The DGA of QSnatch

Archived: 2026-04-05 19:10:24 UTC

QSnatch is a malware that infects QNAP NAS devices. It collects and exfiltrates user credentials from vulnerable devices, and can also load malicious code from its command and control (C2) servers. These C2 servers are resolved by algorithmically generated domains.

The National Cyber Security Centre of Finland (NCSC-FI) published an article about QSnatch in late October 2019 and made the threat known, but samples on Virustotal date back to at least June 2019.

I found seven different QSnatch samples with two different DGAs. Both versions are very similar, with one being a simpler version of the other. I called the more complicated version *Version A* and the simpler one *Version B*. QSnatch is implemented as shell scripts so it is trivial to reimplement the DGA in Python.

Version A

These are two samples with the more complicted version of the DGA from Virustotal. This version of QSnatch comes with a timestamp as version information:

MD5
: 372140d7c2c68dc2c8dc137d1a471e9f

SHA1
: 986f38a04937ede2000e8f25e59ea438ee265e24

SHA256
: 3c38e7bb004b000bd90ad94446437096f46140292a138bfc9f7e44dc136bac8d

Version Timestamp
: 2019-03-20 5:00 UTC

Size
: 41KB, 41655 Bytes

Uploaded to Virustotal
: 2019-11-04 13:39:51 UTC

and this sample (which is also the one currently delivered as of 2019-11-14 11:00 UTC):

MD5
: 60567a1d2b2e02e93ffc162e6a70d60c

SHA1
: 1f1bf0bd2df89029d5267130f014ab5aa133c3ae

SHA256
: 9526ccdeb9bf7cfd9b34d290bdb49ab6a6acefc17bff0e85d9ebb46cca8b9dc2

Version Timestamp
: 2019-05-17 5:00 UTC

Size
: 41KB, 41104 Bytes

Uploaded to Virustotal
: 2019-06-09 22:56:07 UTC

Both samples have a timestamp set to exactly 05:00 UTC, which could mean that the timestamp is in fact a simple date without time information, generated in a timezone UTC+5.

The DGA generates domains like the following:

```
 t2q2r.cf
gc9nz.tk
07tvvc.com
7ubqo.ml
53bcm.de
6zltf.rocks
hv7uv.mx
nypno.biz
qkzccy.net
rassb.cn
```

Bash

The following screenshots shows the shell script lines responsible for generating the domains:

In the following I will briefly discuss the main components of the DGA.

TLDs

The complicated version of the DGA of QSnatch uses a whopping 145 top level domain, including many gTLDS like `rocks`, `mobi`, `today`; and a few sld/tld-tuples where domain registrations are only possible on the third level, like `.com.ua`, `.com.bn`, `.com.by`. The domains are listed in a space separated string as tld/number-pairs divided by colons:

```
domainexts='cf:0 tk:0 com:1 ml:0 de:0 rocks:0 mx:0 biz:0 ...``
```

The number next to the domain specifies the number of extra characters in the hostname. In the analysed sample, `.com`, `.net` and `.org` have 1 extra character specified, meaning their hostname has one extra character compared the remaining tlds, which all have 0 extra characters set.

The script has a bug for the domain `.com.bn` which is listed with a leading dot, resulting in invalid domains with doubled points, e.g., `r4rb..com.bn`.

Domain Timespans

The generated domains have different validities of 15 days (1296000 seconds) down to 1 hour (3600 seconds). The DGA generates domains for all specified intervals starting with the largest timespan.

```
for interval in '1296000' '432000' '86400' '28800' '7200' '3600'; do
```

Hostname Lengths

The hostnames have varying lengths, which are calculated by adding a global length value and the number next to the list of top level domains. The global lengths are 5, 3, then 4. As a result, `.cf` hostnames have a length of 3 to 5, while `.com` hostnames have a length of 4 to 6.

```
for length in 5 3 4; do
```

Iterating over all TLDs

The third loop iterates over all top level domains.

```
n=0; while [ "$n" -lt $domainextcnt ]; do
```

Time permitting, the DGA generates 6 (number of timespans) * 3 (number of hostname lengths) * 145 (number of tlds) = 2610 domains.

Aborting Domain Generation

Before looping over the top level domains, the DGA checks if more than 10 minutes (600 seconds) passed since starting a timespan block. If ten minutes elapsed, then generating domains for that particular timespan is aborted, unless it is the last timespan:

```
test "$(( $timenow - $timestart ))" -gt 600 && test "$interval" != "3600" && break
```

Hostname String Generation

The following command generates the string which is later trimmed into a hostname:

```
hostname=$(echo \
 "$(( $(date +%s) / $interval ))IbjGOEgnuD${ext}" | \
 openssl dgst -sha1 -binary | \
 openssl base64 | \
 sed 'y/ABCDEFGHIJKLMNOPQRSTUVWXYZ-+\//abcdefghijklmnopqrstuvwxyzabc/;s/=//g')
```

1. The command `($(date +%s) / $interval )` divides the current unix timestamp by the timespan length.
2. The seed `IbjGOEgnuD` and the top level domain `${ext}` are appended to the string from Step 1.
3. `openssl dgst -sha1 -binary` generates the SHA1-hash of the string (including a newline character `\n` from the echo command).
4. `openssl base64` converts the hash into base64.
5. `sed 'y/.../;s/=//g')` converts the base64 string into lowercase, replaces `-`, `+` and `\\` with `a`, `b` and `c` respectively, and removes `=`-padding.

Trimming to desired length

Next, the hostname is trimmed to the desired length. The length can not be smaller than 3, which is never the case for the configured lengths:

```
 trycnt=0
while [ ${#host} -gt "$l" ] && [ $trycnt -lt 3 ]; do
  trycnt=$(( $trycnt + 1 ))
  host=${host%?}
done
```

Concatenating the Hostname and SLD

Finally, the hostname and tld are joined to produce the DGA domain:

```
curl --connect-timeout "$curlconntimeout" -m 30 -k -o "$outfile" "https://${host}.${ext}/qnap_firmware.xml?t=$(date +%s)"
```

Python

This is a reimplementation of Version A of the DGA in Python 3:

```
 import time
import hashlib
import base64
import argparse
from datetime import datetime


TLDS = {
    "cf": 0, "tk": 0, "com": 1, "ml": 0, "de": 0, "rocks": 0, "mx": 0,
```

```
    "biz": 0, "net": 1, "cn": 0, "ga": 0, "gq": 0, "org": 1, "top": 0, "nl": 0,
    "men": 0, "ws": 0, "se": 0, "info": 0, "xyz": 0, "today": 0, "ru": 0,
    "ec": 0, "co": 0, "ee": 0, "rs": 0, "com.sv": 0, "com.cy": 0, "co.zw": 0,
    "kg": 0, "com.ge": 0, "tl": 0, "name": 0, "tw": 0, "lv": 0, "bs": 0,
    "li": 0, "ng": 0, "ae": 0, "bt": 0, "tv": 0, "pe": 0, "uz": 0, "me": 0,
    "gy": 0, "am": 0, "kr": 0, "by": 0, "fr": 0, "com.uy": 0, "com.lb": 0,
    "com.br": 0, "vu": 0, "hk": 0, "in": 0, "re": 0, "ch": 0, "af": 0,
    "com.ps": 0, "ug": 0, "dz": 0, "pro": 0, "co.th": 0, "sg": 0, "cd": 0,
    "so": 0, "mo": 0, "co.id": 0, "co.il": 0, "com.do": 0, "ke": 0, "cx": 0,
    "ro": 0, "id": 0, "pm": 0, "hm": 0, "vg": 0, "az": 0, "com.eg": 0, "bz": 0,
    "su": 0, "com.ar": 0, "gg": 0, "com.lr": 0, "pa": 0, "com.ve": 0, "al": 0,
    "fm": 0, "to": 0, "mu": 0, "co.ck": 0, "pk": 0, "co.rs": 0, "cw": 0,
    "nr": 0, "gd": 0, "gl": 0, "ac": 0, "lk": 0, "md": 0, "fi": 0, "sx": 0,
    "lc": 0, "es": 0, "cc": 0, "cm": 0, "la": 0, "co.za": 0, "je": 0, "cz": 0,
    "jp": 0, "ai": 0, "pw": 0, "bg": 0, "nu": 0, "ag": 0, "bm": 0, "eu": 0,
    "com.my": 0, "sc": 0, "ax": 0, "wf": 0, "ly": 0, "qa": 0, "vn": 0, "aq": 0,
    "mobi": 0, "com.tr": 0, "com.ua": 0, "com.py": 0, "hk.org": 0,
    "south.am": 0, "com.kh": 0, "co.zm": 0, "ru.net": 0, "com.km": 0, "tt": 0,
    "kn": 0, "co.ls": 0, "co.fk": 0, "uy.com": 0, "com.gu": 0, ".com.bn": 0,
    "com.pf": 0, "com.fj": 0
}
SEED = "IbjGOEgnuD"


def dga(date):
    def unix(date):
        unix = int(time.mktime(date.timetuple()))
        return unix

    HOUR = 3600
    DAY = 24*HOUR
    for interval in [15*DAY, 5*DAY, 1*DAY, 8*HOUR, 2*HOUR, 1*HOUR]:
        for length in [5, 3, 4]:
            for tld, l in TLDS.items():
                min_length = l + length
                key = f"{unix(date)//interval}{SEED}{tld}\n".encode('ascii')
                key_hash = hashlib.sha1(key).digest()
                key_hash_b64 = base64.b64encode(key_hash).decode('ascii')
                key_hash_b64_noeq_lc = key_hash_b64.rstrip("=").lower()
                trantab = str.maketrans("-+/", "abc")
                hostname_src = key_hash_b64_noeq_lc.translate(trantab)
                hostname_len = max(min_length, 3)
                hostname = hostname_src[:hostname_len]
                domain = f"{hostname}.{tld}"
                yield domain


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="QSnatch dga")
    parser.add_argument(
        "-d", "--datetime",
        help="date time for which to generate domains, e.g., "
             "2019-11-11 18:00:00")
    args = parser.parse_args()
    if args.datetime:
        d = datetime.strptime(args.datetime, "%Y-%m-%d %H:%M:%S")
    else:
        d = datetime.now()
    for domain in dga(d):
        print(domain)
```

Version B

Version B of the DGA is simpler than the previously described version. I found these five samples on Virustotal that use the simpler version of the DGA:

MD5	SHA1	SHA256
8cee2a187198648c199c1d135c918a3a	a9f39f3b832344a79d32d92ac56c50cdaff0b93c	09ab3031796bea1b8b79fcfd2b86dac8f38b1f95f0fce6bd2
c49ac8cfe022ff6acb8eb0036e2fc1a1	e30ce38ff0ce46d8256d06fb3d5e13bf3abb1012	5cb5dce0a1e03fc4d3ffc831e4a356bce80e928423b374fc8
4affa116b27f2d977a756e353f77b8f5	e8bb081056542504b5a69bd5f202cf77fac0a64f	8fd16e639f99cdaa7a2b730fc9af34a203c41fb353eaa250a5
421f006756f72cabc1ffb796c6cdb5c0	5ca92d6f02019519de593758583d7ca5a4bf9f23	5130282cdb4e371b5b9257e6c992fb7c11243b2511a6d418
421240952a097e904df778590caa9668	58523de660632c6b84ffbd243cc75f4fb576980a	15892206207fdef1a60af17684ea18bcaa5434a1c7bdca55f

Examples of domains of this DGA are:

```
 t2q2rs.cf
t2q2rsa.cf
t2q2rsaz.cf
t2q2rsazo.cf
t2q2rsazo1.cf
gc9nzf.tk
gc9nzfb.tk
gc9nzfbt.tk
gc9nzfbt3.tk
gc9nzfbt3i.tk
```

Bash

These are the lines of QSnatch that generate the domains:

The DGA uses fewer tlds, without tld-specific hostname length specifiers. The algorithm only uses one timespan (15 days). It uses more hostname lengths (6, 7, 8, 9 and 10) which are generated one after another for a specific hostname pattern.

The generation of the string for the hostname is the same as for the more complicated DGA version, including the seed `IbjGOEgnuD`. Because Version A mostly uses hostname lengths of 5 and shorter, while Version B uses lengths of 6 up to 10, only some domains from Version A with an extra character overlap with Version B (`.com`, `.net`, `.org`). For example, `07tvvc.com` is a valid domain for both DGAs.

Python

```
 import time
import hashlib
import base64
import argparse
from datetime import datetime


TLDS =[
    'cf', 'tk', 'ml', 'ga', 'gq', 'com', 'biz', 'org', 'de', 'rocks',
    'mx', 'cn', 'top', 'nl', 'men', 'ws', 'se', 'info', 'xyz', 'net', 'today',
    'ru', 'fi', 'name', 'to', 'in', 'com.ua', 'vg', 'vn', 'cd'
]
SEED = "IbjGOEgnuD"


def dga(date):
    def unix(date):
        unix = int(time.mktime(date.timetuple()))
        return unix

    HOUR = 3600
    DAY = 24*HOUR
    INTERVAL = 15*DAY
    for tld in TLDS:
        key = f"{unix(date)//INTERVAL}{SEED}{tld}\n".encode('ascii')
        key_hash = hashlib.sha1(key).digest()
        key_hash_b64 = base64.b64encode(key_hash).decode('ascii')
        key_hash_b64_noeq_lc = key_hash_b64.rstrip("=").lower()
        trantab = str.maketrans("-+/", "abc")
        hostname_src = key_hash_b64_noeq_lc.translate(trantab)
        for hostname_len in range(6, 11):
            hostname = hostname_src[:hostname_len]
            domain = f"{hostname}.{tld}"
            yield domain


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="QSnatch DGA Version 1")
    parser.add_argument(
        "-d", "--datetime",
        help="date time for which to generate domains, e.g., "
             "2019-11-11 18:00:00")
    args = parser.parse_args()
    if args.datetime:
        d = datetime.strptime(args.datetime, "%Y-%m-%d %H:%M:%S")
    else:
        d = datetime.now()
    for domain in dga(d):
        print(domain)
```

You also find both versions of the QSnatch DGA in [my collection of domain generation algorithms on GitHub](#).
