Security Testing Checklist for Web Application

I (Santhosh Tuppad – http://tuppad.com/blog/) have created the checklist for security testing for web application. I have divided it into different components like registration, password, security question and security answer and others. For any discussion you might want to start on security testing I recommend you to register at *http://bangalorehackers.com/* and also thought I would do a bit of marketing to *http://softwaretestingnews.com/* which is a one stop shop for your software testing news.

Generic

- For All web pages which carry confidential data like password, Secret answer for security question should be submitted via HTTPS(SSL).
- Password & security answer needs to be masked with input type = password.
- Server Side Validation for form. Use "Firebug" and "TamperData" to perform this test (You can tamper for minimum length of password, set only new password without old password >> You got to remove the old password element from Firebug from the client-side and then submit it <<)
- Check for SQL Injection for any page in your application that accepts user-supplied information to access a database.
    - A login form, signup form, or "forgot password" form is a good start.
    - A dynamic page that uses URL variables such as ID (product information pages are good for this).
- Check for XSS by searching application for a page that takes user input and outputs it directly to a webpage. Common examples: Forums, Comments, Wikis, Review. Also, check for CSRF.

Password

- Set of rules for setting a password should be same across all the modules like Registration form, Change password, and Forgot password. If these rules differ than hacker might exploit it through brute force method.
  Example: If the registration form does not validate for password minimum length as 8 chars but while changing password from user profile it validates for minimum length or vice versa. Now, as registration form accepts password which are less than 8 chars it becomes easy for hacker to apply brute-force method.
- Password enforcement of alphabets + numeric + special characters should be used in order to protect the account to a greater extent against brute force attack mechanisms.

Forgot your password

- There need to be a restriction on number of forgot password requests sent per day or in "X" hours interval or have a captcha so that automated requests are not sent (To automate the requests you could use "ReloadEvery" add-on which is to be used on http://example.com/user/forgot-password/)

- The URL has to expire on one use after being used to set new password.
- The token associated with the URL should not be guessable or there should be any pattern which could be easily cracked.
- If the URL is not used within "X" hours then it has to expire (**Example:** Once the URL is generated, if it is not used then it has to expire after "72 hours")
- When new token is generated the old ones should expire even if they are not used.
- Example.com should not send the password via e-mails by resetting automatically. There has to be URL which should be used by end-user to set new password of his / her choice.
- While typing secret answer in Forgot Password the secret answer needs to be masked (Secret Answer is also part of authentication which is similar to password, shoulder surfing or auto-complete stuff could be dangerous here compromising the end-user account).
- Once the password is set, you might want to take end-user to logged in state or requesting him / her to login now with the hyperlink (I, personally would recommend taking to login page and requesting him / her to login with new password)

Registration Form

- There needs to be a captcha so that spam bots do not register and spam in discussion forums with illicit content which could be frustrating for your genuine end-users.
- Tamper with the mandatory fields by trying to register without mandatory fields – This is a server-side validation (Add-on on Mozilla Firefox – Tamper Data) **Example:** Can anyone bypass acceptance of terms of conditions and proceed with registration? This could be applied for all the forms and this test idea will not be repeated in other forms.

Change Password

- Once the password is changed successfully. User should not be able to login again with his old password & new password both.
- Login using the credentials on Mozilla Firefox | Login with the same credentials on Google Chrome | Now, change password for the account in Google Chrome | After this, refresh or try to navigate to some webpage which are allowed to be navigated only by logged in end-users | Result: The end-user in Mozilla Firefox web browser has to log out as he / she is in the session which has old password

Security Questions & Secret answer

- Frame the security question in such a fashion that they are not obvious to be known (What's your pet's name? >> Now, is that secret and no wonder we see such questions in famous web applications). It would be good if user is provided with option of choosing customized security question.
- Secret / security answers should be stored in database as hashes and not plain text.

Session Management

- User whose activity is idle for some time should be automatically logged out by expiring his session. (**Example:** User has gone out to fresh room or to have some snacks without logging out. Now, anyone can come to his system & see the user account open & exploit user account.
- No confidential details like password should be saved in cookie.
- Check what information cookie carries & try to tamper with it using Mozilla add-on Tamper Data.

Captcha

- Captcha characters should not be displayed in cyclic fashion.
- Captcha images should not be allowed to download at one time using add-on like "DownThemAll"
- Use http://free-ocr.com/ to see if captcha could be deciphered.
- Every refresh of a webpage should display new captcha every time.
- Do not show the absolute path names of the captcha that is being displayed because it is easy to put assertions identifying the URL and then entering the according characters to pass the captcha.
- I personally insist on using Google reCaptcha for your web application because it has not been cracked till date. There are many captcha third party services out there but, I do not recommend those.
- Usage of question and answers type of captcha in textual format is good but, not good enough.

This is a good checklist but, it could be made much better if you want to. I stop here because I can go on and on generating the test ideas. You are free to use this checklist for your project in your organization and share it with your colleagues owing credits to me. To share this document here is the PDF document which you can download.