Ranbyus's DGA, Revisited - A second version of the Domain Generation Algorithm

Archived: 2026-04-06 00:51:53 UTC

> **Edit Dec. 8th, 2015**: I found two additional samples. One of them uses a different tld ordering and an additional operation on the hardcoded seed. I left the original text as is and put the changes in as edits.
> **Edit Jan. 25, 2016:** found another seed: 0x572473BB **Edit Mar. 2, 2016:** found another seed: 0x17794CF1
> **Edit Apr. 7, 2016:** found another seed: 0x7CB7966E

In May I wrote about the Domain Generation Algorithm (DGA) of the banking trojan *Ranbyus*. This week I stumbled on some new Ranbyus samples that use a significant modification of the DGA. For simplicity's sake I call the DGA from the previous post the *May DGA*, and the DGA in this post the *September DGA*. However, I can't tell if the chronology is correct; the DGA in this post might just as well be an earlier or concurrent version of the DGA reported in May.

The domains of the September version at first glance look like the ones from May. The second level domains consist of the letters a-y; the top level domains are the same and they appear in the same order, i.e., *.in* → *.me* → *.cc* → *.su* → *.tw* → *.net* → *.com* → *.pw*. (**Edit**: this newer sample uses the same TLDs in a different order: *.in* → *.net* → *.org* → *.com* → *.me* → *.su* → *.tw* → *.cc* → *.pw*)

For example, these are the first few domains from this report:

- rftkbenepisfitgdj.in
- xiqmvbjmhmhvmgcmi.me
- wxdunehygeonndttn.cc
- sbghxfgtslfpppqiu.su
- upixinckripequtam.tw
- oamxeavybfwlhqhob.net
- jkkugptcygpwxkjkw.com
- cvorpvaacmkfacelm.pw
- vptafodmeuaxopjbs.in
- eycagukbeduvmjnpx.me

The most striking difference to the May version is the increased length of the second level domains: the May version has 14 letters, while the September version uses 17 letters. As it turns out, the September DGA uses a vastly different algorithm to generate the second level domains.

The DGA

Seeding and Samples

Like the May DGA, the new DGA is seeded with the current date. It also produces 40 fresh domains (almost) every day. In addition to the new domains, the DGA will revisit the domains of up 30 days into the past.

Apart from the current date, the DGA is seeded with a hard-coded magic number, which allows for separate sets of domains. So far, the DGArchive collected seven different seeds for the DGA from May. For the new variant, I found seven seeds so far. **Edit Dec. 8th, 2015**: Some samples, e.g., b625b87a9dfdc345d226e913f9f95d77 and d8c247f95b2784419ffc14c8df8efc07, actually reverse the seed before applying it:

The following table lists the seed after negation so I could leave the reimplementation as is. The *negated* column shows the original seed *before* the NOT-operation.

MD5	**seed**	**negated**
eb35f453b87a2f430f53da4dafb2c968	0F0D5BFA	no
b82bfd9f649e08185a4100ab555ee9b9	F2C72B14	no
72a367560582ccd51be6f2284d92c946	0F0D5BFA	no
293cb29f3009503bebb3f9a4d4362537	F2C72B14	no
b7e7c7b77abbc89922806f4bf42fb30e	AE8714BE	no
b625b87a9dfdc345d226e913f9f95d77	CE7F8514	yes (~31807AEB)
ad9f06a74114dfee3e52d63b6b97ce54	F2C72B14	no
821c05d5c949a9b03ba21973ef9072a1	F2C72B14	no
d8c247f95b2784419ffc14c8df8efc07	572473BB	yes (~A8DB8C44)
1d4edada362f6a289b156d94bff26f41	17794CF1	yes (~E886B30E)
c6665471f52a0a7aba50edf8fc9cc886a	C0E32524	yes (~3F1CDADB)
d9393e7afcae648aa742ecaeefd36e07	7CB7966E	yes (~83486991)

The way the current date influences the domains is different. The May DGA uses the year, month and day directly as variables to generate the letters of the second level domain. The September version condenses the date and the hard-coded magic number into a single 32bit value:

$$X_0 = (\text{year} \cdot \text{month} \cdot \text{day}) \oplus \text{seed}$$

Consequently, all dates that have the same product of year, month and day will generate the same domains. For example, the domains from Januar 24 will be revisited six times the same year: February 12, March 8, April 6, June 4, August 3, and December 12. From a sinkholing perspective, it makes sense to pick a domain from this set.

Python Implementation

The DGA differs in the way the second level characters are picked. While the May version used a custom algorithm to determine the characters, the September edition relies on a pseudo random number generator (PRNG). The PRNG is of the LCG (linear congruential generator) family with common multiplier and increment:

You also find this code, along with a reimplementation of the other Ranbyus version, on my Github).

```
 """
    The DGA of Ranbyus as described here:
        https://bin.re/blog/ranbyuss-dga-revisited/

    Known Seeds are:
        - 0F0D5BFA
        - F2C72B14
        - AE8714BE
        - CE7F8514  (= ~ 31807AEB)
        - 572473BB (= ~ A8DB8C44)
        - 17794CF1 (= ~ E886B30E)
        - C0E32524 (= ~ 3F1CDADB)
"""

import argparse
from datetime import datetime

def to_little_array(val):
    a = 4*[0]
    for i in range(4):
        a[i] = (val & 0xFF)
        val >>= 8
    return a

def pcg_random(r):
    alpha = 0x5851F42D4C957F2D
    inc = 0x14057B7EF767814F

    step1 = alpha*r + inc
    step2 = alpha*step1 + inc
    step3 = alpha*step2 + inc

    tmp = (step3 >> 24) & 0xFFFFFF00 | (step3  & 0xFFFFFFFF) >> 24
    a = (tmp ^ step2) & 0x000FFFFF ^ step2
    b = (step2 >> 32)
    c = (step1 & 0xFFF00000)  | ((step3 >> 32) & 0xFFFFFFFF) >> 12
    d = (step1 >> 32) & 0xFFFFFFFF

    data = 32*[None]
```

```
    data[0:4] = to_little_array(a)
    data[4:8] = to_little_array(b)
    data[8:12] = to_little_array(c)
    data[12:16] = to_little_array(d)
    return step3 & 0xFFFFFFFFFFFFFFFF, data

def dga(year, month, day, seed):
    x = (day*month*year) ^ seed
    tld_index = day
    for _ in range(40):
        random = 32*[None]
        x, random[0:16] = pcg_random(x)
        x, random[16:32] = pcg_random(x)

        domain = ""
        for i in range(17):
            domain += chr(random[i] % 25 + ord('a'))
        if seed == 0xCE7F8514:
            tlds =  ["in", "net", "org", "com", "me", "su", "tw", "cc", "pw"]
        else:
            tlds =  ["in", "me", "cc", "su", "tw", "net", "com", "pw", "org"]
        domain += '.' + tlds[tld_index % (len(tlds) - 1)]
        tld_index += 1
        yield domain

if __name__=="__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("-d", "--date", help="date for which to generate domains")
    parser.add_argument("-s", "--seed", help="seed as hex string", default="0F0D5BFA")
    args = parser.parse_args()
    if args.date:
        d = datetime.strptime(args.date, "%Y-%m-%d")
    else:
        d = datetime.now()
    for domain in dga(d.year, d.month, d.day, int(args.seed, 16)):
        print(domain)
```

Please note that the above Python script only generates the 40 domains of the current day. Like the May version, Ranbyus can also revisit older domains up to 30 days into the past. So to get the full set of domains for any given day, you need to run the script for 31 different days.

Properties

Almost all characteristics of the Ranbyus September DGA are the same as for the May version. The only difference is the increased length of the second level domains:

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	
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	**17 letters** (May version: 14 letters)	

Appendix - Reversing the DGA

Similarities with May version

The new samples share most of the DGA code with the May version. The following graph views show the callback loop from May (left) and September (right):

The basic structure of the DGA itself is also equal:

Most other DGA-related functions stayed the same too, in particular:

- The routine to determine the top level domain *top_level_domain*, i.e., the domains will have the same top level domains in the same order as the DGA from May.
- The routines to determine and handle the current date.
- The data structures to configure the DGA.

Differences to May version

The main difference between the two DGAs is the routine to generate the second level domains:

The May DGA (on the left) uses a custom algorithm inside the loop body to produce a pseudo random number. The September version on the right first generates 32 bytes of random data using the *pcg_random* routine, and then simply accesses this data inside the loop body. Both version take the resulting pseudo random number modulo 25 to get letters from *a* to *y*.

The pseudo random number generator is based on 64bit numbers, which make the routine a little hard to read:

At the core of the above routine is the following linear congruential generator:

$$X_{n+1} = (6364136223846793005 \cdot X_n + 1442695040888963407) \bmod 2^{64}$$

The initial value X_0 is set to the product of *year*, *month*, and *day*, XORed with the hardcoded seed:
