Agile software development under ISO 27001

Leon du Toit

2022-11-26

Abstract

The ISO 27001 standard lists a host of requirements for implementing IT security in an organisation. Conforming to requirements of the standard can raise the cost and friction involved in releasing new services and features in an agile manner. This report will explore which tools and processes can be used to mitigate the potential clash between the goals of standard compliance, and sustaining an agile development culture. The report is organised as a case study at the University of Oslo's Center for IT, where the Services for Sensitive Data is in the process of adopting ISO 27001. *Keywords: Agile, SecDevOps, ISO 27001*

Introduction

Agile software development (Beck (2022)) is described as valuing:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

So while the latter items are valuable, Agile sees the former items as *more valuable*. The University of Oslo's (UiO) Services for Sentitive Data (TSD), a special purpose eInfrastructure, is developed and operated by UiO's Center for IT's Section for Research Services. Agile software development is a well-established practice within the section, and is used to continuously develop new features and services for TSD.

At the same time the Section for Research Servics is working to certify TSD for ISO 27001 ("Information Technology - Security Techniques - Information Security Management Systems - Requirements (ISO/IEC 27001:2013 Including Cor 1:2014 and Cor 2:2015)" (2017)). ISO 27001 stipulates that an organisation should adopt an Information Security Management System (ISMS) in order to manage risks, to ensure that its information security policies are met. The organistion shall identify, and analyse risks and their potential impact, and implement risk-mitigating controls. All of this needs to be documented, and

updated over time. In practice this influences software development processes, and has the potential to uproot an agile development culture, by reversing the order of importance in the Agile values articulated above.

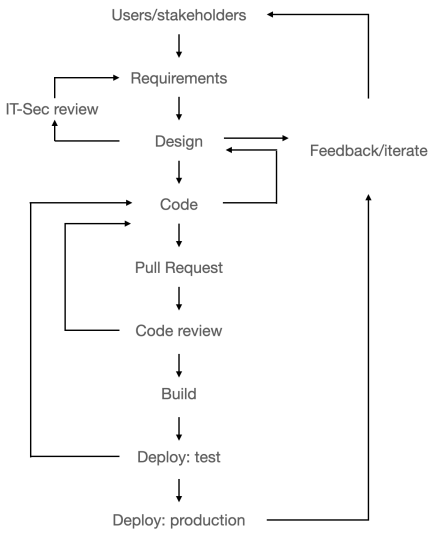
The rest of this report will explain how ISO 27001 certification can adversely affect TSD's agile development culture, and explore how SecDevOps can offer a solution to this conundrum.

TSD - Services for Sensititve Data

TSD is a special purpose eInfrastructure designed for working with sensitive personal data (UiO (2022)). It is hosted at UiO, and operated by the Section for Research Servces at the University of Oslo's Center for IT. In production since 2014, it is a remote-login platform-as-a-service solution supporting more than 1600 research projects, 8000 researchers, and 80 institutions. Development and operations is staffed by 14 full-time employees. TSD is working towards ceritifcation in ISO 9001 and ISO 27001 in 2023. As part of this process, software development processes must be reviewed to ensure compliance with the required controls.

Agile software development at TSD

Figure 1 below illustrates how TSD practices agile software development.



The need for software development originates with the requirements of users and/or stakeholders. It is often necessary to reach agreement on what those

requirements are before any software development happens. This is illustrated by the feedback-loop from requirements to design, and then back to the users and stakeholders. Once developers have a sufficiently clear understanding of the requirements, they decide to what extent the IT Security team needs to be involved. If the requirements translate into a new service, for example, then the design will be presented at the periodic change council. If any changes are needed, the design is updated, and implementation can begin.

While writing code, TSD developers run unit- and integration tests to ensure correctness. Once the feature that covers the requirements is implemented, a new patch is submitted for review by the core team, in the form of a pull request. Reviewers have the chance to comment on the proposed changes, and request additional changes if needed. This would lead to code being modified, and the pull request being updated. Once a reviewer approves of the changes, the changes are merged with the main codebase.

The new version is then built into an executable. The new version is first deployed to a test environment, where more integration tests are performed. If this uncovers any issues that were not caught by local testing during development, or during code review, then the necessary issues are fixed with a new pull request, reviewed, and deployed to test again, until all testing passes. Once deployed to production, users and/or stakehoders have the chance to give feedback. If needed, a new iterative cycle will begin.

TSD has been growing continually at a fast pace in the last five years, with more than one new research project signing up each work day. At the same, the complexity of user requirements increase, as more software and integrations are used in research. Taken together this means that the devlopment team must give high priority to speed in the implementation of new features. Fast release of new services and features is a key component to scaling the service with few employees, but of course this has to be accomplished without compromising the service's information security policies.

TSD has been able to apply agile software development and maintain its information security policies by building on top of a secure, and modular web development platform. At the same time TSD sees the need to pursue compliance with ISO 27001, as this will be an important strategic accomplishment, and will help with sustaining the effort for delivering secure IT services. The open question is how to achieve compliance without grinding the agile development process outlined above to a halt. The next section will explore the details of what compliance might mean for TSD's software development process.

The impact of ISO 27001 on software development

This section reviews the controls listed in Annex A of the ISO 27001 which will have an impact on software development practices in TSD. For each control, the expected impact on TSD's software development process, with reference to

Figure 1, is described.

A.12.5.1 Installation of software on operational systems

"Procedures shall be implemented to control the installation of software on operational systems."

TSD lacks in this regard in the following ways:

- it does not have 100% coverage - there are updates which are not controlled
- where it does have control, there is too large variety in the mechanisms for updates

Getting 100% coverage, and sufficient standardisation has the potential to ease deployment, but it will require up-front investment in new tools and processes. Up until these tools and processes are in place, it will divert effort from releasing new end-user features. This poses a barrier to adoption and can be seen as standing in opposition to the agile principle of "Individuals and interactions over processes and tools". Successful introduction will, therefore, require a cultural change.

A.12.6.1 Management of technical vulnerabilities

"Information about technical vulnerabilities of information systems being used shall be obtained in a timely fashion, the organisation's exposure to such vulnerabilities evaluated and appropriate measures taken to address the associated risk."

TSD uses open source software in its development, and as such is exposed to vulnerabilities in the entire supply chain associated with its open source dependencies. Open source software repositories have varying levels of sophistication in their ability to detect and report on vulnerabilities in their hosted packages. TSD will have to do research to see which tools exist, and how they can be incorporated into the development cycle. In practice this will raise the cost of including any new dependencies and updating existing dependencies.

A.14.1.1 Information security requirements analysis and specification

"*The information security related requirements shall be included in the requirements for new information systems or enhancements to existing information systems."*

TSD already involves the IT security team in the design and implementation phase of new services and signifcant new features. TSD also has a culture of considering the security implications of releasing new features. The additional effort required is to document that this has been done, and to do so *in each and*

every change. This will add some documentation and communication overhead to development.

A.14.2.2 System change control procedures

"Changes to systems within the development lifecycle shall be controlled by the use of formal control procedures."

This is arguably similar to item A.12.5.1, but can be interpreted more broadly: that it applies to systems used *during* development, in addition to those used in delivering actual IT services. In brief, it means more documentation overhead.

A.14.2.4 Restrictions on changes to software packages

"Modifications to software packages shall be discouraged, limited to necessary changes and all changes shall be strictly controlled."

This will introduce more friction to dependency management, as each new dependency will have to be evaluated before adoption. As such it raises the cost of development.

A.14.2.8 System security testing

"Testing of security functionality shall be carried out during development."

TSD already uses both unit- and integration-testing during development, but does not have full coverage on this. This will, therefore, be a change of degree as such testing is rolled out to all components, and as the coverage for any given component is increased. It does, however, increase the setup cost involved in establishing a development environment. If not implemented well, it could be a barrier to entry for new contributors.

A.14.2.9 System acceptance testing

"Acceptance testing programs and related criteria shall be established for new infomation systems, updgrades and new versions."

TSD already has a test environment where integration tests are performed before production release, but as mentioned in the discussion of A.12.5.1, it lacks 100% system coverage. Achieving compliance with this control will require a more feature complete testing environment, more comprehensive test suites, and documentation. This could potentially raise the cost of release, slowing development.

A.15.1.3 Information and communication technology supply chain

"Agreements with suppliers shall include requirements to address the information security risks associated with information and communications technology services and product supply chains."

As mentioned in the discussion of control A.12.6.1, TSD uses open source software in development. The licensing model of open source software gives the user no warranty at all. TSD must therefore, internalise any risk associated with using such software. The implication is that each explicit dependency must be included in a risk assessment. This raises the cost of development.

Problem statement

Analysis of the ISO 27001 control measures that will impact software development show that TSD's current agile process will be impacted in the following ways:

1. Dependency management: vulnerability detection, risk assessment
2. Documentation: policies, procedures
3. Testing: security testing during development, acceptance testing during release

Dependency management is arguably the most significant challenge for TSD, and also the most important in terms of risk management. The dependencies running in production add to the system's Trusted Computing Base (TCB): the hardware and software necessary for enforcing all security rules (Tanenbaum and Bos (2015), p601). If any of the dependencies of the TCB were to be compromised by, for example, a supply-chain attack, then the enforcement of TSD's security rules could be compromised.

TSD's primary programming language for implementing backend web services is Python. Supply-chain attacks have recently been uncovered in the Python package ecosystem. A software supply-chain security company (Phylum-Research (2022)) recently wrote about how attackers copy popular Python packages, rename them using a technique called typosquatting (choosing names that are *almost* the same as the original), and inject malicious code as dependencies. In most cases the malicious code is obfuscated Python code which downloads an executable and runs it on the host machine. An inattentive user could easily include such a package as a dependency, and code would work.

Datadog Security Labs (De Turckheim and Tafani-Dereeper (2022)) recently discovered malicious code injected into a previously non-malicious Python package, `fastapi-toolkit`. The malicious code added an HTTP endpoint to the web framework which allowed attackers to execute arbitrary SQL queries in the web application, given that they knew the value of a specific header. Other than with typosquatting, the package is actually legitimate, but access to the source code had likely been compromised. One can, therefore, be vulnerable to such attacks

even if you used a legitimate package to begin with. These cases illustrate the importance of investing effort into dependency management.

It is clear that identifying vulnerabilities in open source software is a time-consuming, yet important, activity. For TSD it will mean slower development. The same is true for increased amounts of documentation on every change. Documentation, unlike testing, cannot be fully automatated. As such it will take time away from solving development problems.

To address dependency management, documentation and testing will require a cultural change in TSD. Addressing these areas will increase the effort and time needed to release new features. This makes it challenging, therefore, to maintain an Agile development culture while pursing ISO 27001 compliance.

The next section will explore a developement and operational methodology called SecDevOps, to see how it can help offset the potential devlopment costs associated with ISO 27001 compliance.

SecDevOps as a solution

Amazon Web Services, one of the world's leading cloud services providers defines DevOps as "the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity. Under a DevOps model, development and operations teams are no longer 'siloed'. Sometimes, these two teams are merged into a single team where the engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function." (AWS (2022)). DevOps is characterised by a high degree of automation in development and deployment to allow scalable operations.

The high velocity can however, be problematic for maintaining security. The danger is that security is seen as a brake on the pace of development - as is seen in the analysis above. The risk of such a view is that few people pay attention to security requirements in the development and release cycles, and that this exposes vulnerabilities in services. There is a growing recognition that security must be incorporated into DevOps in a holistic manner (Tischart (2022)), and that security must play a role from the outset, before any development happens. In this way SecDevOps can be seen as applying DevOps methodologies to the application lifecycle, but with security concerns being present along the way.

TSD has the ingredients of both DevOps and SecDevOps in its agile development model. The next section will explore which concrete changes TSD can introduce in order to achieve an agile process which is compliant with ISO 27001, by applying a SecDevOps framework to its processes.

Recommendations

SecDevOps provides a conceptual framework for applying security measures throughout the application lifecycle, while maintaining a DevOps/agile approach to software development. The analysis of ISO 27001 controls have identified three areas of which TSD needs to address in this regard:

1. Dependency management: vulnerability detection, risk assessment
2. Documentation: policies, procedures
3. Testing: security testing during development, acceptance testing during release

This section will make concrete recommendations for how SecDevOps can be applied in these areas, given the software development process outlined in Figure 1.

Dependency management

To discover vulnerabilities in existing Python dependencies TSD can adopt the usage of `pip-audit` (Cameron (2022)) in both development and during the build process. The `pip-audit` package allows users to scan their dependencies based on the Python Packaging Advisory Database (PyPI (2022)) - a contributor to the distributed vulnerability database for Open Source (OSV (2022)). This will help detect known issues before the dependencies are used in production.

In addition to adopting this tool, TSD can introduce a new type of pull request and code review: one designated to depenencies. If new dependencies and updates to existing dependencies are made separate from other logical code changes, then reviewers can direct their review efforts to auditing the dependencies themselves. Making this process explicit will help identify potential issues, and it fits well into existing code review practices.

Documentation

When deploying new code, the following has to be documented:

- accpetance tests performed, and passed
- no new risks introduced by the change

TSD can use pull request templates offered by github to structure such information.

Testing

Security testing must be applied during development, and acceptance testing must be performed prior to deployment. TSD's web service environment is a complex set of microservices which interact with one another via a message-oriented middleware. It can, therfore, be time consuming and challenging to set up test environments that match production.

In keeping with the SecDevOps philosophy TSD can work towards containerising all of it services, so that one can specify and provision test environments on-demand using container orchestration tools. This will also solve problems associated with multiple changes being concurrently applied to a given test environment.

Conclusion

The principles of agile development stand in relative opposition to the procedural controls manadated by the ISO 27001 standard. TSD has to reconcile these disparaties by changing its current agile development culture and process. SecDevOps provides a conceptual framework for adopting such changes: use DevOps-style automation with security in mind at each step. Analysis of the controls outlined by ISO 27001 has identified three key areas of change if TSD is to become compliant with the standard: 1) dependency management, 2) documentation, and 3) testing. This report has made recommendations for how TSD, and other organisations facing similar challenges, can leverage tools and processes to address these concerns, thereby maintaining an agile development culture, while at the same time being compliant with the controls mandated by ISO 27001.

References

AWS. 2022. "What Is DevOps?" Amazon Web Services. 2022. https://aws.amazon.com/devops/what-is-devops/.

Beck, Kent. 2022. "The Agile Manifesto." Agile Alliance. 2022. https://www.agilealliance.org.

Cameron, Alex. 2022. "A Tool for Scanning Python Environments for Known Vulnerabilities." PyPI. 2022. https://pypi.org/project/pip-audit/.

De Turckheim, Vladimir, and Christophe Tafani-Dereeper. 2022. "Investigating a Backdoored PyPi Package Targeting FastAPI Applications." Datadog. 2022. https://www.uio.no/english/services/it/research/sensitive-data/.

"Information Technology - Security Techniques - Information Security Management Systems - Requirements (ISO/IEC 27001:2013 Including Cor 1:2014 and Cor 2:2015)." 2017. Standard. International Organization for Standardization.

OSV. 2022. "A Distributed Vulnerability Database for Open Source." OSV. 2022. https://osv.dev/.

Phylum-Research. 2022. "Phylum Discovers Dozens More PyPI Packages Attempting to Deliver W4sp Stealer in Ongoing Supply-Chain Attack." Phylum. 2022. https://blog.phylum.io/phylum-discovers-dozens-more-pypi-packages-attempting-to-deliver-w4sp-stealer-in-ongoing-supply-chain-attack.

PyPI. 2022. "Python Packaging Advisory Database." PyPI. 2022. https://github.com/pypa/advisory-database.

Tanenbaum, Andrew, and Herbert Bos. 2015. *Modern Operating Systems.*

Pearson.
Tischart, Jamie. 2022. “DevOpsSec, SecDevOps, DevSecOps: What’s in a Name?” Cloud Security Alliance. 2022. https://cloudsecurityalliance.org/blog/2016/12/05/devopssec-secdevops-devsecops-whats-name/.
UiO. 2022. “Services for Sensitive Data (TSD).” UiO. 2022. https://www.uio.no/english/services/it/research/sensitive-data/.