A BazarLoader DGA that Breaks Down in the Summer

Archived: 2026-04-06 00:08:36 UTC

André Tavares sent me a Bazar Loader sample whose Domain Generation Algorithm (DGA) shows some interesting behavior. In May, it generates valid domain names with the eponymous top level domain *.bazar*:

But as soon as June comes around, some generated domains contain invalid characters:

And as it gets to July, all domain names are invalid (with very few exceptions):

The DGA also fails during August and September. But when October rolls around, all domains are valid again. This continues until next June, when the DGA has problems all over again.

This short blog post explores what causes the DGA to stop working properly in the summer, of all times.

The Sample Examined

I reverse engineered the DGA of the following sample:

MD5
: 5f11f2db1295fa419b190bd7478d9b23

SHA1
: 96d6c37fa0046a8dc1c520249dc94122e0fb3f52

SHA256
: 86d2aa04988befc74eccca5d99550f67093969b31aafa11cdce3476a4c59ba74

Size
: 248 KB (254474 Bytes)

Compile Timestamp
: 2021-07-13 08:22:30 UTC

Links

MalwareBazaar, Cape, VirusTotal

Filename

5f11f2db1295fa419b190bd7478d9b23.dll (MalwareBazaar), (VirusTotal)

Detections

MalwareBazaar: BazaLoader, **Virustotal**: 47/75 as of 2021-08-05 11:35:35 - Gen:Variant.Razy.892983 (MicroWorld-eScan), Trojan.Agent (CAT-QuickHeal), Backdoor.Win64.Bazdor.ah (Sangfor), Backdoor:Win64/Bazdor.ae3c68af (Alibaba), Trojan (0057f6941) (K7GW), Trojan (0057f6941) (K7AntiVirus), W64/Trojan.FRTN-3244 (Cyren), Win64/BazarLoader.AP (ESET-NOD32), generic.ml (Paloalto), Backdoor.Win64.Bazdor.ah (Kaspersky), Gen:Variant.Razy.892983 (BitDefender), Win64:DropperX-gen [Drp] (Avast), Gen:Variant.Razy.892983 (Ad-Aware), Gen:Variant.Razy.892983 (B) (Emsisoft), Trojan.Agent.Win64.8672 (Zillya), Artemis!Trojan (McAfee-GW-Edition), Trojan.Agent.dkxh (Jiangmin), TR/Redcap.ntozn (Avira), malware (ai score=88) (MAX), Win32.Troj.Undef.(kcloud) (Kingsoft), Trojan.Win64.Agent.oa (Gridinsoft), Trojan:Win64/Cobaltstrike.A!MSR (Microsoft), Backdoor.Win64.Bazdor.ah (ZoneAlarm), Gen:Variant.Razy.892983 (GData), Trojan.Win64.Convagent (VBA32), Gen:Variant.Razy.892983 (ALYac), Trojan.Bazar (Malwarebytes), Trojan.Agent!v7VRXZm6ckQ (Yandex), Trojan.Win64.Bazarloader (Ikarus), Win64:DropperX-gen [Drp] (AVG), Trj/CI.A (Panda)

I have unpacked it to the following state:

MD5

7c64ea7c4a229414b6048d18ab0836fd

SHA1

f10621be9bfee0152931f7790c2cbff022611f62

SHA256

d15dbfb7ef0511556a3527cc98d09145a56302bdd19a6083ee6d007af3352434

Size

113 KB (116224 Bytes)

Compile Timestamp

2021-07-12 13:27:57 UTC

Links

MalwareBazaar, Cape, VirusTotal

Detections

MalwareBazaar: BazaLoader, **Virustotal**: 40/75 as of 2021-08-05 19:07:37 - Trojan.Win32.Razy.4!c (Lionic), Gen:Variant.Razy.891147 (MicroWorld-eScan), Gen:Variant.Razy.891147 (FireEye), Backdoor.Bazdor.Win64.3 (Zillya), Backdoor:Win64/Bazdor.9312a6ac (Alibaba), Trojan (0057f6941) (K7GW), Trojan (0057f6941) (K7AntiVirus), W64/Trojan.QFLC-7900 (Cyren), Win64/BazarLoader.AP (ESET-NOD32), Backdoor.Win64.Bazdor.ax (Kaspersky), Gen:Variant.Razy.891147 (BitDefender), Gen:Variant.Razy.891147 (Ad-Aware), BehavesLike.Win64.Trojan.ch (McAfee-GW-Edition), Gen:Variant.Razy.891147 (B) (Emsisoft), Trojan.Win64.Bazarloader (Ikarus), TR/Redcap.rlvgc (Avira), malware (ai score=81) (MAX), Win32.Hack.Undef.(kcloud) (Kingsoft), Trojan.Win64.Agent.oa (Gridinsoft), Trojan:Win32/Tiggre!rfn (Microsoft), Gen:Variant.Razy.891147 (GData),

Backdoor.Win64.Bazdor (VBA32), Gen:Variant.Razy.891147 (ALYac), Trojan.Bazar (Malwarebytes), Win64.Backdoor.Bazdor.Ajls (Tencent), W64/BazarLoader.AP!tr (Fortinet), Trj/CI.A (Panda)

The Domain Generation Algorithm

The DGA can be easily be located in the unpacked sample based on the *.bazar* TLD, for example with this Yara rule:

```
 rule BazarDGA
{
    strings:
        $bazar_tld= { 2E [4-12] 62 [4-12] 61 [4-12] 7A [4-12] 61 [4-12] 72 }

    condition:
        $bazar_tld
}
```

The rule triggers at the following location, which adds the top level domain to the generated domain (pointed to by `rax`) at the end of the DGA function:

Here is how the DGA works:

1. BazarLoader divides the letters – except J, which was omitted for unknown reasons – into two character classes:
    - the 6 vowels *aeiouy*
    - the 19 consonants *bcdfghklmnpqrstvwxz*
2. The two sets are then combined into all 2·6·19 ordered pairs that contain one vowel and one consonant: `ab`, `ba`, `eb`, `be`, `ib`, `bi`, `ob`, `bo`, …, `oz`, `zo`, `uz`, `zu`, `yz`, `zy`.
3. These 228 pairs are then rearranged with a permutation that is hard-coded into the malware. The permutation is the seed of the BazarLoader DGA and offers the possibility to generate a different set of domains with the same algorithm. The permutation is stored as an array of 228 bytes that represent the one-

line notation of the permutation. So for example, a permutation of 27, 119, 38, … would place the first pair `ab` at position 27, the second pair `ba` at 119, and so on (0 being the first position).

4. Four pairs are then picked from the 228 permutated pairs, and strung together to form the 8 letter long second level domain. Which pairs are selected depends on the current date. The date is formatted as `%m%y`, where `%m` is the zero-padded month and `%y` is the two digit year. For example, December 5, 2035 would be `1235`. The four digits, e.g., 1, 2, 3 and 5, then define which pairs will be selected for the first, second, third and fourth pair respectively.

5. The **first pair** is selected by first splitting the pairs into groups of 19 pairs. The first digit derived from the current date then serves as the index of the groups to select. Since the first digit can only be 0 or 1, only two groups are possible [1]

   BazarLoader then picks a pair at random from the 19 pairs of the given group.

6. The **second pair** is selected like the first pair, except the groups are picked based on the second date digit. This digit can be any value from 0 to 9, so ten different groups are possible:

7. For the **third pair**, the groups only have a size of 4 pairs. Since the third date digit represents the decade, the same group will be selected for years to come.

8. The **fourth pair** is also picked from groups of 4 pairs, based on the least significant digit of the year.

9. The four picked pairs are concatenated into an 8-letter second level domain, and the top level domain `.bazar` is appended.

As can be seen from the illustrations above, pairs at higher positions are selected only as a second pair and only during the summer months. And that is exactly what causes the bug.

The Bug - A Faulty Permutation

The DGA is implemented exactly as described above. The hard-coded permutation, however, is incorrect:

```
 57 63 3A 29 25 0E 1E 5C  04 77 5F 37 02 03 28 51
61 28 39 64 12 1C 49 30  3D 74 06 07 49 0B 10 33
56 10 57 19 4A 3B 2C 2E  36 71 1B 68 24 15 67 5A
50 20 45 6E 4C 54 2F 2B  54 62 4A 0B 59 35 51 23
4D 08 01 45 1A 0A 7B 27  72 55 0C 08 5B 1F 60 32
3C 29 3B 2E 2A 70 3A 0F  17 48 14 2C 4B 25 4E 42
44 15 03 05 7C 26 16 06  24 5A 0D 32 46 39 35 5F
4F 6F 11 0C 34 5B 47 59  4E 42 5D 5E 1C 66 52 53
3F 30 38 21 44 18 00 58  56 1E 40 2A 4B 3E 55 13
3E 65 05 0F 1D 09 36 21  22 6D 2D 12 6A 40 17 19
3F 34 11 2F 5D 63 5E 6B  31 61 69 22 26 33 0D 7A
1D 4D 16 75 7D 0A 4F 02  07 64 79 58 14 1A 53 62
0E 41 18 01 31 2B 47 1F  76 5C 09 04 60 43 37 13
3D 3C 41 48 2D 43 52 38  73 27 23 46 4C 1B 50 6C
78 20 7E 00
```

For the permutation to be valid, i.e., bijective, it would need to contain all numbers from `0x00` to `0xe3` (227). But the largest number in the above list of numbers is only `0x7E` (126). Possibly the wrong data type was chosen when generating the permutation. For example, a signed char to store the numbers 1-228.

Instead of permuting the pairs, the DGA places them all in the first 127 places. Some pairs will therefore be overwritten by another pair placed in the same spot. For instance the first pair `ab` is placed at position `0x57` (first number of the “permutation”) . But since `0x57` appears a second time (35th number of the “permutation”), the pair `ab` will be overwritten.

Similarly, all spots above 127 are never filled. So with the actual “permutation” applied, the illustration for picking the second pair looks as follows, where `?` denotes undefined memory:

All pairs in July, August and September are undefined and will likely result in invalid domains. In June, only 13 out of 19 pairs are undefined, hence some domains come out correct. All other months are not affected by the bug.

Reimplementation in Python

The following Python script will generate all possible domains for a given date. When it is run for months affected by the bug, the resulting domains will contain two `??` that represent characters from undefined memory.

```
 from datetime import datetime
import argparse
from collections import namedtuple

Param = namedtuple('Param', 'block idx')
```

```
pool = (
    "yzewevmeywreomvi"
    "ekwyavygontowaer"
    "udsoyrexvuamtyse"
    "weesuvizpituiqow"
    "uzoretzemuultiaz"
    "icukoqiwolxuykos"
    "upwiymitisneroxe"
    "yxanlekyixxirasi"
    "asxoapuxqaohezwo"
    "oxdigyquziutpave"
    "zohexyvyguqyqidy"
    "ovynumunuwsusyen"
    "xaatyvusivaripfy"
    "oftesaysozuregin"
    "alifkazaadytwuub"
    "zuvoothymivazy"
)

pool +=(10*19*2 - len(pool))*"?"

def dga(date):
    seed = date.strftime("%m%Y")
    params = [
        Param(19, 0),
        Param(19, 1),
        Param(4, 4),
        Param(4, 5)
    ]

    ranges = []
    for p in params:
        s = int(seed[p.idx])
        lower = p.block*s
        upper = lower + p.block
        ranges.append(list(range(lower, upper)))

    domains = set()
    for indices in product(*ranges):
        domain = ""
        for index in indices:
            domain += pool[index*2:index*2 + 2]
        domain += ".bazar"
        domains.add(domain)

    return domains
```

```
if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "-d", "--date", help="date used for seeding, e.g., 2020-06-28",
        default=datetime.now().strftime('%Y-%m-%d'))
    args = parser.parse_args()
    d = datetime.strptime(args.date, "%Y-%m-%d")
    for domain in dga(d):
        print(domain)
```

Characteristics of the DGA

The following table summarizes the properties of the BazarLoader DGA when it is working as intended, i.e., October through May.

property	**value**
type	TDD (time-dependent-deterministic)
generation scheme	arithmetic
seed	current date
domain change frequency	every month
unique domains per month	5776
sequence	random selection, might pick domains multiple times
wait time between domains	none
top level domain	.bazar
second level characters	a-z, without j
regex	[a-ik-z]{8}.bazar
second level domain length	8

1. note that the letters used in the illustrations are randomly placed and not the actual letter pairs that BazarLoader uses. ↩