Incorporating Agile and Scrum into IEC 62443

White Paper
exida
80 N. Main St.
Sellersville, PA
www.exida.com

February 2020

exida White Paper Library
http://www.exida.com/Resources/Whitepapers

Abstract

This paper discusses the use of IEC 62443 Cyber Automation Security standard requirements in an agile environment specifically Scrum. It describes how the security requirements generate activities that, for the most part, are already being done in Scrum however the scope is increased. There are some activities that are IEC 62443 specific and they need to be incorporated into the Scrum process. The main discrepancy that needs to be addressed is that IEC 62443 requires independence of testers while Scrum allows for no sub-teams or independence[1].

Conclusion

It is possible to incorporate agile/Scrum with IEC 62443

Consider security explicitly

Security testers need to be independent. Borrow from other teams while doing security testing.

Introduction

IEC 62443 "Security for Industrial Automation and Control Systems" (IACS) is a standard that considers cyber security in an automation and control system environment. This is different from cyber security in an information technology (IT) environment in that IACS considers availability as the most important aspect while IT considers confidentiality as most important.

There are different ways to develop products. One way that has been used frequently is agile. Scrum is a flavor of agile and is defined at https://www.scrumguides.org/scrum-guide.html. Scrum is a process that contains sprints of 1 week to 1 month for a release along with daily meetings between team members and flexibility in assignments and responsibilities.

This whitepaper shows one way to incorporate the IEC 62443 4-1 requirements for products into agile i.e. Scrum. It uses the approach that there are activities that are already done when using Scrum. These activities are not explicitly called out in the Scrum guidelines however they are performed as part of the Scrum process. This paper describes how to extend these Scrum activities to explicitly consider the IEC 62443 requirements allowing the process to better consider cyber security.

You can see the overall results in Figure 2 - Scrum-like Agile with 62443-4-1. These activities will be described in the paper.

[1] "Scrum recognizes no sub-teams in the Development Team, regardless of domains that need to be addressed like testing, architecture, operations, or business analysis." https://www.scrumguides.org/scrum-guide.html#team-dev

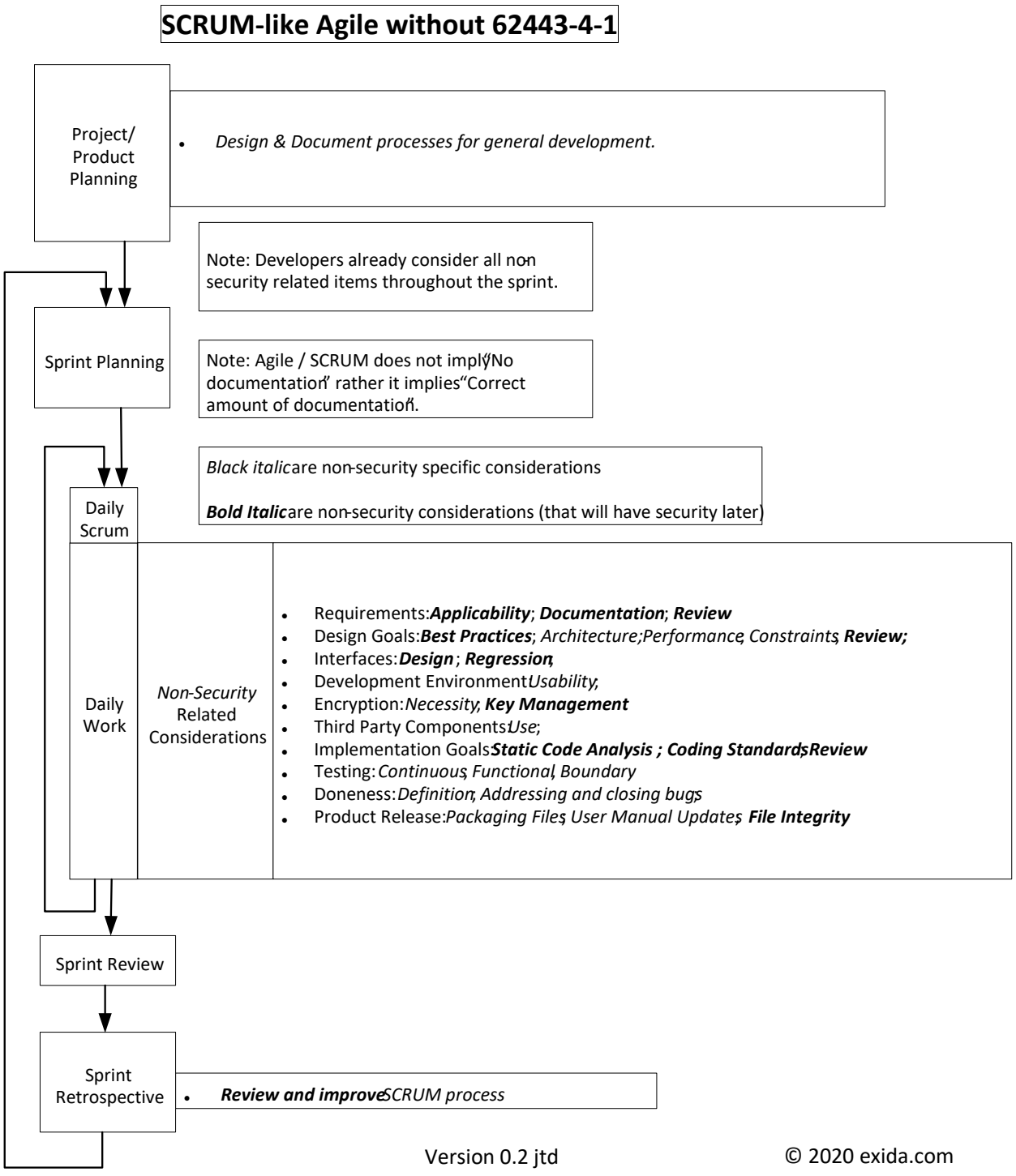

Figure 1 – SCRUM Agile

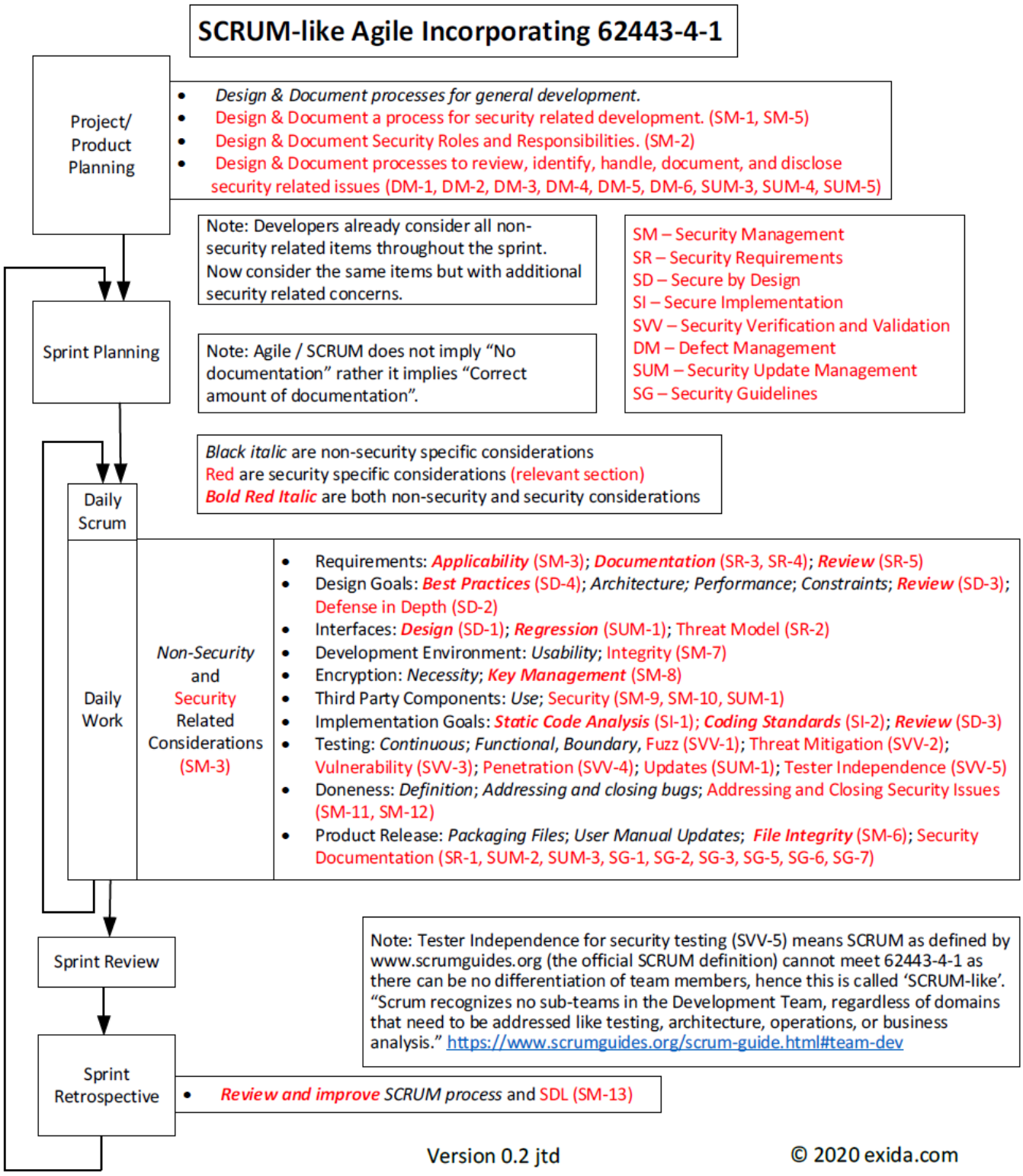

Figure 2 – Scrum-Like Agile with 62443-4-1

Project and Product Planning

Scrum already has a project and product planning phase even though it may not be explicitly recognized by most Scrum members. This planning phase is where the environment is selected or inherited. This includes process, tools, and personnel. The tools used such as compiler, source code control, requirements tools, documentation tools e.g. MS Word, etc. are all chosen explicitly or implicitly. Roles of different team members are usually chosen explicitly but may be implicit based on history and previous projects. The process is usually implicit.

IEC 62443 requires that these processes, tools, and roles are explicitly determined and documented. This results in a development process that considers security explicitly. The process needs to determine what is in or out of the scope of security and to keep track of security related items and issues.

Design & Document a Process for Security Related Development

The process needs to consider configuration management which need to have change controls and audit logging. So, there needs to be version control that has users, privileges for users, logging of changes, etc. This configuration management needs to handle the product definition, requirements, documentation, source code, deliverables, etc. There can be multiple different parts to the configuration management e.g. source code is under Git while requirements are under Doors.

There needs to be a product definition. This is so that everyone understands what is be secured.

There need to security requirements that have traceability as to how they are being met. This is usually done with a tool that contains requirements definitions, assignments to phases or people, links to documents including tests and test results, etc. Remember that agile/Scrum does not disallow documentation it just needs to be the correct level. In the IEC 62443 case these requirements need to be documented.

The process needs to include repeatable security verification and validation testing. There should be an automated set of tests that run for verification (is it doing things right) and validation (is it doing the right [via requirements] thing) tests. There should already be continuous testing as part of the Scrum process so these would fit into that.

The process needs to callout that there will be reviews of different documents, test, requirements, etc. used to create the product. There needs to be reviews of any security related items and issues.

The process itself needs to be periodically reviewed and improved.

Design & Document Security Roles and Responsibilities

There needs to be roles for the different parts of the development process. The same role could handle more than one part of the development process. There needs to be a security tester role. There needs to be someone assigned to each role above. Remember that security tester role people need to be from a group different from the one that developed the security items.

Design & Document Processes to Review, Identify, Handle, Document, and Disclose Security Related Issues

Need to describe how people can report security issues. This is usually already done by a Product

Security and Incident Response Team (PSIRT). In this case the process needs to refer to PSIRT processes. Otherwise there needs to be a process developed and documented that allows anyone, not just customers, to submit security issues.

Need to describe how security issues are investigated, analyzed, and fixed/mitigated. PSIRT should have processes in place however if not then a process needs to be developed and documented that describes how a security issue moves from being submitted through being addressed and communicated to interested parties.

The process needs to consider the security context, severity of the issue, and related security issues when assessing the issue. It also needs to determine root cause, identify any other products containing the issue, and consider the defense-in-depth principles during analysis. It needs to describe how to disclose security issues. The process needs to be periodically reviewed and improved.

Daily Activities

The activities in this section are done every day as part of the typical Scrum work. Most of the activities are extended to explicitly cover security. There are some that are new but not many and the ideas are familiar but probably not usually considered during development.

Requirements Activities

Consideration of non-security related requirements are part of everyday activities. These requirements may be implicit when reading a story or explicit using tools.

Applicability

Each requirement needs to be examined to see if it impacts security. For any security requirements one must make sure that they are applied to the product as development is occurring for the day.

Documentation

Documentation generally has bad connotations especially in the agile/Scrum world. The agile manifesto and Scrum do not say that there is no documentation only that it is the correct level of documentation. In a lot of non-security cases there may not be any documentation needed. However, IEC 62443 requires documentation explicitly for security related items and issues. This documentation needs to consider any security related information including such things as installation, operation, maintenance, and decommissioning of items/features. The documentation needs security requirements including scope and boundaries of any security items. Be sure to include the required capability security level (SL-C).

Review

Review any security requirements and include different roles/people in the reviews. Update the requirements as needed for clarity, verification, threat model, and validity.

Design Goal Activities

Every day there are design decisions made. These impact performance, usability, maintainability, security, etc.

Best Practices

There is usually a set of "best practices" or "effective practices" that depend on the area of development and the product itself. These design goal best practices need to include security aspects. They also need to be updated as new information about how well they work is obtained. An example of design goal would be non-storage of passwords; instead a hash of the password would be used and stored.

Architecture

The architecture is either implicitly or explicitly considered every day. There may be design tradeoffs due to the architecture. Security considerations should be part of the architecture. Examples might be the requirement of firewalls or locked cabinets.

Performance

Performance is a design goal that is usually not considered unless it is obvious there is going to be an issue from the start. Performance is not only an issue for the ordinary use case it can also be a security concern. For example, if one can overload one part of the system it may be possible to bypass a security gate allowing unauthorized user access to an asset.

Constraints

There are always constraints in the system or product. These are considered everyday as design decisions are being made even in the context of implementation. Care must be taken that the constraints do not cause security issues e.g. the choice of a poor key length due to processing power constraints or allowing passwords to be kept in memory accessible by user programs.

Review

The design goals need to be reviewed for security issues. This is best done by reviewing with another member of the team so that a different viewpoint is introduced. If there are shortfalls in security design then they need to be rectified and then brought out to the team so that others are aware and do not make the same design choice.

Interface Activities

Design

Interfaces need to be designed so that they meet the requirements. These requirements usually are ease of use, resource control aware (e.g. who owns memory used in the interface), multitasking limitations, timing requirements, synchronization, etc. Security needs to be explicitly considered from the point of crossing of trust boundaries, sensitivity of information,

Regression

Changes in interface definition can cause regression i.e. bugs. Any changes to an interface need to be explicitly considered from a security point of view. This includes consideration of parameter values as well as ordering of any operation on interfaces e.g. if an "open" needs to be done before a "read" then any changes need to ensure this occurs in the correct order.

Threat Model

Each interface needs to be considered in the threat model. The threat model will use the interface as a path to the asset from threats. These need to be considered as any interfaces are created, changed, and/or deleted. Addition or changes of interfaces expand the attack surface. These need to be considered in updated threat models to ensure the security requirements are still met.

Development Environment Activities

The development environment itself needs to be considered from a security point of view. The tools used may be compromised by an intruder and malware introduced as part of the tool chain. An example could be a compromised compiler that adds a "call home" process to every build.

Usability

The usability of the tool chain should be considered from the start. Choices need to be made to ensure the developers do not try to "get around" the tool chain operation due to it being "too difficult" and then allowing security violations. An example might be having to manually run malware checks on executables before shipping; this is ripe for allowing this step to be missed and possibly allowing virus carrying executables to be delivered to the field.

Integrity

The tools themselves need to be integrity checked. That means they have not been manipulated. The tools include configuration files and scripts as well as any command line used. These need to be checked before each formal build so that there are assurances that the tools themselves don't introduce security issues.

Encryption Activities

Encryption includes any use in the tool chain, artifacts, associated parts of the system (e.g.

password vault), as well as the product itself.

Necessity

The necessity of encryption could change as the product develops or the type or parameters of encryption (symmetric, asymmetric, algorithm, key size, etc) may change. The information encrypted may change as well. These changes need to be broadcast to the team so that all are aware and can determine any impacts to architecture, design, implementation, etc.

Key Management

Key management i.e. how keys are handled, stored, used, etc. must explicitly be considered. This includes during the tool chain use, at rest or in transit, etc. The keys need to be held in hardware at some security levels otherwise can be in software but there needs to be compartmentalization so that only authorized persons or processes can access the keys. There needs to be consideration of how to revoke, reissue, and otherwise manage keys. There should be no hardcoded keys.

Third Party Component Activities

Almost all products these days use third party components. These components need to be considered explicitly from a security point of view for security issues and how the they are handled.

Use

The use of third party components includes how they are integrated into the tool chain, the interfaces, updates, etc. These are sometimes not explicitly considered before the choice of the third party component however they should be. If not explicitly considered then dependencies may be introduced that have a ripple effect on the architecture, design, implementation, and cost that could cause considerable extra work for the team.

Security

All third party components need to be considered in the threat model. Any security issues need to be addressed including how any security updates are handled e.g. by third party or by the product provider. All interfaces, memory accesses, encryption, etc. need to also be explicitly considered as to how it impacts security.

Implementation Goal Activities

Implementation is the act of converting a design into reality e.g. creating software or hardware. The software created needs to be created and analyzed in a way so that security is kept at the required level. There are a few techniques required by the standard to be done.

Static Code Analysis

This is to be done for all software that impacts security. There are many different tools that can be used for static code analysis. Each language has different options. The usual tools for C/C++ are Coverity or Klockwork. These need to be run on the source code and any security deficiencies such as buffer overflows, uninitialized data, or null pointer dereferencing handled.

Coding Standards

Coding standards need to in place and the developers (and tools) need to adhere to them. The reason is that following the coding standards allow for typical security related issues to be bypassed by not allowing them to exist. Examples for C/C++ could be “sprintf” where there is no check on the destination allocation size; each language has its own set of coding standards. These coding standards need to be reviewed and code itself needs to be reviewed and used to ensure developers are following them.

Secure Coding Practices

Use secure coding practices including using proven paradigms for any security related code e.g.

SQL parameterization, checking of parameters limits, etc. Avoid banned functions e.g. 'strcpy' for C/C++. The design should promote good error handling. Use tools when doing analysis and testing.

Review

All security related code needs to be reviewed by members of the team. There needs to be records of these reviews so that when security issues are found they can be studied, and changes made so that they don't occur again. These reviews also allow for changes to the development process and tool chain as needed to keep the security issues from reappearing.

Testing Activities

There should already be testing activities integrated into the development process. The testing should be continuous i.e. part of the tool chain so that as changes are made there is regression testing begin performed as well as testing on the new changes.

Boundary & Fuzz Testing

There should be boundary testing i.e. explicitly checking the minimum and maximum values for all inputs. If there are holes in the inputs, e.g. even numbers allowed only, then test the values at the edges of the holes.

As well, there should be fuzz testing where arbitrary inputs are used for tests. This is to test the handling of malformed data, correctly formatted but unexpected data, and data outside of expected input.

Fuzz testing can be "smart" where the format of the data is known, and testing done taking that formatting, e.g. protocol frames, into account. "Dumb" fuzz testing randomly generates data without taking any format information into account.

Threat Mitigation Testing

Tests need to be created for each identified and valid threat from the threat model and a mitigation for that threat. Any tests created need to be run and be successful at showing the threat has been mitigated as expected.

Vulnerability Testing

Testing for security vulnerabilities using a public well-known, e.g. OWASP, source of known vulnerabilities.

Penetration Testing

Penetration testing here means, from IEC 62443-4-1 9.5 SVV-4, "… via tests that focus on discovering and exploiting security vulnerabilities in the product."

The rational further states "Penetration testing can involve the use of manual techniques, test tools or combinations of the two." So, it is possible to automate IEC 62443 penetration testing.

Update Testing

Whenever updates, i.e. patches or whole replacement, are done they need to be tested to ensure that there are no regressions introduced, i.e. reintroduction of previous bugs or inconsistencies, and that they do not cause problems with operation, safety, or legal constraints.

Tester Independence

One of the main items to be reconciled between Scrum and IEC 62443 is the tester independence requirement. There are different levels of tester independence depending on what is being tested. The following slide shows a table with the requirements.

Test Type	Level of Independence from Developer	Reference
Static Code Analysis	**None (same person can test as developed)**	Security implementation review (SI-1)
Software Composition Analysis		Vulnerability Testing (SVV-3)
Abuse Case	Independent Person (person testing cannot be a developer of the product)	Vulnerability Testing (SVV-3)
Attack Surface		
Known Vulnerability Scanning		
Security Requirements	Independent Department (person testing cannot have same first line manager as developer; alternatively, they could be from QA)	Security Requirements Testing (SVV-1)
Threat Mitigation		Threat Mitigation Testing (SVV2)
Penetration Testing		Penetration Testing (SVV-4)
Penetration Testing	Different Organization (person testing cannot be from same division, legal entity, etc.)	Penetration Testing (SVV-4)

Table 1 - Tester Independence

Doneness

Part of the process is to determine when to declare 'done'. That is to say, when it is time to consider the sprint over. Sometimes there are pieces of functionality not finished or bugs not addressed. These need to be considered when declaring done.

Addressing & Closing Security Issues

Addressing and closing of bugs is a normal activity. Documentation and verification of the fixes should be done but may or may not be explicitly noted as part of day to day operation.

IEC 62443 requires that security issues are explicitly addressed, closed, documented, and verified before the product or patch is released.

Integrity of Files in a Release/Patch

All files that are part of the release/patch need to have explicit integrity checks. This includes executables, scripts, data files, etc. The reason is so that the receiver of the files can verify that the files have not be changed.

Security Documentation

There needs to be documentation about how the product is to be installed, configured, operated, and disposed of so that the security level that is required is met.

There needs to be user manual(s) or portions of other user documentation that describes the information needed to make to device secure.

The information needs to include defense in depth strategy, hardening, external requirements, secure disposal, operations, and accounts management.

The documents need to be reviewed periodically to ensure they are sufficient and correct when the product changes.

Review of the Secure Development Process

In general, there is a review of the development process when the sprint finishes. IEC 62443 requires a review of the security portion of the development process.

The activities may need to be adjusted to be more encompassing or even less encompassing based on how well the requirements are being met.

Tools may need to be better utilized or a different toolset may be found to make development easier yet just as secure.

Adjustment in communication or documentation may be needed.

A balance between all these needs (personnel, process, tools, etc.) to be eventually found that allows for ease of development while producing the required security level capability.

Revision History

Authors: Jeff Davis

exida – Who we are.

exida is one of the world's leading accredited certification and knowledge companies specializing in automation system cybersecurity, safety, and availability. Founded in 2000 by several of the world's top reliability and safety experts, exida is a global company with offices around the world. exida offers training, coaching, project-oriented consulting services, standalone and internet-based safety and cybersecurity engineering tools, detailed product assurance and certification analysis, and a collection of online safety, reliability, and cybersecurity resources. exida maintains a comprehensive failure rate and failure mode database on electrical and mechanical components, as well as automation equipment based on hundreds of field failure data sets representing over 350 billion unit operating hours.

exida Certification is an ANSI (American National Standards Institute) accredited independent certification organization that performs functional safety (IEC 61508 family of standards) and cybersecurity (IEC 62443 family of standards) certification assessments.

exida Engineering provides the users of automation systems with the knowledge to cost-effectively implement automation system cybersecurity, safety, and high availability solutions. The exida team will solve complex issues in the fields of functional safety, cybersecurity, and alarm management, like unique voting arrangement analysis, quantitative consequence analysis, or rare event likelihood analysis, and stands ready to assist when needed.

Training

exida believes that safety, high availability, and cybersecurity are achieved when more people understand the topics. Therefore, exida has developed a successful training suite of online, on-demand, and web-based instructor-led courses and on-site training provided either as part of a project or by standard courses. The course content and subjects range from introductory to advanced. The exida website lists the continuous range of courses offered around the world.

Knowledge Products

exida Innovation has made the process of designing, installing, and maintaining a safety and high availability automation system easier, as well as providing a practical methodology for managing cybersecurity across the entire lifecycle. Years of experience in the industry have allowed a crystallization of the combined knowledge that is converted into useful tools and documents, called knowledge products. Knowledge products include procedures for implementing cybersecurity, the Safety Lifecycle tasks, software tools, and templates for all phases of design.

Tools and Products for End User Support

- exSILentia® – Integrated Safety Lifecycle Tool

  - PHAx™ (Process Hazard Analysis)
  - LOPAx™ (Layer of Protection Analysis)
  - SILAlarm™ (Alarm Management and Rationalization)
  - SILect™ (SIL Selection and Layer of Protection Analysis)
  - Process SRS (PHA based Safety Requirements Specification definition)
  - SILver™ (SIL verification)
  - Design SRS (Conceptual Design based Safety Requirements Specification definition)
  - Cost (Lifecycle Cost Estimator and Cost Benefit Analysis)
  - PTG (Proof Test Generator)
  - SILstat™ (Life Event Recording and Monitoring)
- exSILentia® Cyber- Integrated Cybersecurity Lifecycle Tool
  - CyberPHAx™ (Cybersecurity Vulnerability and Risk Assessment)
  - CyberSL™ (Cyber Security Level Verification)

Tools and Products for Manufacturer Support

- FMEDAx (FMEDA tool including the exida EMCRH database)
- ARCHx (System Analysis tool; Hardware and Software Failure, Dependent Failure, and Cyber Threat Analysis)

For any questions and/or remarks regarding this White Paper or any of the services mentioned, please contact exida:

exida.com LLC

80 N. Main Street

Sellersville, PA, 18960

USA

+1 215 453 1720

+1 215 257 1657 FAX

info@exida.com