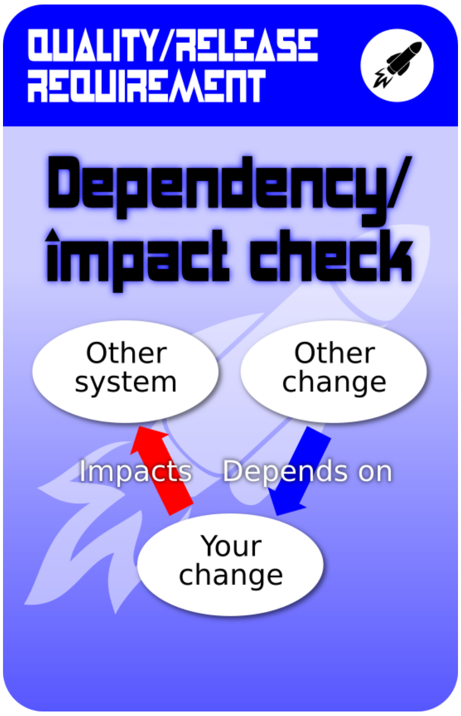
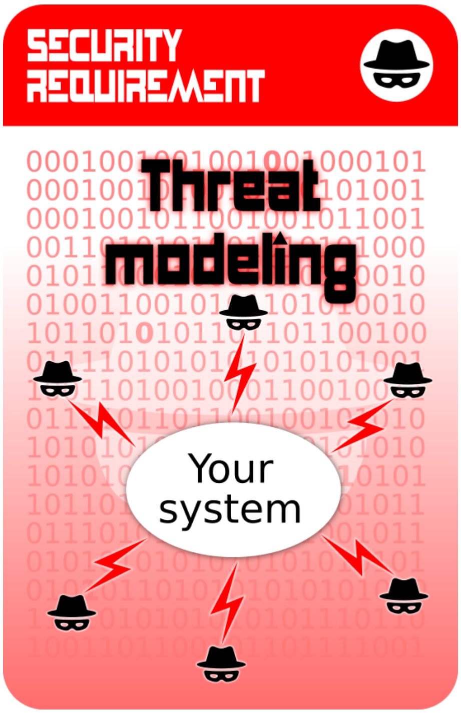
CI-CD-QA Glossary

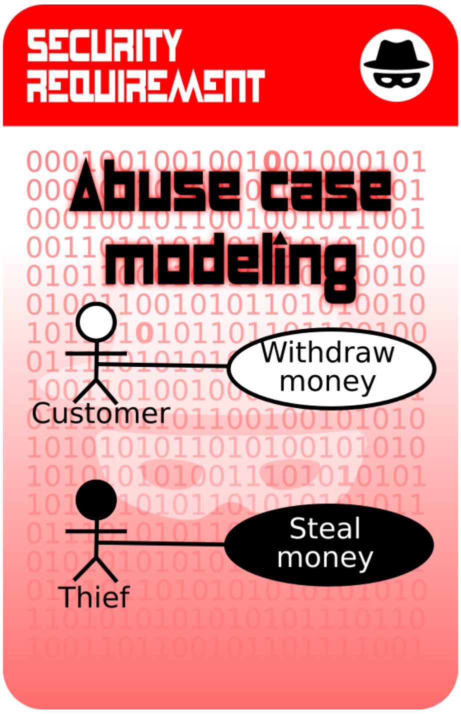
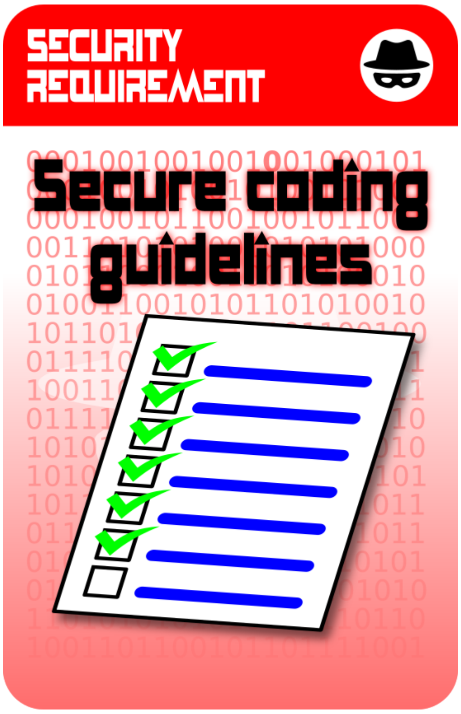
Terminology used in the game

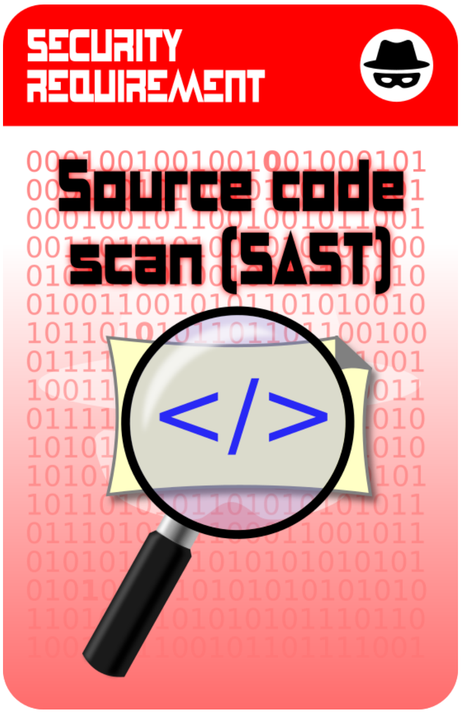
QUALITY/RELEASE REQUIREMENT Determine release date M T W T F S S	Determining the release date is particularly important when still delivering big batch releases in a project driven organization. You determine the release date up-front, so that you can make arrangements with dependencies so that they can deliver their changes before or together with your changes, on that release date. Often this is driven by a predefined calendar. In an agile approach though, this is quite different. You don't deal with a project but with a product and you deliver (preferably) small increments.You deploy (install) frequently and release (activate) on demand. In that case you will determine a release date but not so long up-front.
 QUALITY/RELEASE REQUIREMENT Dependency/ impact check Other system Other change Impacts Depends on Your change	In a traditional, calendar based release approach (and definitely with more monolith types of systems) it is important to know up-front what your dependencies are: which system is impacted by your change and which systems should implement a change for you. Modern applications are built with loose coupling in mind, to make sure that teams can deliver their changes as independently as possible. And if there are dependencies, a good value stream management tool can come handy here.

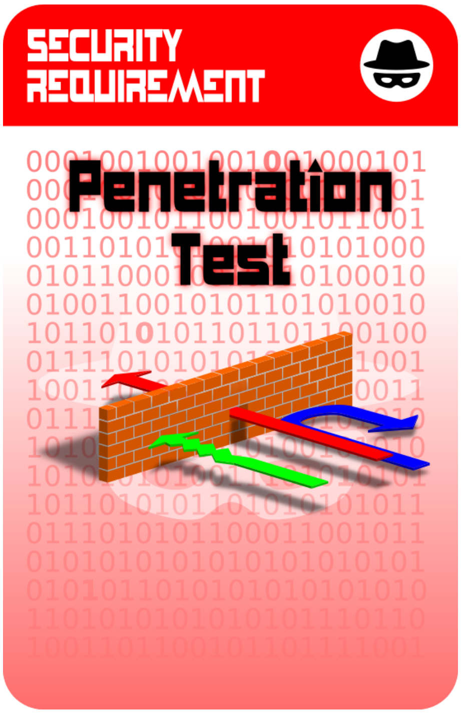
QUALITY/RELEASE REQUIREMENT Test progress report Test 1 Test 2 Test 3 Test 4 Test 5 Test 6 Test 7	When executing tests, you want to see which tests are executed and which of these succeeded and which failed. The test progress report is especially important when executing manual tests, to see . In case of automated tests, you usually execute an entire batch of tests and you will only see the pass/fail status. With automated tests this is always up to date with every executed test cycle.
QUALITY/RELEASE REQUIREMENT Start Business as Usual tests BUSINESS as USUAL	Business as usual tests are a specific test set to validate that the most used features of an application still work properly after the implementation of new or changed features. In a traditional approach these are usually done close to the production release to make sure that no regression was introduced. In case of automated tests, these can be executed at any time, be it when implementing a small change, a large change, refactoring or a bug fix.

 	A bug report is always important: it gives you an idea about the quality of the product under development, the changes being implemented. The number of unsolved bugs and their priority is the most important criterion to evaluate the quality of the release/change.
 	You can develop an application so that it meets the functional requirements - does what it is supposed to do - but if it takes too long before you get a response or response times increase with the number of concurrent users, your application becomes unusable. People will stop. Therefore performance testing is important to see how your application behaves, both with normal expected load and exceptional load. Good tools help you even identify the bottlenecks in your end-to-end request execution, so that you can address the specific performance issue.

	This is kind of typical for a project based approach with long calendar based release cycles. Before you can deploy/release to production, you have to pass the go/no go meeting. Typically things like the test progress, defects list and outcome of the penetration tests are key indicators to decide whether a project (or change set) is ready to go to production. Beware that this does not become a Go/Go meeting (with stakeholders pushing to go to production, because some promises were made or some campaign was already planned).
	Threat modeling is a form of security analysis that makes you think as a hacker. Based on the architecture of the solution to be developed, you try to identify as many vulnerabilities as possible and plan appropriate actions to mitigate these vulnerabilities.

 SECURITY REQUIREMENT Abuse case modeling Withdraw money Customer Steal money Thief	Abuse case modeling is similar to threat modeling, but in this case it is not based on the architecture of the solution but on the use cases. For each use case in your solution you try to identify situations in which malicious users can exploit weaknesses in the features that your application provides.
	These are a set of principles, guidelines, patterns to use or ready to use libraries that avoid security breaches. These guidelines can be the answer to deal with the OWASP top 10 security vulnerabilities. Examples of such guidelines are (not exhaustive): ● do proper secrets management, never have **passwords** in your code ● apply encryption where necessary ● only accept TLS communications, in specific cases even mutual TLS ● never trust user input, always validate to avoid cross site scripting, SQL injection, and other malicious code ● …

SECURITY REQUIREMENT 3rd party library scan lib </>	3rd party library scans are meant for detecting security vulnerabilities in open source libraries your application depends on. It is not a “scan” as such, but instead it checks whether the version of a library you are using is known in a central database of Common Vulnerabilities and Exposures (CVE), and if so, what the severity of the vulnerability is. Tools that do this vulnerability scanning usually also help developers take appropriate actions to mitigate the vulnerability (e.g. suggest which version to use instead, and also generate a pull request for that library in the version management system).
	SAST stands for Static Application Security Testing. The purpose of a SAST tool is to detect security vulnerabilities in your source code, by applying static analysis (scanning source code and trying to identify vulnerable patterns).

SECURITY REQUIREMENT SIEM log aggregation .log .log .log .log Security Information and Event Management system	SIEM stands for Security Information and Event Management. A SIEM system collects logs, analysis logs and correlates information from different log files - both from technical infrastructure and business applications - in order to detect suspicious behavior early, so that appropriate actions can be taken (e.g. detect that a hacker is trying to access your systems and retrieve/destroy information).
	Penetration testing is your last resort for security testing. You let someone (often an external specialized instance) test your application and try to find vulnerabilities that need to be fixed. This kind of test happens with a software version that is as close to the production release as possible and is executed in a non-production environment. Because of the late moment in the development trajectory, it is sometimes hard to fix vulnerabilities that require a lot of effort (which may result in either acceptance of the risk or postponing the release). Penetration testing should therefore be considered as your safety net, if a vulnerability could still sneak through all the preceding actions.