA system) need to be on the same platform as proven web technology before it can be public Internet facing. The applications themselves may need to be upgraded / rebuilt from the ground up as verifiable code and implementing best practices such as the U.S. SKPP or EURO-MILS equivalent specification. Secure inter controller communications with OPCUA needs to be trialed for performance latency. Internet authentication frameworks such as Oath2.0 may have a place for securing inter-enterprise domain traffic.

Upgrade your systems, or don't open them up to enterprise IoT. Security is only as good as the weakest link.

1.3 GENERAL IT CYBER SECURITY V INDUSTRIAL CONTROL CYBER SECURITY; A MISALIGNMENT

Enterprise Automation & IT is run and has been run for decades by professional IT and Automation managers that understand basic IT security and system integration and have the necessary skills and knowledge to prevent the much vaunted Hollywood Die Hard 4.0 Cyber disaster scenario from ever happening on isolated networks. However, guidance from the industrial automation industry is somewhat fragmented and scant in regard to its approach to improving cyber security of its control systems. On the one hand, Idaho National Labs are doing valuable SCADA destructive hacking/testing, and the Aviation industry is forging in-depth secure real-time software development standards (e.g. EURO-MILS). On the other hand ISA (Instrument Society of America) is peddling the same old best practices for accommodating control systems within a secure IT infrastructure, instead of providing a vision as to how control systems should be developed and future proofed, and having no regard to the fact that old software systems could well be full of vulnerabilities.

The ISA white paper called 'The Industrial Cybersecurity Problem' lays out what is probably a good indicator of where the traditional industrial automation industry is , on this subject presently ; out of date. No mention of end to end encryption and its industrial control implementation problems (3rd party key authentication may not be fast enough), and no mention of re-engineering control applications to accommodate secure software development best practices. The reality is, software architecture goes out of date too.

1.4 WHEN IS SECURITY SO COMPLEX AS TO BE UNUSEABLE?

In preparation for this article, I attempted to access the ISAS99 security papers on isa.org. This process tells its own story;

- The latest version of Adobe reader must be installed on my windows 7 PC– This is software with known security vulnerabilities.*
- Adobe reader needed to be enabled as a plug-in, in Google chrome, and the default pdf reader had to be disabled.
- As instructed by ISA.org IT support, I disabled JavaScript global security object protection in Adobe reader. A piece of software which has been used during hackathons, to break out of the sandboxed web browser environment and take control of the PC.

And all this, in order to get access to a security standard in ISA. In the end , I still couldn't access the standard securely and they just emailed me a copy. The security was so complex as to be unusable in my case. This scenario is familiar to the readers and typical of just how complex, the software inter-web has become. Google Chrome is free open source software. Adobe reader is free proprietary software. Windows 7 is paid proprietary software. The secured ISA documentation management system is paid proprietary software. Internet based software system interactions often appear to be a mix of paid and free software. This has implications for Enterprise IoT. Whether or not software is free should be irrelevant. It needs to support the same level of software quality and security assurance before being allowed on the enterprise control system network.

*(http://www.pcworld.com/article/2154940/adobe-patches-critical-flaws-in-reader-acrobat-flash-player-and-illustrator.html)

2 IT ACCESS CONTROL – BUSINESS AS USUAL

Industrial automation control systems are typically firewalled off from main IT business network, with multihomed SCADA/Batch/Application servers acting as the bridge between office IT network and the factory floor. This is largely to facilitate server backup and restore and authentication to the company ADS tree. Yes the Office IT network has internet connectivity..via an authenticated DMZ gateway proxy server / firewall; You have to be logged in and authorised in order to get out onto the web.

Security has developed substantially since the author qualified as a Certified Information Systems Security professional (CISSP) in 2002, but the basics of Access control are still the same;

Identification. Authentication. Authorisation.

Diligence will always be required to ensure that users are members of the correct group and that group has the correct permissions. But that's something that Corporate IT are already good at. OPCUA provides the mechanism to extend this diligence throughout the internet exposed enterprise, if necessary.

On the other hand, the basics of software application architecture are not the same as they were in 2002, having evolved substantially..

2.1 DEVELOPMENTS IN SECURE SOFTWARE ARCHITECTURE; ALL CHANGED, CHANGED UTTERLY.

The basics of secure enterprise software application development have changed substantially in the last decade, notably (in the non consumer sphere) as a result of Microsoft .NET architecture. (Similar Java technology frameworks such as Tridium Niagara, are not discussed here) Software architecture has metamorphosed from windows presentation foundation or windows forms, to ASP .NET web based MVC (model view controller).A program developed in C# for .NET could be deemed to be secure and capable of being monitored correctly by Windows OS processes, but the same cannot be said for older Win32 software technologies (unmanaged software).

The diagram below summarises where and how access control may be implemented and enforced throughout the software domain.

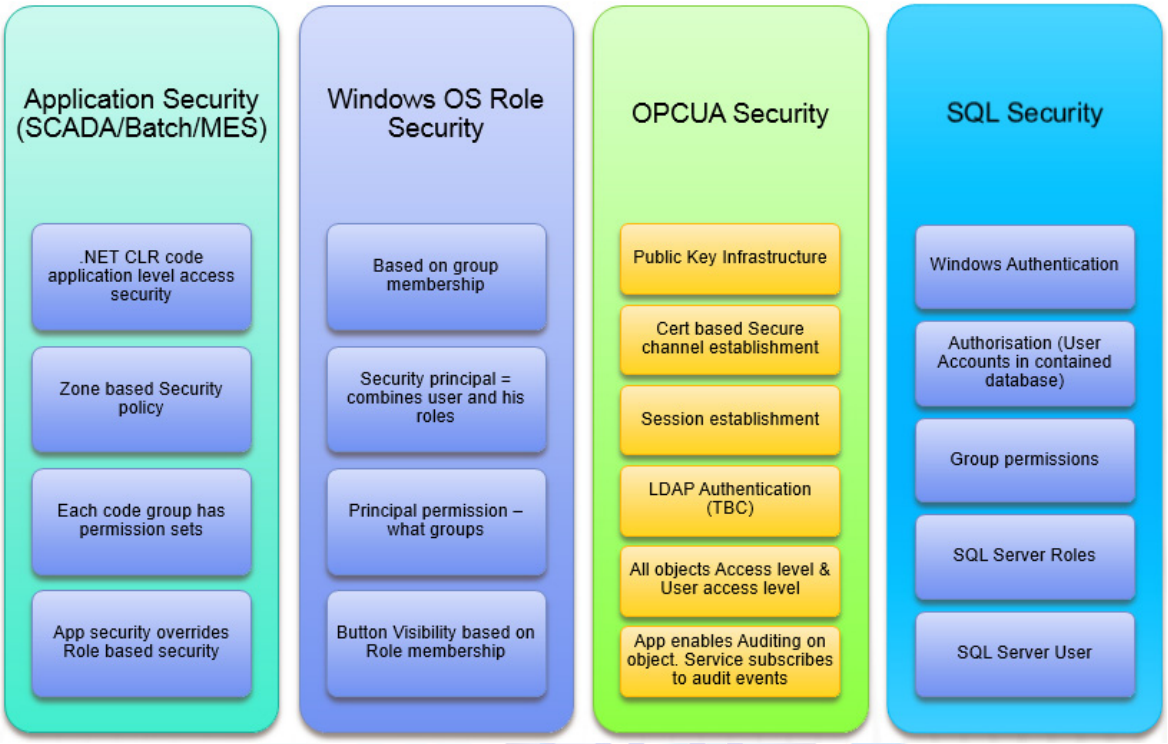


Figure 2: The 4 main areas of application, data , communications and operating system security for industrial automation.

As outlined in diagram above, there are a lot of areas where application, operating systems and directory services security configurations can interact in a way that is unclear to IT administrators, possibly resulting in security vulnerabilities.

2.1.1 'TOO MANY COOKS IN THE KITCHEN' ** THE SOFTWARE IS TOO COMPLEX TO PATCH

> *"As multiple parties (chip manufacturer, motherboard manufacturer, BIOS developer, device manufacturer, etc.) attempt to add value, the product will be subject to orthogonal feature creep. Put together, these extra features create complexity. In order to facilitate interoperability, additional layers of interdependency are added, requiring trust in multiple parties. It is understandably difficult, for example, for one organization to understand, verify, and correct bugs that have been created by another organization, especially since the standardized abstraction does not necessarily represent the real hardware " ***

** NSA Systems and Network Analysis Center Information Assurance Directorate paper on "Separation Kernels on Commodity Workstations 11 March 2010

The size of the task for assuring control system platform security is huge; Aligning multiple Software and hardware vendors to the level of granularity required, to implement a MILS level secure system is impossibly challenging without a single universally accepted security standard. Modern software development, like modern hardware, is also built from 3rd party components. It is not uncommon for a modern ASP .NET application to utilize several JavaScript libraries for the presentation layer, a further few C# libraries for the domain logic , C++ unmanaged .DLLs for the real-time control, and finally, a 3rd party database for the data layer. And some of this software runs on an unknown client front end (The web browser could be Chrome or Internet explorer or Mozilla).

In fact, the NSA recommends against desktop workstation technology entirely , for highly robust environments;

*"For high robustness environments, where a highly capable and motivated adversary is expected to devote significant resources toward compromising the system, it is recommended that complex platforms such as desktop workstations NOT be relied upon. A highly capable adversary is expected to research platform vulnerabilities in commodity workstations, and the platform security of this technology is subject to significant complexity, which stands in the way of robust assurance arguments. Improvements to platform security technology should continue, and as it matures, this technology may be appropriate for many practical situations. However, without significant simplification of the platform, it does not appear that the security arguments of this technology can justify high robustness. Therefore, these systems are not appropriate for certification against the SKPP" ***

Releasing software (that runs control system platforms) as open source, in order to get the community to flush out vulnerabilities (via hacking contests) is another option. Disgruntled ex-employees are good at exposing obfuscated backdoors into control systems. The good hacker PinkiePie exposed a serious security flaw in Google chrome . (http://www.techtimes.com/articles/1489/20131119/pwn2own-2013-hackers-expose-vulnerabilities-ios-android.htm)

"The exploit took advantage of two vulnerabilities-an integer overflow that affects Chrome and another Chrome vulnerability that resulted in a full sandbox escape," HP's senior security content developer, Heather Goudey said about the Pinkie Pie hack. "The implications for this vulnerability are the possibility of remote code execution on the affected device."

In fact, browser vulnerabilities are being exposed with alarming regularity. Very worrying , considering a lot of industrial control applications, including SCADA applications (For e.g. Thingworx or Inductive Automation products) are migrating

towards a web browser client front end in order to save licensing costs. Here an excerpt from the HP run pwn2own 2014 competition:

The following vulnerabilities were successfully presented in the Pwn2Own competition:

By Jüri Aedla: Against Mozilla Firefox, an out-of-bound read/write resulting in code execution.
By Mariusz Mlynski: Against Mozilla Firefox, two vulnerabilities, one allowing privilege escalation within the browser and one bypassing browser security measures.

By Team VUPEN:
Against Adobe Flash, a use-after-free with an IE sandbox bypass resulting in code execution.
Against Adobe Reader, a heap overflow and PDF sandbox escape, resulting in code execution.
Against Microsoft Internet Explorer, a use-after-free causing object confusion in the broker, resulting in sandbox bypass.
Against Mozilla Firefox, a use-after-free resulting in code execution.

(source: http://www.pwn2own.com/2014/03/pwn2own-results-for-wednesday-day-one/)

2.2 HOW THE IT SECURITY DEPARTMENT PREVENTS HACKING ; BACK TO BASICS.

Here is one definition of a security hack;

"The goal of most security attacks is to gain unauthorized access to a computer system by taking control of a vulnerable privileged program. This is done by exploiting bugs that allow overwriting stored program addresses with pointers to malicious code. Today's most prevalent attacks target buffer overflow and format string vulnerabilities."

(http://groups.csail.mit.edu/commit/papers/02/RIO-security-usenix.pdf)

The IT department endeavours to prevent unauthorized access to systems by locking down user , machine and program rights as much as possible , and preventing users from doing anything that could compromise the IT infrastructure, such as installing unproven uncertified untrusted software . It does this be removing install rights from a normal user. This act alone, would have prevented the HAVEX Trojan being installed ,and goes a long way towards justifying the often draconian security restrictions on the modern day corporate IT desktop.

2.2.1 PRINCIPAL BASED SECURITY

The classic principal based security model of MS Windows, focuses on the rights of the user running the code. Most modern hacker attacks are based around impersonating a current system user to execute malicious programs.

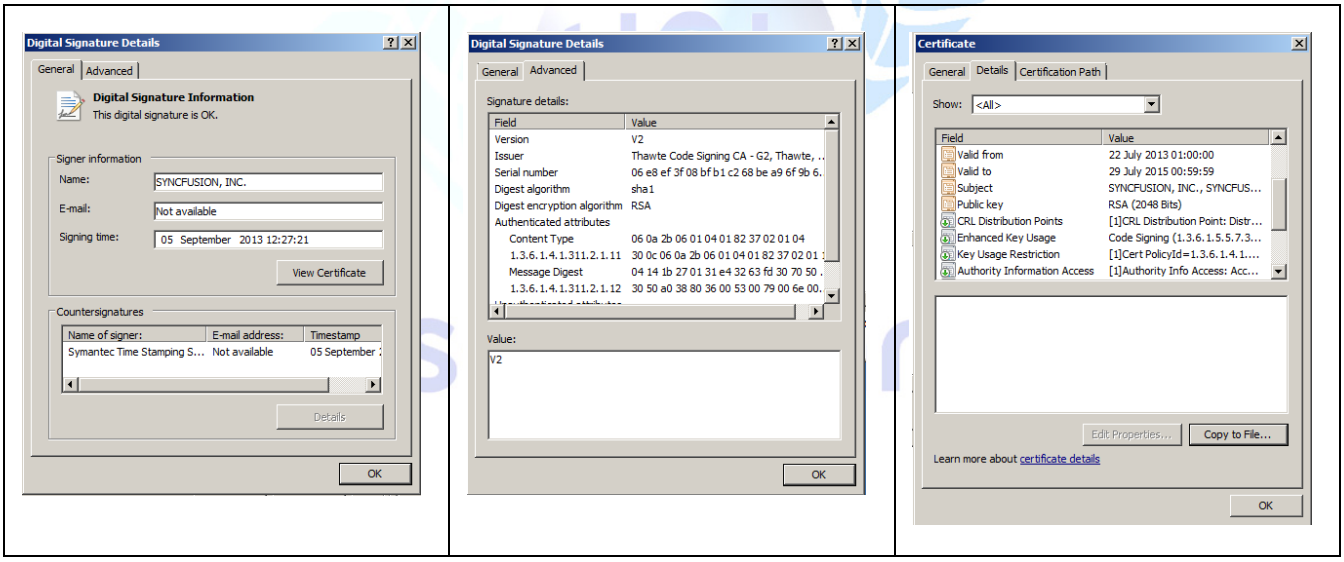
2.3 MICROSOFT .NET SECURITY MODEL DEEP DIVE

(Source for this section: MSDN.Microsoft.com)

The .NET Framework provides a run-time environment called the common language runtime, which runs the code and provides services that make the development process easier. (http://msdn.microsoft.com/en-us/library/8bs2ecf4(v=vs.110).aspx)

During program startup, the Windows operating system loads the executable into a new process address space. The primary process thread calls MSCoreEE.dll to initialise the Microsoft Common Language Runtime (CLR) and load the .exe assembly, then finally calls the program entry point (MAIN). At this point, the managed application is up and running. Managed assemblies need CLR to execute . They take advantage of DEP and address space layout randomisation (ASLR) in Windows. These 2 features improve application security, reducing vulnerability to hijacking. Modern Microsoft C++ source code can actually be compiled as a managed assembly also.

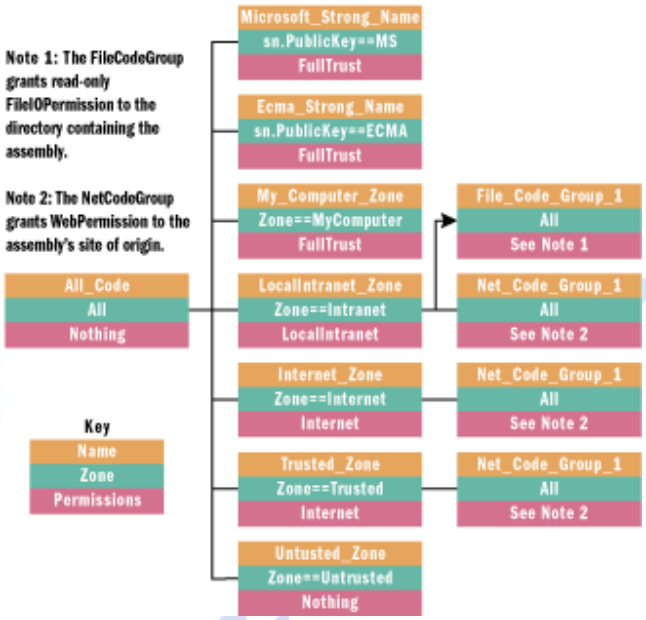
A 'strongly' named assembly (similar to .dll or .exe file) is signed with public/private key pair that uniquely identifies the assemblies publisher. It allows CLR to implement specific code block security. In addition, a hash checksum helps determine if file has been altered. Strongly named assemblies are tamper resistant. The CLR Appdomain concept provides further separation between processes.(virtual address space per application. No direct physical memory access) . ASP.NET creates a new Appdomain per web application.



Make sure all the application .DLL & .Exe files have valid digital signatures

In terms of security and hacker vulnerability, managed code is more secure than unmanaged code.

Microsoft Windows .NET has two security models called code access security, and transparency. Starting with .NET4, transparency is the default model. Transparency is an enforcement mechanism that separates code that runs as part of the application from code that runs as part of the infrastructure. The primary goal of transparency enforcement is to provide a simple, effective mechanism for isolating different groups of code based on privilege.. Details are beyond scope here, but a simplification is ; There must be proof that the code being called is from a safe source and that the user has the rights to call an individual method; The assembly must have an associated safe, trusted public certificate.



Figure; CLI Code categorisation (from MSDN)

The Common Language Infrastructure (CLI) categorizes code into two broad families: verifiable code and non-verifiable code. C# is verifiable code.

ECMA-335 common language infrastructure outlines a specification for the managed code manifest (cryptographic hashes & digital signatures) (http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-335.pdf)

Access rights for applications are determined by two factors: their permissions (the grant set established by their application domain) and their transparency. All partial-trust applications are classified as transparent. Transparent applications do not have to be concerned with security. The Microsoft .NET Framework offers security transparency, code access security and role-based security to address security concerns during software development

2.3.1 .NET SECURITY TRANSPARENCY MODEL

Starting with the .NET Framework version 4, security transparency is the default enforcement mechanism. The common language runtime (CLR) is moving away from providing security policy for computers .Security transparency separates code that runs as part of the application from code that runs as part of the infrastructure. In the .NET Framework 4, machine-wide security policy is turned off by default. Applications that are not hosted (i.e. applications that are executed through Windows Explorer or from a command prompt) now run as full trust.

Historically, the .NET Framework has provided code access security (CAS) policy as a mechanism to tightly control and configure the capabilities of managed code. However Aspects of CAS have been deprecated as they were considered too complex. And CAS policy does not apply to native applications, so its security guarantees are limited. System administrators should look to operating system-level solutions (for e.g. AppLocker) as a replacement for CAS policy. SRP (software Restriction Policies) and AppLocker policies provide simple trust mechanisms that apply to both managed and native code. As a security policy solution, SRP and AppLocker are simpler and provide better security guarantees than CAS.

The primary goal of transparency enforcement is to provide a simple, effective mechanism for isolating different groups of code based on privilege. Within the context of the sandboxing model, these privilege groups are either fully trusted (that is, not restricted) or partially trusted (that is, restricted to the permission set granted to the sandbox).

The three tenets of the .NET transparency model are ; transparent code, security-safe-critical code, and security-critical code. (Each software application code assembly is categorised as one of the above)

Security transparency separates security-sensitive code from non-security-sensitive code. Transparent code is not security critical and cannot perform certain actions.

Security-safe-critical code is fully trusted but is callable by transparent code. It exposes a limited surface area of full-trust code. Correctness and security verifications happen in safe-critical code.

Security-critical code can call any code and is fully trusted, but it cannot be called by transparent code.

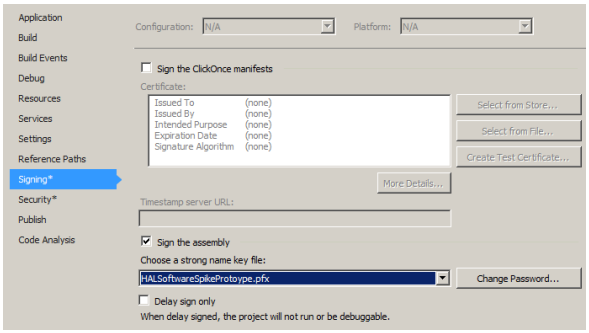
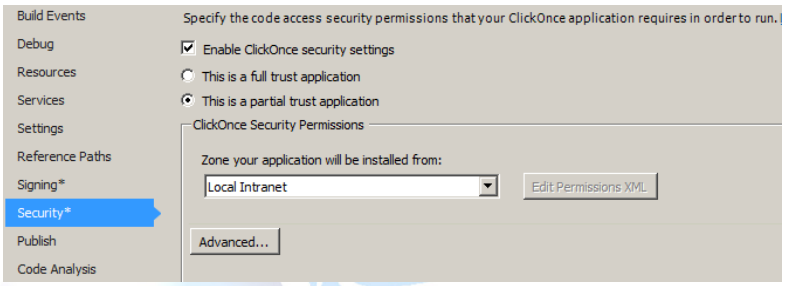
The .NET security transparency model has similarities with a new embedded real-time system architecture called MILS which enables multiple software components with different security levels to share the same platform.

Hosted or sandboxed applications continue to run with trust policies that are decided by their hosts (for example, by Internet Explorer, or ASP.NET). Applications or controls that run in sandboxes are considered partially trusted.

2.3.2 STRONG ASSEMBLIES IN C#/.NET AND VISUAL STUDIO 2013

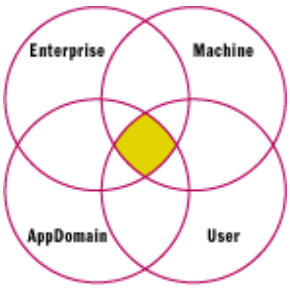
.Net assemblies area assigned a four part numerical version number of the form:

<major>.<minor>.<build>.<revision>. If the assembly also provides public key information, then it is strongly named. A strong assembly also has an embedded digital signature created using a hash of the contents and the private key value.

It is considered best practice to strongly name ALL .NET assemblies. Any assemblies that need to be shared between applications will need to be installed to the GAC (Global assembly cache) and that task requires administrator installation rights.	
Web Browser based programs can be partial trust applications, with limited rights. In addition, ClickOnce applications use certificates to verify authenticity of the application publisher and to sign application files. This reduces tampering possibilities. However the developer has to work a lot harder to ensure that the application will run with the reduced permission set.	

Locking down application rights and permissions requirements at application design time is not easy and the available tools can be cryptic to understand. An Automation application security standard could help in this regard. Either way, all program components should be digitally signed.

2.4 INDUSTRIAL AUTOMATION PC SYSTEMS AND SCADA APPLICATION REALITY



(from http://msdn.microsoft.com/en-us/magazine/cc188939.aspx)

Much current enterprise control system software is not .NET / CLR based, so a blanket code access security policy is not feasible except for new installs. That is, demanding that all assemblies for all installed industrial control software are from a trusted source. Microsoft .NET was only introduced by Microsoft in 2002. Oftentimes, the operating system DEP (data execution prevention) must be disabled before industrial control applications can even be installed on the Microsoft windows platform.

Even more importantly, most Windows / PC based real time control systems are programmed using C++, resulting in 'unmanaged code' from a security perspective. No control systems run on the CLR .(Real-time control is not possible/recommended with managed software – where memory allocation is automatically managed and application task duration is indeterministic). All of these aging SCADA, Batch, Historian, DCS, MES applications do not implement a code access / code transparency security model to the same extent as CLR. If a company is serious about Enterprise IOT, then all of these legacy applications (in terms of software security implementation) are going to have to be upgraded / replaced, with their non real-time critical aspects being ported to a platform such as CLR.

2.4.1 EMBEDDED REAL TIME CONTROL SYSTEMS

Upgrading of embedded real-time control platforms (PLC operating systems such as Wind River or CoDeSys or Windows embedded) to support MILS level security is a key Enterprise IoT concern.

Embedded control system suppliers are becoming aware of their own product limitations. Wind River VxWorks MILS platform has been designed to address the new embedded systems security need. (http://www.windriver.com/products/platforms/vxworks-mils).

Other vendors explicitly state that their equipment is not hardened against cyber attack, so should not be exposed to one. From CoDeSys website: (http://www.codesys.com/details/article/sicherheitsluecke-in-codesys-v23-laufzeitsystem.html)

> *"Security Vulnerability in CODESYS V2.3 Runtime System*
>
> *Kempten, October 2012: Password protection bypass for CODESYS controllers*
>
> *A security vulnerability which affects the CODESYS V2.3 Runtime System is currently being discussed on several different internet platforms: The password protection of a publicly accessible CODESYS controller can be bypassed with the help of an external tool. A password protected controller can then be accessed just like any unprotected PLC and it is possible to execute commands with the controller shell or load applications.Of course, we take this issue very seriously. A fix version which resolves the reported vulnerability is now available for download for our direct OEM customers. In general, we do not offer any standard tools in CODESYS which are to protect the controller from a serious cyber attack. Should the offered password functionality suggest such a protection, this was definitely not our intention. The implementation of standard security mechanisms (firewall, VPN access) is an absolute must when operating a PLC runtime system on a controller accessible through the internet."*

2.5 COMMERCIAL AND STANDARDISED SECURITY SOLUTIONS FOR INDUSTRIAL AUTOMATION

MatrikonOPC provide role based security solutions for the older version of OPC (OPC-DA) which can implement group or role based security right down to the tag level. They also provide encrypted tunneling of OPC communications.

The latest specification of OPC, OPCUA, has built in native encryption and tunneling capability and the ability to independently configure security on each OPCUA object. OPCUA can support PKI. It is up to the OPCUA driver supplier to implement this functionality.

The new Oath 2.0 standard sounds promising for the data security conduits (see S99 section) but the standard has its detractors, there are multiple variants ,and token management is required. There are a lot of flavours of oath implementation.

API metrics has calculated the average latency of around the 300ms range for web based Oath authentication (to return a token).Oath security would need to be trialed on an industrial LAN to see if this could be improved.

(http://apimetrics.io/2014/04/09/oauth-standard-isnt/)

3 NEXT GENERATION SECURITY REQUIREMENTS OF INDUSTRIAL AUTOMATION APPLICATIONS.

Why aren't control systems being implemented with verifiable code? It's an epic challenge for a company to undertake to upgrade all its control systems to this level of application security. As a project , it is bigger than Y2K. Maybe the company is happy to accept the risk ,as to upgrade all systems would be too costly.

If true Enterprise IoT is required of the business, and all industrial control systems must be opened up resulting in increased threat exposure, then the applications themselves may need to be upgraded / rebuilt from the ground up as verifiable code and implementing best practices from those mentioned both here and the EURO-MILS spec.

3.1 THE EASY TO IMPLEMENT REQUIREMENTS

- Applications should be properly vetted during FAT to ensure their ability to authenticate to external active directory services (ADS) , and their ability to confine a users rights to that of the application. The 'separation of concerns' principle advocates that there should be minimal security settings stored on the application server, but instead the settings are linked to groups in the ADS server.
- Sandbox (i.e. make web browser front end , a requirement) as many applications as possible so that they are isolated from underlying infrastructure and are running software whose vulnerabilities have already been exposed.
- Harden the network by locking it down through restrictive policies (for e.g. using Applocker on windows 7).
- A user should have the minimum rights in order to do his job. Only approved software from the IT department should be installed. Encrypt stored data and data communications.
- The PCI security standards council are a great general guide to network security implementation (www.pcisecuritystandards.org).

3.2 THE HARD TO IMPLEMENT REQUIREMENTS

- The industrial automation industry needs an equivalent to PCI that includes the following additions on guidelines ;
    - Regular Certificate expiration best practices.

- Certificate validation optimisation for control networks.
- Self Certificate management guidelines.
- Device to device (controller to controller) encryption guidelines for real-time fast key negotiation. We have OPCUA for this. Now the network latency must be determined. It may be that Oath authentication can be used for some controller to historian data security conduits, and OPCUA for controller to controller data conduits.

- DEP (data execution prevention) compatibility:

  The application should be compatible with operating system DEP (data execution prevention) enabled. This is a break from the present approach whereby the IT security infrastructure is typically modified to accommodate the SCADA applications .

- Disabling 'AutoUpdates' needs a risk assessment.

  Understand the risks associated with Operating system 'autoupdate' disabled, or antivirus signature software auto update disabled. A balance between vigilance and control system reliability must be struck. Guidelines needed.

- Applications must Implement Security transparency (.NET 4) or the MILS equivalent.

  A CLR equivalent code access security /transparency model to be implemented.(Or 'program shepherding' as per Ref 1 MIT article).Software to be verifiable code as defined in Common Language Infrastructure (CLI).

- Change from Windows based to Web browser based control applications. Web browsers have been (and continue to be) community tested for vulnerabilities. Remove WPF, encourage 3 tier web architecture such as ASP .NET MVC. See Sandboxing;

3.3 HAL SOFTWARE SPIKE AND ITS APPLICABILITY TO ROBUST SECURITY DESIGN

The HAL Software 'Spike' application can aid the development of an process control security reference model .

(A reference model is a framework for understanding significant relationships among the entities of some environment, and for the development of consistent standards or specifications supporting the environment. A reference model is based on a small number of unifying concepts and may be used as a basis for educating and explaining standards to a non-specialist.)

3.4 MIGRATION TO WEB BROWSER APPLICATION & ARCHITECTURE DESIGN AND SANDBOXING

Programs that are run from within a browser are typically well sandboxed (i.e. there is a level of security isolation between the application , which runs in a browser, and the underlying operating system) and thus Web SCADA may be a way to leapfrog from the old SCADA application security model to the latest internet hardened sandboxed security model.

For e.g., Microsoft Silverlight technology runs in its own special limited security sandbox appdomain.

Likewise, Microsoft ASP .NET creates a new Appdomain per web application.

Another example is the Chromium project. (http://www.chromium.org/developers/design-documents/sandbox)

You still have to get the access security right, but at least you know that a community tested compromised sandbox can't access the underlying hardware.

Internet facing web servers and their associated DMZ firewall technology and hardened authentication process and database have been at the coal face of hacking attempts for the past 10 years. Many vulnerabilities have already been exposed. ASP.NET MVC and its equivalent java frameworks have been designed around security. If an online Bank or Amazon web servers can survive hacking, then that technology has proven itself in the field. Non web based SCADA , and older generation WPF (windows presentation foundation or windows forms technology) applications are simply not hardened enough to be exposed to internet. There exists too great a risk of vulnerability.

3.4.1 SANDBOXING AND THE .NET SECURITY PERMISSION MODEL

Security involves three interacting pieces: sandboxing, permissions, and enforcement. Sandboxing refers to the practice of creating isolated domains where some code is treated as fully trusted and other code is restricted to the permissions in the grant set for the sandbox. The application code that runs within the grant set of the sandbox is considered to be transparent; that is, it cannot perform any operations that can affect security. The grant set for the sandbox is determined by evidence (the Evidence class). Evidence identifies what specific permissions are required by sandboxes, and what kinds of sandboxes can be created. Enforcement refers to allowing transparent code to execute only within its grant set.

The following list describes the trust model for desktop and hosted applications in the .NET Framework 4. For more information, see Security Changes in the .NET Framework 4.

Desktop applications. As in previous versions of the .NET Framework, managed applications that reside on the desktop (unless they have been downloaded from the Web) are granted full trust.

Hosted applications. Applications that run in a sandbox (for example, Silverlight-based applications) are granted a limited set of permissions that determine which computer resources they can access (for example, which files they are allowed to use). Sandboxes provide the ability to identify some assemblies within the sandbox as being partially trusted and some as being fully trusted. The partial-trust assemblies are granted a specific set of permissions, as determined by the application domain (System.AppDomain) that created the sandbox. Some of the full-trust code in the full-trust libraries can be called by partially trusted code. That trusted code can make calls to protected resources on the computer. However, publicly accessible full-trust types and members that can call into protected resources must have undergone a security audit.

(http://msdn.microsoft.com/en-us/library/ff527276(v=vs.100).aspx)

Microsoft recommends that evidence-based security and code access security be used to implement application security. Most application code can simply use the infrastructure implemented by the .NET Framework. It remains to be seen whether the EURO-MILS specification and .NET security transparency model are aligned.

Recommendation:

> *USE APPLOCKER TO LOCK DOWN THE DESKTOP.*
>
> *USE .NET SECURITY TRANSPARENCY TO LOCK DOWN APPLICATION DEVELOPMENT.*

HAL Software Spike is a real-time web browser / web server based Automation prototyping application. It can provide an introduction to the group rights requirements, and aid visualization of areas of greatest risk.

3.4.2 OPEN SOURCE SOFTWARE SECURITY RISKS

Proprietary software by its nature, is not open source and provides a level of obfuscation making disassembly harder. It is arguably more difficult to tamper with, than open source software and the source is controlled by a single vendor and single quality assurance system, in theory. Though interpreted languages such as C# are easy to disassemble, Microsoft Visual Studio provides plenty of digital signing tools to prevent assembly tampering. Also , as the vendors reputation is at risk, the expectation is that proprietary software goes through a more rigorous quality assurance process. However there are not yet widely accepted 'standardised' software security quality assurance processes (MILS standards are gaining traction)

By contrast , open source software can be examined and in a lot of cases extended / recompiled / forked and re-issued by anyone. Its vulnerabilities can also be challenged by anyone, then published to the community or not published, depending on the programmers motive.

In modern software development, the dividing line between open source and proprietary is increasingly merging . A proprietary C# application can easily call functionality in an open source java assembly (.NET interoperability) . It is not unusual to utilise open source extension libraries to extend a commercial product. For e.g. any web browser based SCADA software that utilises jquery is using an open source JavaScript library. In fact , most modern applications utilising internet based communication , are a combination of open source and proprietary software.

Whether proprietary software is more secure than open source software is an entirely subjective opinion. But the pervasiveness of the internet makes obfuscation immeasurably more difficult, and open source software usage may in fact be increasing. The Microsoft C# compiler is now available open source. Java Eclipse development environment has always been open source.

New industries have been quicker to move to open source software models and web based SCADA, than traditional older industries. For e.g., the wind power turbine industry is comfortable with java platform for distributed SCADA. In addition, water treatment , with its distributed location requirements is comfortable with web enabled SCADA platforms.

3.5 ISA-S99 SECURITY FOR INDUSTRIAL AUTOMATION AND CONTROL SYSTEMS

Here is a brief discussion of the content of these three standards and their relevancy to modern data security from the shop floor to the top floor.

3.5.1 ISA99.01.01 - TERMINOLOGY, CONCEPTS, AND MODEL

There is nothing in this standard that is not already security implementation best practice by a corporate IT group. The emphasis may be different, but the solutions remain the same (confidentiality, integrity, availability -C.I.A). A methodology for risk assessing security of systems (SL - security level) is outlined. There is some mention of rudimentary application level security and security 'zones' or physical / logical areas of demarcation. (Implemented as groups in Active directory services, or possibly even domains within a forest). Inter zone data communication is demarcated with security 'conduits' with its own unique security policies.

The data security conduits may not be amenable to being secured by OPCUA , but may be amenable to Oath style security.

A number of modern web applications (Google, Face book, Twitter) are secured with custom implementations of Oath authentication Web API. As the website APImetrics puts it;

'The idea behind OAuth is simple – a standard way to enable access to protected resources (like a social network), so a user can sign into a web service, and, via a secure exchange of tokens, an app or 3rd party can access that service. All happening without the developer ever seeing the user's username or password. However, in reality, there is a lot of variation between implementations and many traps for the developer.' *(http://apimetrics.io/2014/04/09/oauth-standard-isnt/)*

None of the S99 standards discuss Oath and its suitability for securing non real-time conduit data such as historian data. An automation system trial would be beneficial. Oath could be deemed overkill for data conduits within the Enterprise network, but could be ideal for inter- Enterprise communication.

3.5.2 ISA99.02.01 - ESTABLISHING AN INDUSTRIAL AUTOMATION AND CONTROL SYSTEMS SECURITY PROGRAM

This standard defines the elements necessary to establish a cyber security management system (CSMS) for industrial automation and control systems(IACS) and guidance on how to develop these elements.

It resembles the project management body of knowledge as applied to a system wide security upgrade implementation complete with risk assessments, stakeholder meetings ,leadership buy in etc.

It mentions personnel segregation of duties to maintain appropriate checks and balances ,which is a pillar of security implementation.

This is similar to the 'Chinese walls' **concept from banking security. Unfortunately , reduction in manufacturing staffing numbers is making this increasingly difficult.

**A Chinese wall is an information barrier implemented within a firm organization to prevent exchanges of information that could cause conflicts of interest. For example, a Chinese wall may be erected to separate and isolate persons who make investment decisions from persons who are privy to undisclosed material information which may influence those decisions. .Financial Firms are generally required by law to safeguard insider information and to ensure that no improper trading occurs.

The main thrust of the document is physical or logical system access control;

- Isolate security zones from one another with firewalls/routers.
- Authentication and authorisation. Some samples of waste water systems attacked by disgruntled employees with the correct remote access software are provided.

It notes that risk assessments for traditional corporate IT focus on C.I.A., while control systems risk focuses more on health & safety (HSE) factors, and operational reliability.

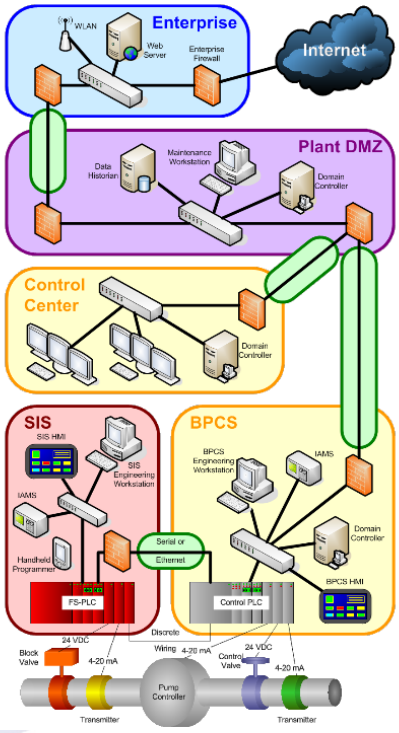
It recommends that wireless not be used for high risk control systems.

The standard notes that older generation device control systems may need to be replaced with newer devices with improved SL (security capability). In addition it provides a methodology for enforcing this, based on the zone security level; All devices within a security zone should support that zones SL.

It notes that some hardening measures on a server cannot be undertaken because the SCADA application vendor will not support the application with the new server configuration. In this case it recommends replacing the software with newer compatible software.

3.5.3 ISA99.03.03 - SYSTEM SECURITY REQUIREMENTS AND SECURITY LEVELS

S99-03 is the most recent standard , the most up to date, and the most useful in terms of new Enterprise IoT security software requirements . It delves into software application security;



(Diagram from : **ANSI/ISA-62443-3-3 (99.03.03)-2013)**

S99-03 talks about 7 foundational software security requirements .The table maps these to the 9 types of actual attack types as evidenced by Verizon)

S99-03	**Addresses this attack category (from Verizon DBIR 2014 major categories of attacks)**
1) Identification and authentication control (IAC),	Insider misuse, Cyber-espionage
2) Use control (UC),	Insider misuse, Crimeware, Cyber-espionage
3) System integrity (SI),	Crimeware
4) Data confidentiality (DC),	Physical theft and loss, insider misuse
5) Restricted data flow (RDF),	Crimeware, insider misuse, Cyber-espionage, Web app attacks
6) Timely response to events (TRE)	This is the ability to know that EIoT systems have been compromised.
7) Resource availability (RA).	DOS attacks

3.5.4 S99.03 CAPABILITY, TARGET, AND ACHIEVED SECURITY LEVELS.

S99.03 defines some security concepts;

- SL-T : target security level (four levels: SL-1, SL-2,SL-3,SL-4)
- SL-C : Capability security level (four levels: SL-1, SL-2,SL-3,SL-4)
- SL-A : Achieved security level (four levels: SL-1, SL-2,SL-3,SL-4)

Security level areas are also defined. A process similar to risk assessment is undertaken to ascertain an areas target v achieved security level.

In addition , it outlines some modern security practices for tightening up security such as 'least privilege';

'The capability to enforce the concept of least privilege shall be provided, with granularity of permissions and flexibility of mapping those permissions to roles sufficient to support it.

The control system shall provide the capability to identify and authenticate all users (humans, software processes or devices) engaged in wireless communication.'

S99.03 mentions that control systems go through unit testing , but this is not the authors experience. There is minimal use of external automated unit testing or test harnesses presently ,in industrial automation. PLC code or SCADA scripts do not presently allow for unit testing to the same extent as visual studio 2013 with its unit testing fakes assembly, for example. A new 'MILS' style software quality assurance program requirement that standardizes unit testing, could address this failing.

3.5.5 S99.03 ,MALICIOUS CODE PROTECTION, & MOBILE SOFTWARE TECHNOLOGIES

S99.03 references the hashing and digital signature of binaries as tamper prevention mechanism. It also references sandboxing, DEP, ASLR. It's the authors experience that DEP & ASLR are security approaches implemented by Microsoft and as of today, few controls systems have implemented this technology. A site wide security audit would clarify this.

S99.03 gives some guidance on modern software development:

'Mobile code technologies include, but are not limited to, Java, JavaScript, ActiveX, portable document format (PDF), Postscript, Shockwave movies, Flash animations and VBScript. Usage restrictions apply to both the selection and use of mobile code installed on servers and mobile code downloaded and executed on individual workstations. Control procedures should prevent the development, acquisition or introduction of unacceptable mobile code within the control system. For example, mobile code exchanges may be disallowed directly with the control system, but may be allowed in a controlled adjacent environment maintained by IACS personnel.' ANSI/ISA-62443-3-3 (99.03.03)-2013 – 40 – 12 August 2013

3.5.6 S99.03 AND CRYPTOGRAPHY

The standard show an awareness of PKI key management problems;

'The security policies and procedures for key management need to address periodic key changes, key destruction, key distribution and encryption key backup in accordance with defined standards'

'Where PKI is utilized, the control system shall provide the capability to operate a PKI according to commonly accepted best practices or obtain public key certificates from an existing PKI. When revocation checking is not available due to control system constraints, mechanisms such as a short certificate lifetime can compensate for the lack of timely revocation information. Note that short lifetime certificates can sometimes create significant operational issues in a control system environment.'

S99.03 notes that timely PKI revocation is a problem for fast controller to controller communications,(but offers no solution) . It also recognises the fact that mobile devices are a challenge and discusses;

- Preventing the use of portable and mobile devices.
- Requiring context specific authorization.
- Restricting code and data transfer to/from portable and mobile devices.

S99.03 does not explicitly discuss whether the OATH form of authentication is suitable for automation and control.

3.5.7 S99.03 AND CONTROL SYSTEM ISLAND MODE

S99.03 requires control systems to support an island mode or Fail close mode whereby all communications across control system boundary is stopped but the control system continues to function. This 'lockdown' mode should be a mandatory control system requirement in the authors view, and needs to be fully tested during SAT.

3.5.8 S99.03 AND SOCIAL APPS

S99 suggests denying face book/twitter style programs on the control network. These applications are 'chatty', creating a lot of network traffic and often having unsafe plugins available for them. Modern industry does tend to allow use of SharePoint messenger style chat programs.

4 WHAT OPCUA GIVES US: SECURITY SUMMARY

OPCUA enables secure communications between industrial automation systems and can run across the internet Security can be implemented right down to object level if necessary, and auditing is built into the protocol.

It uses PKI public key infrastructure using X509 certificates. OPCUA user asymmetric keys for connection establishment, and then faster symmetric keys for data exchange.

HAL Software Spike implements OpenSSL PKI, however 3rd party commercial PKI services (such as VeriSign) can also be used.

4.1 Disdvantages of OPCUA security;

- Regular Certificate expiration is a headache requiring increased IT maintenance.
- Certificate validation takes time. Perhaps too much time for a control network. So only non-time critical data can be exposed to the inherent delays associated with certificate validation, unless the industry comes up with a much faster form of certificate verification for industrial control applications. A trial of PKI OPCUA control system authentication latency similar to that performed by APImetrics for Oath standard, would be very useful.
- Self Certificate management takes resources. But there is always a risk that a 3rd party certificate management company could cease trading.

5 2014 INDUSTRY MOVES

Some embedded control system vendors such as Wind River corporation* are moving towards the complete software security implementation from software object up. Multiple Independent Levels of Security (MILS) programs to fend off growing security concerns are gaining momentum in Avionics. Hopefully the EURO-MILS initiative will address the above points . The initiative is being driven by the Aviation industry and is scheduled to end in 2015.

(MILS is a high-assurance security architecture based on the concepts of separation and controlled information flow. - http://www.euromils.eu/)

*For its part, Wind River introduced the latest VxWorks MILS Platform last year with multi-core features built in. - See more at: http://www.aviationtoday.com/av/topstories/Real-Time-Operating-Systems-Addressing-the-Certification-Security-and-Standards-Dilemmas_81672.html#.U8UICfldXTp

HAL Software is looking forward to see if this or EURO-MILS standard is also applicable to the Process, Chemical, Oil & Gas, Pharmaceutical industry.

5.1 REFERENCES & GLOSSARY:

- INL (Idaho national laboratories) offer advice on how to secure SCADA systems through their National SCADA test bed program.

http://www.inl.gov/research/national-supervisory-control-and-data-acquisition-test-bed/

- Ref 1: Secure Execution via program Shepherding:

http://groups.csail.mit.edu/commit/papers/02/RIO-security-usenix.pdf

- ECMA international software standards:

http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-335.pdf

- In depth common language runtime detail from the book Jeffrey Richter - CLR via C#
- MSDN: Authoritive Windows operating system documentation source.

http://msdn.microsoft.com/en-us/library/

- Windrive VXWorks MILS platform

http://www.windriver.com/products/platforms/vxworks-mils

- CoDeSys technology

http://www.codesys.com

- NSA Article : Separation Kernels on Commodity Workstations

http://www.niap-ccevs.org/announcements/separation%20kernels%20on%20commodity%20workstations.pdf

- Verizon 2014 DATA BREACH INVESTIGATIONS REPORT
- ANSI/ISA–62443-1-1 (99.01.01)–2007 Security for Industrial Automation and Control Systems
- ANSI/ISA–62443-2-1 (99.02.01)–2009 Part 2-1: Establishing an Industrial Automation and Control Systems Security Program
- ANSI/ISA-62443-3-3 (99.03.03)-2013 Part 3-3: System security requirements and security levels
- Oath2.0 Latency : http://apimetrics.io/2014/04/09/oauth-standard-isnt/

5.2 ACRONYMS

SKPP : Separation Kernel Protection Profile
IOT : internet of things
RAT : remote access Trojan
Y2K : Year 2000 bug
EURO-MILs : Secure European virtualisation for trustworthy applications in critical domains.
It : Information technology
ISA : Instrument Society of America
SCADA : Supervisory and control data acquisition
ADS: Active Directory Services
DMZ: Demilitarised Zone
CISSP : Certified information systems security professional
.NET : A Software class framework developed by Microsoft corporation
ASP : Active Server processing.
MVC : Model View Controller
MES : Manufacturing Control System
OS : Operating System
OPCUA: OLE for process control unified architecture
LDAP: Lightweight directory access protocol
MS: Microsoft
CLR : Common language runtime
CLI: Common Language infrastructure
ASLR: Address space layout randomisation
DLL: dynamic link library
MSDN: Microsoft developer network
ECMA: European Computer Manufacturers Association.
DCS: Distributed control system
DEP: Data execution prevention
FAT: Factory Acceptance test
MIT : Massachusetts Institute of technology
WPF: Windows presentation foundation
PKI: Public Key Infrastructure
OpenSSL: Open Secure socket Layer
MILS: Multiple Independent Levels of Security
CIA: confidentiality, integrity, availability
SL : security level /capability
CSMS: Cyber security management system
IACS : Industrial automation and control systems
HSE : health & safety
DEP : Data Execution Prevention (DEP) is a set of hardware and software technologies that perform additional checks on memory to help prevent malicious code from running on a system.