**Pentesting Plan**

Before pentesting, the team must choose the best penetration tool for their web application which contains the following criteria:

* It should be easy to deploy, configure and use.
* It should scan the team’s web application easily.
* It should categorized vulnerabilities based on severity that needs immediate fix.
* It should be able to automate verification of vulnerabilities.
* It should re-verify exploits found previously.
* It should generate detailed vulnerability reports and logs.

Once the team what tests they need to perform, the team can either train the team’s internal test resources or hire expert consultants to do the penetration task for the team.

In pentesting, the goal of the team is to find security holes in their web application. Below are some plans of the team for pentesting.

**1)** Check if web application is able to identify spam attacks on contact forms used in the website.  
**2)** Proxy server – Check if network traffic is monitored by proxy appliances. Proxy server make it difficult for hackers to get internal details of the network thus protecting the system from external attacks.  
**3)** Spam email filters – Verify if incoming and outgoing email traffic is filtered and unsolicited emails are blocked. Many email clients come with in-build spam filters which needs to be configured as per your needs. These configuration rules can be applied on email headers, subject or body.  
**4)** Firewall – Make sure entire network or computers are protected with Firewall. Firewall can be a software or hardware to block unauthorized access to system. Firewall can prevent sending data outside the network without your permission.  
**5)** Try to exploit all servers, desktop systems, printers and network devices.  
**6)** Verify that all usernames and passwords are encrypted and transferred over secured connection like HTTPs.   
**7)** Verify information stored in [website cookies](http://www.softwaretestinghelp.com/website-cookie-testing-test-cases/). It should not be in readable format.  
**8)** Verify previously found vulnerabilities to check if the fix is working.  
**9)** Verify if there is no open port in network.  
**11)** Verify all telephone devices.  
**12)** Verify WIFI network security.  
**13)** Verify all HTTP methods. PUT and DELETE methods should not be enabled on web server.  
**14)** Password should be at least 8 character long containing at least one number and one special character.  
**15)** Username should not be like “admin” or “administrator”.  
**16)** Application login page should be locked upon few unsuccessful login attempts.  
**17)** Error messages should be generic and should not mention specific error details like “Invalid username” or “Invalid password”.  
**19)** Verify if special characters, HTML tags and scripts are handled properly as an input value.  
**20)** Internal system details should not be revealed in any of the error or alert messages.  
**21)** Custom error messages should be displayed to end user in case of web page crash.  
**22)** Verify use of registry entries. Sensitive information should not be kept in registry.  
**23)** All files must be scanned before uploading to server.  
**24)** Sensitive data should not be passed in URLs while communicating with different internal modules of the web application.  
**25)** There should not be any hard coded username or password in the system.  
**26)** Verify all input fields with long input string with and without spaces.  
**27)** Verify if reset password functionality is secure.  
**28)** Verify application for[SQL Injection](http://www.softwaretestinghelp.com/sql-injection-%E2%80%93-how-to-test-application-for-sql-injection-attacks/).  
**29)** Verify application for [Cross Site Scripting](http://www.softwaretestinghelp.com/security-testing-of-web-applications/).  
**31)** Important input validations should be done at server side instead of JavaScript checks at client side.  
**32)** Critical resources in the system should be available to authorized persons and services only.  
**33)** All access logs should be maintained with proper access permissions.  
**34)** Verify user session ends upon log off.  
**35)** Verify that directory browsing is disabled on server.  
**36)** Verify that all applications and database versions are up to date.  
**37)** Verify URL manipulation to check if web application is not showing any unwanted information.  
**38)** Verify memory leak and buffer overflow.  
**39)** Verify if incoming network traffic is scanned to find Trojan attacks.  
**40)** Verify if system is safe from Brute Force Attacks – a trial and error method to find sensitive information like passwords.  
**41)** Verify if system or network is secured from DoS (denial-of-service) attacks. The hacker can target network or single computer with continuous requests due to which resources on target system gets overloaded resulting in denial of service for legit requests.

The team should collect and log all vulnerabilities in their web application. The team must not ignore any step listed above considering that it won’t be executed by end users.

Source: http://www.softwaretestinghelp.com/penetration-testing-guide/