The DGA of Sisron

Archived: 2026-04-06 03:26:32 UTC

Sisron was part of a financial fraud and identity theft botnet. It was taken down by Microsoft in the anti-botnet operation B106. The malware uses a very simple domain generation algorithm (DGA) that produces sets of similar looking domains.

For example, these are some of the domains listed in a Sophos report:

```
 mjawmjiwmtia.com
mjawmjiwmtia.org
mjewmjiwmtia.com
mjewmjiwmtia.org
mjiwmjiwmtia.com
mjiwmjiwmtia.org
mjmwmjiwmtia.com
mjmwmjiwmtia.org
mjqwmjiwmtia.com
mjqwmjiwmtia.org
mtcwmjiwmtia.com
mtcwmjiwmtia.org
mtgwmjiwmtia.com
mtgwmjiwmtia.org
mtkwmjiwmtia.com
mtkwmjiwmtia.org
mtuwmjiwmtia.com
mtywmjiwmtia.com
mtywmjiwmtia.org
```

In this write-up I show:

- when the DGA was most active and what names the AV vendors have given the malware;
- disassembly of the DGA routine and its caller;
- a reimplementation of the DGA in Python as well as a script to determine when a given domain was generated;
- properties of the DGA, including a convoluted yet precise regex to match Sisron domains.

Prevalence and Names

After I reversed the DGA, I went through VirusTotal and collected the reports of 514 scans that contacted one of the generated domains. Each square in the following graphic represents one scan on Virustotal.

The first sample with the DGA was uploaded on April 2013. Summer and fall of 2013 saw the most Sisron samples. Fewer samples were observed throughout 2014. After that, Sisron died off due to the takedown events — almost no samples on Virustotal can be found that use the DGA domains anymore.

Sisron has been detected well by most anti virus software. Many products detected all samples, albeit in some cases only with generic rules. Here is a list of major products with their most common denominations:

product
: names

TrendMicro
: 92% of the samples are detected as **TOMB**

McAfee
: only generic detections (*Generic PWS*; *Generic PUP*; Generic.dx*; *GenericATG*; *Artemis*; …)

ClamAV
: uses the ids **Trojan.Agent-1360** and **Trojan.Agent-267491**

ESET
: all except three samples are labelled **Win32/Agent.WRQ**

Agnitum
: Detects over 300 samples as **Trojan.Scar**. The same name is also used by *Malwarebytes*, *VBA32*, *ViRobot*, *Jiangmin*, *TheHacker* and *AhnLab-V3*

Microsoft
: Microsoft detects 411 samples as **Trojan:Win32/Sisron** and 89 as **Trojan:Win32/Sisron!gmb**

Kaspersky
: Calls the samples either *HEUR:Trojan.Win32.Generic* or *Trojan-Spy.Win32.Zbot.njah*

Sophos
: Uses the designations **Troj/Agent-UYB** and **Troj/Agent-OWJ**

Reverse Engineering

I reversed the following sample from VirusShare.com. The sample was discovered by Daniel Plohmann with the help of Shadowserver:

md5
: e0a9b99f6b05a0cd66cdb5e4f5037980

sha256
: c80b5846fe82d218f496db519d6cfe7b9d7625dd2e00b044850208d28a7eee94

sha1
: c71df833c529b33f10b865ae922ab8c3cb45afb4

source
: VirusShare

Virustotal report
: link

size
: 89'242 bytes

ExIF time stamp
: 2010:09:22 06:56:08-04:00

DGA

The DGA has three parts:

1. Generating a time-based string.
2. Encoding the string with a slightly modified base64 to obtain the second level domain.
3. Appending one of four top level domains.

Time-Based String

The DGA is seeded with the current date — determined by calling `__time64`. Sisron uses a sliding window so that domains are valid for more than one day. The malware achieves this sliding window by backdating the current date a variable number of seconds given in the first function argument. The resulting time value is finally converted into a string of format `<day><month><year>`, e.g., `31052016`.

Base64

The date string is transformed using base64 with three modifications:

1. The characters for index 62 and 63 are both `a`, not the commonly used `+` and `/`, which of course would not be valid domain letters. This modification is not necessary and has no effect. The binary representation of 62 and 63 is `111110` and `111111` respectively. The encoding, however, operates on ASCII digits which at most have 3 consecutive `1`s in their binary representation and start with `00`. So regardless of which sequence of digits is encoded, there is no way to have more than three consecutive `1`s. For the same reason there are no digits in the base64 encoding.
2. Padding uses the letter `a` instead of `=`. Note that the date string always has 8 letters which requires one padding byte. All second level domains therefore end with `a` (see Regex).
3. The resulting string is converted to lowercase, as domains are case insensitive.

Interestingly, even though the encoded string is converted to lowercase, it still can be reversed back to the original date, see Section Reversal.

Top Level Domain

The top level domain is selected from an array of four top level domains *.com*, *.org*, *.net*, and *.info*. The index into the tld array is passed as the second argument of the DGA routine.

DGA Caller

The following routine excerpt calls the DGA:

Register `eax` holds the argument to the function, which is mapped as follows to the parameters of the DGA:

- number of seconds to backdate the date: `86400*(eax % 10)`. 86400 is the number of seconds in a day, so the date is backdated between 0 and 9 days.
- top level domain index: `(eax // 10)`. The register `eax` is assigned the values 0 to 39, as can be seen in the next graph view. This means that the tld index is 0 to 3, which corresponds with the four top level domains.

As can also be seen from the above graph view, the DGA sleeps 3 seconds after each failed DNS request. Once the index reaches 40, it is reset to 0, leading to an infinite loop as long as none of the 40 domains resolve.

Reimplementation and Reversal

Reimplementation

The following Python code generates all 40 domains for a given date (including the domains from the sliding window). You also find this code, as well as reimplementations for other DGAs, on my Github page.

```
 from datetime import datetime, timedelta
import base64
import argparse
```

```
def dga(d, day_index, tld_index):
    tlds = [".com", ".org", ".net", ".info"]
    d -= timedelta(days=day_index)
    ds = d.strftime("%d%m%Y")
    return base64.b64encode(ds).lower().replace("=","a") + tlds[tld_index]

if __name__=="__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("-d", "--date", help="date for which to generate domains")
    args = parser.parse_args()
    if args.date:
        d = datetime.strptime(args.date, "%Y-%m-%d")
    else:
        d = datetime.now()
    for i in range(40):
        print(dga(d, i%10, i//10))
```

Reversal

Although the DGA is converting the base64 encoding to lowercase, e.g., not distinguishing between `a` and `A`, the domains can still be converted back to a unique date. The following code takes a domain and converts it back to the date:

```
 from datetime import datetime, timedelta
import base64
import argparse

def reverse(domain):
    domain = domain.split('.')[0][:-1] + "="
    for c in ['A','E','D','I','M','O','N','Q','U','T','Y']:
        domain = domain.replace(c.lower(),c)
    return datetime.strptime(base64.b64decode(domain), "%d%m%Y")

if __name__=="__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("domain")
    args = parser.parse_args()
    print("The domain is from {}".format(reverse(args.domain).strftime("%x")))
```

For example:

```
 $ python reverse.py mjkwntiwmtya.info
The domain is from 05/29/16
```

Be aware of the sliding window: a given date can be revisited up to 9 days into the future. The domain in the example above will be generated up to June 6th.

Properties

Characteristics

The following tables summarizes the properties of the Sisron DGA.

property	value
type	TDD (time-dependent-deterministic)
generation scheme	pseudo base64 encoding
seed	current date
domain change frequency	daily, with a 10 day sliding window
domains per day	4 (plus 36 revisited)
sequence	sequential
wait time between domains	3 seconds
top level domains	.com, .org, .net, .info
second level characters	very limited set of lower case letters, see Regex
second level domain length	12

Regex

Since the source of input of the base64 encoding are dates — and therefore a very small subset of all 8 byte combinations — there exists a quite specific regular expression for the domains. The following regex holds for all domains for the years 2000 to 2100:

```
[m][djtz][acegikmquy][wx][mno][djtz][i][wx][mno][djtz][acegikmquy][a]\.(com|org|net|info)
```

All second level domains begin with `m` and end with `a`. The remaining letters are also limited. Some letters don't appear at all, i.e., `bfhlprsv`.
