

Eradication of Web Application Security Incidents Checklist

Note: Prior to starting the eradication of web application incidents checklist, Section 1 and Section 2 must be filled with required information.

Section 1: Details of the Organization	
Organization Name:	
Contact Number:	
Website:	
Address:	
<i>Additional Contact Information:</i>	

Section 2: Details of the Incident Responder			
Date Report Received:		Date Report Processing Began:	
Name:		Report Number:	
Title:		Department:	
Email Address:			
Phone Number and, if Applicable, Extension:			
<i>Additional Details (If Any):</i>			

Section 3: Checklist of Eradicating Web Application Security Incidents	
Actions	Completed
Whether a WAF firewall/IDS is employed to filter packets and protect the web server from a variety of attacks	<input type="checkbox"/>
Whether the server's software is updated using patches to keep it up-to-date and protect it from attackers	<input type="checkbox"/>
Whether the user input is sanitized and filtered, the source code for SQL injection is analyzed, and the use of third-party applications is minimized	<input type="checkbox"/>
Whether stored procedures and parameter queries are used to retrieve data	<input type="checkbox"/>
Ensure to disable verbose error messages that can provide attackers with useful information	
Whether custom error pages are used to protect the web applications	<input type="checkbox"/>
Whether a connection using a non-privileged account is formed and least privileges to the database, tables, and columns is granted to avoid SQL injection into the database	<input type="checkbox"/>
Whether commands, such as xp_cmdshell are disabled that can affect the operating system	<input type="checkbox"/>
Whether access control checks are performed before redirecting the authorized user to the requested resource	<input type="checkbox"/>
Whether client-side caching mechanism is avoided	<input type="checkbox"/>
Whether session tokens on the server side are removed when the user logs out	<input type="checkbox"/>
Whether a deny-by-default access approach is implemented for all resources except publicly accessible resources	<input type="checkbox"/>
Whether model access controls such as just-in-time (JIT) are deployed to thwart the risk of persistent privileges	<input type="checkbox"/>

Whether proper key management is used	<input type="checkbox"/>
Whether legacy protocols such as FTP and SMTP are avoided to transfer sensitive data	<input type="checkbox"/>
Whether salted hashing functions such as scrypt and bcrypt are employed to store passwords	<input type="checkbox"/>
Whether the team used a method attribute set to POST	<input type="checkbox"/>
Whether the database service account is accessed with minimal rights	<input type="checkbox"/>
Whether typesafe variables or functions are used by the programmers	<input type="checkbox"/>
Whether the programmers used language-specific libraries that restrict the shell commands and systems calls	<input type="checkbox"/>
Whether parameterized SQL queries are used	<input type="checkbox"/>
Whether modular shell disassociation from the kernel is used	<input type="checkbox"/>
Whether a chroot jail is implemented	<input type="checkbox"/>
Whether PHP wrappers such as PHP filter and PHP ZIP are checked to prevent access to sensitive files in the local server's file system	<input type="checkbox"/>
Whether tier-level segregation on the system and network layers is implemented according to exposure and protection requirements	<input type="checkbox"/>
Whether threat modeling for authentication, and access controls is applied	<input type="checkbox"/>
Whether non-SSL requests to the web pages redirected to the SSL page	<input type="checkbox"/>
Whether it is ensured that the certificate is valid, not expired, and it matches all the domains used by the site	<input type="checkbox"/>

Whether sources like the National Vulnerability Database (NVD) for vulnerabilities are continuously monitored in your components	<input type="checkbox"/>
Whether software component analysis (SCA) tools that perform automated checks for vulnerabilities in open-source software are employed	<input type="checkbox"/>
Ensure that session data is never submitted as part of a GET or POST	<input type="checkbox"/>
Whether a server-side, secure, built-in session manager is used that generates a new random session ID with a high entropy after logging in	<input type="checkbox"/>
Whether CAPTCHA system is used to thwart password guessing attacks	<input type="checkbox"/>
Whether trusted repositories such as npm and Maven are always used for libraries and dependencies	<input type="checkbox"/>
Whether the logs are logged with user context, making them traceable to specific users	<input type="checkbox"/>
Whether it is ensured that raw response bodies from the internal server to clients are never allowed	<input type="checkbox"/>
Whether the systems are configured to log certain changes within the server and track the dates of file changes	<input type="checkbox"/>
Whether public key infrastructure (PKI) is deployed for authentication that checks to ascertain that the script introduced is authenticated	<input type="checkbox"/>
Whether checks/hot fixes are applied to prevent the exploitation of the vulnerability, such as Unicode that can affect the directory traversal	<input type="checkbox"/>
Whether the firewall is configured to deny external ICMP traffic by checking their source address	<input type="checkbox"/>
Whether plug-ins and add-ons in browsers with “click-to-play” are configured to avoid them running automatically	<input type="checkbox"/>
Ensure that the web application encodes all data that is transmitted between the servers and users	<input type="checkbox"/>