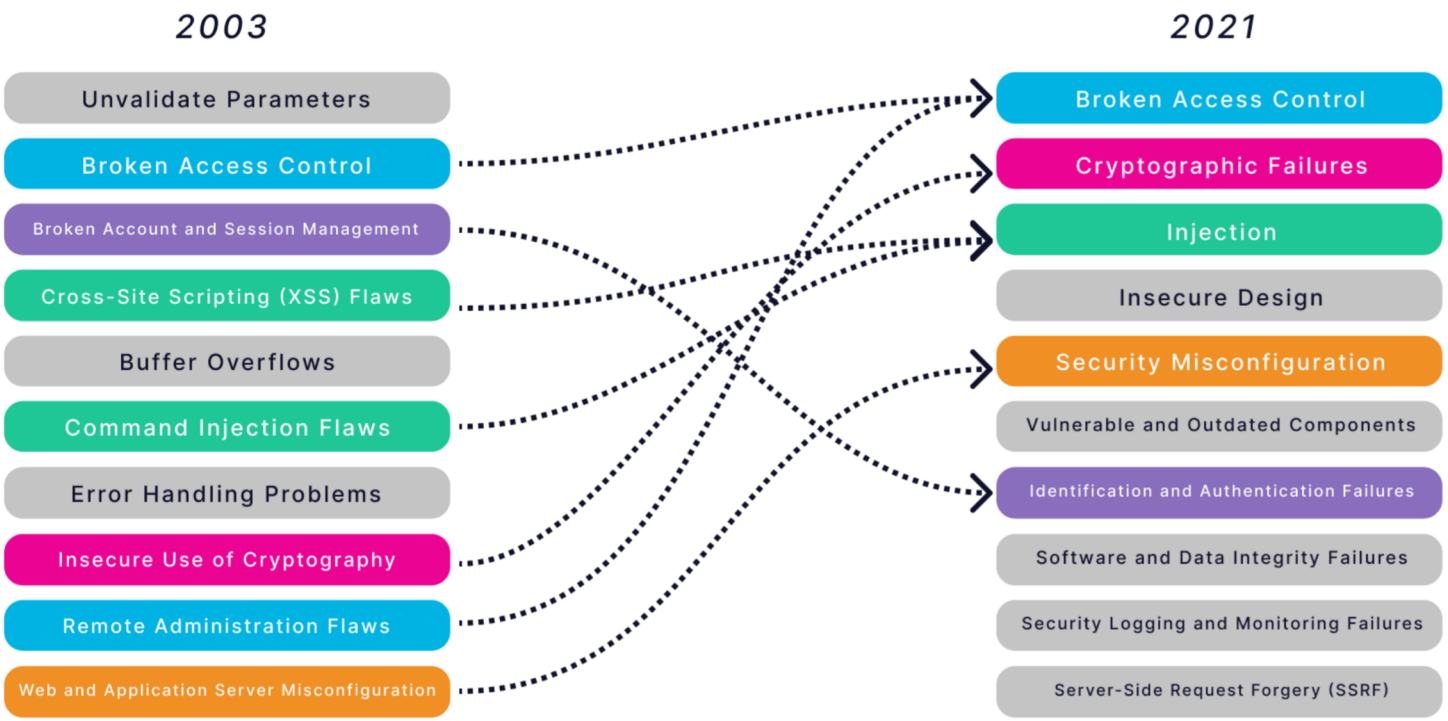
OWASP®

The Ultimate OWASP Top 10 Cheatsheet

GET TO GRIPS WITH THE NEW TOP 10

Since 2003, the OWASP Top 10 has become the de facto generic vulnerability standard for many in the industry. It offers a valuable insight into where we as an industry are heading, as well as which areas we're still struggling to resolve.

We've put this handy cheat sheet together to help you get to grips with the new Top 10 and make the most out of this invaluable tool.



BROKEN ACCESS CONTROL

Access control limits the exposure of resources, including data and application features, to unauthorized entities. This category covers vulnerabilities that allow for unauthorized access, either by a human or by other systems, to the resources you want to protect.

Examples of broken access control

- ✗ Allowing any authenticated user to be able to access the administrative page of the application.
- ✗ Allowing public access to an S3 bucket.
- ✗ Allowing an authenticated user to access another user's account details by entering the other user's ID in the URL. This is also referred to as Insecure Direct Object Reference (or IDOR).

Remediation and prevention

- ✓ Verify an entity requesting access to protected resources, ensuring that they have sufficient permissions or roles to access the requested resource.

Broken access control in action

Vulnerability in Travis CI potentially exposed secret data from public open-source projects on their platform.

In September 2021, a flaw was uncovered in software-testing solution, Travis CI, whereby the secret data associated with build repositories was made accessible to any project that cloned that repository. Appropriate access validation was not performed on the entities that obtained the resources, allowing for unauthorized public access to sensitive data, including certificate private keys (used for digital signing) or API credentials to services used during the build process.

CRYPTOGRAPHIC FAILURES

This category used to be referred to as 'Sensitive Data Exposure'. Protecting sensitive information is an important step to securing an application, and cryptographic functions are often used to do this; for example, by using encryption to encrypt a database password, hashing a user's password, or correctly configuring TLS within an application.

This category covers vulnerabilities where weak or flawed cryptographic configuration and/or implementations have been used, thereby risking the exposure of sensitive data.

Examples of cryptographic failures

- ✗ Storing user passwords as plaintext within a database.
- ✗ Using the SSLv3 protocol over the TLSv1.2 protocol for HTTPS connections.
- ✗ Missing certificate verification.

Remediation and prevention

- ✓ *Don't roll your own crypto.* This is one of the most important things we hear in the security realm, and luckily it makes your life easier. It means you should look to use existing algorithms and implementations that have been appropriately vetted and tested.

You should also only use modern algorithms, as older algorithms typically have known flaws, and appropriate cryptographic or authorization controls to protect any sensitive data.

Cryptographic failures in action

Android fitness app, VeryFitPro, was found to be transmitting sensitive data in plaintext.

In June 2021, it was reported that an Android fitness application with around 10 million downloads was sending its data in plain text. This is a prime example of a cryptographic failure that potentially leaked sensitive data, including passwords, to attackers.

INJECTION

Numerous technologies are capable of interpreting or translating data into a form they can understand and consume. Additional data can be inserted (or injected) into existing data to be interpreted and used by the system. This is known as an injection vulnerability.

Examples of injection

- ❌ SQL injection (SQLi)
- ❌ Cross-site scripting (XSS)
- ❌ JSON injection
- ❌ Template injection

Remediation and prevention

✓ In the past, we placed a lot of emphasis on validating all inputs. This should still be performed where possible, but we must acknowledge that it's becoming increasingly difficult as systems start to have more and more inputs.

Nowadays, we need to ensure the data is properly escaped or encoded when it's being translated into the final data structure; for example, by performing HTML encoding on a field displayed to users, or escaping quotation marks in an SQL query when including an input field in the query.

Injection vulnerabilities in action

SQL injection vulnerabilities found in Sophos' Cyberoam products.

In December 2020, Sophos released patches to fix SQL injection vulnerabilities in their Cyberoam products. This vulnerability allowed attackers to remotely add accounts to the operating system running on these devices if the device's administration interface was publicly exposed.

INSECURE DESIGN

Designing security from the beginning is part of the “shift left” mantra. Ensuring security is baked in from the beginning of the development process will result in a more secure application or feature. Leaving these types of decisions until after implementation will likely be more taxing and costly to address, and in some cases may never be addressed at all. Many insecure design flaws can also be attributed to business logic vulnerabilities.

Examples of insecure design

- ✗ A password reset feature with a flaw that allows anyone to specify which email address the reset email should be sent to.
- ✗ A commerce site that allows users to claim a limited set of discount codes, but wrongly assumes that only humans will be accessing the feature, thereby allowing a bot to claim all the discount codes.

Remediation and prevention

- ✓ Ensure any new feature or application has an appropriate security review at the planning and designing stage to identify any concerns that need to be catered for.
- ✓ Apply threat modeling when planning and designing new requirements.

Insecure design in action

Flaws in Kaseya VSA opened it up to attack from REvil ransomware.

In July 2021 attackers used flaws within Kaseya VSA to distribute the REvil ransomware. It’s believed that CVE-2021-30116 (a credentials leak and business logic vulnerability) was a contributing factor to this high profile attack.

SECURITY MISCONFIGURATION

An application is only as secure as the environment it operates in. Technology has many configuration settings which can be changed and tweaked, some of which are security-focused or can have an impact on the security of the system. Vulnerabilities covered by this category point to where a system configuration was incorrectly or insecurely set.

Examples of security misconfiguration

- ✗ Enabling directory listing on a web server.
- ✗ Disabling authentication on an S3 bucket.
- ✗ Using a certificate verification configuration that disables all certificate verification.

Remediation and prevention

- ✓ Make sure you understand the configuration settings of the system or feature. Always refer to the vendor's documentation as well as other reputable materials to ensure you understand the impact of changing or tweaking settings, and have recommendations for the appropriate configuration. This could also apply to code.

Security misconfiguration in action

Misconfiguration of a cloud database led to the public exposure of over a million records.

In September 2021, it was reported that Indonesia's Ministry of Communications and Informatics did not correctly configure its cloud database. This resulted in a million records from the country's COVID-19 quarantine management system being publicly exposed on the Internet.

VULNERABLE AND OUTDATED COMPONENTS

Over recent years, there has been an explosion of applications and systems leveraging third-party components such as open-source libraries and frameworks. This is fantastic from a delivery perspective since it allows for faster delivery; however, like most code, these components have vulnerabilities of their own or fall out of support. The result is that the application or service which uses the component could also be vulnerable.

Examples

- ✗ Using version 8 of Jetty, which is now End-of-Life (EOL).
- ✗ Using version 2.5.0 of the TensorFlow library, which contains the vulnerability CVE-2021-37678.
- ✗ Running your application on an unpatched version of CentOS 6.

Remediation and prevention

- ✓ The very first step you should take is to appropriately track where components are being used.
- ✓ Once this has been achieved, the next step is to monitor those components for known flaws and vulnerabilities, as well as any indications that the version is no longer going to be supported.
- ✓ The final step is to update components with identified vulnerabilities and flaws. It is also important to update components that will have support dropped for a specific version, to ensure you receive any security updates for that component.

Vulnerable and outdated components in action

Equifax suffered an expensive, highly publicized breach via a vulnerable Apache Struts library.

In September 2017, Equifax notified customers that they had suffered a data breach. The root cause of the breach was the result of Equifax using a vulnerable Apache Struts library. It was estimated that the incident could have cost the company as much as $1.38 billion.

IDENTIFICATION AND AUTHENTICATION FAILURES

Authentication is the process of identifying who is accessing a resource, so it's important that it is robust and effective. If there are any flaws during this identification process it could allow others – including those with malicious intent – to access the resource.

Examples of identification and authentication failures

- ✗ Using a JWT for authentication, but not validating the signature of the token.
- ✗ Not correctly associating an authentication token with the user performing authentication, thereby allowing anyone with a token to authenticate as any user.

Remediation and prevention

- ✓ Where possible, use defined authentication protocols and systems such as Security Assertion Markup Language (SAML).

- ✓ If you decide to implement your own authentication process, ensure that you perform appropriate validation of the entity requesting access to the resource, so they can prove who they say they are. This should include methods such as password and multi-factor authentication, and ensuring any identifiers (such as session cookies or tokens) issued cannot be tampered with.

Identification and authentication failures in action

Flaw in Apple's SSO service allowed full account takeover.

In May 2020 a flaw was identified in Apple's Sign in with Apple SSO service. It was present in the verification of the JSON Web Token (JWT) used as part of this service, allowing an individual to alter an existing JWT to contain the email address of another user. This vulnerability allowed for a full account takeover of victim accounts.

SOFTWARE AND DATA INTEGRITY FAILURES

Applications and services today have become more and more integrated with one another. The result is that the exchange of data has grown significantly. This could include data that is serialized and transmitted to another system or a web application that makes use of externally hosted JavaScript libraries. Ensuring this data has not been tampered with is an important security function, as not doing so could allow attackers to potentially gain full access to the application, service or data.

Examples

- ✗ Leveraging a SaaS-based chat service that involves including JavaScript on your site, but the application does not implement the appropriate Subresource Integrity (SRI) or Content Security Policy (CSP) policies to ensure that it has not been tampered with.
- ✗ An application does not perform appropriate validation on data from external services when deserializing that data.
- ✗ A framework is downloaded from a repository and the signature of the package is not validated.

Remediation and prevention

- ✓ When consuming data and external software, it is important to ensure the data is from a trusted source and has not been tampered with. Integrity checks can take the form of everything from hash digests to digital signatures.

Software and data integrity failures in action

Hundreds of customer networks affected in Codecov breach due to insufficient verification checks.

In April 2021 the build tool CodeCov suffered a security breach which resulted in the bash script used by customers being tampered with. Unfortunately, many customers did not perform appropriate verification checks on the script to ensure that it was not tampered with and thus were impacted by the breach.

SECURITY LOGGING AND MONITORING FAILURES

Attribution and auditing is an important concept in security. Although this will rarely prevent a security incident, it certainly helps with the investigation and recovery. You need to be able to determine who did what and when. Similarly, it is important to monitor logs, especially security-related logs and events, so you are alert to when an attacker is attempting to target the application or service.

Examples

- ✗ A CSP policy has been created on an application, but the report_uri directive has not been set.
- ✗ There is no logging on an authentication page of an application.
- ✗ When an administrator performs an administrative action on an application, this is not logged.

Remediation and prevention

- ✓ Ensure all security-critical actions are logged to a central location. This log should include who, when and where the action was performed. You should also ensure that these logs are protected against being tampered with as well from being destroyed or lost. Backups are essential. These logs may also need to be retained for compliance reasons, and should actively trigger appropriate alerts and events which are monitored and triaged.

Security logging and monitoring failures in action

Failure to apply appropriate security logging and monitoring led to a cryptojacking attack on UK government websites.

In February 2018, a crypto miner was inserted into a service called Browsealoud. This affected several sites including UK government websites. Many of the sites that used this service had not enabled appropriate logging via a control such as Content Security Policy (CSP), which would have triggered and alerted to the issue. Instead, it was identified via manual means.

SECURITY LOGGING AND MONITORING FAILURES

This vulnerability occurs when an application or service retrieves a user-defined resource without validating that is a valid resource. This allows for an unintended interaction with unintended resources.

Examples of SSRF

- ✗ The system defines users as URIs as opposed to simple IDs, using this URI to access the user's details.
- ✗ A system uses a user-supplied URI to access content for a page.

Remediation and prevention

- ✓ Ensure that any user-supplied (human or service) resource is appropriately validated to ensure it is allowed. This is where the use of allow lists will have the most value.

SSRF in action

GitLab's SSRF allowed attackers to send requests to internal servers and services.

In June 2021 GitLab had an SSRF vulnerability within its CLI Lint API library, which is responsible for code handling and managing developer workflows. This allowed an attacker to send requests to an organization's internal servers and services.