

声明：仅用于技术研究，不恰当使用造成的危害后果自负

## 简言

首先，文章里没有直接 rce 的 exp，想要 exp 的大哥抱歉了 23333  
但是由于执行了 groovy 代码，所以瞎搞的话我也不知道会出啥问题

挖jenkins好几个月了，一直莫得比较好用的洞，12.05 公布越权动态调用就搞得心里痒痒，给重新挖了下，此文章写的比较随意，大佬轻喷，欢迎交流

## 分析过程

### 先讲讲偷窥思路

从 orange 大佬12.20公布发现了一个 jenkins 未授权 rce 开始，我就一直在试图将其挖掘出来，一直到 1.16 大佬公布第一部分 Jenkins 动态路由利用这篇文章，才真正拿到触发链，不过到最后 groovy sandbox 绕过实在是不会了.....

在 orange 的文章中，其实帮助最大的还是贴出了官方的一个漏洞通告链接（没有收集漏洞通报的良好习惯2333）和对 Descriptor 的理解还有利用官方在 1.8 号公布了一个通告：<https://jenkins.io/security/advisory/2019-01-08/#jenkins-security-advisory-2019-01-08>

大致讲的是 pipeline 这个插件里对groovy脚本进行解析的时候会出现 sandbox 被绕过的情况以及 REST API 也会直接访问到，但是他是需要 Overall/Read 权限的

这里我思考了一会儿，虽然这通告涉及到 bypass groovy sandbox，一定程度上和 RCE 有关联，但是它需要权限的。Overall/read 在jenkins 中属于比 ANONYMOUS 权限高一丁点的权限，但是它默认是 FALSE 的，就是默认配置安装的 Jenkins 是没有 Overall/read 权限的，不登录的情况下只有 ANONYMOUS 权限，然后呢我们需要的是一个未授权 RCE

，所以这里很有可能是一个最后的一个任意代码执行的地方，那么还得需要去寻找绕过权限检查的触发路由（这里我没有对 REST API 直接访问导致的 RCE 做研究）

在更早的时候，官方有通告如下：

<https://jenkins.io/security/advisory/2018-12-05/>

这也是 CVE-2018-1000861，造成的影响呢是能够一定程度上调用 Jenkins 中的任意 getter 函数，造成了越权的情况

将两者结合起来的话就很有可能是未授权 RCE

首先因为我之前一直在搞 Jenkins

的反序列化黑名单，所以对其路由解析过程算是比较熟悉（因为不熟悉ACL机制，甚至当时根本没听说过，导致没有察觉到这个任意 getter 调用的漏洞点，只是觉得jenkins的路由映射做的很奇怪233333），所以能在拿到官方通报后就能猜到整体触发流程，接下来我会先对 CVE-2018-1000861 做一点简单的分析，然后再从寻找 RCE 的角度去分析挖掘方式

然后还有想说的就是，从通告中可以直接拿到插件代码diff，可以很方便的找到漏洞点

### 动态路由形成过程

CVE-2018-1000861 其实还是和之前 orange 大佬发现的 Jenkins 任意文件读取相关核心：Stapler 有关系

它对于 Jenkins 来说就是一个小型路由生成器

完完全全可以直接从 web.xml 开始跟入 Stapler 类中 service 函数的，因为官方补丁diff的话，反而找不到 Jenkins

路由生成过程，对后面的漏洞挖掘会造成一定理解上的困难

我不贴 service 函数的代码，简单说说就好，service 函数中最后调用到了 invoke 函数，一直跟着 invoke 关键字的话，会进入到 Stapler 中的 tryInvoke 函数中，关键代码块如下：

```
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
289
```

那么关键就是 dispatcher 的生成，跟进 getMetaClass 函数

```
198
199 public MetaClass getMetaClass(Klass<?> c) {
200     if(c==null) return null;
201     synchronized(classMap) {
202         MetaClass mc = classMap.get(c);
203         if(mc==null) {
204             mc = new MetaClass( webApp: this,c);
205             classMap.put(c,mc);
206         }
207         return mc;
208     }
209 }
```

主要是根据传入的 node 变量生成了一个 MetaClass 对象，node 变量经过了一定包装，大致是获取了类相关信息，继续跟进 MetaClass 看看

```
87 /*package*/ MetaClass(WebApp webApp, Klass<?> klass) {
88     this.class = klass.toJavaClass();
89     this.class = klass;
90     this.webApp = webApp;
91     this.baseClass = webApp.getMetaClass(klass.getSuperClass());
92     this.classLoader = MetaClassLoader.get(klass.getClassLoader());
93     buildDispatchers();
94 }
95
```

记录 node 的各种信息，然后调用 buildDispatchers 函数，函数体太长总共301行就不贴出来了，这个函数主要功能就是对 node 对应的 class、此 class 的父类、此 class 继承类（简单来说就是继承家族树中的所有类）进行一个函数信息提取，然后获取指定的函数相关信息，做一个函数反射调用和 url 节点名称的存储，存储在 MetaClass.dispatchers 中，这就是制作路由的过程了，也是 orange 文章中提到的，如下：

整個漏洞要從 Jenkins 的 動態路由 講起，為了給開發者更大的彈性，Jenkins(嚴格來講是 Stapler)使用了一套 Naming Convention 去匹配路由及動態的執行! 首先 Jenkins 以 `/` 為分隔將 URL 符號化，接著由 `jenkins.model.Jenkins` 為入口點開始往下搜尋，如果符號符合 (1) Public 屬性的成員或是 (2) Public 屬性的方法符合下列命名規則，則調用並繼續往下呼叫：

1. get()
2. get(String)
3. get(Int)
4. get(Long)
5. get(StaplerRequest)
6. getDynamic(String, ...)
7. doDynamic(...)
8. do(...)
9. js(...)
10. 擁有 @WebMethod 標註的方法
11. 擁有 @JavaScriptMethod 標註的方法

只要在继承家族树中，任意类满足在以上 11 种规则的函数，统统可以直接在 URL 中访问到（get 的意思是以 get 开始的名字，do 和 js 同理）

然后呢，我随便截一个 dispatcher 的生成过程，如下图：

```

FunctionList getMethods = node.methods.prefix("get");
// check public selector methods of the form NODE.getToken()
for (Function f : getMethods.signature()) {
    if (f.getName().length() < 3)
        continue;

    String name = compile(f.getName().substring(1)); // "getToken" -> "f.getToken"
    final Function ff = f.contextualize(new TraversalMethodContext(name));

    dispatchers.add(new NameBasedDispatcher(name) {
        public boolean doDispatch(RequestImpl req, ResponseImpl rsp, Object node) throws IOException, ServletException, IllegalAccessException, InvocationTargetException {
            if (traceEnable())
                trace("uri=%s, req, node, expression: %s, getMethodName: %s()",
                    req.getUri(), req, node, expression, ff.getName());
            req.getAdapter().invoke(req, rsp, ff.invoke(req, rsp, node));
            return true;
        }
        public String toString() {
            return String.format("%1$s() for uri=%1$s/...", ff.getQualifiedName(), name);
        }
    });
}
}

```

那么这里整个就是一次对当前 node 进行动态路由制作的过程了，如果思路延展一下，可以发现整个是一个迭代的过程，我稍微描述下：访问 <http://target/123/abc>

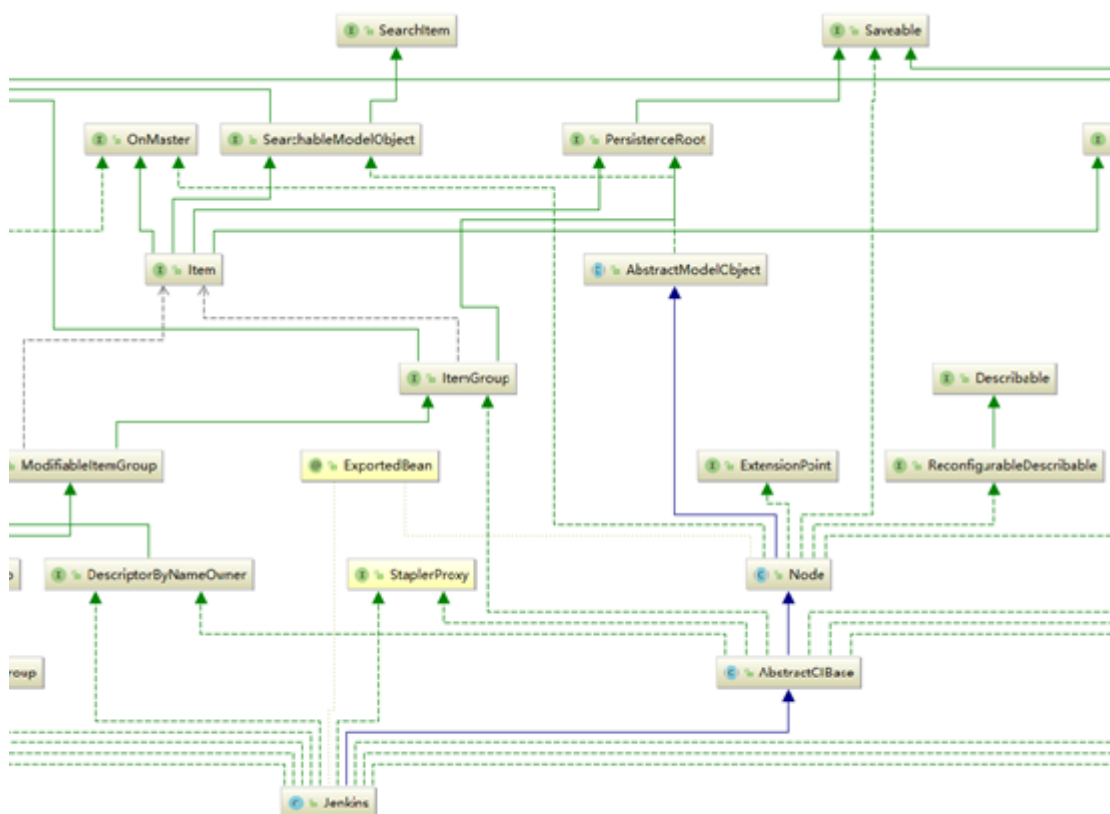
先解析 root 节点（也就是第一次传入的 node），然后对 123 做适配，匹配到的是满足上图中 11 中规则的并且存在于当前 root 节点家族树的函数，并对他进行反射调用，目标函数流程走完后，根据返回结果类型进行下一步处理，因为在上图中的 req.getStapler().invoke 调用中，目标函数的返回结果传递给了第三个形参，如下：

```
void invoke(RequestImpl req, ResponseImpl rsp, Object node ) throws IOException, ServletException {
    if(node==null) {
        // node is null
        if(!Dispatcher.isTraceEnabled(req)) {
```

其实就是一个新的 node，那么对 abc 做适配的，就是新 node 的家族树中的函数了，以此递归下去，直到匹配出错或者所有节点匹配完成

- 如果 url 是以 /\$stapler/bound/ 开头的话, 就 org.kohsuke.stapler.bind.BoundObjectTable 为 root
- 如果其他的话就是以 Jenkins.model.Jenkins 为 root

上者和 Object 绑定功能相关，默认情况下是不会有利用点的。稍微看看 Jenkins 类的一部分家族树，如下图：



所以我们访问 Jenkins 的时候，第一个解析的路由节点，就是上图中满足那11个规则的函数

## 绕过路由访问限制

那么限制在哪儿呢？

还是在 Stapler 类中 tryInvoke 函数中，函数一开头就做了一个操作如下：

```
684 boolean tryInvoke(RequestImpl req, ResponseImpl rsp, Object node) throws IOException, ServletException {
685     if(traceable())
686         traceEval(req,rsp,node);
687
688     if(node instanceof StaplerProxy) {
689         if(traceable())
690             traceEval(req,rsp,node, prefix:"((StaplerProxy)", suffix:".getTarget()");
691         Object n = null;
692         try {
693             n = ((StaplerProxy)node).getTarget();
694         } catch (RuntimeException e) {
695             if (function.renderResponse(req,rsp,node,e))
696                 return true; // Let the exception serve the request and we are done
697             else
698                 throw e; // unprocessed exception
699     }
```

如果是 StaplerProxy 的实现类，那么就会调用当前 node 的 getTarget() 函数，从家族树中看见 Jenkins 这个类确实实现了 StaplerProxy，那么它的 getTarget 函数如下：

```
public Object getTarget() {
    try {
        checkPermission(READ);
    } catch (AccessDeniedException e) {
        if (!isSubjectToMandatoryReadPermissionCheck(Stapler.getCurrentRequest().getRestOfPath())) {
            return this;
        }
        throw e;
    }
    return this;
}
```

上图中会检查当前用户是否拥有 READ 权限，如果没有的话会抛出异常，然后进入 isSubjectToMandatoryReadPermissionCheck 函数中，并且带入了当前访问的路由节点名，如下：

```
public boolean isSubjectToMandatoryReadPermissionCheck(String restOfPath) {
    for (String name : ALWAYS_READABLE_PATHS) {
        if (restOfPath.startsWith(name)) {
            return false;
        }
    }

    for (String name : getUnprotectedRootActions()) {
        if (restOfPath.startsWith("/") + name + "/" || restOfPath.equals("/") + name)) {
            return false;
        }
    }

    // TODO SlaveComputer.doSlaveAgentJnlp; there should be an annotation to request unprotected access
    if (restOfPath.matches(regex: "/computer/[^/]+/slave-agent[.]jnlp")
        && "true".equals(Stapler.getCurrentRequest().getParameter("encrypt"))) {
        return false;
    }

    return true;
}
```

满足上面三个条件的路由节点名，都会放行当前请求通过，如果都不满足则记录请求并转到 login 窗口

查看一下 ALWAYS\_READABLE\_PATHS：

```
/**
 * Urls that are always visible without READ permission.
 *
 * <p>See also: {@link #getUnprotectedRootActions}</p>
 */
private static final ImmutableSet<String> ALWAYS_READABLE_PATHS = ImmutableSet.of(
    "/login",
    "/logout",
    "/accessDenied",
    "/adjuncts/",
    "/error",
    "/oops",
    "/signup",
    "/tcpSlaveAgentListener",
    "/federatedLoginService/",
    "/securityRealm",
    "/instance-identity"
);
```

上图中的节点路由都可以通过，不过这数量少得可怜。但是其中 securityRealm 就是绕过 ACL 限制的跳板入口。

这里不禁想起来自己挖洞的时候记录2333333，如下：

```
113 ALWAYS_READABLE_PATHS = ImmutableSet.of(
114     "/login",
115     "/logout",
116     "/accessDenied",
117     "/adjuncts/",
118     "/error",
119     "/oops",
120     "/signup",
121     "/tcpSlaveAgentListener",
122     "/federatedLoginService/",
123     "/securityRealm",
124     "/instance-identity"
125 );
126 以上的路由都访问过，没有大作用
127
```

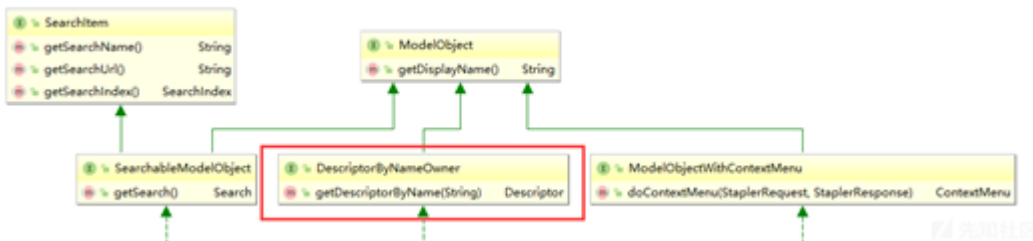
疯狂打脸233333，因为当时草草看了下，一心对反序列化黑名单着迷，没有对相关函数的返回类型做家族树调查

跳-跳-跳

对 /securityRealm 访问时进行动态调试发现，返回的是 Hudson.security.HudsonPrivateSecurityRealm 类，我们跟过去，查看其中的 getUser 函数，如下：

```
423 public User getUser(String id) {
424     return User.getById(id, create: true);
425 }
```

这里根据传入的下一节点名当做 id，然后生成一个 User 出来，稍微测试了下，不存在的用户也能正常生成 User，未对这个原因进行深究，此时目标函数返回的是 User 类。我们看看 User 类的家族树，找到一个关键点如下：



查看 getDescriptorByName 如下：

```
public interface DescriptorByNameOwner extends ModelObject {
    default Descriptor getDescriptorByName(String id) {
        return Jenkins.getInstance().getDescriptorByName(id);
    }
}
```

其实就是调用了 Jenkins.getDescriptorByName，这个函数主要根据传入的 id（String），然后获取到程序中所有继承了 Descriptor 的子类

总结下这里的利用类连续跳动过程：

Jenkins -> HudsonPrivateSecurityRealm -> User -> DescriptorByNameOwner -> Jenkins -> Descriptor

Descriptor 绕过 ACL 的主角

Descriptor，从这个类名都能感觉到，是描述功能相关，并且其中拥有大量的 getter，从设计上思考的话，这个 Descriptor 很有可能是会对很多功能点的相关描述

动态调了下，默认配置的 Jenkins 拥有约579个 Descriptor

```
▼ descriptors = {Iterators$6@10423}
▼ iterables = {Iterable[2]@10490}
> 0 = {ExtensionList@10492} size = 579
```



## 如何寻找 RCE

其实在研究动态路由的过程中，就发现了，想要 RCE 还是要依靠插件中的一些脚本解析功能才行，但是突然懒癌发作，看着一堆插件就不想动手去分析了23333

这里我们还是简单一点，根据官方漏洞通告寻找补丁 diff：

<https://github.com/jenkinsci/workflow-cps-plugin/commit/d09583eda7898eafdd15297697abdd939c6ba5b6>

从中看见几个修改的类文件：

src/main/java/org/jenkinsci/plugins/workflow/cps/CpsFlowDefinition.java  
src/main/java/org/jenkinsci/plugins/workflow/cps/CpsGroovyShellFactory.java  
src/main/java/org/jenkinsci/plugins/workflow/cps/replay/ReplayAction.java

稍微筛选下，就能找出 CpsFlowDefinition 才是主角（虽然通过 ReplayAction 也能够触发，但是根据我的跟踪中发现需要一定权限才可以）  
查看关键点 CpsFlowDefinition\$DescriptorImpl 如下：

```
@Extension
public static class DescriptorImpl extends FlowDefinitionDescriptor {

    @Override
    public String getDisplayName() { return "Pipeline script"; }

    public FormValidation doCheckScript(@QueryParameter String value, @QueryParameter boolean sandbox) {
        return sandbox ? FormValidation.ok() : ScriptApproval.get().checking(value, GroovyLanguage.get());
    }

    public JSON doCheckScriptCompile(@QueryParameter String value) {
        try {
            CpsGroovyShell trusted = new CpsGroovyShellFactory( @execution null).forTrusted().build();
            new CpsGroovyShellFactory( @execution null).withParent(trusted).build().getClassLoader().parseClass(value);
        } catch (CompilationFailedException x) {
            return JSONArray.fromObject(CpsFlowDefinitionValidator.toCheckStatus(x).toArray());
        }
        return CpsFlowDefinitionValidator.CheckStatus.SUCCESS.asJSON();
        // Approval requirements are managed by regular stapler form validation (via doCheckScript)
    }
}
```

继承的 FlowDefinitionDescriptor，这个类继承自 Descriptor，上图中有两个满足那11个规则的函数，其中带上 @QueryParameter 注解的参数都可以通过参数请求传递进来

OK，到此为止已经拿到了 Jenkins 的无限 RCE 触发链，但是最终它是解析 Groovy 脚本的，并且似乎上了沙盒，虽然官方补丁 diff 中含有一点 bypass sandbox 的技术点，但是我对 groovy 是一窍不通，搞了好几天都没办法，各位师傅如果有经验的话，试试呢？

目前为止的效果

都是官方补丁 diff 的 bypass

使用 Grab 注解如下：

```
host:8080/security/Realm/User/orich1/descriptorByName/org.jenkinsci.plugins.workflow.cps.CpsFlowDefinition/checkScriptCompile?value=@Grab%28group%3D%27foo%27%2C%20module%3D%27bar%27%2C%20include%3D%27%27%29

Post data: ☐ Enable Referrer
[{"column":0,"line":1,"message":"startup failed:\r\nGeneral error during conversion: Error grabbing Grapes -- {unresolved dependency: foo@bar:1.0: not found}\r\n\r\n\tat sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)\r\n\r\n\tat sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:43)\r\n\r\n\tat java.lang.reflect.Constructor.newInstance(Constructor.java:423)\r\n\r\n\tat org.codehaus.groovy.reflection.CachedConstructor.invoke(CachedConstructor.java:77)\r\n\r\n\tat org.codehaus.groovy.runtime.callsite.ConstructorSite$ConstructorSiteNoUnwrap.callConstructor(ConstructorSite.java:84)\r\n\r\n\tat org.codehaus.groovy.runtime.callsite.CallSiteArray.defaultCallConstructor(CallSiteArray.java:60)\r\n\r\n\tat org.codehaus.groovy.runtime.callsite.AbstractCallSite.callConstructor(AbstractCallSite.java:247)\r\n\r\n\tat groovy.grape.GrapeIvy.getDependency(GrapeIvy.java:100)\r\n\r\n\tat sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)\r\n\r\n\tat sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)\r\n\r\n\tat sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)\r\n\r\n\tat java.lang.reflect.Method.invoke(Method.java:498)\r\n\r\n\tat org.codehaus.groovy.runtime.callsite.PogoMetaMethodSite$PogoCachedMethodSite.invoke(PogoMetaMethodSite.java:169)\r\n\r\n\tat org.codehaus.groovy.runtime.callsite.PogoMetaMethodSite.callCurrent(PogoMetaMethodSite.java:59)\r\n\r\n\tat org.codehaus.groovy.runtime.callsite.CallSiteArray.defaultCallCurrent(CallSiteArray.java:104)\r\n\r\n\tat org.codehaus.groovy.runtime.callsite.AbstractCallSite.callCurrent(AbstractCallSite.java:154)\r\n\r\n\tat groovy.grape.GrapeIvy.resolve(GrapeIvy.java:100)\r\n\r\n\tat groovy.grape.GrapeIvy$resolve$1.callCurrent(Unknown Source)\r\n\r\n\tat org.codehaus.groovy.runtime.callsite.CallSiteArray.defaultCallCurrent(CallSiteArray.java:104)"}]
```

使用 ASTTest 注解如下：

```
[{"column":1,"line":1,"message":"unable to resolve class org.jenkinsci.plugins.workflow.libs.Library$","status":"fail"}, {"column":1,"line":1,"message":"unable to resolve class org.jenkinsci.plugins.workflow.job.WorkflowJob$","status":"fail"}]
```

总结（我菜如狗）

Stapler 的动态路由制作过程

Jenkins 本身的白名单路由

Descriptor 的利用，这里的利用过程相当曲折，从 Jenkins 入口跳出去最终再跳到 Jenkins 自己这里获取 Descriptor，然后再从各种继承类中找到 groovy 解析的利用点，膜 orange

Groovy sandbox 的绕过，实在是不会弄了，Orz，求大哥们教教

Links：

hacking-Jenkins-part1-play-with-dynamic-routing：

<https://devco.re/blog/2019/01/16/hacking-Jenkins-part1-play-with-dynamic-routing/>

jenkins官方通告：

<https://jenkins.io/security/advisory/2019-01-08/#jenkins-security-advisory-2019-01-08>

<https://jenkins.io/security/advisory/2018-12-05/>

pipeline-groovy插件相关：

<https://plugins.jenkins.io/workflow-cps>

<https://github.com/jenkinsci/workflow-cps-plugin/commit/d09583eda7898eafdd15297697abdd939c6ba5b6>

点击收藏 | 2 关注 | 2

[上一篇：过D盾webshell分享](#) [下一篇：安全工具——wfuzz](#)

1. 2 条回复



[leveryd](#) 2019-01-29 14:45:42

大佬有没有挖别的Java框架的？

0 回复Ta



[orich1](#) 2019-02-09 10:59:01

[@leveryd](#) 挖不动挖不动

0 回复Ta

---

[登录](#) 后跟帖

先知社区

---

[现在登录](#)

热门节点

---

[技术文章](#)

[社区小黑板](#)

目录

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)