Attacking Application Logic

- Code is nothing but discrete logic translated to a medium that computers can understand and process
- Because these vulns are quite subtle,they are ignored by testers and scanners alike.

- The Nature of Logic Flaws

→ No common signature is associated with a logic flaw unlike other vulns.
→ The definitive characteristic is that any implemented logic is flawed in a certain way.
→ Often occurs due to anomalies that the dev did not address
→ Observe the functions of the webapp and think how these assumptions could be broken.

- Real-World Logic Flaws

****

HACK STEPS:

→ Context: Same encryption used for permananet cookie and screen name. Hence by manipulating the screen name, the admin cookie can be retrieved and replayed. Hence “Encryption Oracle”
→ Look for locations where encryption is used in the application. Try to substitute encryted values encountered. Try to cause an error within the app that reveals decrypted value or it is displayed on-screen.
→ Look for `oracle reveal` vuln.
→ Look for `oracle encrypt` to encrypt plaintext and replay.

****

****

HACK STEPS:

→ When probing key functionality for logic flaws, try removing and adding parameters in requests.
→ Be sure to delete the name of the param.
→ Attack only one param at a time
→ If it is a multistage process, foloow up completely

****

****

HACK STEPS:

Forced Browsing, involves circumventing controls imposed by in-browser navigation on sequence of the functions of an application.
→ When a multistage process requires a sequence, try to ssubmit requests out-of-turn. Try skipping, accessing a single stage more than once, and accessing earlier stages after later ones.
→ Use params or distinct URL(s) to achieve this.
→ From the context try to understand what assumptions are made

****

****

HACK STEPS:

→ Whenever an app implemets a key action across stages, take params that are submitted at one one stage adn try submitting it to different stages. If relevant items are updates, explore any exploitation possible.
→ Take params from different users and submit them using other users,if any are accepted probe for any possibilities for epxloitation.

****

****

HACK STEPS:

→ In a complex app invloving either horizontal or vertical privilege segregation, try to locate any instances where an user can accumulate an amount of state within his session that relates in some way to identity.
→ Try to step through areas of functionality and switch to unrelated areas to determine the effects on a web app.
****

****
HACK STEPS:
When encountered with buinsness limits,
→ Try entering negative values and see if app accepts them and processs them.
→ May require several steps to acheive the intended result by attacker.
****

****
HACK STEPS:
→ When adjustments are based on user input or actions, recheck the validity of these adjustments.
→ Functionalities can be fooled by simply undoing actions that invoke actions.
****

****
HACK STEPS:
→ Whenever you probe app for command injection, having attempted to insert the relevant metacharacters int the data, always try backslashes to escape the escaping characters.
****

****
HACK STEPS:
→ If data is stripped once, determine whether you can submit a string that compensates this.
→ If data validation is a set order of operations, then manipulate the payload accordingly
****

****
HACK STEPS:
While performing black-box testing try for subtle thread safety issues of this kind.
→ Target selected items of key functionality (logins, password change and funds transfer).
→ Identify a single request or a small number that can be used to perform a single action, for each function tested.
→ Use a VPS to launch a scripted attack to see if you can manipulate applicaion behaviour
****

- Avoiding Logic Flaws
→ There is no silver bullet to protect apps the same way there is no unique signature for the flaws.
→ But some steps to take to mitigate these flaws are
⇒ Ensure the apps design is documented with all assumptions made.
⇒ Reference all components that use these fucntionalitites employing these assumptions
⇒ Duting security reviews, reflect on design assumptions and circumstances.
⇒ Think laterally about two key areas

- The ways in which app will handle unexpected user behaviour
- Potential side effects of any dependencies and interoperation between code compenents.