**Bug bounty methodology**

* **configure burpsuite**
  + **identify the target scope**
    - target->scope->add the website you want to test
    - target ->sitemap->click on the long phrase -> show only in scope items // to show only the target site in the site mapper
    - proxy->options->intercept client request->check on is in target scope  // to only intercept target requests
  + **configure the scanner to automate and identify the vulnerabilities to scan**
    - scanner ->options -> active scanning area -> choose the vuln you want the scanner to search for
    - scanner -> live scanning ->use suite scope [defined in target scope]  // to only scan the selected target website identified in target scope
    - scanner is not good at finding logical vulns and access control vulnerabilities
* **mapping the application**
  + subdomain enumeration
    - [virustotal.com](http://virustotal.com/)
    - google dorks
    - bruteforcing
  + get IP's for subdomain founds and scan for open ports in each ip
  + crawling the website
    - manually : try to browse the entire app in a normal way by visiting every link , submitting every form
    - automatic with burp spider or zap , webscarab
  + directory bruteforcing : to search for hidden files , backup files , and admin interfaces
  + know the technology used on the site
    - [www.builtwith.com](http://www.builtwith.com/)
    - wappalyzer addon
    - whatweb tool in linux
  + look at source code
  + Try to understand the functionality of the website
  + Extra work
    - search  for sensitive info in github repositories (gitgraber is a tool that automate this process)
      * python gitgraber.py -k wordlist.txt -q "companyname"
    - search in internet archive for old version of website
* **search for vulnerabilities**
  + Test for access control logical vulnerabilities manually
    - try to access administrative pages
    - try to have multiple accounts to test access controll and check if one account can access data of other accounts
    - Test for path traversal
    - Test for bypassing authorization schema
    - Test for vertical Access control problems (a.k.a. Privilege Escalation)
    - Test for horizontal Access control problems (between two users at the same privilege level)
    - Test for missing authorization
  + test for session management vulnerabilities
    - token generation
      * test for token meaning
      * test for token predictability
    - token handling
      * check for insecure transmission
      * test session fixation
      * test session termination
        + Check session termination after a maximum lifetime
        + Check session termination after relative timeout
        + Check session termination after logout
      * Check session tokens for cookie flags (httpOnly and secure)
      * Check session cookie scope (path and domain)
      * Check session cookie duration (expires and max-age)
    - test for csrf
    - Extra
      * Establish how session management is handled in the application (eg, tokens in cookies, token in URL)
      * Test to see if users can have multiple simultaneous sessions
      * Confirm that new session tokens are issued on login, role change and logout
      * Test for consistent session management across applications with shared session management
      * Test for session puzzling
  + test for authentication mechanism vulnerabilities
    - Test for user enumeration
    - Test for authentication bypass
    - Test for bruteforce protection
    - Test password quality rules
    - Test remember me functionality
    - Test for autocomplete on password forms/input
    - Test password reset and/or recovery
    - Test password change process
    - Test CAPTCHA
    - Test multi factor authentication
    - Test for logout functionality presence
    - Test for cache management on HTTP (eg Pragma, Expires, Max-age)
    - Test for default logins
    - Test for user-accessible authentication history
    - Test for out-of channel notification of account lockouts and successful password changes
    - Test for consistent authentication across applications with shared authentication schema / SSO
  + Automate other vulnerabilities with scanner (input based vulnerabilities)
* **Exploiting the vulnerabilities found**
  + just poc (proof of concept ) don't harm the web app
* **report the vulnerabilities**
  + try to take notes with everything you have found and screen shots as many as you can