OWASP Top 10

OWASP

Open Web Application
Security Project

About OWASP

- The Open Web Application Security Project (OWASP), an online community, produces freely-available articles, methodologies, documentation, tools, and technologies in the field of web application security.

- Founded in 2001, headquartered in the United States

OWASP Top 10

- List of the top ten web application vulnerabilities. It does not cover every web application security vulnerability
- The Top 10 is a fantastic foundation on which to build an application security plan that also considers the needs of the application and organization
- Determined by OWASP and the security community at large
- First release in 2003. Released every few years. Most recently released in 2017. Plan to release in 2021.

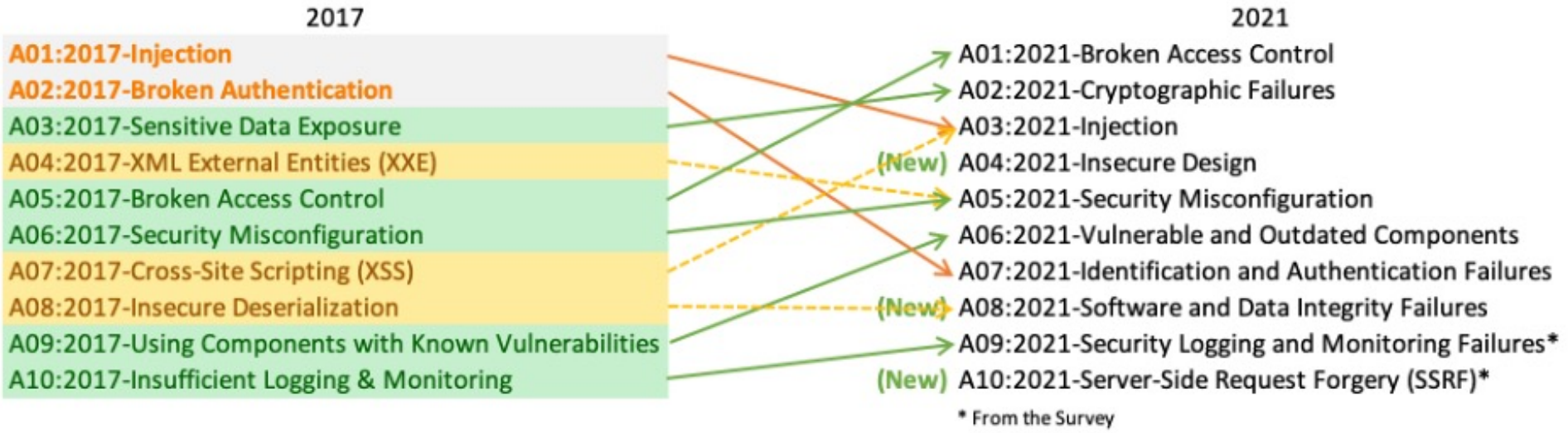
OWASP Top 10 changes from 2013 to 2017

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

OWASP Top 10 Vulnerabilities for 2017

A1: Injection
A2: Broken Authentication
A3: Sensitive Data Exposure
A4: XML External Entities (XEE)
A5: Broken Access Control
A6: Security Misconfiguration
A7: Cross-Site Scripting
A8: Insecure Deserialization
A9: Using Components with Known Vulnerabilities
A10: Insufficient Logging and Monitoring

OWASP Top 10 Vulnerabilities for 2021



Broken Access Control

- Authorization vulnerabilities are centered around flaws that allow a user to have access to data and application functionality that the developers did not intend

- Privilege Escalation

- Motivation: Gaining rights, accesses and control you are not supposed or authorized to have

- Prevention:
  - After logout, volumes and cookies should not be valid
  - Restricting resources on the server side

Cryptographic Failures

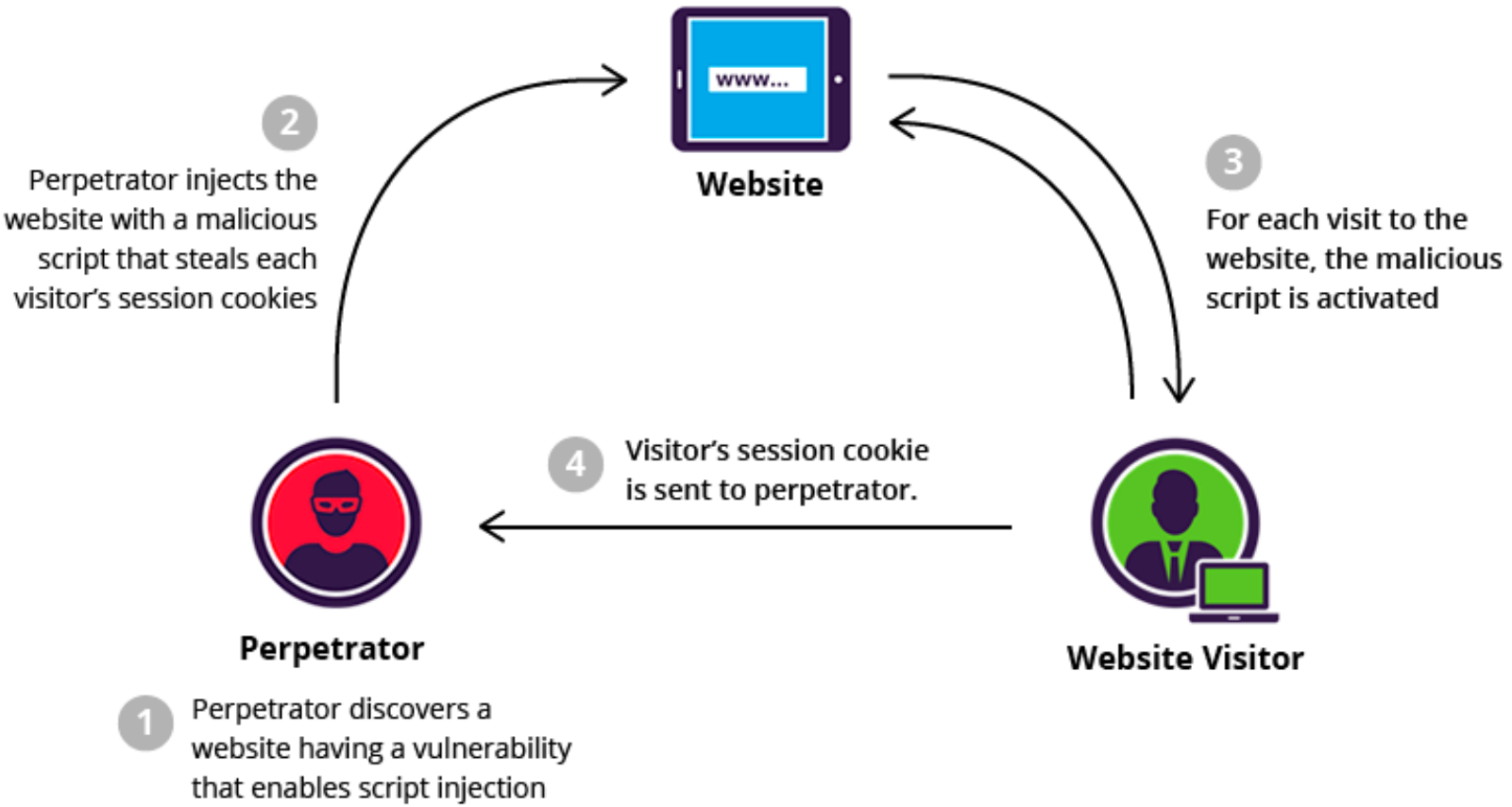
- Previously known as A3:2017-Sensitive Data Exposure, which was broad symptom rather than a root cause. The renewed name focuses on failures related to cryptography.
- This category often leads to sensitive data exposure or system compromise.
- Covers the display of data, data at rest, and data in transit.
- Sensitive data that does not need to be kept, should not be. Data should be protected in accordance with how sensitive it is.
- Can be done via man in the middle attack. Usually when the application has no strong encryption and Transport Layer Security
- Motivation: Identifying sensitive data bits and exploit them
- Prevention:
  - Regularly use safe and secure protocols and algorithms
  - Encryption of all data, both static and transit

Injection

- Occurs anytime untrusted input is used as an execution command. In OWASP Top 10 2021, Cross-site Scripting is now part of this category.
- Examples are in SQL queries, PHP queries, OS command, etc.
- Where injected: Username & password fields, textbox field, comment box, URL etc
- Motivation: To check if the application is vulnerable
- Prevention:
  - Using safe APIs
  - Sanitization or whitlisting

Cross Site Scripting (XSS)

- Occurs in applications that do not properly handle untrusted input
- Malicious scripts into web pages for different purposes
- Reflected, Stored and Document Object Model (DOM)-based
- Motivation: Acquire user's information, or attack and deface webpages or websites
- Prevention
  - Content-Security-Policy should be put in place
  - Escaping untrusted characters and output encoding


www...
Website
2
Perpetrator injects the website with a malicious script that steals each visitor's session cookies
3
For each visit to the website, the malicious script is activated
4
Visitor's session cookie is sent to perpetrator.
Perpetrator
Website Visitor
1
Perpetrator discovers a website having a vulnerability that enables script injection

Insecure Design

- This is a new category for 2021, with a focus on risks related to design flaws.
- The phrase “shift left” or “move left” has become an often used enterprise technology buzzword. It refers to the agile software development methodology of “moving security testing and controls earlier in the application lifecycle” to detect vulnerabilities as soon as possible
- If we genuinely want to “move left” as an industry, we need more threat modeling, secure design patterns and principles, and reference architectures.
- An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks.
- Will discuss it later in the Software Security session.

Security Misconfiguration

- With more shifts into highly configurable software, it's not surprising to see this category move up. The former category for A4:2017-XML External Entities (XXE) is now part of this risk category.
- Occurs anytime an insecure default setting goes ignored or a server or application is configured without security in mind
- Examples include the application returning stack traces or other default messages to the client and vulnerabilities such as Web Cache Deception
- Default Configurations on the Application
- Prevention:
  - Hardening applications and hardware
  - Check configurations from time to time (Updates)

XML External Entities (XXE)

- Occurs when Extensible Markup Language (XML) parsers allow loading of external entities. Commonly occurs in older XML processors, as they are configured to allow loading of external entities by default.

- Can be used to steal data, perform denial of service attacks, or map out the application and its environment. Application vulnerable to XXE attacks when enabled users to upload a malicious XML that further exploits the vulnerable code and dependencies

- Motivation: Malicious attacks like steal data, run malicious codes etc.

- Prevention:
  - Reviewing codes
  - Avoid serialization of delicate and sensitive data

Example #1: The attacker attempts to extract data from the server

```
<?xml version="1.0" encoding="ISO-8859-1"?> <!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]> <foo>&xxe;</foo>
```

Example #2: An attacker probes the server's private network by changing the above ENTITY line to

```
<!ENTITY xxe SYSTEM "https://192.168.1.1/private" >]>
```

Example #3: An attacker attempts a denial-of-service attack by including a potentially endless file

```
<!ENTITY xxe SYSTEM "file:///dev/random" >]>
```

Vulnerable and Outdated Components

- This was previously titled Using Components with Known Vulnerabilities.
- Vulnerabilities can pop up in 3rd party code and tools. If the code is still supported, generally a patch can be applied. If it's no longer supported, a replacement or work-around may be required.
- Libraries, coding frameworks, network frameworks vulnerable functions, etc.
- Prevention:
  - Patch frequently
  - Be aware of new vulnerabilities(CVE Numbers) by always checking online, check whether any include components in your system and fix it

Identification and Authentication Failures

- It was previously named Broken Authentication. This category is still an integral part of the Top 10, but the increased availability of standardized frameworks seems to be helping.

- A broad category that covers issues such as Credential Stuffing, Insecure Password Reset, Session Management Issues, Insufficient Password Complexity, Session Cookies, etc.

- Motivation: Either get into someone else's session or use a session which has been ended by the user or steal session related information.

- Prevention:
  - Using two or more factor authentications
  - Session expiry after idle moments

Software and Data Integrity Failures

- This is a new category for 2021, focusing on making assumptions related to software updates, critical data, and CI/CD pipelines without verifying integrity.

- A8:2017-Insecure Deserialization is now a part of this larger category.

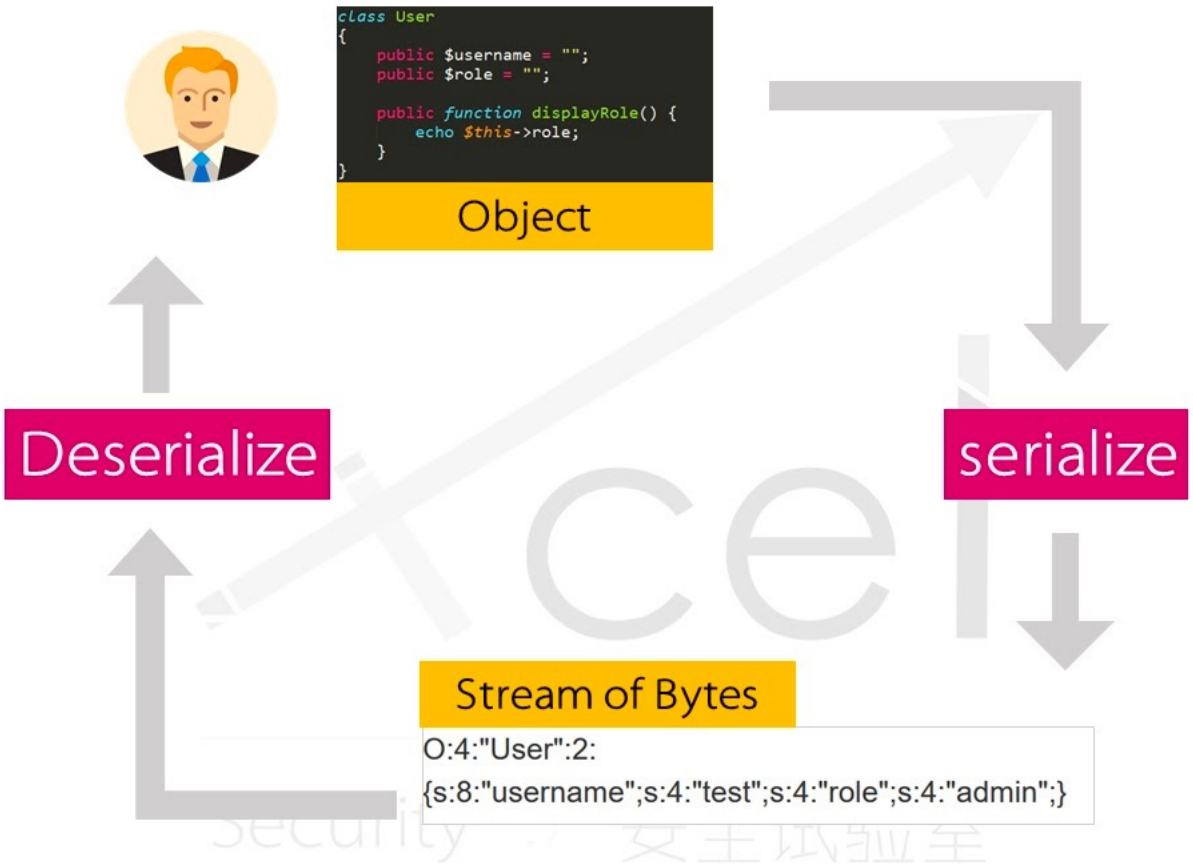
Insecure Deserialization

- Deserialization is a process where structured data is taken and turned into an object. Applications that use weak deserialization methods are vulnerable to Insecure Deserialization. Native language serialization formats are often weak

- Makes it possible for data to be interpreted as code, or in a way that an attacker can take advantage of. Often leads to remote code execution. Even if not, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks

- Some applications save data on the client side and they may be using object serialization. Applications that depend on the client to maintain state could allow meddling of serialized data.

- Prevention:
  - Serialized data should be encrypted
  - Deserialize data with least privileges


class User
{
public $username = "";
public $role = "";
public function displayRole() {
echo $this->role;
}
}
Object
Deserialize
serialize
Stream of Bytes
O:4:"User":2:
{s:8:"username";s:4:"test";s:4:"role";s:4:"admin";}

Security Logging and Monitoring Failures

- It was previously A10:2017-Insufficient Logging & Monitoring. This category is expanded to include more types of failures, is challenging to test for, and isn't well represented in the CVE/CVSS data. However, failures in this category can directly impact visibility, incident alerting, and forensics.

- Logging and monitoring is often overlooked. Proper logging provides valuable information to developers and security teams that can be used to improve weak points.

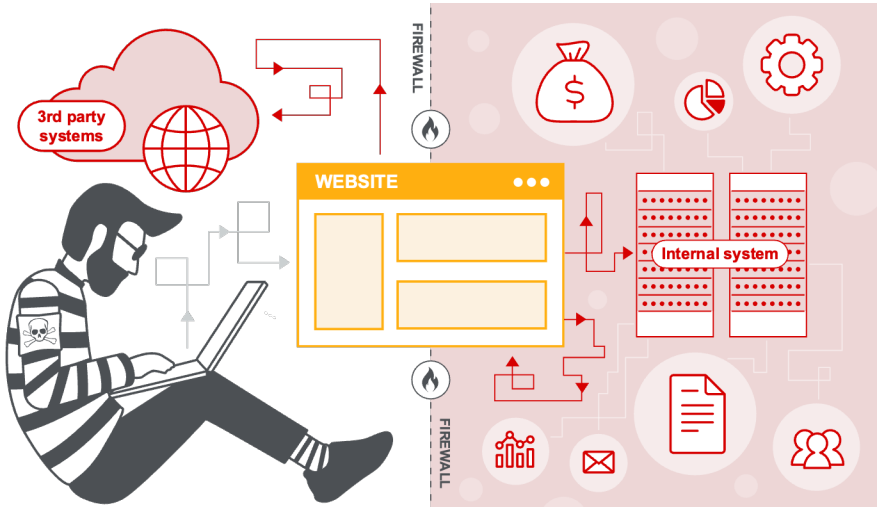
- In the event of a breach, logging and monitoring data can be used to assist with quicker response times, reducing impact.

- 100% security is not feasible. Goal is to secure the system as hard as possible, frustrate the attacker and buy time for detection and recovery.

- Prevention:
  - Monitoring log analysis and application traffic 24/7
  - Being ready to react effectively to any security breach any time of the day and night

Server-Side Request Forgery (SSRF)

- SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to force the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or ACL.



https://portswigger.net/web-security/ssrf

Server-Side Request Forgery (SSRF)

- Prevention:
  - **From Network layer**
    - Segment remote resource access functionality in separate networks to reduce the impact of SSRF
    - Enforce “deny by default” firewall policies or network access control rules to block all but essential intranet traffic.
  - **From Application layer:**
    - Sanitize and validate all client-supplied input data
    - Enforce the URL schema, port, and destination with a positive allow list
    - Do not send raw responses to clients
    - Disable HTTP redirections
    - Be aware of the URL consistency to avoid attacks such as DNS rebinding and “time of check, time of use” (TOCTOU) race conditions

Reference

- https://owasp.org/Top10/
- https://www.greycampus.com/blog/information-security/owasp-top-vulnerabilities-in-web-applications