The DGA of PadCrypt - Versions 2.2.86.0 and 2.2.97.1

Archived: 2026-04-10 02:54:20 UTC

EDIT 2016-03-06: I completely missed that Lawrence Abrams of BleepinComputer.com not only reversed the DGA of PadCrypt long before me, but also tweeted an updated version of the DGA. Many thanks to MalwareHunterTeam for letting me know, and apologies to Lawrence Abrams for my redundant post. I updated the post and my reimplementation to reflect the second DGA too.

EDIT 2: Someone asked me on Twitter how to reverse the DGA. See the Appendix for a short tutorial how to very easily decompile PadCrypt.

PadCrypt is a Ransomware first mentioned in a tweet of @abuse_ch. BleepinComputer.com wrote two articles articles on PadCrypt. The *MalwareHunterTeam* mentioned the Domain Generation Algorithm (DGA) PadCrypt version 2.2.86.1 in a tweet:

BleepinComputer.com tweeted version 2.2.97.0:

Here are the 24 distinct domains generated on March 6, 2016 for version 2.2.86.1:

And these are the 72 distinct domains generated on March 6, 2016 for version 2.2.97.0:

To get a reimplementation of the DGA, I looked at this sample of PadCrypt 2.2.86.1 from Hybrid-Analysis:

MD5
: f58072b9413886e7b79e826ad94bd89e

SHA-256
: 1ad70a64863c2c59390e4e956d97065524acf5e2ae53f786770d2cf6bd602141

upload date
: 2015-03-01 10:55:57 (CST)

filename
: package.pdcr.bin

PadCrypt version
: 2.2.86.1

filesize
: 1.3 MB

link
: link

The following sample of PadCrypt 2.2.97.0 was provided by Lawrence Abrams here, you can download it for example from Virusshare.com. The sample features a modified version of the DGA with subtle changes (see tweet above).

MD5
: 7c0f7a734014022f249338a23881dc24

SHA-256
: f0f8ef9a0cd40363aa4b06ba4c0fa286f23010830caa931addefacb087d7c678

upload date
: 2016-03-06 19:52:54

filename
: PadCrypt.exe

PadCrypt version
: 2.2.97.0

filesize
: 1.4 MB

link
: link

DGA Design

The DGA of PadCrypt 2.2.86.1 generates 24 domains per day, the DGA of version 2.2.97.0 three times that (72). The DGAs use SHA256 hashing as generation scheme. Other malware families that are based on hashing include Bamital, Murofet, Gameover, Pushdo (all MD5) and Dyre (SHA-256).

PadCrypt does not use magic numbers for seeding, only the current date along with the domain number (0 to 23) are hashed. The only difference between 2.2.86.1 and 2.2.97.0 is the separator between date and domain number.

For the older version they are separated by `:` (e.g., `6-3-2016:17`); the newer version uses `|` (e.g., `6-3-2017|17`)

The last nibble of the hash value determines the top level domain. The 4 bit value is used as an index into a hard-coded list of top level domains. The list only has 11 domains and for greater indices the first top level domain is used instead. This makes ".com" — the first domain — much more common than the rest. The nibbles 3 to 18 determine the 16 second level characters. The second level characters are picked by mapping 0000 to 1001 to a hard-coded list of characters:

The remaining values 1010 to 1111 are mapped to the corresponding hex representation, i.e., *"a"* to *"f"*. Because the letters *a*, *b*, *c*, *d* and *f* are also in the hard-coded mapping they occur twice as common as the remaining letters "enolmk".

Reimplementation

I reimplemented the DGA in Python. Use the `-d` or `--date` argument to set the date. If no date is provided then the domains for the current day are generated. You can select the DGA version with `-v` or `--version`. You also find the reimplementation with potential future improvements on my Github page.

```
 """
    The DGA of PadCrypt

    See
    - https://twitter.com/BleepinComputer/status/705813885673201665
    - http://www.bleepingcomputer.com/news/security/padcrypt-the-first-ransomware-with-live-support-chat-and-an-
    - http://www.bleepingcomputer.com/news/security/the-padcrypt-ransomware-is-still-alive-and-kicking/
    - https://bin.re/blog/the-dga-of-padcrypt/

"""

import argparse
import hashlib
from datetime import datetime

configs = {
    "2.2.86.1" : {
        'nr_domains': 24,
        'tlds': ['com', 'co.uk', 'de', 'org', 'net', 'eu', 'info', 'online',
            'co', 'cc', 'website'],
        'digit_mapping': "abcdnfolmk",
```

```
        'separator': ':',
        },
    "2.2.97.0" : {
        'nr_domains': 24*3,
        'tlds': ['com', 'co.uk', 'de', 'org', 'net', 'eu', 'info', 'online',
            'co', 'cc', 'website'],
        'digit_mapping': "abcdnfolmk",
        'separator': '|'
        }
}

def dga(date, config_nr):
    config = configs[config_nr]
    dm = config['digit_mapping']
    tlds = config['tlds']
    for i in range(config['nr_domains']):
        seed_str = "{}-{}-{}{}{}".format(date.day, date.month, date.year,
                config['separator'], i)
        h = hashlib.sha256(seed_str.encode('ascii')).hexdigest()
        domain = ""
        for hh in h[3:16+3]:
            domain += dm[int(hh)] if '0' <= hh <= '9' else hh
        tld_index = int(h[-1], 16)
        tld_index = 0 if tld_index >= len(tlds) else tld_index
        domain += "." + config['tlds'][tld_index]
        yield domain

if __name__=="__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("-d", "--date", help="date for which to generate domains")
    parser.add_argument("-v", "--version", help="dga version",
            choices=["2.2.86.1", "2.2.96.1"], default="2.2.86.1")
    args = parser.parse_args()
    if args.date:
        d = datetime.strptime(args.date, "%Y-%m-%d")
    else:
        d = datetime.now()
    for domain in dga(d, args.version):
        print(domain)
```

Characteristics

Those are the characteristics of the PadCrypt DGA:

property	value
type	TDD (time-dependent deterministic)
generation scheme	hashing
seed	current date
domain change frequency	1 day
domains per day	24 (version 2.2.86.1), 72 (version 2.2.97.0)
sequence	sequential
wait time between domains	none
top level domains	`com` , `co.uk` , `de` , `org` , `net` , `eu` , `info` , `online` , `co` , `cc` , `website`
second level characters	letters: “abcdefnolmk”
second level domain length	16

Only 11 characters make up the character set of the second level domains. Together with the fixed length of 16 characters, this makes the domain easy to attribute to PadCrypt. The distribution is:

```
 a: 12.5 %
b: 12.5 %
c: 12.5 %
d: 12.5 %
e:  6.25 %
f: 12.5 %
k:  6.25 %
l:  6.25 %
m:  6.25 %
n:  6.25 %
o:  6.25 %
```

Among the top level domains, `.com` is picked 37.5% of the time. The remaining 10 tlds are picked with probability 6.25%.

Appendix - How to Reverse

Since I got asked by private message on Twitter, here’s how to easily reverse PadCrypt.

PadCrypt is written for .NET version 4.5. This allows us to open the binary in a .NET decompiler. I'm using the excellent and free dnSpy. Naturally, the authors of PadCrypt are aware that decompiling intermediate languages is easy, so they use a protector. Both PadCrypt 2.2.86.1, 2.2.97.0 and later version 2.2.120.0 use DeepSea 4.1. Opening the obfuscated binary in dnSpy gives you very messy code. The following screenshot shows the obfuscated DGA:

The decompiled routine takes up 268 lines and is impossible to read. Fortunately, there are deobfuscators for many known obfuscators. I'm relying on the free and open soure de4dot. It comes with 21 deobfuscators and often works out of the box. I ran the following command on PadCrypt:

```
de4dot.exe padcrypt_2.2.97.0.exe -o padcrypt_2.2.97.0_deobf.exe
```

Again looking at the DGA routine reveals a much shorter, very easy to ready C# code:

Or, if you prefer, Visual Basic:
