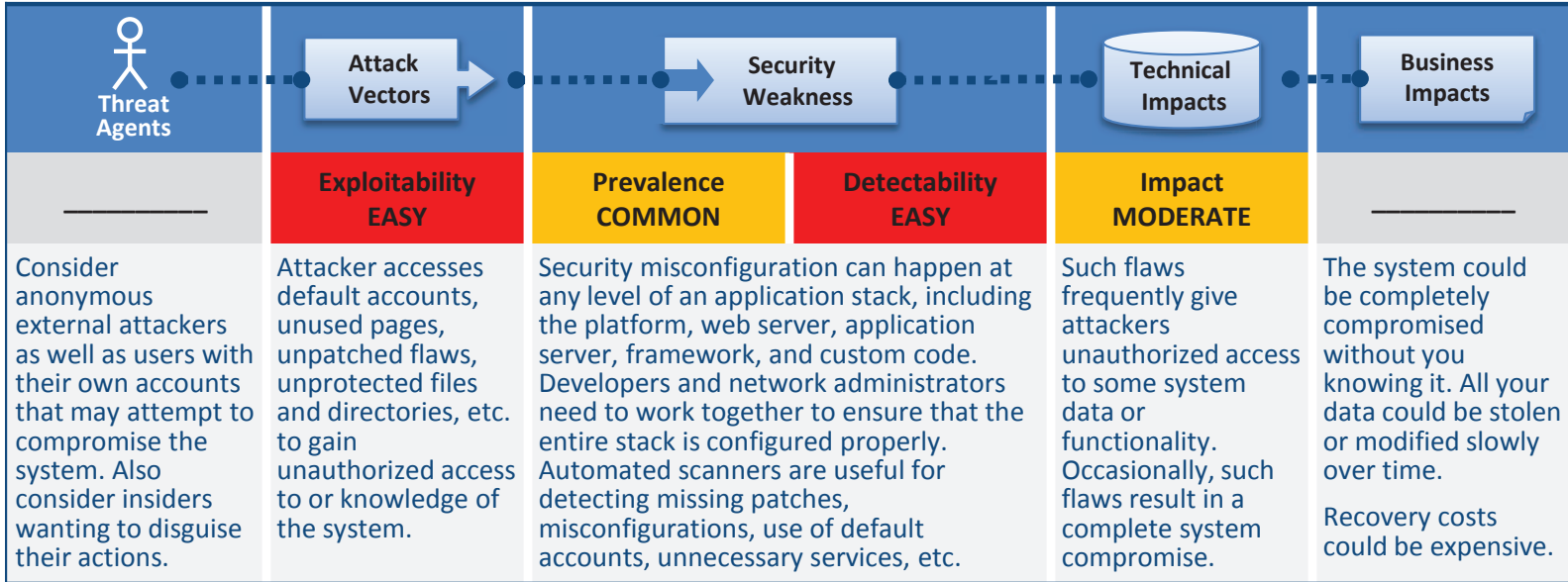


A6

Security Misconfiguration



Am I Vulnerable?

Have you performed the proper security hardening across the entire application stack?

1. Do you have a process for keeping all your software up to date? This includes the OS, Web/App Server, DBMS, applications, and **all code libraries**.
2. Is everything unnecessary disabled, removed, or not installed (e.g. ports, services, pages, accounts, privileges)?
3. Are default account passwords changed or disabled?
4. Is your error handling set up to prevent stack traces and other overly informative error messages from leaking?
5. Are the security settings in your development frameworks (e.g., Struts, Spring, ASP.NET) and libraries understood and configured properly?

A concerted, repeatable process is required to develop and maintain a proper application security configuration.

How Do I Prevent This?

The primary recommendations are to establish all of the following:

1. A repeatable hardening process that makes it fast and easy to deploy another environment that is properly locked down. Development, QA, and production environments should all be configured identically. This process should be automated to minimize the effort required to setup a new secure environment.
2. A process for keeping abreast of and deploying all new software updates and patches in a timely manner to each deployed environment. This needs to include **all code libraries as well**, which are frequently overlooked.
3. A strong application architecture that provides good separation and security between components.
4. Consider running scans and doing audits periodically to help detect future misconfigurations or missing patches.

Example Attack Scenarios

Scenario #1: Your application relies on a powerful framework like Struts or Spring. XSS flaws are found in these framework components you rely on. An update is released to fix these flaws but you don't update your libraries. Until you do, attackers can easily find and exploit these flaws in your app.

Scenario #2: The app server admin console is automatically installed and not removed. Default accounts aren't changed. Attacker discovers the standard admin pages are on your server, logs in with default passwords, and takes over.

Scenario #3: Directory listing is not disabled on your server. Attacker discovers she can simply list directories to find any file. Attacker finds and downloads all your compiled Java classes, which she reverses to get all your custom code. She then finds a serious access control flaw in your application.

Scenario #4: App server configuration allows stack traces to be returned to users, potentially exposing underlying flaws. Attackers love the extra information error messages provide.

References

OWASP

- [OWASP Development Guide: Chapter on Configuration](#)
- [OWASP Code Review Guide: Chapter on Error Handling](#)
- [OWASP Testing Guide: Configuration Management](#)
- [OWASP Testing Guide: Testing for Error Codes](#)
- [OWASP Top 10 2004 - Insecure Configuration Management](#)

For additional requirements in this area, see the [ASVS requirements area for Security Configuration \(V12\)](#).

External

- [PC Magazine Article on Web Server Hardening](#)
- [CWE Entry 2 on Environmental Security Flaws](#)
- [CIS Security Configuration Guides/Benchmarks](#)