12.2.9 Web Application Countermeasure Facts

Web applications capture, store, process, and transmit data. Significant damage can occur when an attack breaches web application data's security. There are a number of countermeasures you can use to prevent attacks from happening or at least mitigate the impact in the event of an attack.

This lesson covers the following topics:

- Web application attack countermeasures
- Web application security tools
- Web application penetration testing
- Web application penetration tools

Web Application Attack Countermeasures

Basic countermeasures to a web application attack include:

- Perform input validation on all data entered into the application.
- Implement a firewall to filter packets and deny external ICMP traffic access.
- Configure all security features and turn off all ports and services that are not being used.
- Scan for the most up-to-date security vulnerabilities and ensure that the appropriate patches and updates have been implemented.
- Filter user input to double-check the integrity before it's sent to the application.
- Use non-privileged accounts with the lowest necessary permissions to access the database.

The following table describes defensive countermeasure for specific web application attacks.

Attack	Defense
Command injection flaws	Perform input validation. Do not permit dangerous characters in the input. Perform input and output coding. Use language-specific libraries. Use parameterized SQL queries.
Cross-site scripting (XSS) attacks	Use rigid specifications to validate all headers, cookies query strings, hidden fields, and form fields. Use a web application firewall. Do not trust all websites that use HTTPS. Encode input and output filters. Filter all metacharacters that are input. Use testing tools during the design phase to detect and prevent errors in the application before it's put into production. Design standard scripts using private and public keys that will check for authenticated scripts.
DoS attacks	Secure remote administration and connectivity testing. Perform extensive input validation. Configure the firewall to deny ICMP traffic. Stop data processed by the attacker from being executed.
Web services attacks	Configure Web Services Description Language (WSDL) access control permissions. Use multiple security credentials. Configure firewalls and IDS systems to filter Simple Object Access Protocol (SOAP) and Zoho Markup Language (ZML)syntax and to detect web services anomalies. Maintain an updated and secure store of XML schemas. Implement centralized in-line requests and responses schema validation.
Unvalidated redirects and forwards	Do not use redirects and forwards. If you cannot avoid using them, be sure that the supplied values are valid and that the user has appropriate authorization.
Cross-site request forgery	Log off immediately after using a web application. Clear the history after using a web application, and don't allow your browser to save your login details.
Insecure cryptographic storage	Do not create or use weak cryptographic algorithms. Generate and store encryption keys offline.
Broken authentication and session management	Use SSL for all authenticated parts of an application. Verify that user information is stored in a hashed format. Do not submit session data as part of a GET or POST.
Insufficient Transport layer protection	Set the secure flag on all sensitive cookies. Ensure that certificates are valid and are not expired. All non-SSL web page requests should be directed to the SSL page.
Directory traversal	Update web servers with security patches on a regular basis. Limit access to the secure areas of the website.
Cookie/session poisoning	Do not store plain text or poorly encrypted passwords in a cookie. Implement timeouts for cookies. All cookie authentication credentials should be associated with an IP address. A logout option should always be provided.
Security	Configure all security mechanisms and remove all unused services. Carefully set up roles and permissions and disable default

misconfiguration	accounts.

Web Application Security Tools

The following table lists several web application security tools.

Tool	Description
Acunetix Web Vulnerability Scanner	Acunetix WVS checks for SQL injections and cross-site scripting. It also includes advanced penetration testing tools.
Watcher Web Security Tool	Watcher is a plugin that quietly audits a web application to find any errors or compliance issues.
Netsparker	Netsparker runs automated scans to look for SQL injection, cross-site scripting, and remote code injection.
N-Stalker	N-Stalker is a suite of assessment checks that can improve the overall security of your web applications. It contains assessment checks for code injection, cross-site scripting, parameter tampering, and web server vulnerabilities.
VampireScan	VampireScan is a tool that allows users to test the web infrastructure and application for vulnerabilities. It provides insight on addressing risk vulnerabilities by rating them on a low, medium, or high scale.
dotDefender	dotDefender is a software web application firewall that inspects HTTP and HTTPS traffic. It also detects and blocks SQL injection attacks.
ServerDefender VP	ServerDefender VP is a software firewall that provides security against web attacks.

Web Application Penetration Testing

The purpose of testing a web application is to identify, analyze, and document any vulnerabilities including input validation, buffer overflows, authentication, and injection opportunities. At times, penetration testing may focus on general vulnerabilities. Other times, it may be a response to unusual activity or be initiated with a specific objective in mind. Having an overall objective will let you know where to start and when you are done.

The following table presents a possible penetration testing routine. You should see some correlation with the web server attack methodology. Once again, you can address these items in a different order, and you can add or remove a step here and there. As long as you follow a set routine, you should be able to obtain consistent and accurate results.

Phase	Description
Information gathering	Gather as much information as you can about the target system or network. Start by searching advanced search engines for active and cached pages from the web server. This can help you identify any issues with the web application structure and view the error pages that are produced. Error messages can provide information on software versions, databases, and other network resources. Next, identify application entry points using cookie information, status codes, and server errors. Identify web applications that are being used by reviewing current and outdated versions of files and artifacts.
Configuration management testing	Many web application attacks are a result of improper configuration. It's important to scour your settings to seek out any vulnerabilities that a potential hacker may be able to exploit. Complete application configuration management testing to obtain information on the source code, log files, and any default error codes. This can be done by using CGI scanners and reviewing web server content, configurations files, and log files. Test for hidden file extensions and check for the location of old backup and configuration files. Perform file enumeration and directory enumeration to locate administrative interfaces that could be used to gain access to administrative-level permissions. As with application attacks, a common exploitation occurs through the code. Review HTML and source code to locate any coding errors. Look for tags or comments the developer embedded into the code, and remove comments as necessary.
Authentication testing	Authentication testing involves attempting to bypass the application's authentication requirements. Begin with the reset password option. Social engineering or simple guessing can provide the answers to a secret question if it is used. See if a session lets you reuse a session after you've logged out or after you've been idle for a long period of time. Check to see how much and what type of data is left in the browser cache after the session is closed. Another thing to check is that the password error or lockout feature works appropriately. Applications should be configured to provide a simple, generic error message, such as invalid login or password.
Session management	Session management testing includes testing for cookie attributes, session fixation, exposed session variables, and cross-site forgery requests. Because cookies are an important part of the web application, you'll want to familiarize yourself with how they

testing	work and how to best secure them. Cookie attributes can be set to make them safer to use. These attributes include HttpOnly and Expires. Ensure that cookies are sent only over a secure connection. HTTPOnly permits a cookie to be accessed by HTTP, but not other scripts like JavaScript. The Expires attribute ensures that the cookie is deleted once it has expired. A cookie can be used by that browser as long as the cookie is still valid. If this attribute is not set, the cookie will be deleted when the session closes.
Denial-of-service testing	Part of denial-of-service testing involves testing for account lockouts, buffer overflows, and user object allocation. Verify that an account locks down after a certain number of failed login attempts. Review your application to see what determines valid or invalid attempts. Repeatedly send requests to the web application until the server is overwhelmed. Observe what happens when this occurs.
Data validation testing	Many web applications fail to have a proper data validation method. Conduct data validation testing to detect, analyze, and try to exploit input vulnerabilities. Verify that there is input validation for all input that can be provided by a user, such as character type or character count. Although it may be hard to determine the restrictions for some entries, these restrictions are necessary. Having no restrictions opens the door to future problems. Use various tools to search for vulnerabilities to injection attacks.
Authorization testing	Test the authorization features of the application. Determine how the web application authorizes its users, then attempt to exploit any vulnerabilities that are present. Try escalating permissions to gain access to administrative pages and other restricted areas.
Documentation	Finally, compile all of the information that you gathered during the various testing phases. Be sure to include the date, the types of tests that were done, the tools that were used, and the outcome of each test. In addition to providing information on current vulnerabilities, this information will help to serve as a benchmark for future testing.

Web Application Penetration Tools

The following table describes several web application penetration testing tools.

Framework	Description
Kali Linux	Kali Linux is an advanced framework used for penetration testing and security auditing. It is a collection of more than 300 penetration testing tools.
Metasploit	Metasploit is a toolkit for penetration testers that contains hundreds of remote exploits for use on many different platforms. Metasploit helps to verify and manage vulnerabilities and security assessments.
Browser Exploitation Framework	Browser Exploitation Framework is an open source penetration testing tool. It is designed to test web applications and browsers for vulnerabilities and exploitation opportunities.