OWASP
Top10 :2021

A01:2021 – Broken Access Control:

OVERVIEW

Moving up from the fifth position, 94% of applications were tested for some form of broken access control with the average incidence rate of 3.81%, and has the most occurrences in the contributed dataset with over 318k. Notable Common Weakness Enumerations (CWEs) included are CWE-200: Exposure of Sensitive Information to an Unauthorized Actor, CWE-201: Insertion of Sensitive Information Into Sent Data, and CWE-352: Cross-Site Request Forgery.

DESCRIPTION

*Violation of the principle of least privilege or deny by default, where access should only be granted for particular capabilities, roles, or users, but is available to anyone.
*Bypassing access control checks by modifying the URL (parameter tampering or force browsing), internal application state, or the HTML page, or by using an attack tool modifying API requests.

A02:2021 – Cryptographic Failures:

OVERVIEW

Shifting up one position to #2, previously known as Sensitive Data Exposure, which is more of a broad symptom rather than a root cause, the focus is on failures related to cryptography (or lack thereof). Which often lead to exposure of sensitive data. Notable Common Weakness Enumerations (CWEs) included are CWE-259: Use of Hard-coded Password, CWE-327: Broken or Risky Crypto Algorithm, and CWE-331 Insufficient Entropy.

DESCRIPTION

The first thing is to determine the protection needs of data in transit and at rest. For example, passwords, credit card numbers, health records, personal information, and business secrets require extra protection, mainly if that data falls under privacy laws, e.g., EU's General Data Protection Regulation (GDPR), or regulations, e.g., financial data protection such as PCI Data Security Standard (PCI DSS).

A03:2021 – Injection:

OVERVIEW

Injection slides down to the third position. 94% of the applications were tested for some form of injection with a max incidence rate of 19%, an average incidence rate of 3%, and 274k occurrences. Notable Common Weakness Enumerations (CWEs) included are CWE-79: Cross-site Scripting, CWE-89: SQL Injection, and CWE-73: External Control of File Name or Path.

DESCRIPTION
Some of the more common injections are SQL, NoSQL, OS command, Object Relational Mapping (ORM), LDAP, and Expression Language (EL) or Object Graph Navigation Library (OGNL) injection. The concept is identical among all interpreters. Source code review is the best method of detecting if applications are vulnerable to injections. Automated testing of all parameters, headers, URL, cookies, JSON, SOAP, and XML data inputs is strongly encouraged. Organizations can include static (SAST), dynamic (DAST), and interactive (IAST) application security testing tools into the CI/CD pipeline to identify introduced injection flaws before production deployment.

A04:2021 – Insecure Design:

OVERVIEW
A new category for 2021 focuses on risks related to design and architectural flaws, with a call for more use of threat modeling, secure design patterns, and reference architectures. As a community we need to move beyond "shift-left" in the coding space to pre-code activities that are critical for the principles of Secure by Design. Notable Common Weakness Enumerations (CWEs) include CWE-209: Generation of Error Message Containing Sensitive Information, CWE-256: Unprotected Storage of Credentials, CWE-501: Trust Boundary Violation, and CWE-522: Insufficiently Protected Credentials.

DESCRIPTION
Insecure design is a broad category representing different weaknesses, expressed as “missing or ineffective control design.” Insecure design is not the source for all other Top 10 risk categories. There is a difference between insecure design and insecure implementation. We differentiate between design flaws and implementation defects for a reason, they have different root causes and remediation. A secure design can still have implementation defects leading to vulnerabilities that may be exploited. An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks. One of the factors that contribute to insecure design is the lack of business risk profiling inherent in the software or system being developed, and thus the failure to determine what level of security design is required.

A05:2021 – Security Misconfiguration:

OVERVIEW
Moving up from #6 in the previous edition, 90% of applications were tested for some form of misconfiguration, with an average incidence rate of 4.%, and over 208k occurrences of a Common Weakness Enumeration (CWE) in this risk category. With more shifts into highly configurable software, it's not surprising to see this category move up. Notable CWEs included are CWE-16 Configuration and CWE-611 Improper Restriction of XML External Entity Reference.

DESCRIPTION
The application might be vulnerable if the application is:
*Missing appropriate security hardening across any part of the application stack or improperly configured permissions on cloud services.

*Unnecessary features are enabled or installed (e.g., unnecessary ports, services, pages, accounts, or privileges).
*Default accounts and their passwords are still enabled and unchanged.

A06:2021 – Vulnerable and Outdated Components:

OVERVIEW

It was #2 from the Top 10 community survey but also had enough data to make the Top 10 via data. Vulnerable Components are a known issue that we struggle to test and assess risk and is the only category to not have any Common Vulnerability and Exposures (CVEs) mapped to the included CWEs, so a default exploits/impact weight of 5.0 is used. Notable CWEs included are CWE-1104: Use of Unmaintained Third-Party Components and the two CWEs from Top 10 2013 and 2017.

DESCRIPTION

You are likely vulnerable:
*If you do not know the versions of all components you use (both client-side and server-side). This includes components you directly use as well as nested dependencies.
*If the software is vulnerable, unsupported, or out of date. This includes the OS, web/application server, database management system (DBMS), applications, APIs and all components, runtime environments, and libraries.
*If you do not scan for vulnerabilities regularly and subscribe to security bulletins related to the components you use.

A07:2021 – Identification and Authentication Failures:

OVERVIEW

Previously known as Broken Authentication, this category slid down from the second position and now includes Common Weakness Enumerations (CWEs) related to identification failures. Notable CWEs included are CWE-297: Improper Validation of Certificate with Host Mismatch, CWE-287: Improper Authentication, and CWE-384: Session Fixation.

DESCRIPTION

*Confirmation of the user's identity, authentication, and session management is critical to protect against authentication-related attacks. There may be authentication weaknesses if the application:
*Permits automated attacks such as credential stuffing, where the attacker has a list of valid usernames and passwords.
*Permits brute force or other automated attacks.

A08:2021 – Software and Data Integrity Failures:

OVERVIEW

A new category for 2021 focuses on making assumptions related to software updates, critical data, and CI/CD pipelines without verifying integrity. One of the highest weighted impacts from Common Vulnerability and Exposures/Common Vulnerability Scoring System (CVE/CVSS) data. Notable Common Weakness Enumerations (CWEs) include CWE-829: Inclusion of Functionality from Untrusted Control Sphere, CWE-494: Download of Code Without Integrity Check, and CWE-502: Deserialization of Untrusted Data.

DESCRIPTION
Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks (CDNs). An insecure CI/CD pipeline can introduce the potential for unauthorized access, malicious code, or system compromise. Lastly, many applications now include auto-update functionality, where updates are downloaded without sufficient integrity verification and applied to the previously trusted application. Attackers could potentially upload their own updates to be distributed and run on all installations. Another example is where objects or data are encoded or serialized into a structure that an attacker can see and modify is vulnerable to insecure deserialization.

A09:2021 – Security Logging and Monitoring Failures:

OVERVIEW
Security logging and monitoring came from the Top 10 community survey (#3), up slightly from the tenth position in the OWASP Top 10 2017. Logging and monitoring can be challenging to test, often involving interviews or asking if attacks were detected during a penetration test. There isn't much CVE/CVSS data for this category, but detecting and responding to breaches is critical. Still, it can be very impactful for accountability, visibility, incident alerting, and forensics. This category expands beyond CWE-778 Insufficient Logging to include CWE-117 Improper Output Neutralization for Logs, CWE-223 Omission of Security-relevant Information, and CWE-532 Insertion of Sensitive Information into Log File.

DESCRIPTION
Returning to the OWASP Top 10 2021, this category is to help detect, escalate, and respond to active breaches. Without logging and monitoring, breaches cannot be detected.
Insufficient logging, detection, monitoring, and active response occurs any time:
*Auditable events, such as logins, failed logins, and high-value transactions, are not logged.
*Warnings and errors generate no, inadequate, or unclear log messages.
*Logs of applications and APIs are not monitored for suspicious activity.

A10:2021 – Server-Side Request Forgery (SSRF):

OVERVIEW
This category is added from the Top 10 community survey (#1). The data shows a relatively low incidence rate with above average testing coverage and above-average Exploit and Impact potential ratings. As new entries are likely to be a single or small cluster of Common Weakness Enumerations (CWEs) for attention and awareness, the hope is that they are subject to focus and can be rolled into a larger category in a future edition.

DESCRIPTION
SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL).
As modern web applications provide end-users with convenient features, fetching a URL becomes a common scenario. As a result, the incidence of SSRF is increasing. Also, the

severity of SSRF is becoming higher due to cloud services and the complexity of architectures.