Main app methodology

BY UNCLE RAT

Agenda

How to use this document

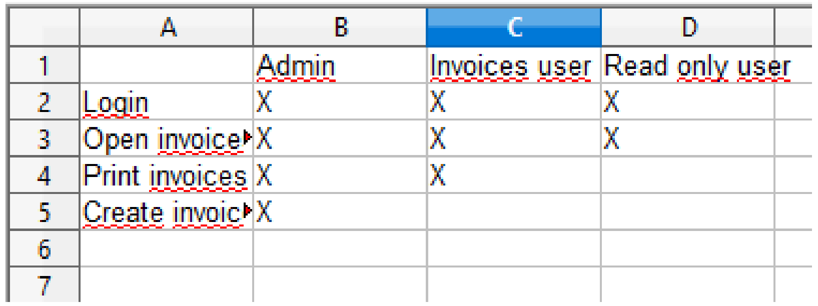
How to use this document

1. Use this document with the video for the main app methodology
2. First go through the intracies of bug bounties section, don't skip this
3. Make note of what parameters are tested for what vulnerabilities
4. Explore the vulnerabilities in their own sections
5. Practice your methodology to develop an intuition
6. Re-read this document
7. Imagine all the ways in which we can create impact with these vulnerabilties

Preperation

Preperation

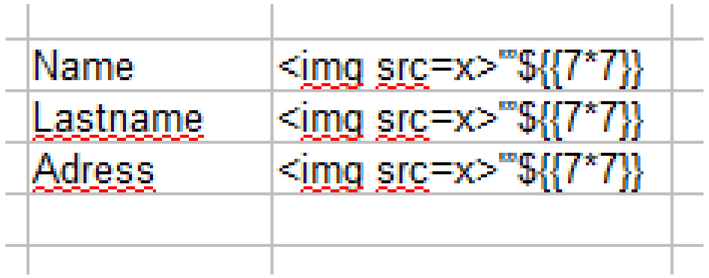
- Manually explore your target
  - At least a couple of hours
  - Uncle rat says: 8 hours +
- Keep Burp suite open in the background
  - Scope set properly
  - Will later on be used to explore the requests
- Make a mindmap of the functionality
- Take note of the privilege levels



	A	B	C	D
1		Admin	Invoices user	Read only user
2	Login	X	X	X
3	Open invoice	X	X	X
4	Print invoices	X	X	
5	Create invoic	X		
6				
7				

Preperation

- Read any manual you can find
- Register your accounts
  - While you do use an XSS attack vector in every possible field
    - • Triggers integration issues if existent
  - Also SSTI
  - Ex. **<img src=x>'"${{7*7}}**
    - This will test for HTML injection
    - This will test for JS context XSS
    - This will test for SSTI
    - This will test for CSTI

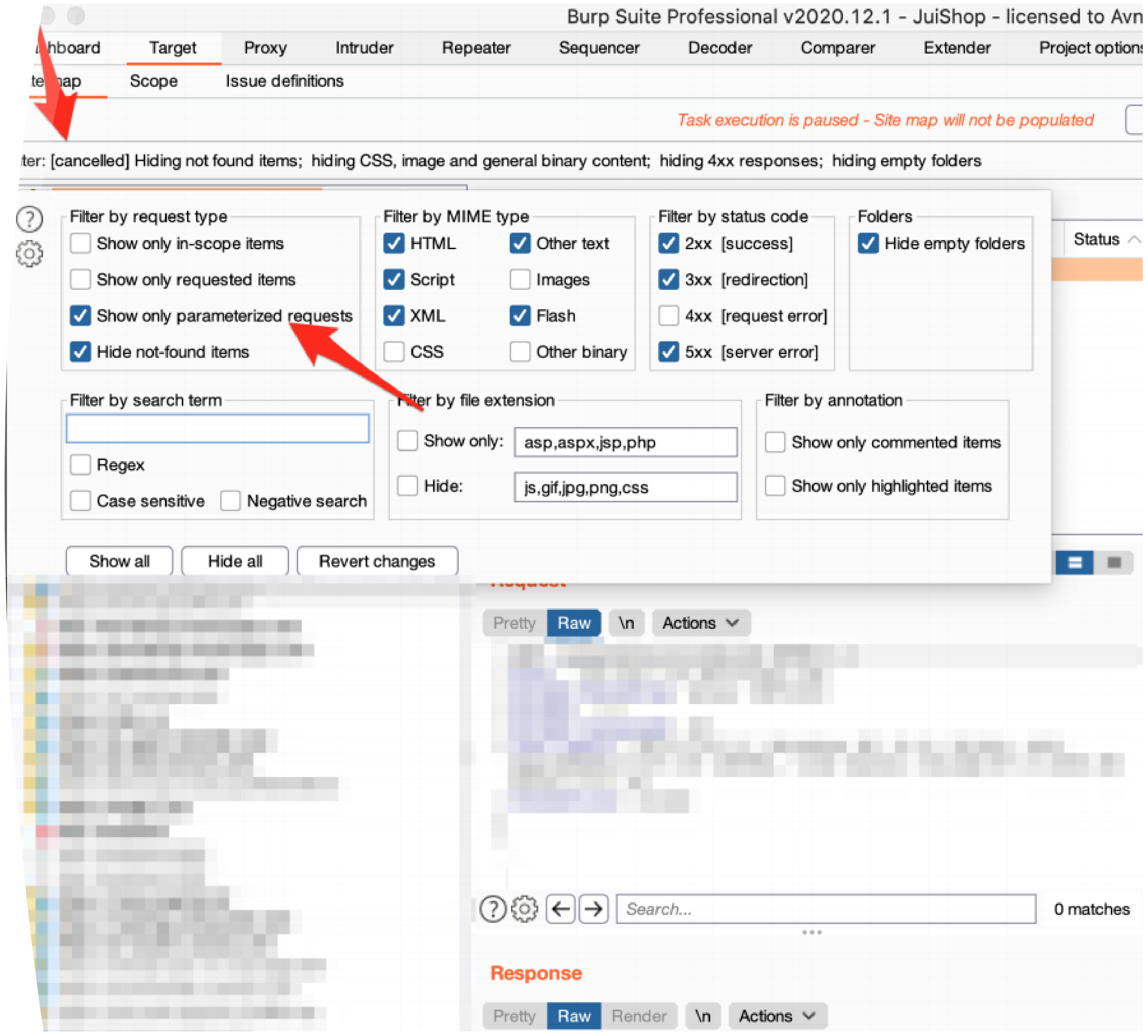


Name	<img src=x>'"${{7*7}}
Lastname	<img src=x>'"${{7*7}}
Adress	<img src=x>'"${{7*7}}

Exploring the requests

Exploring the requests

- Burp: Filter on all requests with parameters
  - Study each request and parameter
  - Think of what it does
  - Think of how to break it functionally
  - Does our bug have security impact if we find any?



Task execution is paused - Site map will not be populated

ter: [cancelled] Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

Filter by request type
- [ ] Show only in-scope items
- [ ] Show only requested items
- [x] Show only parameterized requests
- [x] Hide not-found items

Filter by MIME type
- [x] HTML
- [x] Other text
- [x] Script
- [ ] Images
- [x] XML
- [x] Flash
- [ ] CSS
- [ ] Other binary

Filter by status code
- [x] 2xx [success]
- [x] 3xx [redirection]
- [ ] 4xx [request error]
- [x] 5xx [server error]

Folders
- [x] Hide empty folders

Filter by search term
- [ ] Regex
- [ ] Case sensitive
- [ ] Negative search

Filter by file extension
- [ ] Show only: asp,aspx,jsp,php
- [ ] Hide: js,gif,jpg,png,css

Filter by annotation
- [ ] Show only commented items
- [ ] Show only highlighted items

Parameter analysis

Parameter analysis

- Explore the application: Business logic vulnerabilities
- URL parameters that gets resolved: SSRF
- Parameters grabbing files either locally or remotely: LFI/RFI
- CSRF parameters: CSRF
- Uploading an image: XXE via SVG
- Uploading documents: XXE via DOCX/XLSX
- Looking for hidden fields, both in the request and the response

Broken Access Control

Broken Access Control

- Some Terms
  - Account = Organisation such as google for example
  - User = Person such as an employee
- Preperation
  - Create 2 accounts
  - Invite at least 2 users per account
- Test for
  - IDOR
    - Between 2 accounts
    - Between 2 users within 1 account

Broken Access Control

- Test for
  - BAC
    - Between all users from all privilege levels
    - Within 1 account
    - Within 2 users from different accounts
- Use tools
  - Authorize
    - Built for BAC testing
    - Has more speciliased settings
  - Auto repeater
    - Less specialised
    - Can be customised further

SQL Injection

SQL Injection

- Test
  - Every database read or write
- Basic check
  - Enter ' and " wherever you can
  - If you get a SQL error, run SQLmap
- Advanced check
  - Investigate DB systems and build wordlist

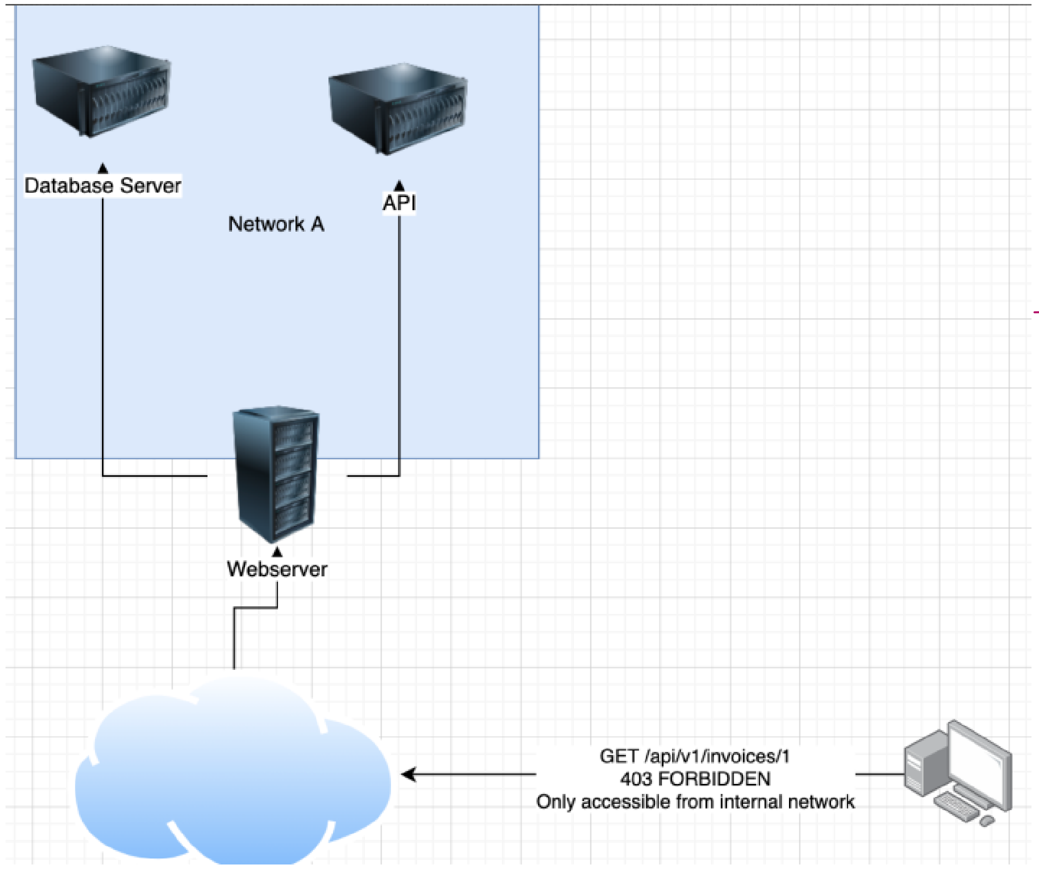
Business Logic Vulnerabilities

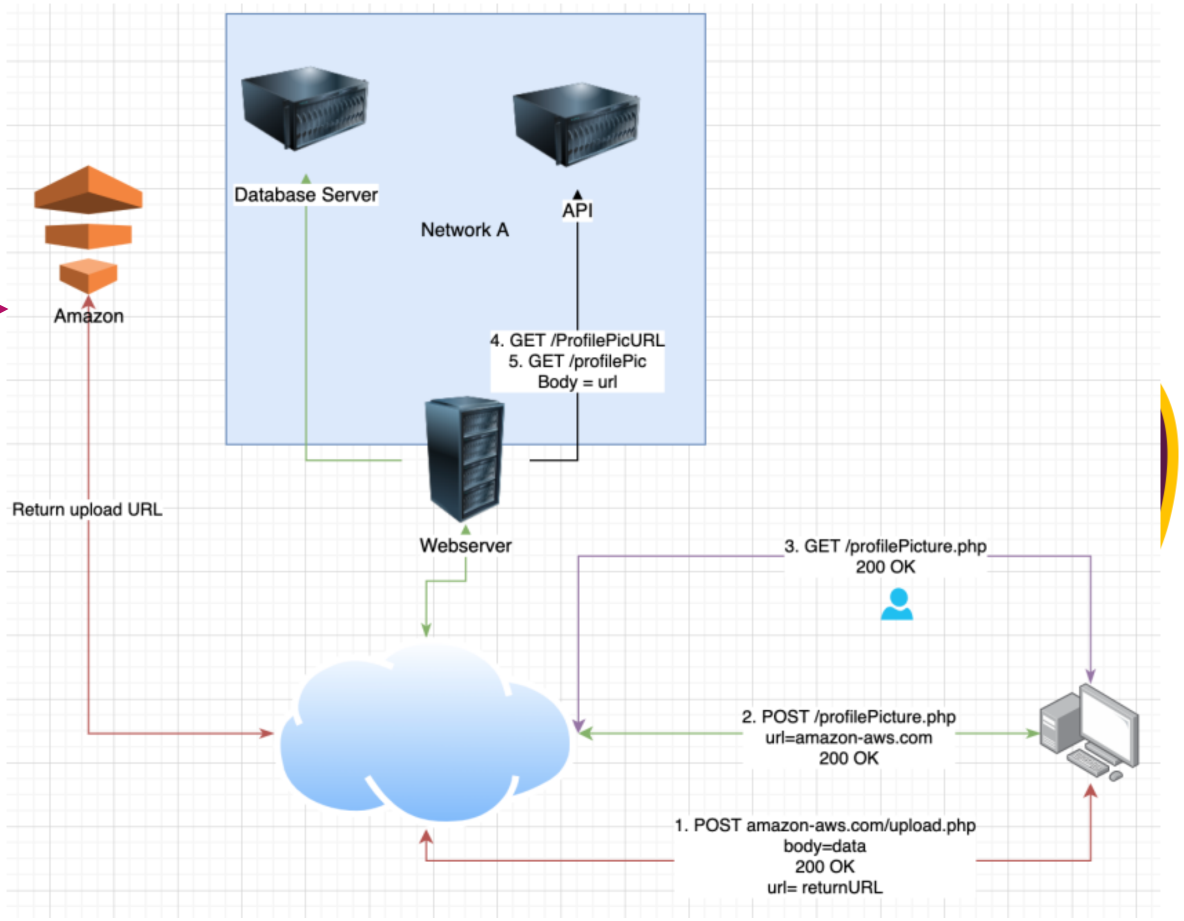
Business Logic Vulnerabilities

- Whatever the manual tells you not to do, do it
- Mess with every single parameter and analyse results
  - Look for the developer restraints and try to look for edge values
  - I.E. if the developer wants us to rate from 1 to 5
    - Try 0
    - Try 6
    - Try –1
    - Try a
    - Try ...
- Mess with the order of requests
- Copy parameters from the response to the request to see if they can be changed

Server Side Request Forgery

Server Side Request Forgery

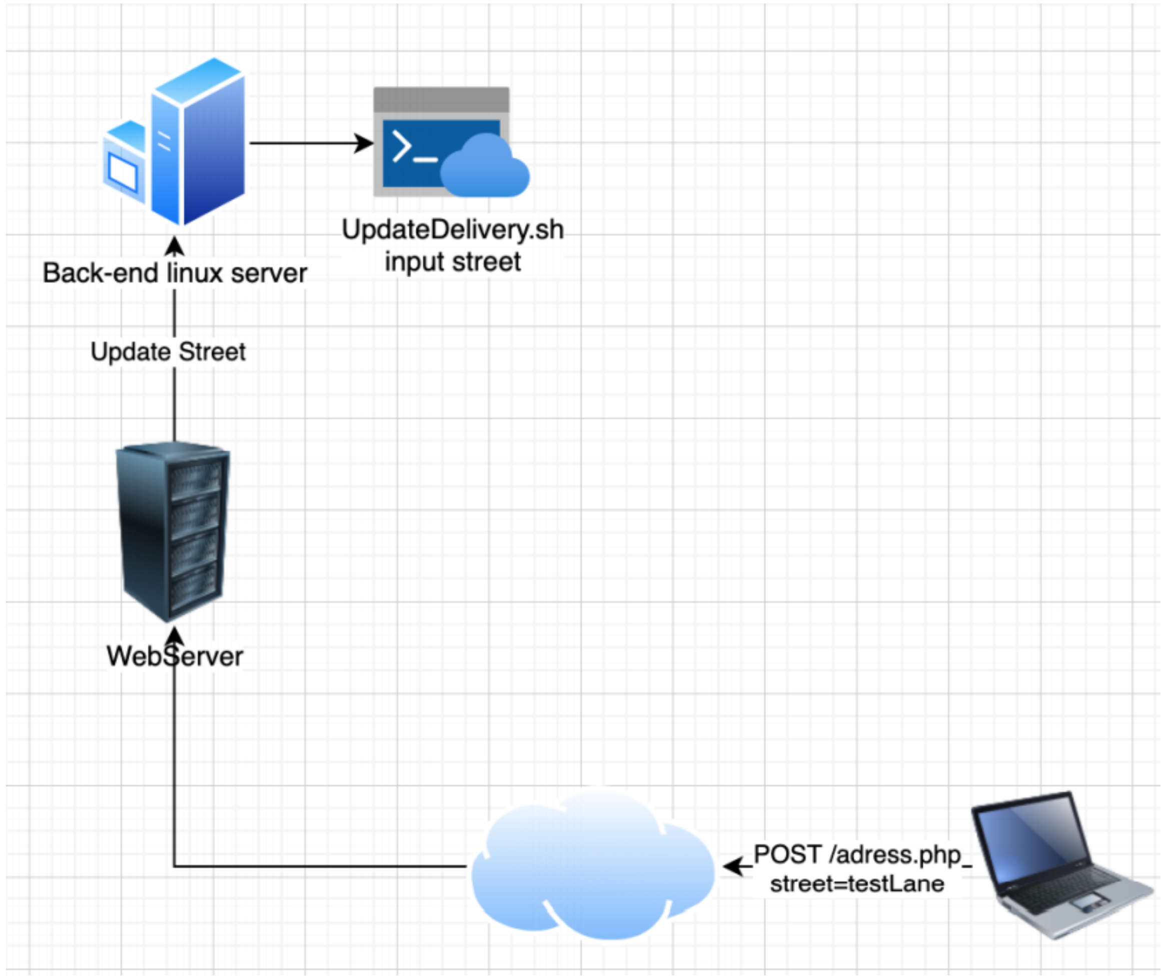




Server Side Request Forgery

- SSRF gives us access to internal servers we should not be able to access
- test
  - any URL that gets resolved by the server and that we can control
  - Partial URLs in the body instead of a full URL
  - URLs within data files such as XML files or CSV files (import functionality)
  - The referer header can sometimes contain SSRF defects
- Test for
  - SSRF against the server itself
  - SSRF against other backend systems
  - Blind SSRF
- Requires extensive knowledge of the network or brute forcing

OS Command injection



OS Command injection

OS Command injection

- Input sometimes gets sent to back-end sh scripts
- This input needs to be properly sanitised
  - The filtering is often blacklist based
  - If developer forgets one thing on blacklist, we have an entry point
- Test
  - Combine all command separators with all commands to make list
  - Encode the list
  - Test every single parameter
    - Tool: Burp intruder + self created list

CSRF

CSRF

- Test using
  - match and replace
  - Autorepeater plugin in burp
  - CSRF scanner burp

CSRF

- Match and replace
  - Add a match and replace rule
    - Type: Request body
    - Match: “CSRF=*” (replace this v alue with how your target does CSRF tokens)
    - Replace: “CSRF=“ << SEE NEXT SLIDE
    - Regex match: True

CSRF

- Test techniques
  - Remove the CSRF token from requests
  - Replace the CSRF token with a random value (for example 1)
  - Replace the CSRF token with a random token of the same restraints
  - Leave CSRF Parameter empty
  - Use a CSRF token that has been used before
    - See if you can request a CSRF by executing the call manually and use that token for the request

Finding more endpoints

Finding more endpoints

- Javascript analysis
- Wayback URLs
- Google dorking
- API documentation