The Domain Generation Algorithm of BazarLoader - A DGA based on the Emercoin TLD .bazar

Archived: 2026-04-05 21:26:44 UTC

Edit 2020-07-19: Cybereason published an excellent article A Bazar of Tricks: Following Team9's Development Cycles. They only show the seeding part of the domain generation algorithm, however, the listing of generated bazar domains matches the algorithm in this blog post (apart from the first two domains `alztwfdicu.bazar` and `ocgjqlaspr.bazar` which are hardcoded). The article shows that the DGA is part of Bazar Loader, which will try to download Bazar Backdoor. I therefore renamed most instances of BazarBackdoor to BazarLoader.

Edit 2020-07-14: I have documented some additions to the DGA of BazarLoader:

- There exists a version of BazarLoader with a faulty DGA. I documented it in a separate blog post.
- The attackers registered four of the DGA domains, which @Securityinbits long before I published this post. I listed them here.
- The malware transforms the A RR of registered DGA domains, see the paragraph on sinkholing

BazarLoader (also known as Bazar Loader, Bazar Backdoor or Team9 Backdoor) is a module of the dreaded TrickBot Trojan. It is mostly used to gain a foothold in compromised enterprise networks [1] [2] [3] [4]. The malware is named after the C&C domains with top level domain *.bazar*. This TLD is provided by EmerDNS, a peer-to-peer decentralized domain name system in OpenNIC. This makes it very difficult, if not impossible, for law enforcement to take over these domains.

BazarLoader has been using a handful of hard-coded domains such as `bestgame.bazar`, `forgame.bazar` or `newgame.bazar` in the past, but today a sample was uploaded to Virustotal that tries a plethora of domains such as:

```
 ecfgjkehhgjm.bazar
afhhjkakjhjm.bazar
beggklbjigkn.bazar
cfhhjkckjhjm.bazar
bdehklbighkn.bazar
dcegjldhggjn.bazar
adggjkaiigjm.bazar
dcghkldhihkn.bazar
dehhikdjjhim.bazar
ceegkkcjggkm.bazar
eeegjkejggjm.bazar
cfehjkckghjm.bazar
cehhimcjjhio.bazar
ddfgjldihgjn.bazar
afhijlakjijn.bazar
```

```
ccfgimchhgio.bazar
eefhklejhhkn.bazar
acgiimahiiio.bazar
ecghjkehihjm.bazar
cehiklcjjikn.bazar
```

These look like algorithmically generated — and it turns out they are. This blog post shows how the underlying domain generation algorithm works.

Sample

I analysed the aforementioned sample from Virustotal:

MD5
: fdffbfa1380ab1a0ee2e26ff1be432b1

SHA1
: 5a004286c5b97afd97beec4b1332777c494d6ff1

SHA256
: e77e27630277a31276539c379671f54095d6b735f0568a3c457ac6a189c4c5b4

Size
: 288 KB (295424 Bytes)

Compile Timestamp
: 2020-06-12 09:35:14 UTC

Links
: MalwareBazaar, Cape, VirusTotal

Filenames
: BthCxn.exe, v86.exe_ (VirusTotal)

Detections
: **MalwareBazaar**: BazaLoader, **Virustotal**: 23/76 as of 2020-07-10 04:00:39 - Trojan:Win32/Trickbot.KB (Microsoft), Trojan.Trickbot!8.E313 (CLOUD) (Rising)

Many AV classify the sample as malicious, but only Microsoft and Rising also name the sample, both as *Trickbot* — which is at correct in the broader sense. The binary unpacks to this:

MD5
: 599b72d329b4b876390ae0567991da01

SHA1
: a8128b487bf6efd80b78c453e24a3447208008dd

SHA256
: 6b24ebfb84665cb844410ec9f948cfcf7f6d08f4ede16d52930c53236390848f

Size
: 141 KB (144896 Bytes)

Compile Timestamp
: 2020-06-12 09:34:11 UTC

Links

MalwareBazaar, VirusTotal

Filenames

none

Detections

Virustotal: 10/75 as of 2020-07-10 13:52:07

The detection for this sample is worse and none of the AV products assigns a non-generic name. The sample will finally inject the following executable, which only 3 out of 76 products even classify as malicious.

MD5

d1c4d25673be94db051dcd5271c64ae1

SHA1

cc4d30072bbd16fbdc387eb546aeb4dc38a5ea4a

SHA256

b4b7f0fd63cda1269ee937fa398fb80b6655d205066e6593fefceda7e3b09f6b

Size

132 KB (135168 Bytes)

Compile Timestamp

2020-06-11 11:16:31 UTC

Links

MalwareBazaar,VirusTotal

Filenames

none

Detections

Virustotal: 3/76 as of 2020-07-10 13:52:05

DGA Disassembly

The domain generation algorithm of BazarLoader is in a single function, including seeding (click to enlarge):

The algorithm roughly consists of these three steps:

1. Determine the first six letters of the second level domain at random.
2. Generate a seed based on the current date
3. Calculate the last six letters of the second level domain based on the first six and the seed.

Step 1: First Six Letters

The (simplified) decompilation of the first step is as follows:

```
 i = 0;
do
{
    r = (GetTickCount() % 25) / (i + 6);
    i_1 = i++;
    *domain++ = 2 * i_1 + r + 'a';
}
while ( i < 6 );
```

The function `GetTickCount` usually doesn't bode well for DGAs: since this functions is largely unpredictable, it almost always means that the generated domains will be unpredictable as well. However, this algorithm is different. The tick count is mapped to a value between 0 and 24, which is then divided by 6+i. The division shrinks down the range of numbers, and ultimately the range of potential letters. The character set is also offset by double the loop index, leading to these choices of letters:

index	random number range	potential letters
0	0–4	abcde
1	0–3	cdef
2	0–3	efgh

index	random number range	potential letters
3	0–2	ghi
4	0–2	ijk
5	0–2	klm

So even though the exact letters can't be predicted, there are only 2160 combinations. Since the malware continuously tries to contact newly generated domains, registering a few of the 2160 domains is probably enough to get lucky with `GetTickCount` within a couple of minutes of malware runtime.

Step 2: Seeding

Seeding is based on on the current date, which is determined by `GetDateFormatA`.

```
 if ( !seeding_done )
{
  GetDateFormatA(LOCALE_INVARIANT, 0, 0i64, 0i64, lpCurrentDate, 24);
  szMonth = lpCurrentDate[0];
  szYear = *&lpCurrentDate[3];
  v15 = 0;
  v17 = 0;
  str_to_int = resolve_api(0i64, 19i64, 2865918183i64, 534i64);
  if ( str_to_int )
      nYear = str_to_int(&szYear);
  else
      nYear = 0;
  str_to_int_0 = resolve_api(0i64, 19i64, 2865918183i64, 534i64);
  if ( str_to_int_0 )
      nMonth = str_to_int_0(&szMonth);
  else
      nMonth = 0;
  LODWORD(nYearMinus18) = nYear - 18;
  wnsprintfA(szSeedStr, 7, "%.2d%d", (12 - nMonth), nYearMinus18);
  seeding_done = 1;
}
```

The function `GetDateFormatA` with `LOCALE_INVARIANT` locale returns the current date formatted as `<month>/<day>/<year>`, so for example `07/10/2020` for July 10, 2020. The month and year are taken from this string and converted into integers. The month is then turned into a two-digit number by calculating `a = 12 - month` and padding it with zero if necessary. The year is transformed into a four-digit number according to `b = year - 18`. The two values are then concatenated into a string. For example, July 2020 turns into `052002`.

Step 3: Last Six Letters

The last six characters of the second level domain are based on the seed and the first six letters:

```
 j = 6;
[...]

szDomainPlusSix = szDomain + 6;
do
{
    ascii_code = *(szDomainPlusSix - 6) + szDomainPlusSix[szSeedStr - szDomain - 6] - '0';
    if ( ascii_code < 'a' )
       ascii_code = 'z';
    *szDomainPlusSix++ = ascii_code;
    --j;
}
while ( j );
szDomain[12] = 0;
```

This simply treats each character of the seed string as an integer, and uses that to offset the characters of the first six letters to form the last six letters. For instance, let's look at the seed string `052002` applied to the randomly picked first six letters `cfhiilc`:

1. Add 0 to `c` to get `c`.
2. Add 5 to `f` to get `k`.
3. Add 2 to `h` to get `j`.
4. Add 0 to `i` to get `i`.
5. Add 0 to `i` to get `i`.
6. Add 2 to `l` to get `n`.

The malware's check if the letter falls before `a` is actualy not necessary, as the offset is always positive between 0 and 9. A check to see if the letter falls beyond `z` would be more reasonable, but is also unnecessary: the "largest" letter in the first half is "m", which offset by 9 leads to "v". The following image illustrates the procedure:

Python Reimplementation

When implemented in Python, the DGA looks something like this:

```
 import argparse
from datetime import datetime
from itertools import product


def dga(date):
    month = date.month
    year = date.year
    date_str = "{0:02d}{1:04d}".format(12-month, year-18)

    valid_chars = [
      "abcde",
      "cdef",
      "efgh",
      "ghi",
      "ijk",
      "klm"
    ]
    valid_chars = [list(_) for _ in valid_chars]
    for part1 in product(*valid_chars):
        domain = "".join(part1)
        for i, c in enumerate(part1):
            domain += chr(ord(c) + int(date_str[i]) )
        domain += ".bazar"
        yield domain



if __name__=="__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("-d", "--date", help="date when domains are generated")
    args = parser.parse_args()
    if args.date:
        d = datetime.strptime(args.date, "%Y-%m-%d")
    else:
        d = datetime.now()
    for domain in dga(d):
        print(domain)
```

Regex

The possible letters per position is very limited. Assuming that the year is between 2020 and 2029, the domains match the following regular expression (**Edit 2020-11-10**: Fixed the regular expression, thanks to Luca Corbatto for providing the correct regex):

```
[a-e][c-f][e-h][g-i][i-k][k-m][a-f][c-o][g-j][g-i][i-l][k-v]\.bazar
```

Characteristics

The following table summarizes the properties of BazarLoader's DGA.

property	**value**
type	TDD (time-dependent-deterministic), to some extend TDN (time-dependent non-deterministic)
generation scheme	arithmetic
seed	current date
domain change frequency	every month
domains per day	2160
sequence	random selection, might pick domains multiple times
wait time between domains	None
top level domain	`.bazar`
second level characters	a-v
regex	`[a-e][c-f][e-h][g-i][i-k][k-m][a-f][c-o][g-j][i-k][k-m][k-v]\.bazar`
second level domain length	12

Domain to Seed

Since the function to determine the second half of the domain is reversible, the month and year can be calculated from the domains. I wrote a small Javascript form that does just that. For those of you who block Javascript, here's a screenshot. The same code in Python can also be found on my GitHub page

Registered Domains

As far as I can tell, the attackers registered five domains using the Emercoin address ETQERUknhW2A5cBmfHN4VBqL7VGiFnKQRh. Also see tweets by Brad @malware_traffic and Security-in-bits @Securityinbits:

date	domain	valid for
2020-05-18 10:24:32 UTC	cdghilckihin.bazar	May 2020
2020-05-18 10:24:32 UTC	cefgilclhgin.bazar	May 2020
2020-07-03 11:08:26 UTC	defikldjhikn.bazar	July 2020
2020-07-03 11:14:24 UTC	aeehjkajghjm.bazar	July 2020
2020-07-14 14:13:16 UTC	cdfhimcihhio.bazar	July 2020

Sinkholing

The IP resource record of the DGA domains are XOR decrypted with key `0xFE` to get the real IP of the C2 servers. You can use this Javascript form to calculate the transformation.

---

1. In-depth analysis of the new Team9 malware family ↩
2. BazarBackdoor: TrickBot gang's new stealthy network-hacking malware ↩
3. TrickBot BazarLoader In-Depth ↩
4. Group Behind TrickBot Spreads Fileless BazarBackdoor ↩

Source: https://johannesbader.ch/blog/the-dga-of-bazarbackdoor/