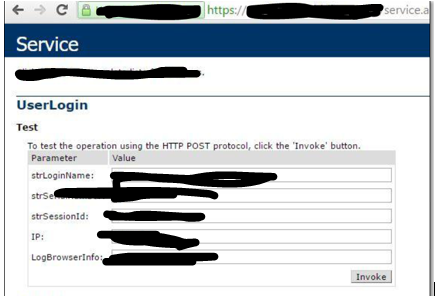
“How to Detect”

1. **A1 How to Detect**, “Broken Object Level Authorization”

Broken object level authorization on API is common if there is no any proper ACL defined “Access Control List” or a white listing of user functions, below is one of the way to detect A1 if the API is openly exposed to end user, note: it is mandatory that the API should directly accessible to the tester after that he/she would be able to detect or penetrate further,

1. Access API via browser, check if it is accessible first before detecting/executing the A1
2. Use any interception tool (this will apply if any mobile app using the API) to intercept the communication and check API url if it is accessible to you,
3. Let’s assume the API is accessible and there is a vulnerability A1 now how attacker can penetrate further please use below method to detect,
4. <https://newmobileapp.owasp.com/service.asmx?op=UserLogin>

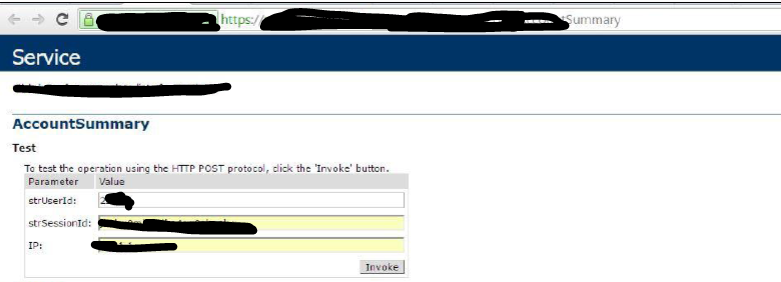
I have invoked the web service function with the parameters seen in figure below:



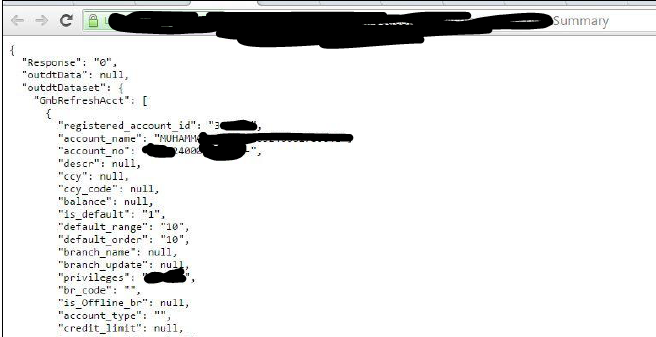
The request generated a response with user id and Session which can be seen in the figure below:



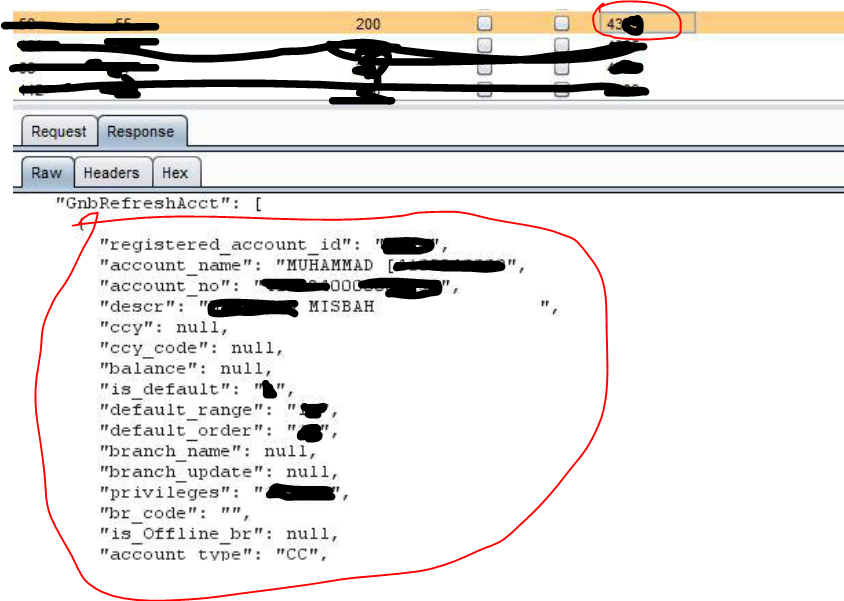
I have entered the retrived paramaetr of user id 21\*\*\* and Session in account services web method which can be seen in the figure below:



Once I have executed the above user id I got the following information from server,



Now the next phase is to check the A1 “broken Object Level Authorization” to do so I have change the parameter **having the same authorized object** which was 21\*\* now changed it to 43\*\* nd I got the other object details shown in following figure.



======================END==================================

**Detecting A4:2019** - Lack of Resources & Rate Limiting

Getting benefit of missing Rate limit on object or parameter I have start the brute force attack to enumerate the user as much as I can, as a result I have discovered hundreds of user id by starting the brute force on the API, below is the way of detecting rate limit or lack of resources,

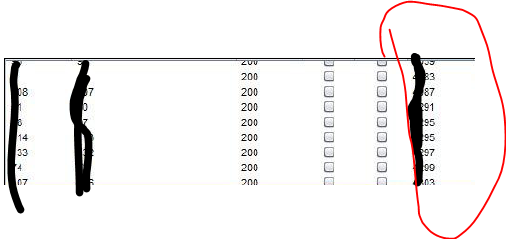
1. First check if the API is accessible directly using any browser
2. Check if there is any parameter passing in the URL that could be used to start brute-forcing
3. Parameter can also be detected using any tool to spider the directories.

Lets tack an example of below URL,

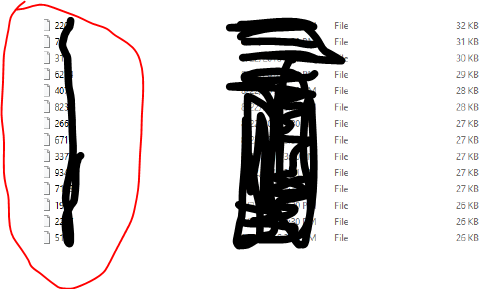
There is API with parameter 2376 and there is no any rate limit or lack of resources

[https://newmobileapp.Owasp.com/service.asmx?op=userid=2376](https://newmobileapp.owasp.com/service.asmx?op=userid=2376),

i start brute forcing on above url and retrieved hundreds of customer data within few minutes which can be used for further penetrating.



User details can be extracted from these files,

================END=====================================