The DGA of Ranbyus

Archived: 2026-04-05 16:33:21 UTC

Ranbyus is a trojan that steals banking information — among other personal data. End of April 2015 I first noticed samples of Ranbyus that use a Domain Generation Algorithm (DGA) to generate the domains for its command and control (C2) traffic:

```
 hcfoopojnuqxho.su
undrdsbhivryqn.tw
dkehliueofdued.net
mpuakxjqpscfpj.com
eelolbwmfmtkae.pw
noppsmyiijqujh.in
joxrsxwdybbgqb.me
(...)
```

In this post I show how the DGA works by reversing the following sample:

filename
: `_RANDOMNUM_6_11__vozvrat.exe`

filetype
: PE32 executable (GUI) Intel 80386, for MS Windows

md5
: fa57f601402aab8168dea94c7c5f029f

sha256
: dc4f3340ca8e623a5a77eb95411696fc25a7e6f5ef657ac9fd76eb4bc11c16b4

malwr
: link

I focused my efforts exclusively on the domain generation part of Ranbyus. Refer to the blog posts of Aleksandr Matrosov here and here for an in-depth analysis of Ranbyus.

Algorithm

This section shows the algorithm behind the domains of Ranbyus and its seeding and parametrization.

Callback Loop

The next image represents the part of the Ranbyus that tries to find a valid C2 target. It consists of an outer loop (`month_loop`) and an inner loop. The register `edi` holds the index of the outer loop. It runs from 0 down to -30. The number of iterations for the inner loop is specified by a parameter of the DGA (set to 40 in all analysed samples):

The first act of the outer loop is to get the current time:

Ranbyus then subtracts days from the current date according to the index of the outer loop:

The resulting date will be used to seed the DGA with a granularity of one day. In the first iteration, the DGA uses the current date. In the next iteration — when the index is -1 — yesterday's date is used. This continues up to 30 days in the past if need be. So even though the DGA generates a fresh set of domains every day, it also checks the domains of past days. This gives the DGA the benefit of fast changing domains in case domains get blocked or sinkholed, while at the same time enabling older domains to be used for up to one month if they still work.

The inner loop generates the domains for the day with `the_dga` and makes the callback. In case of failure, Ranbyus sleeps for `wait_time` milliseconds (500 in my samples) and retries up to `nr_of_domains` (40) different domains.

DGA Parameters and Seed

Apart from the current date, the DGA is seeded with a hardcoded magic number:

The number of domains per day is hardcoded to 40:

The wait time after a failed callback attempt is set to 500 ms:

Ranbyus also uses a hard-coded list of top level domains:

The top level domains are: *.in*, *.me*, *.cc*, *.su*, *.tw*, *.net*, *.com*, *.pw*, and *.org*. The last domain *.org* is never used due to a bug of the DGA. The top level domains are tested one after another (except the last one), starting at a day-dependent offset:

The error of subtracting 1 from the modulus is repeated also when picking the letters of the second level domain.

The DGA

This is the disassembly of the DGA routine:

The subroutine generates domains in two independent parts:

1. the top level domain is picked from the hardcoded list shown above
2. the second level domain is generated.

The following disassembly shows how the top level domain is picked:

Starting at the day dependent offset determined earlier, the algorithm picks the domains in a circular fashion, omitting the last domain because the DGA wrongly subtracts one from the modulus.

```
 [".in", ".me", ".cc", ".su", ".tw", ".net", ".com", ".pw", ".org"][offset % (9-1)]
offset++
```

The disassembly for the second level domain looks as follows. It generates 14 different letters based on the DGA's seed, and the value of `day`, `month` and `year`. Note that these names are misleading: although theses values are initialized with the current or past dates, the values are modified by each call to the routine.

This pseudo-code summarizes the algorithm:

```
FOR i = 0 TO 13
   day = (day >> 15) ^ 16 * (day & 0x1FFF ^ 4 * (seed ^ day))
   year = ((year & 0xFFFFFFF0) << 17) ^ ((year ^ (7 * year)) >> 11)
   month = 14 * (month & 0xFFFFFFFE) ^ ((month ^ (4 * month)) >> 8)
   seed = (seed >> 6) ^ ((day + 8 * seed) << 8) & 0x3FFFF00
   int x = ((day ^ month ^ year) % 25) + 'a'
   domain[i] = x;
```

The malware authors repeated their modulus error: like for the tld, the modulus needed to be increased by one. As it stands, ‘z’ is no reachable. Ranbyus shares this bug with the DGAs of Ramnig and Necurs.

Seed the end of this blog post for a C-implementation of the DGA.

Observed Seeds

The following tables lists some of the samples from malwr.com that are Ranbyus with the described DGA. All samples use the same parametrization, only the seed varies.

md5	seed
4b04f6baaf967e9c534e962c98496497	65BA0743
087b19ce441295207052a610d1435b03	65BA0743
28474761f28538a05453375635a53982	65BA0743
b309eab0277af32d7a344b8a8b91bd73	C5F128F3
4c7057ce783b2e4fb5d1662a5cb1312a	C5F128F3
b309eab0277af32d7a344b8a8b91bd73	C5F128F3
7cbc671bcb97122e0ec5b448f0251dc0	C5F128F3
437028f94ceea4cab0d302d9ac6973eb	C5F128F3
b309eab0277af32d7a344b8a8b91bd73	C5F128F3
6378b7af643e87c063f69ddfb498d852	B6354BC3
fa57f601402aab8168dea94c7c5f029f	B6354BC3
9f2c89ad17e9b6cf386028a8c9189264	0478620C

Summary

DGA Characteristics

The characteristics of Ranbyus' DGA are:

property	value
seed	magic number and current date
granularity	1 day, with a 31 day sliding window
domains per seed and day	40
domains per sliding window	1240
sequence	sequential

property	value
wait time between domains	500 ms
top level domains	.in, .me, .cc, .su, .tw, .net, .com, .pw
second level characters	lower case letters except ‘z’
second level domain length	14 letters

Decompiled Code

The following C code generates the domains for a given day and seed. In order to generate all domains that the malware can generate for any given seed and date, one would also need to run the code for all dates going 30 days in the past.

Edit 23.5.2015: The following code had contained a bug that led to a wrong sequence of top level domains, thanks to Anthony Kasza for sharing that with me.

```
 #include <stdio.h>
#include <stdlib.h>

char* dga(unsigned int day, unsigned int month, unsigned int year,
        unsigned int seed, unsigned int nr)
{
    char *tlds[] = {"in", "me", "cc", "su", "tw", "net", "com", "pw", "org"};
    char domain[15];
    int d;
    int tld_index = day;
    for(d = 0; d < nr; d++)
    {
        unsigned int i;
        for(i = 0; i < 14; i++)
        {
            day = (day >> 15) ^ 16 * (day & 0x1FFF ^ 4 * (seed ^ day));
            year = ((year & 0xFFFFFFF0) << 17) ^ ((year ^ (7 * year)) >> 11);
            month = 14 * (month & 0xFFFFFFFE) ^ ((month ^ (4 * month)) >> 8);
            seed = (seed >> 6) ^ ((day + 8 * seed) << 8) & 0x3FFFF00;
            int x = ((day ^ month ^ year) % 25) + 97;
            domain[i] = x;
        }
        printf("%s.%s\n", domain, tlds[tld_index++ % 8]);
    }
}

main (int argc, char *argv[])
{
```

```
    if(argc != 5) {
        printf("Usage: dga <day> <month> <year> <seed>\n");
        printf("Example: dga 14 5 2015 b6354bc3\n");
        exit(0);
    }

    dga(atoi(argv[1]), atoi(argv[2]), atoi(argv[3]),
            strtoul(argv[4], NULL, 16), 40);
}
```

Note: *I removed the Disqus integration in an effort to cut down on bloat. The following comments were retrieved with the export functionality of Disqus. If you have comments, please reach out to me by Twitter or email.*
