Ebook

invicti

Application Security According to ISO 27001

A guide to applying ISO 27001/27002 recommendations to secure in-house and outsourced application development

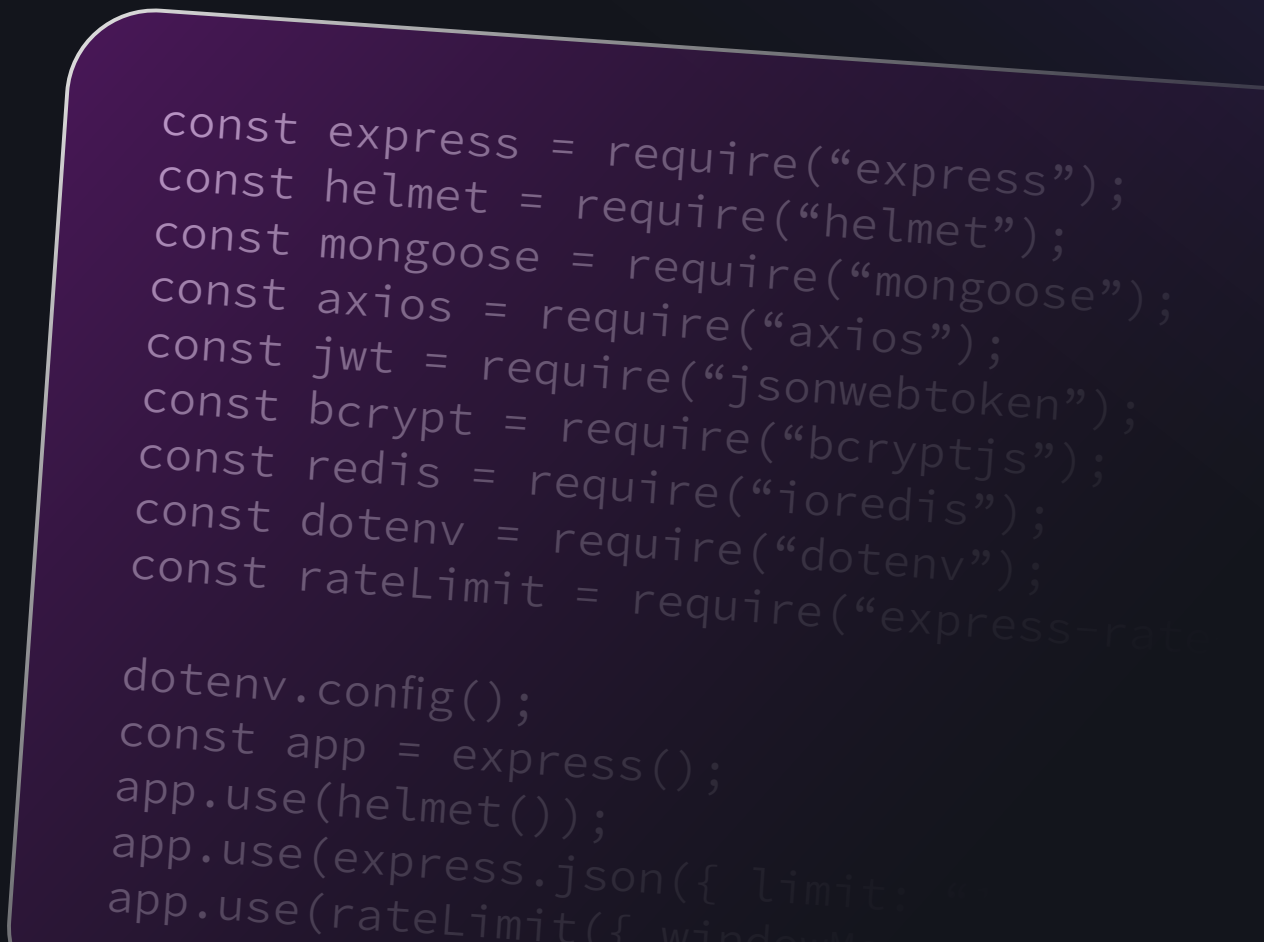


Table of contents

Executive summary 3

ISO 27001 puts application security front and center 4
- What was new in ISO 27001:2022 5
- Secure SDLC controls 6
- Vulnerability scanning is a compliance necessity 7

Adopting a security mindset from design to deployment 8
- A security-first approach 8
- Security in agile development workflows 9

Secure coding practices 10
- Phase 1: Planning and before coding 11
- Phase 2: During coding 11
- Phase 3: Review and maintenance 12
- Developer education 12

Application security testing in the SDLC 13
- Multiple testing methods for securing the SDLC 14
- Dynamic security testing from first build to production 15

Securing development workflows 16
- Where does your code come from? 17
- Have you tested that app in real life? 18

Software supply chain security 19
- Third-party software still needs security testing 20
- Guidelines for securing outsourced code 21
- Security requirements for cloud services and platforms 21

ISO 27001 is for life, not just for certification 22

invicti

Executive summary

Information security professionals are well aware of the importance of ISO 27001 compliance in demonstrating an organization's ability to secure its data. But where does application security come into play in this popular compliance framework? With cyber threats constantly evolving and becoming ever more sophisticated and pervasive, relying on traditional network security measures is no longer enough to protect your data and systems. Accordingly, application security has become a crucial component of ISO 27001 compliance, but the standard can also provide more general insights and best practices.

Incorporating application security measures into your broader infosec strategy can help mitigate the risks posed by vulnerabilities across your software, and web applications in particular. This includes everything from enforcing secure coding practices during development to integrating security testing into the software development lifecycle (SDLC) and performing vulnerability scanning on all your web assets. To make this a reality, technical solutions must be accompanied by the right policies and procedures to ensure that security is baked into every aspect of your software development process, running from secure design principles right through to managing third-party code.

In this ebook, we dive deeper into the place of application security in the ISO 27001 standard. We'll explore some of the key challenges that organizations face when planning and implementing application security controls, and provide practical guidance on how to overcome them.

Whether you're involved in ensuring ISO 27001 compliance or are simply looking to the standard for best practices on securing application development, this ebook provides insights and actionable strategies to help you protect your organization's data in line with ISO standards for information security management.

Apart from vulnerability scanning, which is now a crucial consideration for ISO 27001 compliance, you will also learn about the importance of other aspects of application security, including:

- Testing throughout the software development lifecycle to ensure that security controls are effective and properly implemented
- Implementing secure coding practices to reduce the risk of vulnerabilities being introduced during development
- Ensuring that third-party software and code used in applications are secure and do not introduce vulnerabilities

ISO 27001 puts application security front and center

In October 2022, ISO 27001 was updated for the first time since 2013. Bringing information security management in line with the modern realities of software development meant defining security controls that encompass agile workflows and cloud-based environments across the entire SDLC.

In response to unrelenting threats to their increasingly web-based software infrastructure, organizations need to build cybersecurity management systems that reach all the way back into software development practices to ensure that the applications they build are inherently more secure. To that end, the ISO 27001 information security, cybersecurity, and privacy protection standard (and its companion document ISO 27002) provides a framework for incorporating development security into the wider information security strategy. The previous 14 information security categories were reorganized into four (People, Organizational, Technological, and Physical), and for the first time, the standard directly addressed the full software development life cycle.

In a nutshell

- The October 2022 updates to ISO 27001, the first since 2013, recognized technological changes such as the rise of cloud computing and agile development.
- For application development and security, the standard defined controls that encompassed the full software development life cycle for the first time.
- Updated security controls mandated security testing at multiple points during development and after deployment, making vulnerability scanning a practical necessity.

What was new in ISO 27001:2022

The main text of the ISO 27001 standard describes in broad strokes the goals and characteristics of an overarching secure management system, while the annex lists in ISO 27002 describe in detail the security controls (processes, policies, and logical controls) that organizations can use to achieve their cybersecurity goals. Building on the *Software Development Environment* control from 2013, the 2022 update reorganized multiple granular controls within a newly created SDLC control.

Many of the new control requirements were prompted by innovations in both technology and cyberattacks over the past decade. For example, among the standard's new requirements was one for defining security responsibilities between a cloud provider and an organization, which would have had limited applicability a decade earlier. Other controls added in 2022 included:

- Building threat intelligence by collecting and analyzing data about existing or emerging threats
- Ensuring IT departments are prepared for business continuity
- Continuously monitoring physical premises to prevent unauthorized access
- Deleting stored information when no longer needed
- Masking/encrypting data to limit exposure of sensitive information
- Monitoring networks, systems, and applications for deviations from a defined baseline of normal activities
- Restricting access to external websites that may compromise data

> The 2022 version was long overdue, especially because it provided so much more value when it comes to the framework organizations need to protect modern information technologies
>
> **Matthew Sciberras**
> CISO and VP of Information Security, Invicti

Secure SDLC controls

The biggest impact of the updated standard for software development teams was that the new control for secure SDLC wrapped a number of previous controls into a coherent set of requirements for the software life cycle. Specifically, ISO 27002 lists the following as requirements necessary for a secure SDLC and then links to one or more controls that expand on each requirement:

While many of the listed controls were already included in the older version of the standard (if scattered among multiple categories), the secure coding control was new. This specified activities and best practices for establishing a secure development environment, defining and following coding standards, and continually maintaining the security of production code. Additionally, the standard requires organizations to hold third-party and open-source software to the same coding standards as they use internally.

- Separating development, test, and production environments
- Defining security requirements in the specification and design phase
- Applying secure system architecture and engineering principles
- Performing system and security testing on deployed code, such as regression testing, code scanning, and penetration tests
- Using project management principles to address risks at any stage in the SDLC
- Defining secure coding guidelines for each programming language
- Building developer expertise in secure coding techniques and in finding and fixing vulnerabilities
- Creating secure repositories with restricted access to source code
- Securely protecting software, hardware, services, and network configurations
- Setting up secure version control with formal rules for managing changes to existing systems
- Applying security requirements to any outsourced development

> Everything in the updated standard represents practices that every good software engineering shop should be doing, but very few are doing all of these things
>
> **Matthew Sciberras**
> CISO and VP of Information Security, Invicti

Vulnerability scanning is a compliance necessity

Even assuming the strictest design and coding practices, vulnerabilities can still creep into an application at multiple stages of the SDLC. To address this, the standard emphasizes vulnerability scanning both during development and testing and after deployment. In addition to the overall SDLC control, there are two controls that directly call out vulnerability testing:

- **Management of Technical Vulnerability:** Specifies that the organization's exposure to attacks should be assessed and remedied. To accomplish this, the control recommends using vulnerability scanning tools as well as penetration testing.

- **Security Testing in Development and Acceptance:** Recommends performing vulnerability scanning and penetration testing throughout the SDLC to verify that security requirements have been met.

Four main types of application security testing tools are in common use and work synergistically: software composition analysis (SCA), static application security testing (SAST), interactive application security testing (IAST), and dynamic application security testing (DAST). SCA, SAST, and IAST look for vulnerabilities on the code side, with SCA checking for known vulnerable components. DAST performs black-box testing on a running application, allowing it to be used both during development and after deployment.

The unique advantage of DAST tools is that they can be incorporated into the SDLC and the continuous integration/continuous deployment (CI/CD) workflows that are the backbone of DevOps. DAST is also technology-agnostic to scan any type of web application, service, or API. Additionally, advanced DAST tools can discover web assets in the crawling phase as well as identify outdated technology stack components, including runtimes, frameworks, databases, libraries, and web servers.

> The ISO standards tell you what to do but not how to do it. Any vulnerability and penetration testing method would be acceptable, but the automated capabilities that DAST brings to the table will lead most organizations towards DAST tools
>
> **Matthew Sciberras**
> CISO and VP of Information Security, Invicti

Adopting a security mindset from design to deployment

ISO 27001 clearly states that security must be a consideration all across the software development lifecycle, including calling out “security in the software development methodology” and “security requirements in the specification and design phase.” While there is no one-size-fits-all set of workflows and goals, incorporating security best practices all through design and implementation is crucial to prevent vulnerabilities that leave web applications open to attack.

In a nutshell

- ISO 27001 outlines a roadmap to incorporate security considerations into the entire development workflow and infrastructure.
- Embedding systematic security testing throughout the pipeline can minimize the number of vulnerabilities that make it into production.
- Collaboration among developers, security teams, and project management is vital to integrate application security as a core aspect of development rather than an add-on.

A security-first approach

Infusing security into application specifications and development workflows requires a deliberate plan since bolting on security at a later stage will likely lead to an insecure final product. From the initial design phase, the security workflow should be well-documented to track progress, flag issues, and document improvements. The design process may start with asking basic risk assessment questions:

- How will user accounts be protected and encrypted?
- Will the user require a password and, if so, what kinds of passwords are acceptable?
- Does accessing accounts require multifactor authentication?

Initially, software engineers can answer those fundamental questions by defining security requirements that, for example, require strong passwords during account creation or enable two-factor authentication by default.

As development goes on, the initial answers to such questions may become more complex and need to factor in vulnerabilities specific to the type of application or the way it’s deployed. Notably, this may mean paying special attention to designing, documenting, and testing application program interfaces (APIs) for cloud-based applications. Vague security goals or poor threat assessment can lead to extra work and delays as teams struggle to address unclear security concerns or deal with the results of security testing that was brought in too late in the process.

A security-first approach is also important during code reviews to minimize the risk of new or modified code adding new vulnerabilities. Code reviews that emphasize security also create opportunities for application developers to share knowledge and best practices with the rest of the team, which can strengthen the security of future designs.

Security in agile development workflows

The most common development workflow used today is agile and incremental, replacing the traditional waterfall methods that typically relied on an isolated testing phase performed after development work was complete. If they are to be effective, agile workflows require ongoing collaboration between developers, security teams, and other stakeholders to ensure that security is a permanent part of the project from day one. In fact, this is the only way to ensure application security in sprint-oriented DevOps environments where stopping to wait for test results and then backtracking to remedy issues is not an option.

To fold security into agile development without compromising release schedules, tools and processes for security testing and issue remediation need to be vetted and set up before coding (let alone testing) even begins. ISO 27001 calls for vulnerability scanning throughout the SDLC, and while there are many approaches and tool combinations that can satisfy this requirement, the versatility of modern DAST solutions can be a particular boon. When integrated with DAST, an agile workflow enables web apps to be developed and updated continuously without compromising security or slowing down release cycles.

Crucially, early dynamic testing gives developers ample time to discover and fix vulnerabilities before they become tougher and more expensive problems. Organizations can't afford to wait until release to test their web apps for vulnerabilities, so security must be "designed and implemented within the secure development life cycle of software and systems," as the standard puts it.

> CISOs and security managers are being bombarded with tasks and initiatives. Having a security strategy is crucial to the success of your security program
>
> **Matthew Sciberras**
> CISO and VP of Information Security, Invicti

Secure coding practices

The ISO 27001 information security standard includes compliance requirements that call for developer education as the foundation of application security. By combining educational efforts with policies and tools to consistently enforce secure coding practices, organizations have a better chance of preventing vulnerabilities from being introduced in the first place.

Winston Churchill once said, “Those that fail to learn from history are doomed to repeat it.” If Churchill were a chief security officer today, he might say, “Those that don’t learn to write secure code are doomed to repeatedly get the same vulnerabilities.” While good software security testing can uncover such vulnerabilities, organizations could save a lot of the time and money spent on fixing issues if they developed more secure code in the first place.

Research shows that developer security education is lacking, and organizations need to urgently address this not only to minimize risk but also to comply with ISO 27001/27002 requirements. The standard clearly states that “secure coding principles should be applied to software development” and lays out an extensive set of requirements that apply to writing secure code.

In a nutshell

- ISO 27001 defines guidelines for developing secure code, with a special emphasis on developer education.
- To be effective, developer security education needs to focus on practical issues based around the most common software security weaknesses.
- Motivating and empowering developers to create secure code remains the main challenge.

The ISO standard separates secure coding requirements into three phases (planning, during coding, and review and maintenance), also calling for implementing these practices throughout the SDLC. Further, secure coding principles must apply not only to in-house development but also to any open-source, third-party, and outsourced code. To do this, organizations need to be aware of the real-world threats they face and understand how software weaknesses can open the door to attackers.

Phase 1: Planning and before coding

ISO 27002 advises that the planning phase be used to draw up principles and expectations for secure coding for both in-house and outsourced development. Organizations should pay special attention to establishing developer competence in creating secure code. This will likely require developer training.

The standard also advises that development tools be regularly updated and properly configured to help enforce the coding standards. This includes defining strict access rights to ensure the privacy and security of code while it's being written. Threat modeling should play an integral role in the architecture and design of the application. This could entail defining use cases where the system is attacked or otherwise compromised.

Phase 2: During coding

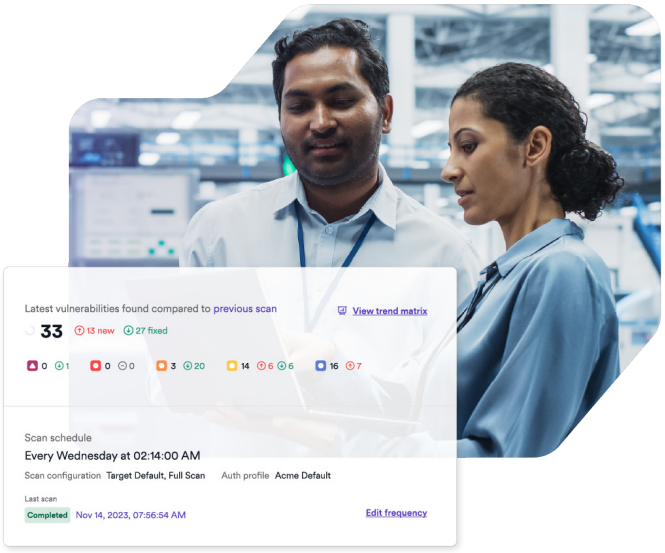
The ISO standard mandates defining "secure coding practices specific to the programming languages and techniques being used" and "prohibiting the use of insecure design techniques." Overall, secure coding practices span all levels of development. Some are language-independent, others, such as those related to proper memory management, only apply to C or C++, and others still apply to interpreted rather than compiled languages. Crucially, some are specific to web applications. One overriding best practice is to thoroughly validate any input retrieved from the user or an external source. Failure to validate incoming data opens the door to attacks such as SQL injection, cross-site scripting (XSS), and server-side request forgery (SSRF).

Other common weaknesses result from failed or insecure authentication of users. One cardinal rule is not to store passwords in a program where they can be read by an attacker. Instead, passwords should be stored in an external encrypted file or (preferably) only as hashes. Other errors include improper access control and failure to encrypt sensitive data. This can expose private, financial, or corporate data. Failure to encrypt sensitive personal data is not only a weakness, but it also can be illegal. ISO recommends applying the principle of least privilege, that is granting only the lowest level of access required to do a job.

As an ongoing check, the ISO standard requires at least SAST during development to verify that the code does not contain known security weaknesses and DAST later in the SDLC to catch dynamic issues that static analysis cannot find. Integrating these tools into the development pipeline allows security testing and remediation to become a routine part of development.

Phase 3: Review and maintenance

After deployment, the organization should keep monitoring for new threats, comparing these with its production applications, and respond, as needed, with updated coding standards. Attack logs can be a resource for determining necessary code adjustments to protect against new emerging threats. Regular vulnerability scanning and penetration testing can also reveal weaknesses that need to be eliminated from existing and future code.



Developer education

In the end, deploying secure applications depends on developers who are both able and willing to write secure code. This is hampered both by inadequate security training offered to developers (or demanded of them, in most organizations) and by the way that security is still often treated as an isolated concern. That separation can lead developers to believe that security simply isn't their responsibility. While formal training can be useful, in practice, educational resources such as Invicti Learn can be more effective in showing how to write secure code. Invicti Learn explains the most critical vulnerabilities and configuration errors that can open web applications to attacks and provides guidance to remedy and prevent them.

Providing hands-on and minimally intrusive educational assets can be crucial for motivating developers to consider security issues in their code, as can empowering developers to take the extra time and effort required to do so. In many cases, delivering features on schedule takes precedence over reworking perfectly functional code for security reasons. To maintain a solid security posture and minimize security issues in the long run, organizations need to establish a culture and process where security is an inherent part of software quality and where development teams are fully supported in all efforts to eliminate application vulnerabilities.

> Organizations need to put a lot of effort into developer education, but even more important than education is convincing developers that security works in their favor so they don't treat it as a chore. Attitude is often the biggest problem
>
> **Matthew Sciberras**
> CISO and VP of Information Security, Invicti

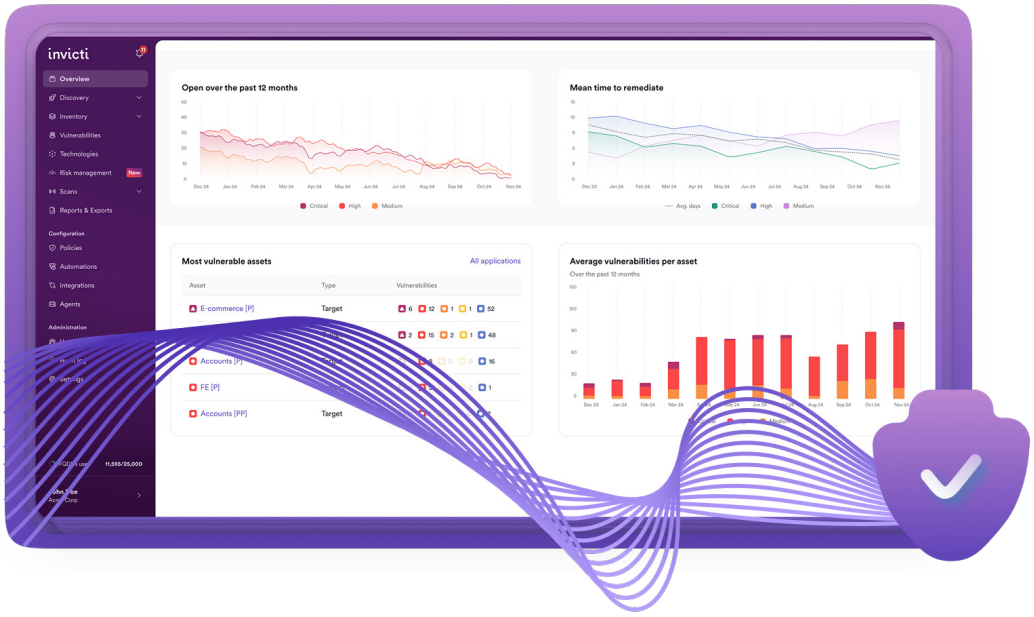
Application security testing in the SDLC

Ensuring comprehensive application security in compliance with ISO 27001 and 27002 requires the coordinated use of multiple approaches to security testing and application protection.

The web attack surface of modern organizations has grown dramatically during the decade since the 2013 edition of ISO 27001, as has their reliance on web applications to process sensitive data and provide business-critical functionality. At the same time, at least some in-house development is now the norm rather than the exception, lending weight to the well-worn cliché that every organization is now a software organization. Accordingly, ISO 27001 not only mentions the need for security testing throughout the SDLC but states outright: “Security testing processes should be defined and implemented in the development life cycle.”

In a nutshell

- ISO 27001 specifically calls for ensuring application security at all points in the software development life cycle.
- While vulnerability testing is one of the key requirements, a comprehensive approach that combines multiple methodologies is needed for compliance.
- Several common approaches to application security testing are in use and should be coordinated for maximum coverage.



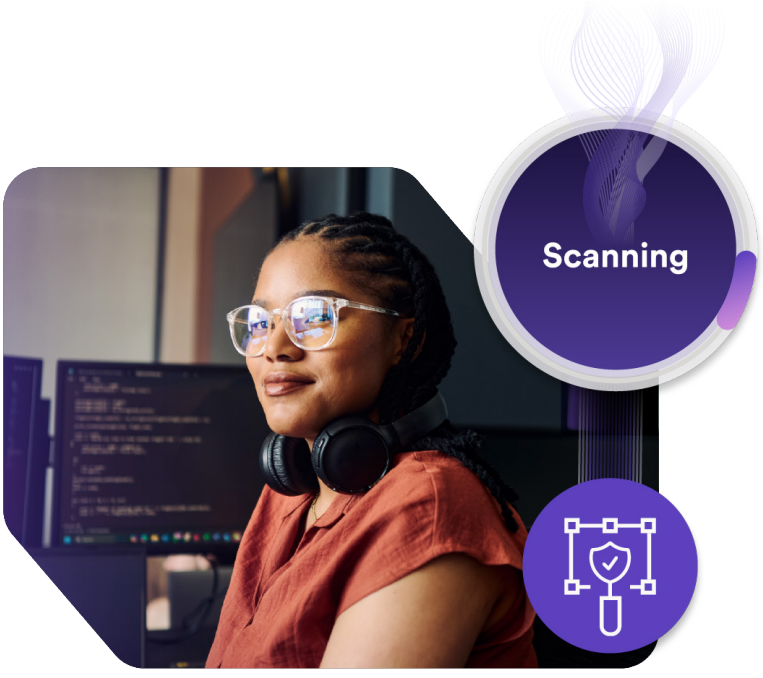
Multiple testing methods for securing the SDLC

The complexity and opacity of modern web application frameworks and architectures mean that security testing needs to cover multiple aspects and phases of development. Many tools are available to test application security at various stages of the SDLC—some look for weaknesses in code, others test user inputs in a running program, and yet others monitor a running program for possible intrusions. None of these tools is sufficient on its own, and each category of security testing tool has its place in a comprehensive security testing program.

To catch vulnerable code constructs and components early on in the coding cycle, before the application is in a runnable state, organizations typically look to static application security testing (SAST) and software composition analysis (SCA) tools. Because both require access to the source code, both types come under the umbrella of white-box (or inside-out) testing.

SAST tools scan the source code for potentially insecure syntax and data flows. They come in various shapes and sizes, from lint-type plugins for integrated development environments (IDEs) to standalone analysis programs. Most will integrate directly into build systems and issue-tracking systems to make security testing an integral part of the SDLC rather than an extra step that developers might be tempted to skip. One drawback of SAST tools is that, because they analyze source code but don't know the developer's intent, they tend to report a high proportion of false positives that need to be manually dismissed or painstakingly tuned out.

SCA tools don't analyze the source code directly but rather check for vulnerable open-source software components. With open-source software making up, on average, 78% of the code base of all deployed web applications, ensuring that you are not using a component with known vulnerabilities is crucial. Depending on the tool and your specific requirements, SCA may also include license compliance checking to make sure your use of open-source software doesn't violate company policy or the relevant open-source licenses.



Dynamic security testing from first build to production

SAST and SCA tools operate only on the static source code and its components, both for initial coding and for validating software updates in a continuous integration/continuous delivery (CI/CD) environment. Once an application is in a runnable state, whether in a test environment or deployed to a production system, you add dynamic and interactive application security testing into the mix. Both DAST and IAST tools interact with a running web application to detect vulnerabilities and provide runtime insights but differ in the scope of testing and requirements for source code access.

DAST tools perform black-box (outside-in) testing that does not require access to the source code, is technology-agnostic, and can be performed on any runnable application or prototype. While historically DAST was only run in staging and production, the integration and automation capabilities of the latest generation of tools allow it to be plugged into the pipeline in much the same way as SAST.

The limitation of requiring a running application to test is obviated by web frameworks now providing all the scaffolding to build runnable prototypes from an early stage of development. Depending on the solution, DAST can also incorporate code-level insights provided by interactive security testing.

IAST (gray-box) tools cover the middle ground between SAST and DAST to combine elements of dynamic and static testing. The category covers a wide variety of solutions and approaches, from scanner extensions to standalone products, with some requiring code modifications (instrumentation) to extract runtime insights during testing and others being triggered by a DAST scan. An example of a DAST-driven approach that provides true interaction between dynamic analysis and code insights is the Invicti IAST component. This runs as a server-side agent and exchanges information with the DAST scanner throughout the scan process. Notably for this approach, no code modifications are needed, and the performance impact is minimal.

> A comprehensive, multi-level and multi-methodology approach is the only way to truly improve the security of the SDLC
>
> **Matthew Sciberras**
> CISO and VP of Information Security, Invicti

Securing development workflows

The ISO 27001 standard requires IT organizations to define a comprehensive, overarching information security policy that incorporates the entire development workflow.

The most wide-ranging cyberattack on US government agencies and major software companies, namely the 2020 SolarWinds breach, was the result of cyber attackers hijacking part of the software development supply chain of a third-party network monitoring tool vendor. Similar attacks against open-source projects have been successful as well, such as the one involving the NetBeans Java development environment, which for years unwittingly shipped malware that had been introduced into its build system.

We will discuss supply-chain security in detail in the next chapter, but it's only one aspect of a bigger point: every organization that develops software must develop a policy to secure its entire development pipeline. This holds true both for internal and customer-facing software, and covers all platforms and technologies, notably including web applications. ISO 27001 provides a starting point for understanding how to develop security controls and policies for software development as part of the wider information security picture.

The standard (specifically its Requirement 6.2) calls for the development of a comprehensive, overarching information security policy and applicable objectives, "taking into account the information security requirements, results from risk assessment, and treatment." Objectives should be measurable, monitored, communicated, updated, and made available as documented information.

In a nutshell

- Software development and deployment involve multiple, often complex steps that can create opportunities for malware or vulnerabilities to go undetected in a system.
- Gaps in security can occur because of lax security enforcement, underappreciation of the possible risks a procedure or IT asset poses, or insufficient security testing coverage.
- The best protection comes from incorporating the entire development workflow and infrastructure into a detailed IT security plan that is enforced, monitored, and regularly updated.

When implementing policy changes, the IT organization must determine "what will be done, what resources will be required, who will be responsible, when it will be completed, and how the results will be evaluated."

As these steps illustrate, the standard is not an abstract normative document but rather a framework that requires active implementation. ISO 27001 makes it clear that the implementation of the security policy must be a living process that is properly communicated, enforced, and updated. Such vigilance can help staff spot and immediately address unanticipated gaps in security coverage and staff knowledge.

Where does your code come from?

Developers frequently search the web for answers to coding problems they encounter—problems as simple as how to use a data structure in a given language or as complex as how to implement a difficult algorithm. Forums such as StackOverflow are popular for these kinds of discussions, where contributors who answer queries will post the full code intended to remedy the problems at hand. In turn, many developers will copy and paste the supplied code, unchanged, into their product code. With AI tools advancing in leaps and bounds, automatically generated code carries the same risks on an even greater scale.

The possibility of unwittingly inserting insecure or downright malicious code is clearly a serious threat. But there are two other hidden risks. The first has to do with licensing: If the copied code comes from an open-source project, then the code is subject to the terms of an open-source license. In the most harmless scenario, this requires a statement distributed with the product acknowledging that some of its code is used under a specific license. However, for the widely used GPL and AGPL licenses, the code of the entire application must be released to all users, which may rule out some commercial usage. Static code analyzers today can spot code that is likely taken from an open-source project, and a policy must be in place to use such tools on a regular basis across the entire codebase to minimize the risk of non-compliance.

A related threat arises when developers bring in dynamic dependencies that incorporate third-party code into the application. This is a particularly common practice in JavaScript code in web applications. In this setup, the code is brought into the program every time the application is run. While there is a risk that the code could be modified for malicious purposes, it can also be modified with no evil intent and prevent an application from running correctly or even running at all. In an extreme example, back in 2016, a developer deleted from his personal repository a simple 11-line function that enabled characters to be added to the start of a string. Thousands of web applications, including those of several global technology giants, suddenly stopped working until the deleted lines were restored.

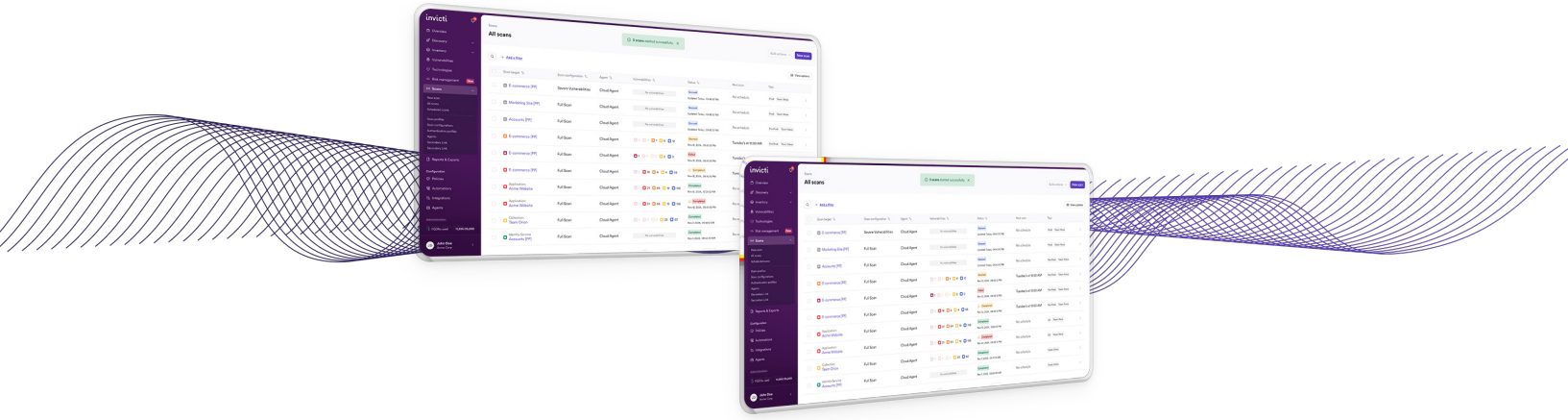
Have you tested that app in real life?

Developers understand the importance of testing their code and functionality, with unit tests, integration tests, and user-acceptance tests all being established practices. But if security is to leave no gaps, code-level testing must be followed by security checks on running web applications. DAST scanners search for entry points, vulnerabilities, and other weaknesses that could be exploited by malicious actors. While DAST tools can and should be run after deployment, restricting them to this stage could leave a window for attacks against new releases. The best practice is thus to also test each web app in a staging environment that mirrors the production environment as closely as possible to find a DAST tool to search for vulnerabilities before moving into production.

Spotting and eliminating application security gaps starts with secure coding practices, code reviews, and static analysis. Then dynamic testing comes in, overlapping with static testing in the development phase and (ideally) progressively reducing the number of vulnerabilities at each subsequent stage. But even after passing all security tests in staging, web applications still need regular scanning in production, for two main reasons.

Firstly, the production environment may be configured differently than in staging and is likely to change even more as time goes by. Secondly, and more importantly, new vulnerabilities and attack techniques are discovered daily, so it's crucial to regularly rescan your entire web environment with a well-maintained tool that tests for all the latest tricks available to attackers.

> Companies nowadays are required to adopt best practices around SDLC, including software supply chain management. Security is as strong as it's weakest link, thus obtaining a secure pipeline is fundamental in today's risk appetite.
>
> **Matthew Sciberras**
> CISO and VP of Information Security, Invicti



Software supply chain security

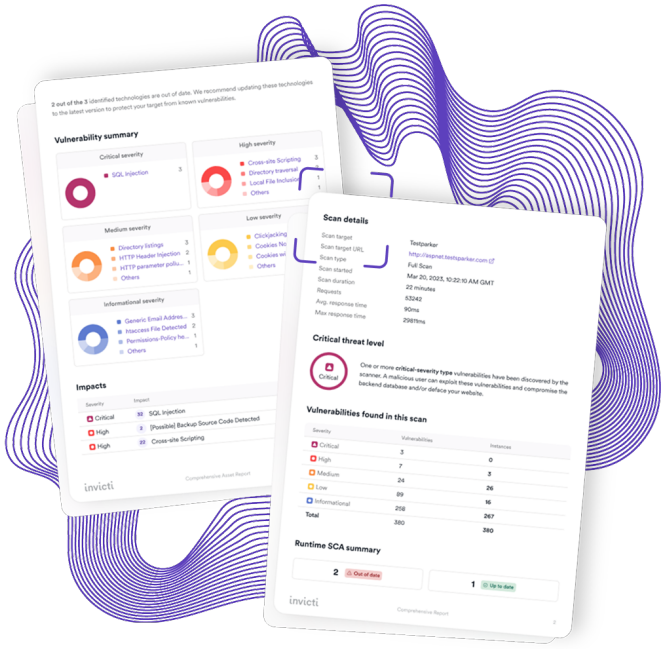
The majority of your web application code is most likely written by third parties—but you're still the one responsible for security. ISO 27001 requirements and guidelines for information security management thus also include securing third-party software and cloud infrastructure.

It's no secret that all your code is not your own, with the vast majority of web application code consisting of open-source and third-party libraries (including any outsourced code) alongside a small percentage of code developed in-house. Moreover, not only do you not own all your application code, but there's a good chance the platform on which the application runs is also made up of third-party software: cloud services, web servers, networking software, and operating systems. And yet, if there's a data breach, your customers don't care whether some third party wrote the software that was compromised—they'll hold you responsible.

The collaborative nature of modern software is clearly recognized in ISO 27001, with organizations required to "identify and implement processes and procedures to address security risks associated with the use of products and services provided by suppliers." Although this is a daunting task, both documents lay out guiding principles for protecting outsourced and third-party code as well as cloud services.

In a nutshell

- ISO 27001 includes requirements to negotiate responsibilities with an outsourcing supplier for delivering secure code.
- Organizations are required to test the security of third-party components even with no access to the source code, making DAST and manual penetration testing essential.
- Securing the cloud platforms your applications run on requires a close partnership with the cloud service provider to define and monitor security controls.



Third-party software still needs security testing

It makes good business sense for an organization to use third-party libraries for common tasks such as handling network operations or rendering the user interface.

Such pre-written code is often stable, debugged, and ready to run, saving a lot of development and testing effort. But widely-used code can also make an easy target for attackers looking for a big payback on their efforts. Fortunately, the security community continually monitors popular platforms and software for weaknesses or security breaches.

ISO recommends that organizations keep an eye on disclosures and apply patches and updates promptly when available. Regression testing must follow to verify that existing code still works as intended.

> An organization cannot accept third-party software as-is—it must perform security testing. SAST and SCA are a good starting point for open-source code, but for libraries accessed through an API where the source is unavailable, automated DAST and manual penetration testing are the only options.
>
> **Matthew Sciberras**
> CISO and VP of Information Security, Invicti

Guidelines for securing outsourced code

The advantages to outsourcing development are many, but the main advantage is that the outsourcing supplier can contribute skills lacking in your organization. As with code developed in-house, however, that outsourced code can carry security risks. The responsibility for protecting data remains with your organization, so it's worth looking to ISO 27002 for security best practices for all stages of outsourced development.

The first step ISO recommends is researching the outsourcing supplier: its reputation, documentation, and certifications. Given that the supplier will have access to your organization's data, you should pay special attention to documented security practices.

Next, it's time to negotiate a strong contract. ISO says the contract should clearly delineate the responsibilities of both parties, including non-disclosure agreements where appropriate. The contract should also establish ownership of the completed code and intellectual property. Procedures and policies for secure design, coding, and testing should be written into the contract, with an option to audit those procedures.

Access control is another crucial consideration. During development, your organization should provide the appropriate access level for any resources needed by the supplier, and both parties should establish secure procedures for code delivery. Upon termination of the contract (either because the software was delivered or the supplier failed in its obligations), your organization should remove any access rights granted to the supplier, and the supplier should destroy all copies of your data and return any assets. And if at any time the outsourcing supplier becomes aware of a data breach involving its code, it should be contractually obligated to promptly notify your organization and work with you to remedy the situation.

Crucially, both the supplier and your organization need to perform security testing. SAST can be used during development because you will have access to the source code, but DAST is also essential both during development and after deployment. Once the code is deployed, you should continue to monitor the supplier's security procedures and practices to keep up with any reported vulnerabilities affecting third-party software used in the supplier's code.

Security requirements for cloud services and platforms

When it comes to cloud infrastructure, ISO 27002 recommends that you negotiate a dedicated agreement with your cloud service provider to incorporate security. In the agreement, the provider should be required to use industry-standard architecture and infrastructure. It must also protect your organization's data by applying secure access controls and ensuring appropriate handling for any sensitive data. The provider's obligations should also include monitoring for intrusions and malware as well as ensuring dedicated support in gathering evidence should a breach occur. If the provider subcontracts any of its services, the same contractual terms must apply to subcontractors. And to cover the entire lifecycle, upon contract termination, the provider must return all data and configuration files to the organization and properly remove your data from its systems.

ISO 27001 is for life, not just for certification

No matter how complex their business processes, software environments, and development workflows, organizations bear full responsibility for the security of their data—and that of their customers. With web applications constituting your most exposed attack surface, securing application development and deployment is now a non-negotiable priority for both information security professionals and engineering leaders. While ISO 27001 is most often discussed purely in terms of ensuring compliance and certification, it can also provide valuable high-level insights and guidelines for designing and implementing an effective application security program.

One of the biggest changes in the current edition of ISO 27001 was to reorganize and spell out controls for ensuring application security all across the software lifecycle, notably requiring security testing at every phase of development and deployment, regardless of where the code originated. This means a comprehensive approach to testing that combines multiple methodologies, including SAST and SCA, and incorporates vulnerability scanning using DAST solutions as the only way to automatically test all running applications, regardless of origin, mode of deployment, or underlying technologies.

The bottom line is that vulnerabilities cannot be ignored at any point in the software development life cycle. Certainly, identifying and managing all possible risks can be daunting, but when faced with the possibility of an attack that could derail development or, worse, cause a breach for customers of deployed software, testing for vulnerabilities both early and often is more than an obvious solution—it's the specified standard.

Invicti Security provides a centralized application security platform that helps organizations prove and reduce real application risk with zero noise. Combining application security posture management (ASPM) with discovery and scanning, Invicti gives security and development teams a single, correlated view of exploitable vulnerabilities across their application frontends and APIs. With its best-of-breed dynamic application security testing (DAST) acting as the fact-checker, Invicti integrates into CI/CD pipelines to deliver proof-based results supported by AI-powered prioritization and remediation.

Built on more than 20 years of DAST innovation through Acunetix and Netsparker and further strengthened by the acquisition of Kondukto ASPM, Invicti operates globally across more than 11 countries and serves over 4,000 customer organizations.

Acunetix & Netsparker & kondukto

ARE NOW

invicti

Invicti's solutions automate application vulnerability identification, confirmation, and management to help keep sensitive information and critical infrastructure secure.

Find Us LinkedIn X (Twitter) Facebook Instagram invicti.com Contact Us