

---

# CODEGATE 2013 Write-Up

12<sup>th</sup> 내숨씨를보여줄때가왔군 Team

---

*Written by PnP (Pwn&Play 前TeamWANG)*

2013. 03. 03

---

---

# Table of Contents

---

---

## 1. Rank

## ~~2. Solutions – Vulnerab~~

## 2. Solutions – Binary

- i. 100 Point
- ii. 200 Point
- iii. 300 Point

## 3. Solutions – Web

- i. 100 Point
- ii. 200 Point
- iii. 300 Point
- iv. 400 Point
- v. 500 Point

## 4. Solutions – Forensics

- i. 100 Point
- ii. 200 Point
- iii. 300 Point
- iv. 400 Point
- v. 500 Point

## 5. Solutions – Misc

- i. 100 Point
- ii. 200 Point – 1
- iii. 200 Point – 2
- iv. 300 Point – 1
- v. 300 Point – 2

# 1. Rank

내숨씨를보여줄때가왔군 (South Korea)

Vulnerab	Binary	Web	Forensics	Misc
100 41/580	100 159/580	100 107/580	100 67/580	100 287/580
200 49/580	200 57/580	200 110/580	200 42/580	200 86/580
300 41/580	300 25/580	300 31/580	300 23/580	200 23/580
400 13/580	400 10/580	400 71/580	400 8/580	300 81/580
500 12/580	500 3/580	500 63/580	500 22/580	300 27/580

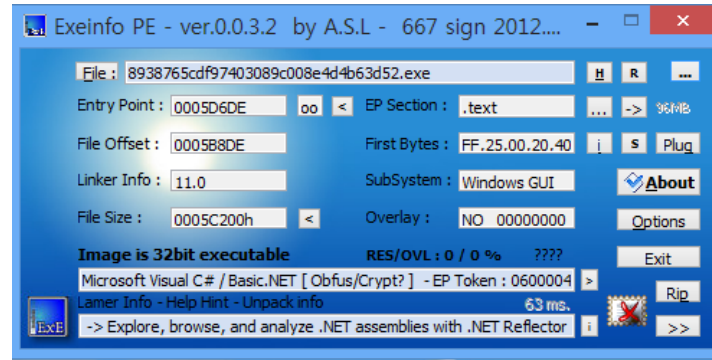
Rank

1		European NO..	7106
2		GoN	6614
3		More Smoked..	6209
4		PPP	6020
5		Team ClevCode	5905
6		WBC	5806
7		sutegoma2	5711
8		MachoMan	5707
9		ForbiddenBITS	5704
10		int3pids	5300
11		plus	5100
12		내숨씨를보여줄때가왔군	5100

1		European NO..	7106
2		GoN	6614
3		More Smoked..	6209
4		PPP	6020
5		Team ClevCode	5905
6		WBC	5806
7		sutegoma2	5711
8		MachoMan	5707
9		ForbiddenBITS	5704
10		int3pids	5300
11		plus	5100
12		내숨씨를보여줄때가왔군	5100
13		코드레드	4911
14		ufologists	4900
15		CLGT	4705
16		B10S	4611
17		Null@Root	4504

## 2. Solutions – Binary

### i. 100 Point



C# 으로 컴파일 된 바이너리 파일입니다.

작년과 마찬가지로 Binary100 문제가 C# 이네요.

C# 의 경우 .NET Reflector 로 디컴파일을 할 수 있습니다.

```
public void Transformable(string Data)
{
    if (Data.Length == 0x10)
    {
        if (this.xorToString(AESCrypt.Encrypt(this.r.Text, KeyValue)) == this.lowkey)
        {
            MessageBox.Show(AESCrypt.Decrypt(this.StringToXOR(this.a.ByteToString_t(this.c)), KeyValue));
        }
        else
        {
            MessageBox.Show("Do you know ? " + AESCrypt.Decrypt(this.StringToXOR(this.a.ByteToString_t(this.d)), KeyValue));
        }
    }
}
```

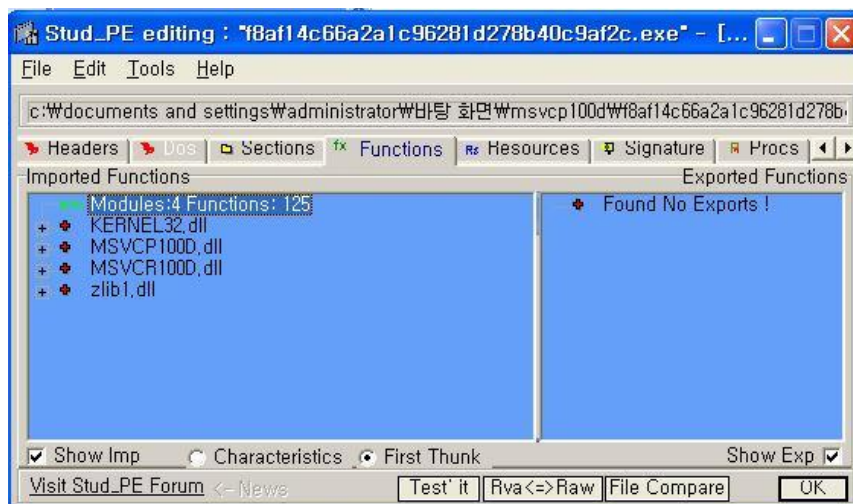
키 값을 출력하는 루틴이네요. 그대로 복사하여 C# 컴파일을 하면 됩니다.

**Key : code9ate2013 Start**

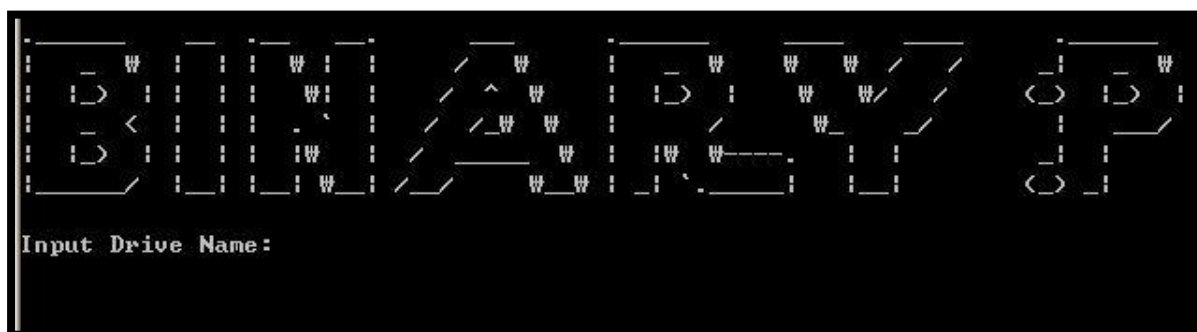
## ii. 200 Point



VS 10으로 컴파일 되어 있으며 패킹은 되어있지 않습니다. 먼저 이 프로그램이 로드 하는 모듈을 살펴 보았습니다.



압축 라이브러리인 zlib1.dll 을 로드하네요.



실행을 해보면 다음과 같습니다. 아무 값이나 입력해 보았습니다.

```

Status: 0x0      StartOfPartition1: 0x0      StartOfPartition2: 0x0 0x0
Type: 0x0      EndOfPartition1: 0x0      EndOfPartition2: 0x0 0x0
StartOfSector: 0x0 0x0 0x0 0x0      NumberOfSector: 0x0 0x0 0x0 0x0
System Message: 8pFHHoMssjtoucpX4EdPgcrdzuKXgEFU7iNur4YzDrOdf yNOA/bp7lX=
-----
Offset(h)      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F      ASCII Value
0x00000000      EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00      .R.NTFS      .....
0x00000010      00 00 00 00 00 F8 00 00 3F 00 FF 00 3F 00 00 00      .....?...?...
0x00000020      00 00 00 00 80 00 80 00 E5 D5 FF 04 00 00 00 00      .....

```

하드디스크 섹터와 그에 해당하는 hex dump 를 출력합니다. 그런데 수상해 보이는 System Message 라는 문자열이 있네요.

Base64 로 인코딩 된 형태 같습니다. System Message 를 출력하는 루틴으로 가보니 처음부터 저 상태로 정해져 있는 값을 출력할 뿐이 었습니다.

```

0x0000001E0      20 72 b5 73 74 t
0x0000001F0      00 00 00 00 00 t
Message: 8pHsLEdImJ==

```

아래로 가보니 Message 라며 출력하는 값이 따로 있네요. 이 값은 사용자의 Input 에 따라 값이 달라지는 듯 합니다.

Message 값과 System Message 의 연관성을 알아보기 위해 Input Drive Name : 부분을 우회해봅니다.

```

0041430= 60 40 4100 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065 1066 1067 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083 1084 1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096 1097 1098 1099 1100 1101 1102 1103 1104 1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115 1116 1117 1118 1119 1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131 1132 1133 1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146 1147 1148 1149 1150 1151 1152 1153 1154 1155 1156 1157 1158 1159 1160 1161 1162 1163 1164 1165 1166 1167 1168 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181 1182 1183 1184 1185 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197 1198 1199 1200 1201 1202 1203 1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301 1302 1303 1304 1305 1306 1307 1308 1309 1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328 1329 1330 1331 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341 1342 1343 1344 1345 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 1358 1359 1360 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392 1393 1394 1395 1396 1397 1398 1399 1400 1401 1402 1403 1404 1405 1406 1407 1408 1409 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425 1426 1427 1428 1429 1430 1431 1432 1433 1434 1435 1436 1437 1438 1439 1440 1441 1442 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454 1455 1456 1457 1458 1459 1460 1461 1462 1463 1464 1465 1466 1467 1468 1469 1470 1471 1472 1473 1474 1475 1476 1477 1478 1479 1480 1481 1482 1483 1484 1485 1486 1487 1488 1489 1490 1491 1492 1493 1494 1495 1496 1497 1498 1499 1500 1501 1502 1503 1504 1505 1506 1507 1508 1509 1510 1511 1512 1513 1514 1515 1516 1517 1518 1519 1520 1521 1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540 1541 1542 1543 1544 1545 1546 1547 1548 1549 1550 1551 1552 1553 1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565 1566 1567 1568 1569 1570 1571 1572 1573 1574 1575 1576 1577 1578 1579 1580 1581 1582 1583 1584 1585 1586 1587 1588 1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1610 1611 1612 1613 1614 1615 1616 1617 1618 1619 1620 1621 1622 1623 1624 1625 1626 1627 1628 1629 1630 1631 1632 1633 1634 1635 1636 1637 1638 1639 1640 1641 1642 1643 1644 1645 1646 1647 1648 1649 1650 1651 1652 1653 1654 1655 1656 1657 1658 1659 1660 1661 1662 1663 1664 1665 1666 1667 1668 1669 1670 1671 1672 1673 1674 1675 1676 1677 1678 1679 1680 1681 1682 1683 1684 1685 1686 1687 1688 1689 1690 1691 1692 1693 1694 1695 1696 1697 1698 1699 1700 1701 1702 1703 1704 1705 1706 1707 1708 1709 1710 1711 1712 1713 1714 1715 1716 1717 1718 1719 1720 1721 1722 1723 1724 1725 1726 1727 1728 1729 1730 1731 1732 1733 1734 1735 1736 1737 1738 1739 1740 1741 1742 1743 1744 1745 1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760 1761 1762 1763 1764 1765 1766 1767 1768 1769 1770 1771 1772 1773 1774 1775 1776 1777 1778 1779 1780 1781 1782 1783 1784 1785 1786 1787 1788 1789 1790 1791 1792 1793 1794 1795 1796 1797 1798 1799 1800 1801 1802 1803 1804 1805 1806 1807 1808 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818 1819 1820 1821 1822 1823 1824 1825 1826 1827 1828 1829 1830 1831 1832 1833 1834 1835 1836 1837 1838 1839 1840 1841 1842 1843 1844 1845 1846 1847 1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860 1861 1862 1863 1864 1865 1866 1867 1868 1869 1870 1871 1872 1873 1874 1875 1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889 1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903 1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917 1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931 1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026 2027 2028 2029 2030 2031 2032 2033 2034 2035 2036 2037 2038 2039 2040 2041 2042 2043 2044 2045 2046 2047 2048 2049 2050 2051 2052 2053 2054 2055 2056 2057 2058 2059 2060 2061 2062 2063 2064 2065 2066 2067 2068 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103 2104 2105 2106 2107 2108 2109 2110 2111 2112 2113 2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144 2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158 2159 2160 2161 2162 2163 2164 2165 2166 2167 2168 2169 2170 2171 2172 2173 2174 2175 2176 2177 2178 2179 2180 2181 2182 2183 2184 2185 2186 2187 2188 2189 2190 2191 2192 2193 2194 2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221 2222 2223 2224 2225 2226 2227 2228 2229 2230 2231 2232 2233 2234 2235 2236 2237 2238 2239 2240 2241 2242 2243 2244 2245 2246 2247 2248 2249 2250 2251 2252 2253 2254 2255 2256 2257 2258 2259 2260 2261 2262 2263 2264 2265 2266 2267 2268 2269 2270 2271 2272 2273 2274 2275 2276 2277 2278 2279 2280 2281 2282 2283 2284 2285 2286 2287 2288 2289 2290 2291 2292 2293 2294 2295 2296 2297 2298 2299 2300 2301 2302 2303 2304 2305 2306 2307 2308 2309 2310 2311 2312 2313 2314 2315 2316 2317 2318 2319 2320 2321 2322 2323 2324 2325 2326 2327 2328 2329 2330 2331 2332 2333 2334 2335 2336 2337 2338 2339 2340 2341 2342 2343 2344 2345 2346 2347 2348 2349 2350 2351 2352 2353 2354 2355 2356 2357 2358 2359 2360 2361 2362 2363 2364 2365 2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390 2391 2392 2393 2394 2395 2396 2397 2398 2399 2400 2401 2402 2403 2404 2405 2406 2407 2408 2409 2410 2411 2412 2413 2414 2415 2416 2417 2418 2419 2420 2421 2422 2423 2424 2425 2426 2427 2428 2429 2430 2431 2432 2433 2434 2435 2436 2437 2438 2439 2440 2441 2442 2443 2444 2445 2446 2447 2448 2449 2450 2451 2452 2453 2454 2455 2456 2457 2458 2459 2460 2461 2462 2463 2464 2465 2466 2467 2468 2469 2470 2471 2472 2473 2474 2475 2476 2477 2478 2479 2480 2481 2482 2483 2484 2485 2486 2487 2488 2489 2490 2491 2492 2493 2494 2495 2496 2497 2498 2499 2500 2501 2502 2503 2504 2505 2506 2507 2508 2509 2510 2511 2512 2513 2514 2515 2516 2517 2518 2519 2520 2521 2522 2523 2524 2525 2526 2527 2528 2529 2530 2531 2532 2533 2534 2535 2536 2537 2538 2539 2540 2541 2542 2543 2544 2545 2546 2547 2548 2549 2550 2551 2552 2553 2554 2555 25
```

사용자가 입력한 값을 암호화 하는 루틴을 살펴보았습니다.

```
PUSH EAX
CALL f8af14c6.0041114A          memset
ADD ESP,0C
MOV EAX,DWORD PTR SS:[EBP-290]  strlen 값 (원본 stream 의 크기)
PUSH EAX
LEA ECX,DWORD PTR SS:[EBP-284]  입력한값 ( 원본 stream 의 주소)
PUSH ECX
LEA EDX,DWORD PTR SS:[EBP-2A8]  압축될 stream 이 저장될 곳의 크기가 저장된주소(12)
PUSH EDX
MOV EAX,DWORD PTR SS:[EBP-2B4]  압축될 stream 이 저장될 곳의 주소
PUSH EAX
CALL f8af14c6.004113A7          string 압축 zlib_compress
ADD ESP,10
```

제일 처음 만나게 되는 암호화 함수는 zlib 라이브러리의 compress 라는 함수 입니다.

그 다음 만나는 함수는 00411A60 함수인데 64bit 의 키를 사용하는 것으로 보아 base64 입니다.

그런데 사용하는 키 테이블이 일반적인 키테이블(ABCD~...7890-/+ ) 이 아니라

“QwErTyUiOpAsDfGhJkLzXcVbNm0246813579qWeRtYuIoPaSdFgHjKlZxCvBnM+ /=” 네요.

따라서 암호화 된 System Message를 복호화 하는 순서는 다음과 같습니다

#### CustomBase64 -> Zlib.Decompress

몇 번의 삽질 후에 customBase64를 Original Base64 변환 하는 파이썬 스크립트를 코딩했습니다.

```
#CustomBase64Code

from string import maketrans

base64fixTable =
maketrans("QwErTyUiOpAsDfGhJkLzXcVbNm0246813579qWeRtYuIoPaSdFgHjKlZxCvBnM+ /=",
"ABCDEFGHijklmnopqrstuvwxyz0123456789+/=");
def correctbase64(str):
    return str.translate(base64fixTable)

print
correctbase64('8pFHHoMssjtoucpX4EdPgcrdzuKXgEFV7iNur4YzDr0dfyNOA/bp7lX=')
```

```
#Zlib.DecompressCode

import zlib
str_object1 = open('file', 'rb').read()
```

```
dc_obj = zlib.decompressobj()
out = dc_obj.decompress(str_object1)
f = open('filedecomp', 'wb')
f.write(out)
f.close()
```

Result :



**Key : BuRn 2013 VuLn**

### iii. 300 Point

게임을 생성하는 루틴(sub\_401B800)을 xRef를 따라가서 분석 하다보면 전역 변수가 있는데 이것을 분석하여 difficulty에서 생성된 키들을 출력하는게 문제의 목표였습니다.

저 같은 경우 다른 잡다한 함수에서 difficultyLevel가 초기화 되는것을 보고 루틴과 변수를 찾아 내었습니다.

또한, 전역 변수에 대해 자세히 분석 하시다 보면 dword\_416E30가 보이는데 어레이를 추출해서 파싱하여 뜯어내다 보면 답이 "EYE IS PRECIOUS"라는 것을 알 수 있습니다

**Key : EYE IS PRECIOUS**



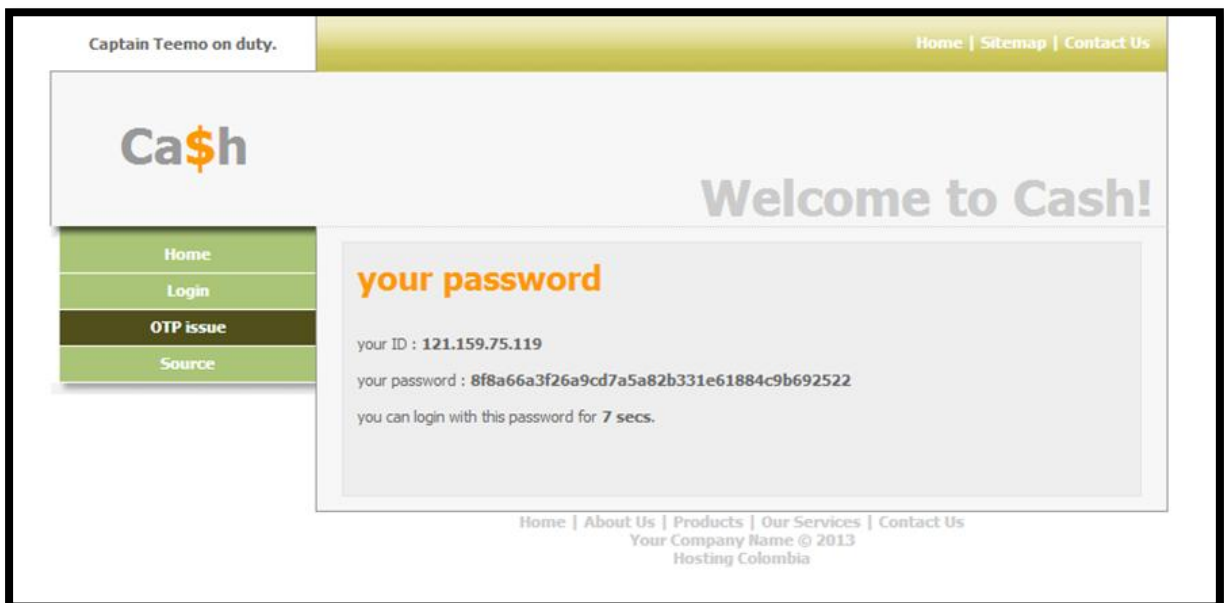
### 3. Solutions – Web

#### i. 100 Point

간단한 SQL 인젝션 문제입니다. raw injection 이 가능하므로 hash함수값을 계속 증가하여 'or' 이 발견되는 부분을 찾아서 password 에 넣어서 admin 을 로그인하면 답이 출력 되는 것을 알 수 있습니다.

Key : DAER0NG\_DAER0NG\_APPLE\_TR33

#### ii. 200 Point



이 문제를 살펴보면 소스를 다운받는 부분을 볼 수 있습니다. 다운 받아서 보면 PHP파일이 총 4개가 있는데 login\_ok.php와 otp\_util.php를 살펴보았습니다.

먼저 아래는 login\_ok.php의 소스코드.

```

1 <?php
2     include("../otp_util.php");
3
4     $flag = file_get_contents($flag_file);
5
6     if (isset($_POST["id"]) && isset($_POST["ps"])) {
7         $password = make_otp($_POST["id"]);
8         sleep(3); // do not brute force
9
10        if (strcmp($password, $_POST["ps"]) == 0) {
11            echo "welcome, <b>".$_POST["id"]."</b><br />";
12            echo "<input type='button' value='back' onclick='history.back();' />";
13
14            if ($_POST["id"] == "127.0.0.1") {
15                echo "<hr /><b>".$flag."</b><br />";
16            }
17        } else {
18            echo "<script>alert('login failed..');history.back();</script>";
19        }
20    }
21
22 ?>

```

그리고 otp\_util.php의 소스코드입니다.

```

1 <?php
2
3     $flag_file = "/flag.txt";
4
5     function make_otp($user) {
6         // access for 20secs.
7         $time = (int)(time()/20);
8         $seed = md5(file_get_contents($flag_file)).md5($_SERVER['HTTP_USER_AGENT']);
9         $password = sha1($time.$user.$seed);
10        return $password;
11    }
12
13 ?>

```

저 부분을 보고 로컬에서 돌리면서 나온 문자열을 계속 login\_ok.php에 Brute Forcing을 했더니 key가 나왔습니다.

key : LL1K3\_caitlyn\_LOLOL

### iii. 300 Point

파일 안의 js의 난독화를 풀면 어드민 페이지가 보입니다. 그리고 contact 부분의 blind SQL 인젝션이 가능함을 이용해서 스크립트를 짜서 아이디와 비밀번호를 가져오면 md5 화 되어있으나 간단한 비밀번호를 사용하는 유저때문에 로그인이 가능합니다.

그 후 로그인을 하면 마이페이지에서 해킹한 시간을 알 수 있습니다.

Key : Edward Van Coon(2013-02-07)

## iv. 400 Point

Math Challenge가 나옵니다.

가입해서 Challenge를 보면 p=문제번호&k=답 이런 식입니다. 문제 번호와 답이 int라는것을 감안하면 쿼터가 없다는 것을 예상하여, Gp=1||1&k1||1# 를 시도해서 SQL인젝션을 하면 클리어 됩니다.

클리어 하였을 때 사용했던 소스코드입니다.

```
Private Sub Form_Load()  
    Dim winhttp As New WinHttpRequest  
    For i = 0 To 31  
        winhttp.Open "GET",  
"http://58.229.122.14/site/page/auth.php?p=2||2&k=1||1"  
        winhttp.SetRequestHeader "Cookie", "PHPSESSID=세션"  
        winhttp.SetRequestHeader "Content-Type", "application/x-www-form-  
urlencoded"  
        winhttp.Send  
        winhttp.WaitForResponse  
        MsgBox StrConv(winhttp.ResponseBody, vbUnicode)  
    Next i  
End Sub
```

Key : someday\_day\_one\_day\_milian

## v. 500 Point

JavaScript를 복호화 하면 sha1을 이용하여 페이지와 그의 체크 섬을 확인합니다.

체크 섬을 조작하여 simulator.php를 조작하면 SQLite 경로가 보입니다.

여기서 추가적으로 체크 섬을 한번 더 조작해서 GM의 SQLite에서 memo를 gzuncompress(gunzip)이용하여 클리어 하시면 됩니다.

Key : W3LC0M3\_T0\_L0L0L0L

## 4. Solutions – Forensics

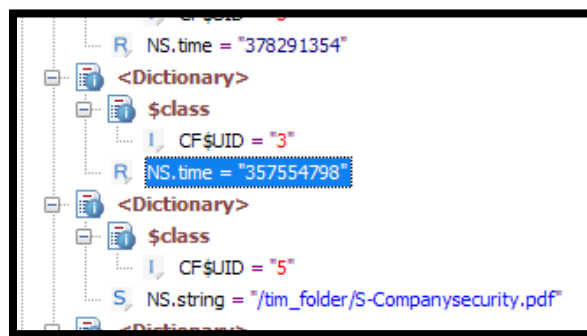
### i. 100 Point

아이폰 이미지 에서 유출된 문서를 찾으라는 문제입니다.

아이폰 어플 중 5BB3AF5D-01CC-45D9-947D-977DB30DD439 에서 Dropbox 라는 어플이 의심스러워서 찾던 도중 라이브러리 폴더에서 스냅샷을 찍은 사진이 있는 것을 보고, 이 파일이 유출된 문서라 추측하고 최종시간과 파일크기가 맞다고 생각 할 수 있습니다.

그리고 나서 Upload 폴더안에 sqlite 파일들을 읽어보면 MODIFIED 시간을 보면 그 파일의 수정 시간 인 것을 알 수 있습니다. 하지만, 시간 값이 온전하지 못한 것을 볼 수 있는데, 이를 MAC 타임스탬프로 변환 하면 수정시간이 2012-12-27 17:55:54 인 것을 알 수 있습니다.

그리고나서 업로드 시간은 Caches 폴더의 cache.db의 base64 를 디코딩 한 후 plist를 Oxygen을 이용해 읽어보면 그 업로드된 시간이 캐시 되어 남아있는 것을 알 수 있습니다.

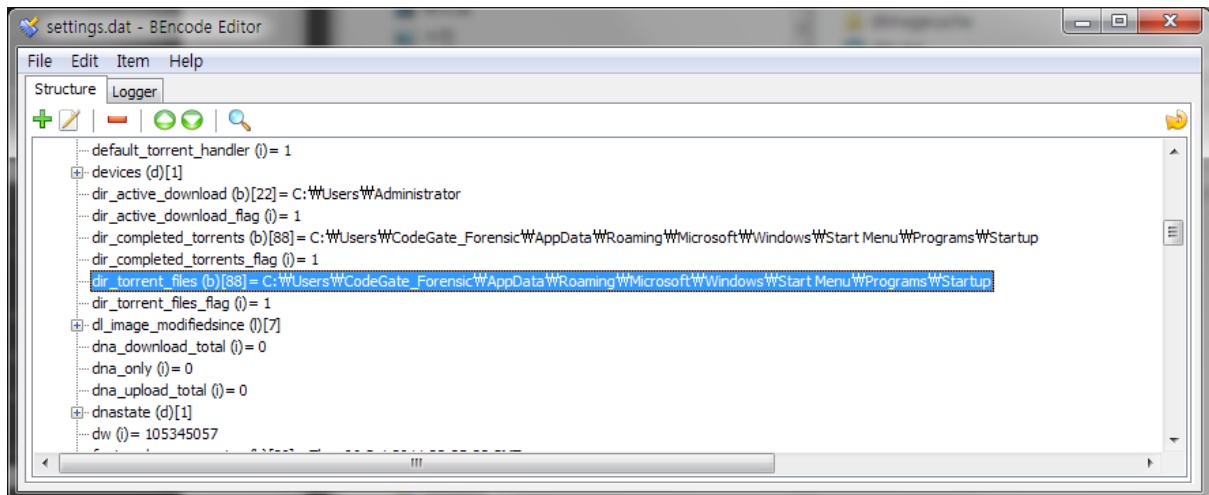


Key : 2012-12-27 17:55:54 \_2012-05-01 17:46:38 \_S-Companysecurity.pdf\_2.1MB

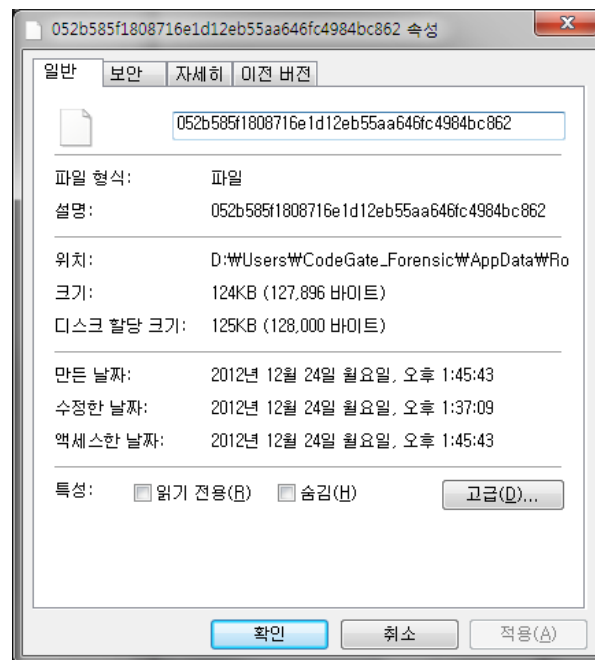
### ii. 200 Point

문제에서 요구하는 파일을 찾기 위해서 어디서 다운로드 되었나 생각 해보아야 하는데 인터넷 캐시는 없으므로 프로그램을 통해, 즉 Torrent 를 통해 받았음을 짐작할 수 있습니다.

어떤 파일을 받았는지 알아보기 위해 BEncoder를 이용해 uTorrent의 설정 파일을 열어보면 아래와 같이 다운로드 완료된 토렌트 파일의 위치를 알 수 있습니다.



해당 경로로 이동하여 파일을 생성시각을 동영상 다운로드 시각이라 생각하고 인증할 수 있었습니다.



결과적으로 위 파일의 MD5 체크섬 값인 449529c93ef6477533be01459c7ee2b4D와, 생성 시각(또는 접근 시각) 2012/12/24 13:45:43을 통해 아래 값을 만들어 낼 수 있습니다.

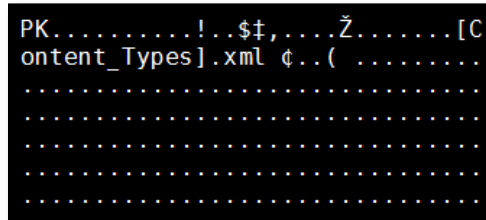
SHA-1(449529c93ef6477533be01459c7ee2b4D\_2012/12/24 13:45:43)

**Key : 20789d8dae4efe2ece9194ef08b1c752c04b5e47**

### iii. 300 Point

먼저 문제의 워드 파일을 열어보니 스테가노그래피인 것 같아서 내부에 있는 이미지 파일을 가져 오기 위해 zip형식으로 변환한 후, word\media에서 이미지들을 추출할 수 있었습니다.

헌데, emf 파일들이 썸썸해서 모두 Hex Editor로 보던 중, image6.emf 파일이 doc 파일임을 알 수 있었습니다.



```
PK.....!...$!,...Ž.....[C
ontent_Types].xml  ..( .....
.....
.....
.....
.....
.....
```

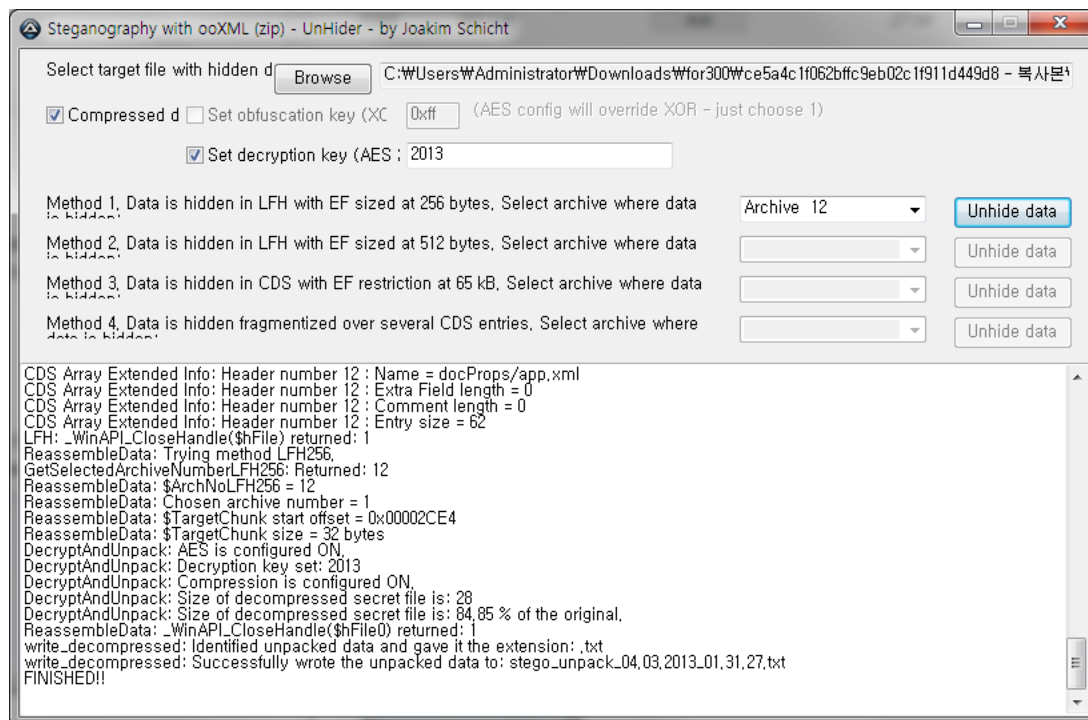
파일을 열어보니 빨간 글씨로 2013이라고 적혀있는 것을 볼 수 있었습니다(위협).



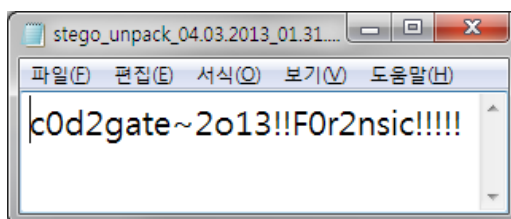
이를 통해 doc steganography에 관한 정보를 찾던 중 New Steganographic Techniques for the OOXML File Format라는 OOXML형식의 파일 포맷에 대한 스테가노그래피 기법을 간략히 설명해놓은 PDF를 볼 수 있었습니다

([https://www.google.co.kr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CDAQFjAA&url=http%3A%2F%2Fforensic.korea.ac.kr%2Fxe%2F%3Fmodule%3Dfile%26act%3DprocFileDownload%26file\\_srl%3D17283%26sid%3D1d210b83346fa16b7b1bb1abdc9bad6d&ei=sYYzUZfbILGujAKIjoH4BQ&usg=AFQjCNGD8GagP1FiQNt\\_0rpkuXEXhOBXcA&bvm=bv.43148975,d.cGE&cad=rjt](https://www.google.co.kr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CDAQFjAA&url=http%3A%2F%2Fforensic.korea.ac.kr%2Fxe%2F%3Fmodule%3Dfile%26act%3DprocFileDownload%26file_srl%3D17283%26sid%3D1d210b83346fa16b7b1bb1abdc9bad6d&ei=sYYzUZfbILGujAKIjoH4BQ&usg=AFQjCNGD8GagP1FiQNt_0rpkuXEXhOBXcA&bvm=bv.43148975,d.cGE&cad=rjt))

이 문서를 통해 ooXML 관련 Steganography 프로그램을 찾다가 ooXML Steganography UnHider라는 프로그램을 찾을 수 있었고 이를 이용하여 아래와 같이 문제를 해결할 수 있었습니다.



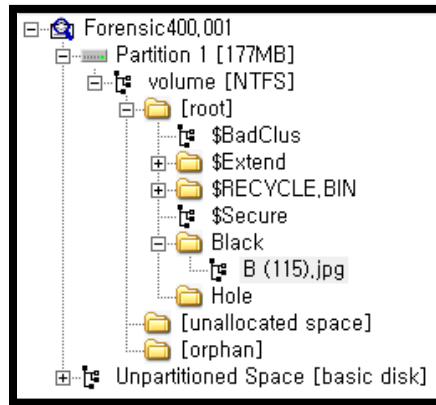
먼저, 아까 전 emf 파일로 위장한 워드 파일 쓰여있던 ‘2013’을 비밀번호로 넣고 Unhide하면 아래와 같은 텍스트가 나오게 되는데, 이 값이 바로 인증키가 됩니다.



**Key : c0d2gate~2o13!!F0r2nsic!!!!**

#### iv. 400 Point

문제파일을 받아 이미지 마운트 시키기 전에 안에 있던 파일들 뒤적거리다 보니 Black 폴더의 “B (115).jpg”에 ADS가 걸려있음을 알 수 있었습니다. 곧바로 숨겨진 파일을 추출 후 Hex Editor로 분석 시도하려 했지만 암호화가 되어 있는 것 같아 분석을 보류해두었습니다.



이후 Black=older, Hole=p/w, compare 라는 힌트가 나오고 나서 여러 방법을 시도해 볼 수 있었습니다.

처음에는 Beyond Compare를 이용해 푸는 방법이 아닐까하고 일일이 비교해보았으나 진전이 없어 다른 방향의 분석을 시도하게 되었습니다.

그러던 중 Hole 폴더 내 모든 이미지 파일의 EOF 부근에 1Byte크기의 값이 공통적으로 들어간다는 것을 발견할 수 있었고, 수정된 시간을 기반으로 하여 7y\_5n4\_10221CuPw7R73\_rrK9Pd라는 배치를 만들어 낼 수 있었습니다.

당시 대회 종료 1시간 전이라 워낙 정신이 없어서 기술적으로 푼다는 생각보다 추측으로 풀어야겠다는 생각이 컸기에(가능성은 거의 없었지만서도-\_-;;) 문자 배열을 추측하기 시작했습니다

문자열을 만드는 규칙은 다음과 같았습니다.

1. 133t체라는 가정 하에 각 숫자에 맞는 알파벳을 조합.
2. 단어마다 구분자로 언더 바(\_) 사용.

규칙 하에 문자열을 만들어 내는 도중 PASSwORd(P422w0Rd)라는 문자를 찾을 수 있었고, 이후 STrInG(57r1n9)문자열을 찾아내어 아래와 같이 24개의 추측 테이블을 구성할 수 있었습니다.

7ru3CyP7_P422w0Rd_57r1n9_K1
7ru3CyP7_P422w0Rd_K1_57r1n9
7ru3CyP7_57r1n9_P422w0Rd_K1
7ru3CyP7_57r1n9_K1_P422w0Rd
...
K1_P422w0Rd_7ru3CyP7_57r1n9
K1_P422w0Rd_57r1n9_7ru3CyP7
K1_57r1n9_7ru3CyP7_P422w0Rd
K1_57r1n9_P422w0Rd_7ru3CyP7

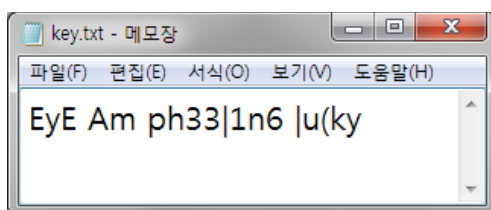
테이블은 추출된 단어 “7ru3CyP7” , “P422w0Rd” , “57r1n9” , “K1” 를 각각 1, 2, 3, 4로 바꿔 프로그래밍을 통해 치환하였습니다.



이후 문자열에 있는 7ru3CyP7를 통해 TrueCrypt라는 문자를 나타냄을 알 수 있었고 가끔가끔 TrueCrypt를 분석을 하는 이유로 기본적인 사용 방법은 알고 있었기에 곧 ADS 걸렸던 파일이 TrueCrypt 볼륨임을 알 수 있었습니다.

TrueCrypt를 이용해 볼륨을 선택하고 위 스트링 테이블 중 하나를 비밀번호로써 입력해야 하는데 어이없게도 첫 번째 스트링인 “7ru3CyP7\_P422w0Rd\_57r1n9\_K1” 에서 풀려버리더군요(..)

그렇게 해서 인식된 드라이브에 접근하여 key.txt를 읽으면 답이 나옵니다.



**Key : EyE Am ph33|1n6 |u(ky**

## v. 500 Point

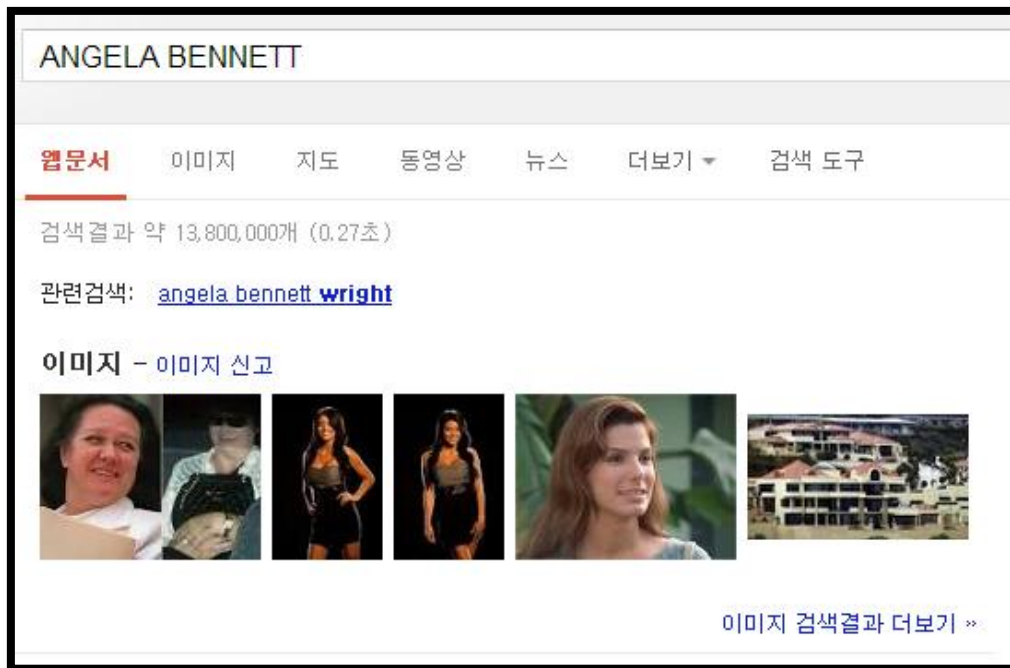
R-Studio를 이용해 이미지 파일을 Scan하여 Recognized 8 영역을 탐색하면 k3y2013.JPG 파일을 볼 수 있습니다. 해당 파일의 MD5 체크섬을 인증하면 답이 됩니다.

**Key : 597eb84759c836cf9889e07770ffacf7**

## 5. Solutions – Misc

### i. 100 Point

구글 번역기로 돌려보니 ‘안젤라 베넷 로그인 미국 기관. 에너지 원자력위원회’ 다음과 해석됩니다. 그냥 한번 안젤라 베넷이 뭐 하는 가스나인가 검색해보니 다음의 검색결과가 나옵니다.



위 처럼 나오는데 세 번째 사진에 ‘산드라 블록’ 이 있습니다. 제가 얼마 전에(2/28) 안 랩에서 주최한 V스쿨이라는 캠프에 다녀왔는데 그 캠프에서 ‘영화 속 보안’ 이라는 강의를 했었습니다.

그 강의에서 저 여자가 나온 영화를 소개했었는데 세 번째 사진을 보자마자 그 강의를 생각났으며 검색결과를 토대로 영화를 찾아봤는데 역시나 ‘The Net’ 라는 해킹영화가 있더군요.

영화를 다운받아서 키 값을 찾아야되나 고민을 하고 있었는데 브레이크 쓰루가 탐나 막막해서 진짜, 진짜 그냥 구글에다가 ‘movie the net password’ 라고 검색해보니 다음과 요상한 사이트가 있었습니다. -> <http://mike.passwall.com/uselesstrivia/thenet.htm>

들어가보니 영화에 대한 정보가 나와있었으며 쪽 스크롤을 내려보면서 찾다보니 문제의 키 값을 찾을 수 있었습니다.

0 1:0 4	Key	Praetorian <TAB> Suspect ANGEL
		is near - advise immediately <CR>
		Archangel
0 0:4 6	Ae	/TALK/
0 0:3 9	Key	/Password:/ natoar23ae
0 0:1 9	Ae	/LOGIN/
0 0:0 6	Ae	/TELNET/

키 값은 ‘natoar23ae’ 이며 정말 빨리 풀었는데도 아쉽게 브레이크 쓰루를 실패한 문제였습니다.

**Key : natoar23ae**

## ii. 200 Point - 1

배열 내부의 개체 수가 91개가 존재하는 것을 보고 base91이라는것을 바로 눈치챈 후 클리어 하였습니다.

Key : HahA-LUCKY-Se7vN

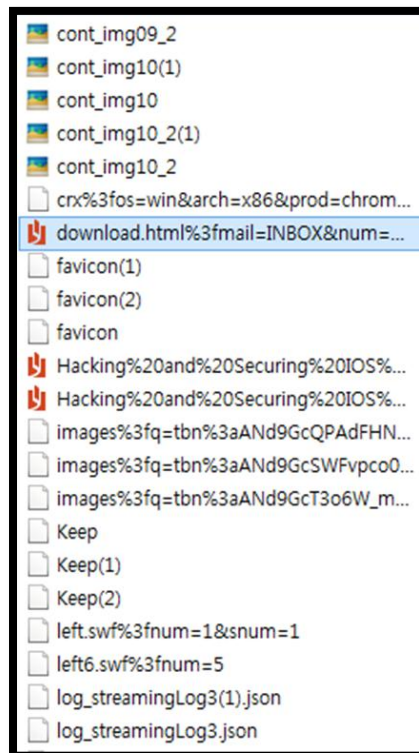
## iii. 200 Point - 2

요상한 바이너리 파일이 하나 있습니다. 일단 궁금증을 해결하기 위해 윈헥스로 실행해봤습니다.

00000010	FF FF FF FF FF FF FF FF	03 00 1C 00 33 32 2D 62	yyyyyyyy 32-b
00000020	69 74 20 57 69 6E 64 6F	77 73 20 37 2C 20 62 75	it Windows 7, bu
00000030	69 6C 64 20 37 36 30 30	04 00 2D 00 44 75 6D 70	ild 7600 - Dump
00000040	63 61 70 20 31 2E 38 2E	34 20 28 53 56 4E 20 52	cap 1.8.4 (SVN R
00000050	65 76 20 34 36 32 35 30	20 66 72 6F 6D 20 2F 74	ev 46250 from /t
00000060	72 75 6E 6B 2D 31 2E 38	29 00 00 00 00 00 00 00	runk-1.8)
00000070	74 00 00 00 01 00 00 00	78 00 00 00 01 00 00 00	t x
00000080	FF FF 00 00 02 00 32 00	5C 44 65 76 69 63 65 5C	yy 2 \Device\
00000090	4E 50 46 5F 7B 39 35 31	33 34 30 35 36 2D 41 39	NPF_{95134056-A9
000000A0	30 42 2D 34 30 41 35 2D	39 34 44 39 2D 35 30 37	0B-40A5-94D9-507
000000B0	46 33 43 34 31 41 36 38	44 7D 00 00 09 00 01 00	F3C41A68D}
000000C0	06 00 00 00 0C 00 1C 00	33 32 2D 62 69 74 20 57	32-bit W
000000D0	69 6E 64 6F 77 73 20 37	2C 20 62 75 69 6C 64 20	indows 7, build
000000E0	37 36 30 30 00 00 00 00	78 00 00 00 06 00 00 00	7600 x

다른건 모르겠고 맨 윗부분인데 느낌이 웬지 그냥 패킷파일 같았습니다.

그래서 파일 확장자를 '.pcap' 를 수정하였더니 정상적인 패킷 파일 이였습니다. 와이어샤크로 실행해 와이어샤크 기능 중 하나를 이용해 파일을 출력하였더니 다음과 같은 많은 파일들이 출력 되었습니다.



힌트도 벌써 두 개씩이나 나와있었던군요.

1- **[Misc3(200) Hint]** You can solve the question off-line.

2- **[Misc3(200) Hint]** Find out document.

솔직히 첫 번째 힌트는 무슨 소리인지 못 알아먹었었고 두 번째 힌트를 보니 추출해왔던 PDF 파일들이 심각히 의심스러웠습니다. 여기 있던 PDF는 ‘코드게이트 롤’과 ‘IOS 해킹’을 주제로 작성된 문서였는데 처음에는 IOS 해킹과 관련이 있으려나 하고 계속 찾아봤는데 문제가 없어서 멘붕하고 있다가 그냥 코드게이트 롤 PDF 파일이 심각히 의심스러워 윈헥스로 또 열어보았습니다.

한번 윈헥스를 이용해 파일을 추출해보았더니 코드게이트 로고 이미지와 또 코드게이트 롤 PDF 파일이 나왔습니다. 처음에는 그냥 문서 내에 ‘뭔가 있겠지, 뭔가 있어야 돼’ 하고 감이 안 와서 코드게이트 사이트 롤 원본 내용과 PDF의 내용을 일일이 비교하다가 아무것도 없어서 심각한 뻘찜을 느낀 뒤.. 문서를 밑으로 내려보니 키 값이.....있었습니다.....

- B. [ Preliminary match Rules ]
- o Questions consist of 5 categories (Vulnerability, Binary, Web, Forensics and miscellanea).
  - o Each category has 5 questions (100 points, 200 points, 300 points, 400 points, 500 points) according to its degree of difficulty.
  - o 4 categories (Vulnerability, Binary, Web and Forensic)'s all 100 points questions will be opened at the match starts.
  - o Yut Challenge Organizer will open the next questions randomly, no matter order of level. (No level's sequential limitation)
  - o In case of Miscellanea questions, they will be opened at the discretion of YUT Challenge organizer every 5 hours.
  - o Top 3 fastest teams to solve each question will be given bonus points: 3 points for 1st team, 2 points for 2nd team and 1 point for 3rd team. This Bonus points will be used to distinguish the teams' rank that has the same total score.
  - o Top ranked 8 teams will be decided according to their scores
- C. [ Final match Rules ]
- o After Preliminary match, it will be notified at YUT Challenge website

Key : DOUBT EVERY DOCUMENT FILE

Key : DOUBT EVERY DOCUMENT FILE

#### iv. 300 Point - 1

해당 파일을 PDFStreamDumper로 열어서 확인하였습니다.

처음 나오는 키 값은 12 오브젝트의 메타데이터로 존재함을 확인할 수 있었습니다(pdf를 실행시키고 파일 속성을 봐도 보임).

```
/Title (Confidential  
/Keywords (PDF, Miscellane  
/1st_key (nn@LiC!oU$) /Pro  
>>
```

2번째 키의 경우 5번 내용을 보면

**! Confidential Documents ! --- 2nd\_key is combination of strings in three objects;  
strlen(2nd\_key) == 14;**

라는 내용이 존재 하는데 이를 통해 3개의 오브젝트를 찾는 것이 중요한 것 같다고 생각할 수 있습니다.

이때 다른 오브젝트에는 내용이 존재하거나 font정보, page정보 등이 있지만 6,7,8 오브젝트의 경우 단순한 데이터만 존재한다는 것을 알 수 있습니다.

이 데이터를 확인하고 출력해보면 2nd\_Key가 나옵니다.

세번째 키의 경우 기존 파일의 11번 오브젝트에 보면 다시 %PDF-1.3으로 시작하는 내용이 존재한다는 것을 알 수 있습니다. PDF안에 PDF가 있다고 판단되어 해당 내용을 빼내고 내용을 확인하면 다시 내부의 내용을 확인할 수 있습니다.

이때 7번 오브젝트에서 자바스크립트 액션을 확인할 수 있는데, 5번에 있는 데이터의 내용을 cipher라는 변수에 집어넣고 다시 실행시키면 답이 나오게 됩니다.

Key : nn@LiC!oU\$\_PpPDdD[F\_F]ile\_4n4ly5i5

## v. 300 Point - 2

파일의 색이 나타낸것을 보아 RGB 를 통해 ascii area 로 추출 후 복호화 해보면 수상한 문자열이 reverse 된 것을 lower(md5(xor(hex(id),hex(password)))) 패턴을 통해 답을 추출할 수 있었습니다.

이름은 Kim jong-eun 인 것을 계성할 수 있고 또 저 안에서 AND eht tooB 가 password 옆에 있으므로 이것을 reverse 하여 모두 input 하면 답이 나옵니다.

```
è\,ÆÈŽØ,,ŽÀ¹ÿ{ä{ø¤&^TrlesÇEGFéíÿ
„.ŽÀ»'+.µ±+k,Ń7.Ži.&VE7F.$ö2.&ö÷
F-æráââZ"Î^íf..í.K.Í..1.<Ń.F°·N¹
²P.Í.'÷¬Ń.âqè]¹ºÉóŃèb¾®.,„»>¬¿0...
.<»>ãŃ¿º..{è..»¹¾0.ði.Ç¾†¾féýÿ¾ž
.†¾'è.5h¤<t<ŃÍ.Féòÿ^ÂPQRW'Í.<¤As
ÈuN?'AS..>È...GA'ìŃ.ž.üeé.IOP°<ŃÍ.
º'ìŃ..¹Ń.Žæ/öö¥•Æ5.5.ex¢Hf0.Ç
AF0.ùÇ^ööâ•µÆ;ìqè†ÿÍ.))drowssa
p(xeh,)di(xeh(rox(5dm(rewolAND e
ht tooBWelcome to RedStarOS ;)ÚÚ
You must Input the Name and Pass
ÚÚName : ÚÚThe Key is ÚÚPass : n
ue-gnoj miInvalid ID or Passwor
d ;(
```

Key : 38df6e4a037792070fabd08d5ca75fc3