The DGA of BumbleBee

bin.re/blog/the-dga-of-bumblebee/

The following Tweet by @Artilllerie caught my attention because it mentions a “Possible DGA on .life domains”:

Artilllerie ☣
@Artilllerie

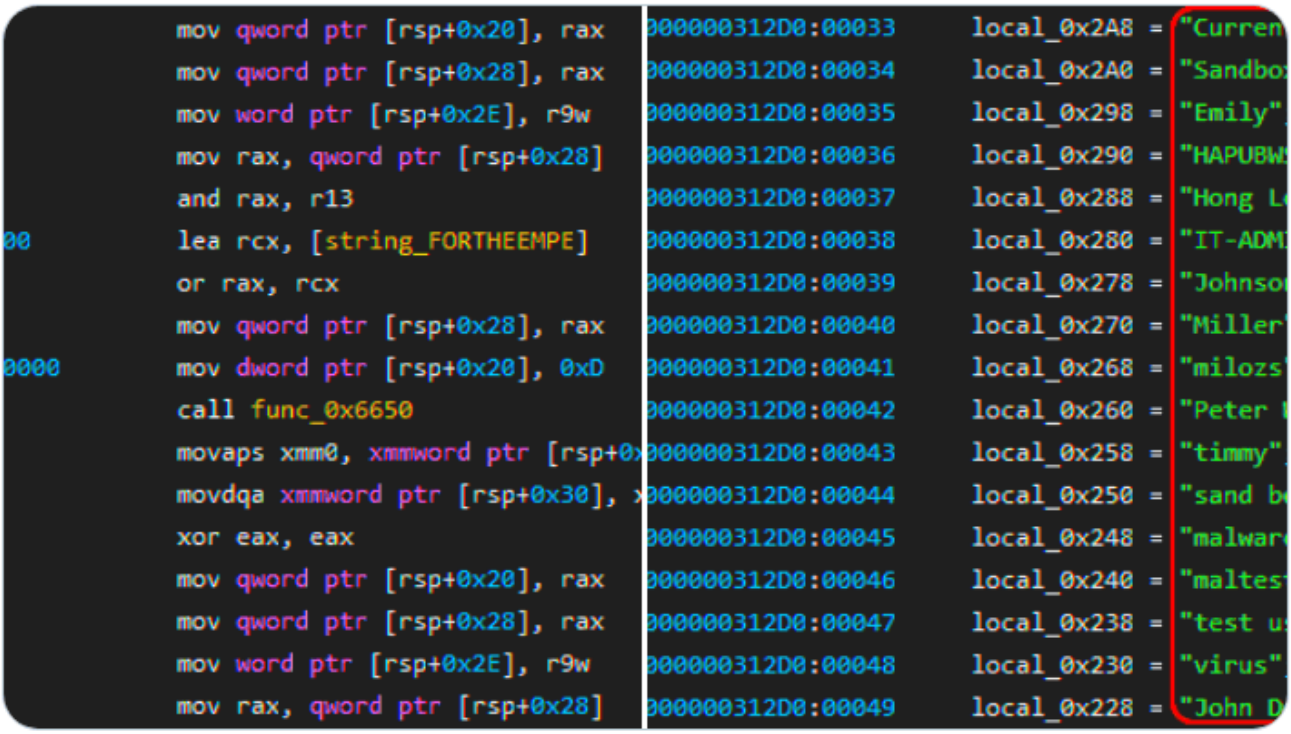
#BumbleBee

🧬Well detected (49/71) because 2023-09-06
virustotal.com/gui/file/fe3c9...

▶️group_name: lnk1
▶️4 well-known .life C&C domains

🧩Mysterious "FORTHEEMPEROR" string (ping) 🤔
🧩Many anti-sandbox with 17 usernames checks 🧐
🧩Possible DGA on .life domains

@malwrhunterteam



JAMESWT

5:04 pm · 11 Sep 2023 · **6,361** Views

This is the mentioned file

File type
fe3c93db5bfab8423d142e07b5adc73620d8a492f2ac67f4ade1e40bf3abd7cc: PE32+ executable (GUI) x86-64, for MS Windows

MD5
cf19e55c9604d5c002ac7b9770c529de

SHA1
34a3c780ba2decb6c676723fbcb916c007bacb8c

SHA256
fe3c93db5bfab8423d142e07b5adc73620d8a492f2ac67f4ade1e40bf3abd7cc

Size
595 KB (609280 Bytes)

Compile Timestamp
2023-09-06 10:43:36 UTC

Links
MalwareBazaar, Cape, VirusTotal

Filenames
7.exe (MalwareBazaar), 7.exe (VirusTotal)

Detections
MalwareBazaar: BumbleBee, **Virustotal**: 51/73 as of 2023-09-12 03:16:46 - Trojan.Win32.BumbleBee.4!c (Lionic), Gen:Variant.Lazy.339495 (MicroWorld-eScan), TrojanDownloader.Win64 (CAT-QuickHeal), Gen:Variant.Lazy.339495 (ALYac), Downloader.Win64.Bumblebee.Vle3 (Sangfor), Gen:Variant.Lazy.339495 (BitDefender), Trojan.Lazy.D52E27 (Arcabit), Trojan-Downloader.Win64.BumbleBee.aik (Kaspersky), Downloader.BumbleBee!8.15B7F (TFE:5:54W0iAL9QiL) (Rising), Trojan.TR/AD.BumbleBee.lfbzg (F-Secure), Gen:Variant.Lazy.339495 (VIPRE), Trojan.Win64.BUMBLELOADER.YXDIHZ (TrendMicro), W32.Trojan.TR.AD.BumbleBee.lfbz (Webroot), TR/AD.BumbleBee.lfbzg (Avira), Trojan[Downloader]/Win64.Bumblebee (Antiy-AVL), Trojan:Win32/Synder!ic (Microsoft), Trojan-Downloader.Win64.BumbleBee.aik (ZoneAlarm), Gen:Variant.Lazy.339495 (GData), TrojanDownloader.BumbleBee (VBA32), Trojan.BumbleBee (Malwarebytes), Trj/Chgt.AD (Panda), Trojan.Win64.BUMBLELOADER.YXDIHZ (TrendMicro-HouseCall)

This is the file after unpacking:

File type
17E93000000.bin: PE32+ executable (GUI) x86-64, for MS Windows

MD5
0b7b1d42e60ceeff49ef796d4d4f5b14

SHA1

54a528d2f62b1af4c331f2c5ae03efdabc793833

SHA256

af59ce785e062bf0d198eb4e3bdbc1ee57d58164de6dc1faf38836c670ef6f7d

Size

1 MB (1048576 Bytes)

Compile Timestamp

2023-09-04 16:02:26 UTC

Links

MalwareBazaar, Twitter, Dropped_by_md5, Cape, VirusTotal

Filenames

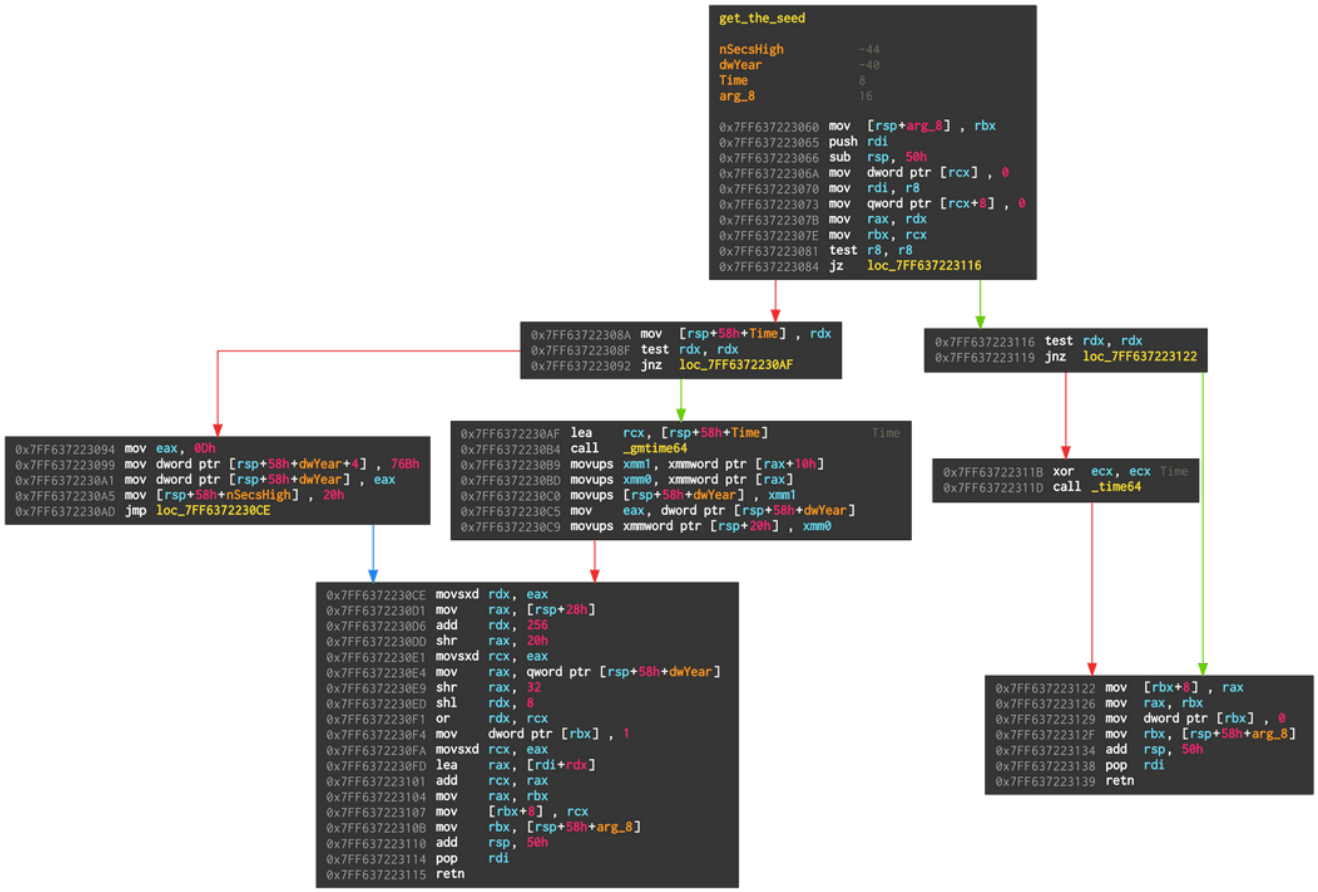
17E93000000.bin (MalwareBazaar), 17E93000000.bin (VirusTotal)

Detections

MalwareBazaar: BumbleBee, **Virustotal**: 25/75 as of 2023-09-15 17:56:30 - Windows.Trojan.Bumblebee (Elastic), a variant of Win64/Bumblebee.M (ESET-NOD32)

Reverse Engineering

This is the seeding of the DGA:

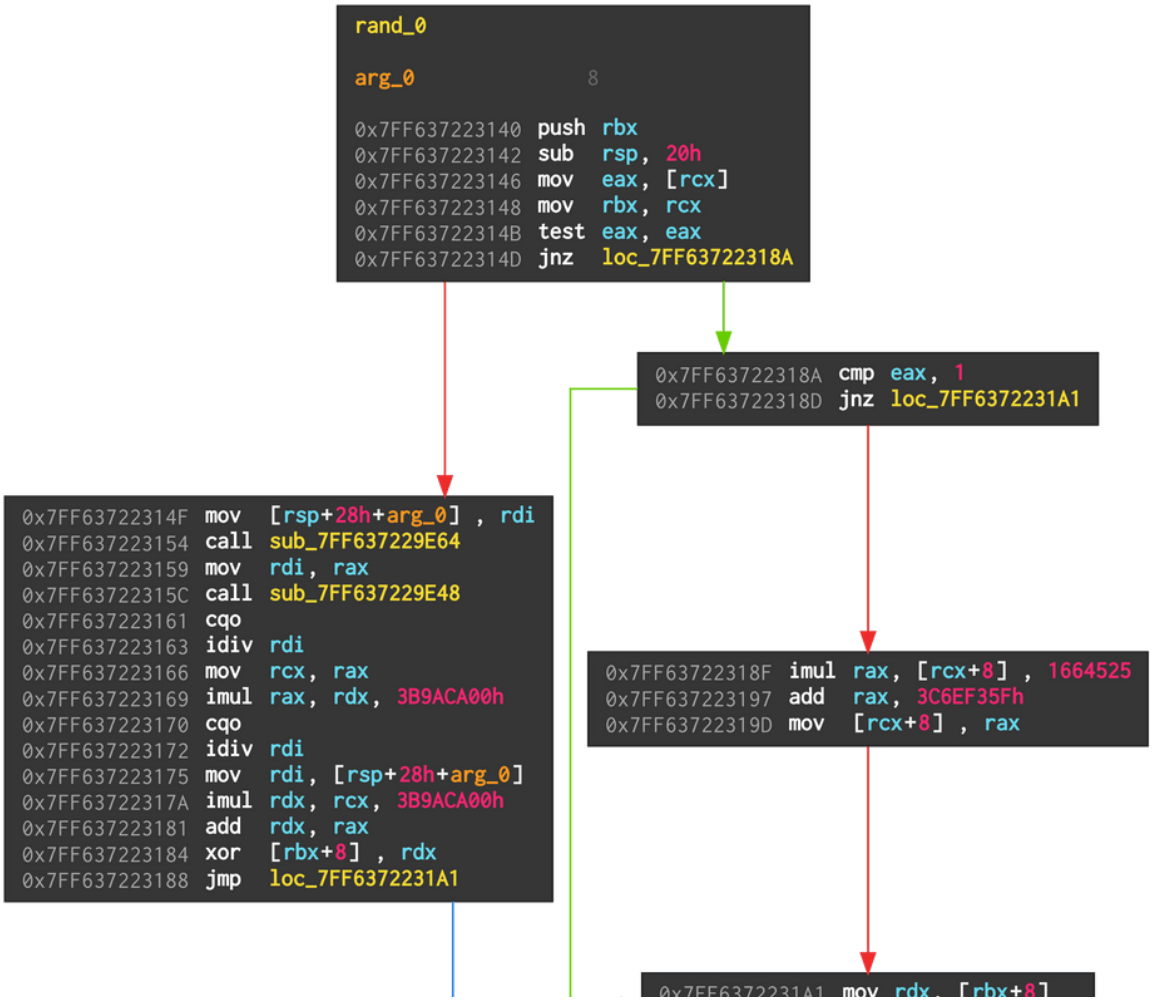


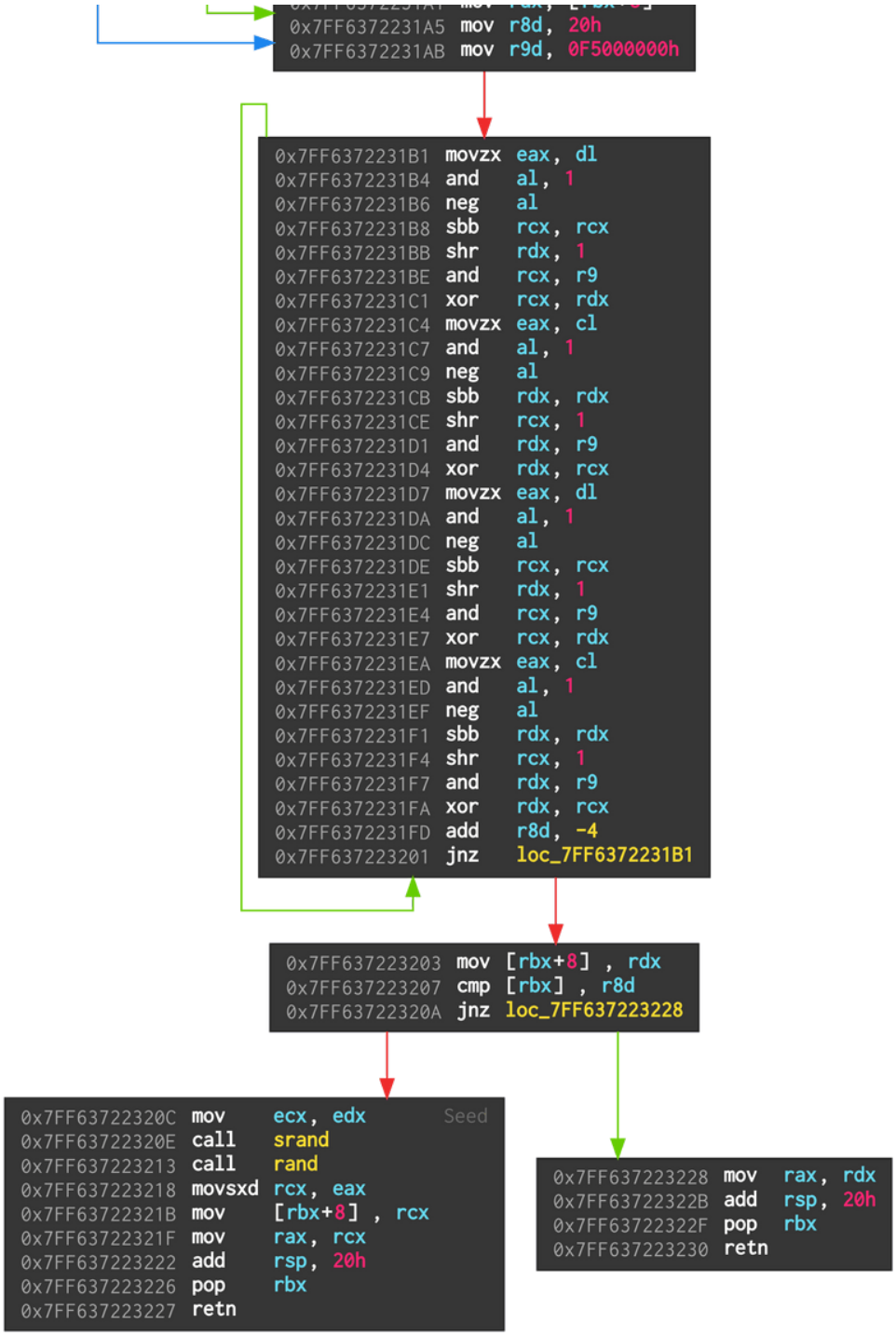
The seed can either be based on a time that is passed to the function, or be time-independent. If the seed is time dependent, then the seed changes based on the current year, month, and second. A magic seed value can be used to create different sets of domains for the same dates (interestingly, the magic seed for the analysed sample is TEST_SEE). Here is a reimplementation of the seeding:

```
def seed(magic: int, time: Optional[datetime] = None) -> int:
    if time:
        secs = time.second
        month = time.month - 1
        year = time.year
    else:
        secs = 32
        month = 13
        year = 1899
    return magic + (secs | ((month + 256) << 8)) + year
```

Not shown is the seeding when the magic value is 0. In this case, the seed is using the current unix timestamp as the seed, likely creating domains that are unpredictable to the attackers and any feasible sinkholing attempts.

The seed is then used for the random number generator:





Only the right path of the first branch is relevant. It is a simple LCG (Linear congruential generator) with the common parameters 1664525 as the multiplier and 1013904223 as the increment as proposed in the book Numerical Recipes .

The strange loop towards the end extracts the high DWORD of the random number by dividing it 32 times by 2. But it also XORs the result with 0xF5000000 if the division has a remainder. I didn't not figure out what the purpose of these XOR operations is, but it is likely part of library and improves the RNG.

```
def rand(r: int) -> int:
    r = r*1664525 + 1013904223
    r &= (2**64 - 1)
    for _ in range(32):
        if r & 1:
            r = (r // 2) ^ 0xF5000000
        else:
            r = (r // 2)
    return r
```

The random number generator is finally used to generate the domain names:

Figure 1The core of BumbleBee's

The character set `szCharSet` is the lowercase alphabet followed by the digits. The DGA picks 11 characters based on the generated pseudo random numbers, then tacks on the TLD `.life`. In total, 100 domains are generated.

Reimplementation

Here is a reimplementation of the DGA in Python:

```
from datetime import datetime
import argparse
from typing import Optional
import string

charset = string.ascii_lowercase + string.digits

def seed(magic: int, time: Optional[datetime] = None) -> int:
    if time:
        secs = time.second
        month = time.month - 1
        year = time.year
        secs = 0
        month = 0
        year = 0
    else:
        secs = 32
        month = 13
        year = 1899
    return magic + (secs | ((month + 256) << 8)) + year

def rand(r: int) -> int:
    r = r*1664525 + 1013904223
    r &= (2**64 - 1)
    for _ in range(32):
        if r & 1:
            r = (r // 2) ^ 0xF5000000
        else:
            r = (r // 2)
    return r

def dga(seed: int):
    r = seed
    for _ in range(100):
        domain = ""
        for _ in range(11):
            r = rand(r)
            domain += charset[r % len(charset)]
        yield domain + ".life"

if __name__=="__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--magic", "-m", type=str, default="TEST_SEE")
    help_msg = (
        "time for which to generate domains, e.g., "
        "2020-06-28 13:37:12. If not specified, the static DGA is used"
    )
    parser.add_argument("--time", "-t", help=help_msg")
    args = parser.parse_args()
    if args.time:
        time = datetime.strptime(args.time, "%Y-%m-%d %H:%M:%S")
    else:
```

```
        time = None
    magic = sum(ord(v) << i*8 for i, v in enumerate(args.magic))
    s = seed(magic=magic, time=time)
    for domain in dga(s):
        print(domain)
```

Characteristics

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

property	value
type	time-independent-deterministic, or time-dependent - deterministic
generation scheme	arithmetic
seed	current date
domain change frequency	never or every month and second (repeating each minute)
domains per day	100 if time-independent, 6000 if time-dependent
sequence	in order
wait time between domains	unknown
top level domain	`.level`
second level characters	a-z0-9
regex	`[a-z0-9]{11}\.life`
second level domain length	11

Example Domains

cmid1s1zeiu.life
itszko2ot5u.life
3v1n35i5kwx.life
newdnq1xnl9.life
jkyj6awt1ao.life
ddrjv6y42b8.life
1pnhp5o5za1.life
y13iqvlfjl5.life
xp0btfgegbo.life
gpv3uw5tmy4.life
5d7rdf3layn.life
2aed6bvquxs.life
5t9oknzu433.life
sy53gmpuq1i.life
09cwff8wgdh.life
4elhq2521mw.life
b4arp834sch.life
s3iug4uiy7t.life
q1cvhi9onpu.life
m3j4htyodnu.life
dzzrhn9rvqa.life
uriqas6zede.life
tv45x1ukt9w.life
9dnuk0xl7yc.life
zro95b8zb3r.life
9da1kshoyuq.life
zph13yx1leo.life
0q6mvuo4wl6.life
nyoqtkpub9x.life
l1bnym8lg65.life
d63hq5crsun.life
f4te7v7fi28.life
oi27t509pny.life
xg2mddk9qrj.life
9uknixukwim.life
5ejt5qpx2oh.life
v9y5rypfhdj.life
aq59tsppo18.life
vdnizm8lcke.life
knof8y1kufn.life
mhwv3bpckbi.life
b4ycw3b0ztx.life
tu0t62osn5m.life
pkgbfa9ati6.life
wd60v3x8mun.life
qpgomg0nfob.life
9619skmuswk.life
10fa4glizbq.life
h9cgsquxt5t.life
cpjeg06jqj7.life
tuaksrh3m4v.life
pnkk456mk55.life

bryfg80da8m.life
4c9takty1zx.life
17afrof66rf.life
keoauupcj2n.life
okxar0c3d29.life
759lhww6ixh.life
br40ztd8bya.life
vdug3t5r2cz.life
6j0uqybrqj4.life
km87l2nqldk.life
d421obfpnmh.life
hsk3pjutatd.life
iudmgiv2ndb.life
vf9bknmns0b.life
325g1cipn4m.life
g3z3h2xzdfv.life
i4hmyqc1p69.life
r967duebyji.life
f83jeqe01vd.life
sbprbiukvhf.life
lc2q21q7nd4.life
co7hu2019oy.life
ue9panfagh0.life
fby66hp7jm0.life
njg6qfp2lfa.life
mb1hy4vi0q7.life
7jemrghylwb.life
yxz60ai05jv.life
v68i3v975xq.life
67xsof7l8ak.life
q886dsegew3.life
16nqnk7hvgs.life
we5x2dfevhn.life
88kwlc3k73o.life
p2xo397h86f.life
njljnzf5c20.life
2g6py8d93tm.life
dz8bw5q6jy2.life
gflfug3a9lb.life
rssaelatar7.life
35l9tvici4l.life
lqhjkq5lfiu.life
3t3qouhmhww.life
fuwisezq1sl.life
ibm2bld58ah.life
h02pknjmc6v.life
enenfxgn3fh.life
zcf8nrpzrqk.life