OWASP - APPLICATION SECURITY RISK 2017 - CHECKLIST

A1:2017 INJECTION

Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

	VULNERABILITY	YES	NO	N/A	NOTE
1	User-supplied data is not validated, filtered, or sanitized by the application		No		It prevents the most common types of vulnerabilities including Cross Site Scripting, Injection Flaws, and Malicious File Execution. Enforces good Software Engineering practices. (Model-View-Controller design, Server-side form validation, postbacks) that make the code more readable, scalable, and maintainable.
2	Dynamic queries or non-parameterized calls without contextaware escaping are used directly in the interpreter.		NO		It prevents the most common types of vulnerabilities including Cross Site Scripting, Injection Flaws, and Malicious File Execution. Enforces good Software Engineering practices. (Model-View-Controller design, Server-side form validation, postbacks) that make the code more readable, scalable, and maintainable.
3	Hostile data is used within object-relational mapping (ORM) search parameters to extract additional, sensitive records.		NO		It prevents the most common types of vulnerabilities including Cross Site Scripting, Injection Flaws, and Malicious File Execution. Enforces good Software Engineering practices. (Model-View-Controller design, Server-side form validation, postbacks) that make the code more readable, scalable, and maintainable.
4	Hostile data is directly used or concatenated, such that the SQL or command contains both structure and hostile data in dynamic queries, commands, or stored procedures.		NO		It prevents the most common types of vulnerabilities including Cross Site Scripting, Injection Flaws, and Malicious File Execution. Enforces good Software Engineering practices. (Model-View-Controller design, Server-side form validation, postbacks) that make the code more readable, scalable, and maintainable.
	PREVENTION				
1	The preferred option is to use a safe API, which avoids the use of the interpreter entirely or provides a parameterized interface, or migrate to use Object Relational Mapping Tools (ORMs)	YES			Includes a Database Abstraction Layer that writes SQL for you in real time
2	Use positive or "whitelist" server-side input validation. This is not a complete defense as many applications require special characters, such as text areas or APIs for mobile applications	YES			Our DB includes a Database Abstraction Layer that writes SQL for you in real time
3	special characters using the specific	YES			Our DB includes a Database Abstraction Layer that writes SQL for you in real time

4	queries to prevent mass disclosure of	YES			Our DB includes a Database Abstraction Layer that writes SQL for you in real time

A2:2017 BROKEN AUTHENTICATION	Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently

	VULNERABILITY	YES	NO	N/A	NOTE
1	credential stuffing, where the attacker has		NO		
2	attacks		NO		It depends on the configuration of the web2 py application
3	passwords, such as "Password1" or		NO		We need to strengthen the application
4	recovery and forgotpassword processes,		NO		We use an email service for user confirmation
5	hashed passwords (see A3:2017-Sensitive		NO		
6	authentication.		NO		
7	rewriting).		NO		
8	successful login		NO		
9	User sessions or authentication tokens		NO		
	PREVENTION				
1	authentication to prevent automated,	YES			
2	credentials, particularly for admin users	YES			
3	testing new or changed passwords against		NO		
4	rotation policies with NIST 800-63 B's	YES			We need to strengthen the application
5	recovery, and API pathways are	YES			
6	attempts. Log all failures and alert	YES			
7	manager that generates a new random	YES			

A3:2017 SENSITIVE DATA EXPOSURE	Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser

	VULNERABILITY	YES	NO	N/A	NOTE
1	concerns protocols such as HTTP, SMTP,		NO		
2	algorithms used either by default or in		NO		Proxy Biella e TO5
3	crypto keys generated or re-used, or is		NO		
4	user agent (browser) security directives or		NO		
5	not verify if the received server certificate		NO		
	PREVENTION				
1	transmitted by an application. Identify	YES			
2	Apply controls as per the classification.	YES			
3	Discard it as soon as possible or use PCI	YES			
4	rest.	YES			
5	algorithms, protocols, and keys are in	YES			
6	protocols such as TLS with perfect forward	YES			

7	sensitive data	YES			
8	salted hashing functions with a work	YES			
9	configuration and settings	YES			

A4:2017 XML EXTERNAL ENTITIES	Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

	VULNERABILITY	YES	NO	N/A	NOTE
1	XML uploads, especially from untrusted			NOT	
2	application or SOAP based web services			NOT	
3	processing within federated security or			NOT	
4	version 1.2, it is likely susceptible to XXE			NOT	
	PREVENTION				
1	formats such as JSON, and avoiding	YES			
2	libraries in use by the application or on the			NOT	
3	processing in all XML parsers in the			NOT	
4	side input validation, filtering, or			NOT	
5	functionality validates incoming XML using			NOT	
6	code, although manual code review is the			NOT	

A5:2017 BROKEN ACCESS CONTROL	Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

	VULNERABILITY	YES	NO	N/A	NOTE
1	modifying the URL, internal application		NO		
2	another's users record, permitting viewing		NO		
3	without being logged in, or acting as an		NO		
4	or tampering with a JSON Web Token		NO		
5	unauthorized API access		NO		
6	an unauthenticated user or to privileged		NO		
	PREVENTION				
1	deny by default.		YES		
2	once and re-use them throughout the		YES		
3	record ownership, rather than accepting		YES		
4	requirements should be enforced by		YES		
5	ensure file metadata (e.g. .git) and backup		YES		
6	when appropriate (e.g. repeated failures)		YES		
7	minimize the harm from automated attack		YES		
8	server after logout. Developers and QA		YES		

A6:2017 SECURITY MISCONFIGURATION	Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.

	VULNERABILITY	YES	NO	N/A	NOTE
1	across any part of the application stack, or		NO		
2	installed (e.g. unnecessary ports, services,		NO		
3	enabled and unchanged		NO		
4	other overly informative error messages		NO		
5	features are disabled or not configured		NO		
6	servers, application frameworks (e.g.		NO		
7	or directives or they are not set to secure		NO		
8	(see A9:2017-Using Components with		NO		
	PREVENTION				
	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, with different credentials used in each environment. This process should be automated to minimize the effort required to setup a new secure				
1	environment	YES			
2	unnecessary features, components,	YES			
3	configurations appropriate to all security	YES			
4	provides effective, secure separation	YES			
5	e.g. Security Headers.	YES			
6	effectiveness of the configurations and	YES			

A7:2017 CROSS-SITE SCRIPTING XSS	XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites

	VULNERABILITY	YES	NO	N/A	NOTE
1	includes unvalidated and unescaped user		NO		
2	unsanitized user input that is viewed at a		NO		
3	page applications, and APIs that		NO		
	PREVENTION				
1	escape XSS by design, such as the latest	YES			
2	based on the context in the HTML output	YES			

3	modifying the browser document on the	YES			
4	a defense-in-depth mitigating control	YES			

A8:2017 INSECURE DESERIALIZATION	Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks

	VULNERABILITY	YES	NO	N/A	NOTE
1	where the attacker modifies application			NOT	
2	access-control-related attacks where			NOT	
2.1	(RPC/IPC)			NOT	
2.2	brokers			NOT	
2.3	Caching/Persistence			NOT	
	Databases, cache servers, file systems			NOT	
2.4	authentication tokens			NOT	
	PREVENTION				
1	digital signatures on any serialized objects			NOT	
2	deserialization before object creation as			NOT	
3	deserializes in low privilege environments			NOT	
4	such as where the incoming type is not the			NOT	
5	outgoing network connectivity from			NOT	
6	user deserializes constantly.			NOT	

A9:2017 USING COMPONENTS WITH KNOWN VULNERABILITIES	Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts

	VULNERABILITY	YES	NO	N/A	NOTE
	Your application is vulnerable:				
1	components you use (both client-side and		NO		
2	out of date. This includes the OS,		NO		
	regularly and subscribe to security		NO		
3	platform, frameworks, and dependencies		NO		
4	compatibility of updated, upgraded, or		NO		
5	configurations (see A6:2017-Security		NO		
	PREVENTION				
1	unnecessary features, components, files,	YES			
2	both client-side and server-side	YES			
3	sources over secure links. Prefer signed	YES			
4	are unmaintained or do not create security	YES			

A10:2017 INSUFFICIENT MONITORING &	Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack

LOGGING	systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring

	VULNERABILITY	YES	NO	N/A	NOTE
1	logins, and high-value transactions are not		NO		
2	inadequate, or unclear log messages		NO		
3	monitored for suspicious activity		NO		
4	Logs are only stored locally		NO		
5	response escalation processes are not in		NO		
6	by DAST tools (such as OWASP ZAP) do not		NO		
7	escalate, or alert for active attacks in real		NO		
	PREVENTION				
1	and server-side input validation failures	YES			
2	that can be easily consumed by a	YES			
3	audit trail with integrity controls to	YES			
4	such that suspicious activities are detected	YES			
5	and recovery plan, such as NIST 800-61 rev	YES			