# Security Standard.

# Background

Having a secure approach to development has never been so important.

The way we build software and systems is rapidly evolving, becoming more and more automated and integrated. This results in a need to have some standards and guidance around security. To save our time looking up and picking ways and more importantly to keep things uniform so things are done in the best way across the board

This guidance will help you understand the security implications of modern code development and deployment practices.

Principles

Secure development principles

1. **Secure development is everyone's concern**

Genuine security benefits can only be realised when delivery teams weave security into their everyday working practices.

1. **Keep your security knowledge sharp**

Creating code that is capable of withstanding attack requires an understanding of attack types and of defensive security practices.

1. **Produce clean & maintainable code**

If your code lacks consistency, is poorly laid out and undocumented, you're adding to the overall complexity of your system.

1. **Secure your development environment**

There is sometimes a perceived conflict between security and usability. This situation is highlighted in the case of end user devices and the environments used to support software development.

1. **Protect your code repository**

Your code is only as secure as the systems used to create it. As the central point at which your code is stored and managed, it's crucial that the repository is sufficiently secure.

1. **Secure the build and deployment pipeline**

Continuous integration, delivery and deployment are modern approaches to the building, testing and deployment of IT systems.

1. **Continually test your security**

Security testing can be manual, but it can also be automated.

1. **Plan for security flaws**

All but the very simplest software is likely to contain bugs, some of which may have a security impact.

For more detailed information on the above principals see here:

<https://www.ncsc.gov.uk/collection/developers-collection?curPage=/collection/developers-collection/principles>

Standards

Coding security standards are defined in this document:



It is taken from the OWASP here:

<https://www.owasp.org/index.php/OWASP_Secure_Coding_Practices_-_Quick_Reference_Guide>

Guidance

* Logically architect the layout of the code  
  Writing clean and maintainable code is much easier when it's clear which components belong where. The architecture should take into account potential expansion. The [SOLID](https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)) principles provide an example of this approach.
* Coding standards  
  Follow secure coding standards where available to avoid potential security vulnerabilities.
* Use self-explanatory naming conventions  
  To help the reader understand what is going on, classes, methods, functions, file names and folder names should all be self-explanatory
* Provide good supportive tooling to developers  
  This may include modern IDEs with plugins that support the developer with capabilities such as pre-defined code snippets, debugging, unit testing, security hints, 'linting', specification checking and annotations.
* Maintain a consistent coding style  
  Understanding code becomes more difficult when different coding styles used by different developers mix and intertwine.
* Clearly outline code block responsibilities  
  Security issues can arise when one code component incorrectly assumes another has taken responsibility for an action. For example, when validating potentially malicious input at the border of your application. One way to achieve this is to have a comment block at the top of every method or function.
* Separate secret credentials  
  Keep secrets such as passwords and private keys logically isolated from the core code base. This will help prevent them being checked in to public code repositories. Hard coding credentials in source code is bad practice.
* Do small and regular code commits  
  Performing effective review becomes more difficult when large changes are made with each commit. Small and clearly labelled commits simplify the review and roll back process.
* Attribute code changes to an author  
  It should be clear who has authored a code change, and strong authentication controls should be used to do so. It should not be possible to falsify the code's author or its review status.
* Police and critique each other's work through peer review  
  Encourage a culture that does not accept complicated, confusing or insecure coding practices. Peer review helps prevent such issues being incorporated into your code base. Feedback helps support education within your team. Using pull requests and comments is one way to achieve this.
* Team communications  
  When multiple team members are working on the same code base, there should be strong and regular communication channels between them. The aim here is to avoid the following scenario: 'I thought you were securing that component!'. Keeping teams physically close to one-another, or providing real-time chat channels are two ways to achieve this.
* Document and comment clearly and consistently  
  Clear and concise documentation should support your product. This may be as a result of self-documenting code, code comments, or supportive material. Documentation should be kept up to date, as a system evolves. Old and out-of-date documentation is difficult to trust and could be damaging for security if it's interpreted incorrectly.
* Support new team members  
  Developers and other team members may come and go over the life span of a product. To ensure adequate knowledge of the product is maintained, provide good support and documentation to new team members. After all, who will fix security vulnerabilities left behind by previous developers?
* Check return values and handle errors appropriately  
  Checking for errors at the point of a call and handling them immediately means that your code doesn't continue running in a potentially unstable state. This will make your code more robust and secure.