
BUG BOUNTY

Basic Bug Hunting Methodology

1- **Philosophy**
2- **Discovery**
3- **Mapping**
4- **Authorization & Session**
5- **Subdomain Takeover**
6- **Clickjacking & spf**
7- **XSS**
8- **GitHub Information**
9- **SQLi**
10- **File upload**
11- **CSRF**
12- **Privilege logic transport**
13- **Mobile**
14- **Auxiliary info**

i. **Reference**

1- Philosophy

- 1st party bug bounties = Google Paypal, etc
- 2nd party bug bounties = Bugcrowd, H1, Synack, etc

2- Discovery

- Find the application less tested.
- Wide Scoped
- Find Subdomain
- Port scan
- Find acquisitions
- Functionality changes or re-design
- Mobile websites
- New mobile app version
- Search parent company by trademark or privacy policy
- Tool
    i. Recon - https://github.com/jhaddix/domain
    ii. Sub-domain enumeration
        1. Horizontal sub-domain enumeration
            a. gobuster
            b. massdns
            c. subbrute.sh
            d. subbrute-big.sh
            e. ct.sh
            f. amass
            g. subfinder
            h. Sublist3r
            i. aquatone
            j. domlink
            k. https://www.virustotal.com
            l. https://censys.io/
            m. https://dnsdumpster.com/
            n. https://securitytrails.com/dns-trails
            o. https://www.shodan.io/
            p. Google dorks
        2. Vertical sub-domain enumeration
            a. https://bgp.he.net/
            b. https://whois.arin.net/ui/query.do
            c. https://apps.db.ripe.net/db-web-ui/#/fulltextsearch
            d. https://viewdns.info : DNS and WHOIS.
            e. https://reverse.report
            f. Google dork
    iii. Acquisitions
        1. www.crunchbase.com/
    iv. Port Scanning
        1. Masscan
        2. Nmap
        3. Sparta
        4. Brutespray

v. Linked Discovery
vi. CSP- Header

3- Mapping

- directory brute forcing
    i. dirbuster
    ii. dirb
    iii. gobuster
    iv. RobotsDisallowed
    v. snallygaster
- Platform Identify
    i. Wappalyzer : Extension for chrome/firefox.
    ii. Builtwith
    iii. Vulners Web Scanner
- 401/403 response
    - WayBackUrls
    - WayBackUnifier
    - ReconCat
- Screenshots
    - EyeWitness
    - WebScreenshot
    - wmap : Chrome extension
- Linkfinder
    - LinkFinder
    - JSParser
    - relative-url-extractor
- WPscan
    - wpscan — url www.example.com
- Cmsmap
    - cmsmap.py -t https://example.com -o output.txt
    - cmsmap.py -t https://example.com -u admin -p passwords.txt
    - cmsmap.py -k hashes.txt -w passwords.txt
- WAF
    - wafwoof
    - https://censys.io/
    - https://dnsdumpster.com/
    - https://securitytrails.com/dns-trails
    - http://viewdns.info/iphistory/?domain=tesla.com
    - bypass-firewalls-by-dns-history

4- Authorization & Session

- Auth
    i. Auth Related (more in logic, priv, and transport sections)

        1. User/pass discrepancy flaw
        2. Registration page harvesting
        3. Login page harvesting
        4. Password reset page harvesting
        5. No account lockout
        6. Weak password policy
        7. Password not required for account updates
        8. Password reset tokens (no expiry or re-use)

- Session
    - Failure to invalidate old cookies
    - No new cookies on login/logout/timeout
    - Never ending cookie length
    - Multiple sessions allowed
    - Easily reversible cookie (base64 most often)

5- Subdomain Takeover

- https://github.com/EdOverflow/can-i-take-over-xyz
- https://github.com/haccer/subjack
- https://github.com/Ice3man543/SubOver
    i. AWS - Buckets
        1. slurp
        2. S3scanner
        3. teh_s3_bucketeer

6- Clickjacking & spf

- Clickjacking
    i. https://github.com/abhinavporwal/ClickJacking-Bug-Testing
- Spf
    i. https://mxtoolbox.com/
    ii. https://www.kitterman.com/spf/validate.html

7- XSS

- https://github.com/abhinavporwal/xss-cheat-sheet
- Polyglot payloads
- Input Vectors
    i. Customizable Themes & Profiles via CSS
    ii. Event or meeting names
    iii. URI based
    iv. Imported from a 3rd party (think Facebook integration)
    v. JSON POST Values (check returning content type)
    vi. File Upload names
    vii. Uploaded files (swf, HTML, ++)
    viii. Custom Error pages
    ix. fake params - ?realparam=1&foo=bar'+alert(/XSS/)+'
    x. Login and Forgot password forms

8- GitHub Information

- https://github.com/techgaun/github-dorks
- Gitrob
    i. ./gitrob google
- Trufflehog https://github.com/SeppPenner/postgres.git
- https://github.com/techgaun/github-dorks/blob/master/github-dorks.txt

9- SQLi

- SQLi polyglots
- You can also leverage the large database of fuzzlists from Seclists (https://github.com/danielmiessler/SecLists)
- SQLMap

10- File upload

- Malicious File Upload
    i. Upload unexpected file format to achieve code exec (swf, html, php, php3, aspx, ++) Web shells or...
    ii. Execute XSS via same types of files. Images as well!
    iii. Attack the parser to DoS the site or XSS via storing payloads in metadata or file header
    iv. Bypass security zones and store malware on target site via file polyglots
- File Upload Attack
    i. content type spoofing
    ii. extension trickery
    iii. [File in the hole! presentaion] (https://www.nds.rub.de/media/attachments/files/2012/11/File-in-the-hole.pdf)
- Local File Inclusion
    i. https://github.com/rotlogix/liffy
    ii. https://github.com/danielmiessler/SecLists/blob/master/Fuzzing/JHADDIX_LFI.txt
- Remote file inclusion & redirects
    i. Common blacklist bypasses
        - escape "/" with "/" or “//” with “//”
        - try single "/" instead of "//"
        - remove http i.e. "continue=//google.com"
        - “//\” , “|/” , “/%09/”
        - encode, slashes
        - ”./” CHANGE TO “..//”
        - ”../” CHANGE TO “....//”
        - ”/” CHANGE TO “//”
    ii. Redirections Common Parameters or Injection points
        - dest=
        - continue=
        - redirect=
        - url= (or anything with “url” in it)
        - uri= (same as above)
        - window=
        - next=
    iii. RFI Common Parameters or Injection points:
        - File=

- document=
- Folder=
- root=
- Path=
- pg=
- style=
- pdf=
- template=
- php_path=
- doc=

11- CSRF

- Testing CSRF On Application :
    i. Csrf Normal
    ii. Chnage Method To GET-Based
    iii. Change Value Of CSRF-Token To undefined
    iv. Delete CSRF Token Value Or Delete Token Parameter
    v. Use The same CSRF Value In Different Accounts
    vi. Replace Value CSRF Token with Same Length Characters
    vii. Change Content-Type from application/json to text/plain
    viii. Use Vulnerable-Subdomain To Bypass CSRF Token

12- Privilege logic transport

- Privilege
    i. Admin has power
    ii. User/peon has no power
    iii. User/peon can use function only meant for admin
    iv. Find site functionality that is restricted to certain user types
    v. Try accessing those functions with lesser/other user roles
    vi. Try to directly browse to views with sensitive information as a lesser priv user
    vii. Autorize Burp plugin is pretty neat [here] (https://github.com/Quitten/Autorize).
- Common function
    i. Add user function
    ii. Delete user function
    iii. start project / campaign / etc function
    iv. change account info (pass, CC, etc) function
    v. customer analytics view
    vi. payment processing view
    vii. any view with PII
- IDOR
    i. Find any and all UIDs
        1. increment
        2. decrement
        3. negative values
        4. Attempt to perform sensitive functions substituting another UID
            a. change password
            b. forgot password
            c. admin only functions
    ii. Common Functions , Views, or Files:
        1- Everything from the CSRF Table, trying cross account attacks
        2- Sub: UIDs, user hashes, or emails
        3- Images that are non-public
        4- Receipts
        5- Private Files (pdfs, ++)
        6- Shipping info & Purchase Orders
        7- Sending / Deleting messages
- Transport
    i. Sensitive images transported over HTTP
    ii. Analytics with session data / PII leaked over HTTP

iii. (ForceSSL) [https://github.com/arvinddoraiswamy/mywebappscripts/tree/master/ForceSSL]

- Business Logic Flow
  i. Logic flaws that are tricky, mostly manual:

     1. substituting hashed parameters
     2. step manipulation
     3. use negatives in quantities
     4. authentication bypass
     5. application level DoS
     6. Timing attacks

13- Mobile

- Its common to see mobile apps not applying encryption to the files that store PII.
- Common places to find PII unencrypted
    i. Phone system logs (avail to all apps)
    ii. webkit cache (cache.db)
    iii. plists, dbs, etc
    iv. hardcoded in the binary
- Quick spin-up for iOS (Daniel Mayers' idb) [https://github.com/dmayer/idb]

14- Auxiliary info

- Content Spoofing or HTML injection
- Referer leakage
- security headers
- path disclosure
- Rate Limiting on reset password & email verification

Reference

i. https://github.com/Quikko/Recon-Methodology
ii. https://blog.usejournal.com/bug-hunting-methodology-part-1-91295b2d2066
iii. https://github.com/jhaddix/tbhm
iv. https://medium.com/@trapp3rhat/bug-hunting-methodology-part-3-457eaf9768a5