A Death Match of Domain Generation Algorithms

By Yuriy Yuzifovich

Published: 2022-01-19 · Archived: 2026-04-05 16:06:45 UTC

Authors: Yuriy Yuzifovich and Hongliang Liu, originally published on December 29, 2017

The dictionary definition of Domain generation algorithms (DGA) is "*algorithms seen in various families of malware that are used to periodically generate a large number of domain names that can be used as rendezvous points with their command and control servers*" (https://en.wikipedia.org/wiki/Domain_generation_algorithm). In the real-life recursive DNS traffic we monitor at Nominum (- now part of Akamai) we observe a lot of 'strange' DNS queries, many of them generated by malware DGAs; below are some examples for domains generated by the Dyre banking trojan (aka Dyreza):

```
t3622c4773260c097e2e9b26705212ab85.ws.
u83ccf36d9f02e9ea79a9d16c0336677e4.to.
v02bec0c090508bc76b3ea81dfc2198a71.in.
wa9e4628c334324e181e40f33f878c153f.hk.
xdcc5481252db5f38d5fc18c9ad3b2f7fd.cn.
yf32d9ac7f0a9f463e8da4736b12d7044a.tk.
```

Malware creators use algorithmically generated domains as a diversion mechanism: they flood the DNS stream with requests for thousands of DGA-based domains but select only a few domains to provide the true C&C service, where the malware can find its mothership and communicate for instruction. Meanwhile, poor security researchers get overloaded with work trying to discover and block the selected few.

Other than creating a diversion, malware creators use DGA's because they are harder to detect compared to hardcoded IPs or domain names; by not hardcoding the location of the C&C in the malware binary itself, the attacker can better hide and protect the mothership; Attackers keep creating new DGAs, and once again — create work overload for security researchers, who need to reverse-engineer binaries, or use different machine intelligence driven methods, in order to discover the DGAs.

In this article, we are going to discuss this deathmatch between attackers and security researchers on DGA battleground.

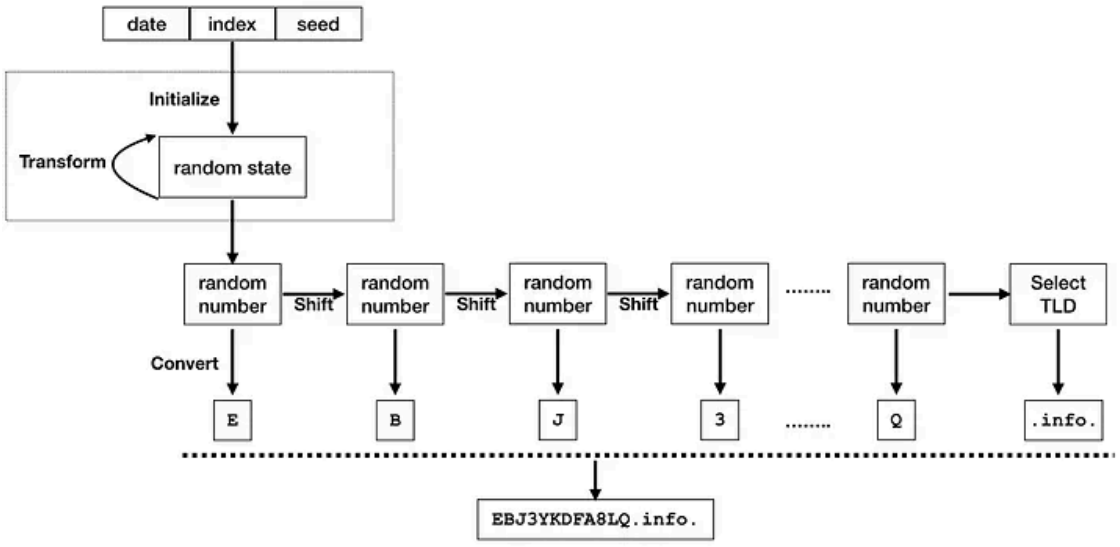
THE DGA BATTLEGROUND

The general methodology of any DGA is using a deterministic pseudo-random generator (PRNG) to generate a list of candidate domain names. The seed of a PRNG can be the current date, some magic numbers, an exchange rate, etc. This random generator can be a single uniform distribution generator, e.g. use a combination of bitshift, xor,

divide, multiply, and modulo operations to generate a string sequence as the domain name (such as in Conficker, Ramnit, and others); it can also be a rule generator, which selects from some knowledge base (such as in Suppobox). For example, the following DGA algorithm uses the current date as the seed, and a PRNG to generate a char sequence for the DGA domains (https://en.wikipedia.org/wiki/Domain_generation_algorithm):

```
generate_domain(year, month, day):
 """Generates a domain name for the given date."""
domain = ""
for i in range(16):
year = ((year ^ 8 * year) >> 11) ^ ((year & 0xFFFFFFF0) << 17)
month = ((month ^ 4 * month) >> 25) ^ 16 * (month & 0xFFFFFFF8)
day = ((day ^ (day << 13)) >> 19) ^ ((day & 0xFFFFFFFE) << 12)
domain += chr(((year ^ month ^ day) % 25) + 97)
return domain
```

Press enter or click to view image in full size



The key to the PRNG-based DGA methodology is a deterministic random generator, where the DGA sequence is predictable from both the malware and the attacker, so the attacker can generate and select some of these domains for C&C service, and the malware just needs to loop over and reach the chosen C&C.

This part requires some agreement: both the DGA and the seed must be known by both sides before generating DGA domains. However, this agreement exchange isn't only to the malware and the attacker; anyone, like we security researchers, can replicate it even after the infection. This non-exclusive feature provides the breakthrough point for security research: by intercepting both the DGA and the seeds, one can predict the malware DGA domains and block them.

To obtain the DGA algorithm itself, security researchers might need to reverse engineer the malware binary after capturing the malicious binary code. Many DGA algorithms are reverse engineered and reported by multiple projects and security blogs, such as:

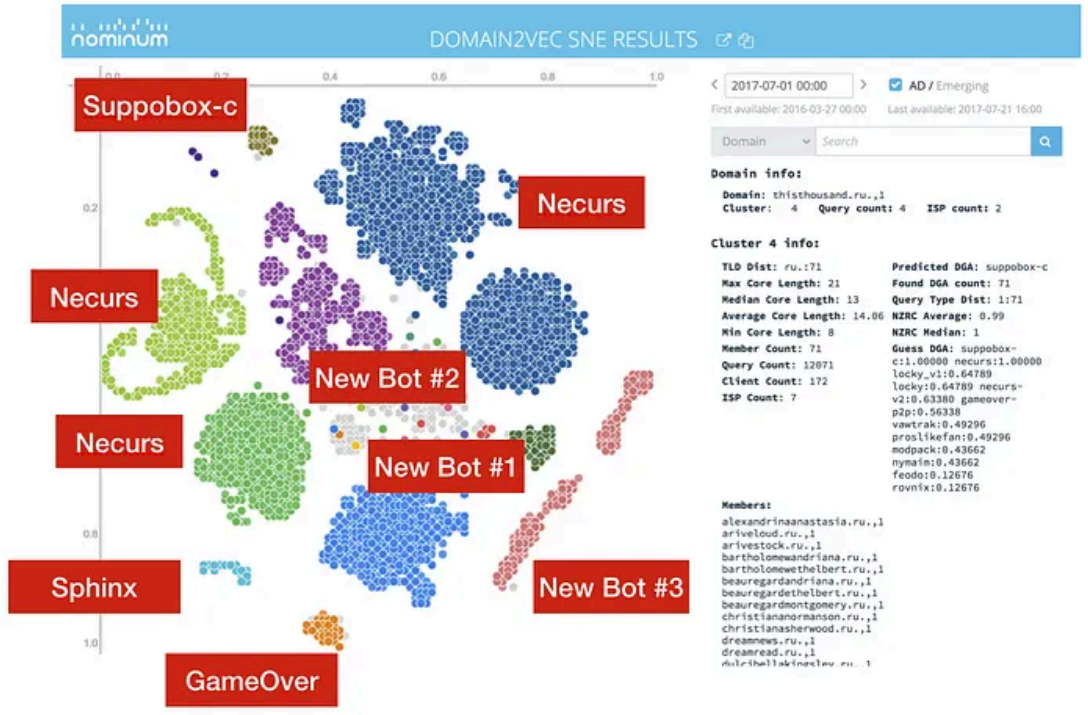
- DGArchive https://dgarchive.caad.fkie.fraunhofer.de/
- 360netlab's DGA project https://github.com/360netlab/DGA
- Johannes Bacher's reversing efforts https://github.com/baderj/domain_generation_algorithms.

Having the DGA algorithm and knowing the DGA seed is a sufficient condition to predict DGA domains, but is not a necessary one to have the DGA domain list: we can reduce the problem to separate DGA traffic from legitimate traffic, and obtain the DGA domain list from the traffic. In DNS traffic, we can model a feature phase space where DGA domain queries and other legitimate queries are separable, where the ground truth of DGA (algorithm and seeds) are not needed, and the task can be abstracted as finding this "golden phase" space.

There are several advanced machine learning methods to find this phase space and separate the malicious DGA from the legitimate, without reverse engineering the binary. Most of these methods use client IP vs domain visit graph features; for example:

Our team's Domain2vec correlation engine uses representation learning to discover DGA clusters in real-time DNS traffic. This method builds a sequence model to learn the domain correlation and captures the malware activity since the malware needs to loop over DGA names. (See also — "Augmented Intelligence to Scale Humans Fighting Botnets", https://www.botconf.eu/2017/augmented-intelligence-to-scale-humans-fighting-botnets/):

Press enter or click to view image in full size



Press enter or click to view image in full size

These methods have reduced the strong condition to a loose yet more general condition and has solved the difficulty of obtaining both DGA and the seeds. In the later part of this post, we will talk about some cases which only these loose condition methods can detect.

After obtaining the DGA algorithms, the battleground now moves on to the random seed front…

MAGIC NUMBER SEEDS

Some DGAs only use the current date as the seed, and hardcode some numbers in the binary; these DGA's can be easily predicted when the algorithm is reverse-engineered, for example, in Conficker families, Nymaim etc.

Since the attacker's goal is to avoid detection, it becomes practical to use magic numbers as dynamic seeds. The magic number technique is very common today, and Necurs (the backdoor), Locky (the ransomware) (https://test-nominum.pantheonsite.io/unlocking-locky/) are good examples of the combined usage of date/time and magic numbers. Magic numbers are usually combined with the date in bit-shifting and provide additional variance. Popular malware like Locky can deploy many variants with different magic numbers each to evade detection (see also — https://blogs.forcepoint.com/security-labs/lockys-new-dga-seeding-new-domains).

Since DGA can generate DNS traffic with seeds, astute researchers can get the seeds by using DNS traffic and the DGA. To capture the dynamic magic number seeds, researchers usually use "replay attack" technique by

reproducing possible DGA domains and validating it in the DNS traffic. OpenDNS, for instance, has a brute force method to search the possible numerical seeds (magic numbers) in the DNS traffic by generating all 2^{32} sets of Ramnit names (https://www.slideshare.net/OpenDNS/using-algorithms-to-brute-force-algorithms-a-journey-through-time-and-namespace).

This method works well not only for ramnit but also for Necurs and other DGAs, especially when the magic numbers are small. However, this method is not always useful because generating all 2^{32} names can be expensive, and the malware can easily escape it by upgrading to a 2^{64} seed, as already happened with the Murofet's DGA. Our team has proposed and implemented a more sophisticated hash collision method, primarily to crack down Locky's dynamic seeds (https://www.botconf.eu/2017/math-gpu-dns-cracking-locky-seeds-in-real-time-without-analyzing-samples/). Instead of using brute force linear test on all seeds with domains in the DNS traffic, this method uses GPU computing and collide the hash value of possible Locky DNS queries with real-time DNS traffic for detecting the new seeds.

Beyond magic numbers, magic strings or magic domain names are also used for generating DGA domains. Currently, there are not many effective methods to detect these seeds beyond reverse engineering the binary.

Other types of seeds

The exchange rate of currency can be used as random seeds. Bedep the Ad/Click fraud botnet, for example, use foreign currency exchange rate as their seed (https://www.arbornetworks.com/blog/asert/bedeps-dga-trading-foreign-exchange-for-malware-domains/). Some botnets use the most popular hashtag on twitter as the DGA seed, as reported by Cybereason (http://go.cybereason.com/rs/996-YZT-709/images/Cybereason-Lab-Analysis-Dissecting-DGAs-Eight-Real-World-DGA-Variants.pdf). The common idea behind these seeds is that they are hard to reproduce and that the seed may not be a simple number.

GOOD DGA, BAD DGA

The creators of DGA algorithms want to keep the uniqueness of the DGA so they can distinguish their C&C traffic from legitimate traffic, and also avoid collision with other DGAs. Our research has shown us that some DGAs are smarter than others.

Dictionary based DGA

A little twist in the way algorithmically generated domains are created in the dictionary based method. As we've seen, security researchers use features in the DNS string to separate malicious DGA traffic from legitimate traffic. The modeling work looks at attributes such as randomness, entropy and other lexical string features, which frequently generate domains with a 'random', 'non-human readable' look. (see for example https://www.r-bloggers.com/building-a-dga-classifier-part-1-data-preparation/). Some cleverly designed DGAs such as Suppobox try to evade this randomness by using dictionary words:

High collision DGA

DGAs like Pykspa and Virut are getting lower grades in our notebook: they have strong collisions with other legitimate names and other DGAs.

Pykspa is a worm whose DGA is reverse-engineered at https://www.johannesbader.ch/2015/03/the-dga-of-pykspa/. This DGA generates thousands of possible DGA domains using common TLDs like com, biz, net, org, info and cc, and its core domain has 6–15 chars. These thousands of domains flood the recursive DNS traffic. Because of the common TLD set and the short domain length for these huge amounts of domains, security researchers have a hard time to clearly identify and block them, even if they know the DGA + seed to predict. For example, some short domains like `wgxodod.info.` `ydnpxkv.info.` `hrvxccq.org.` have a good chance to collide with other DGAs (such as Locky), or with legitimate `.com` names.

Virut is another type of DGA where the domain name only has 6 a-z chars with `.com` TLD, and the algorithm itself has a simplistic design, so the chance of a generated domain colliding with a legitimate service is very high. We have observed many domains like `wenxin.com`, which was a legitimate domain, yet it was reported as Virut by some security researcher (https://twitter.com/DGAFeedAlerts/status/917181597400600576). And by the way, the domain `akamai.com` follows the exact pattern of a Virut DGA. But don't get too concerned…

Blocking these high collision DGA domains in a safe way requires security researchers to combine the domain prediction method with DNS traffic; Our team has recently implemented a real-time new core domain detection system (for domains never seen before), where only the predicted DGA are blocked only if identified also as a new core domain.

Non DGA

In DNS traffic, we've observed many 'DGA-look-alike', which are not in fact DGA domains. For example, in recent traffic we saw these 7 char `.ru` domains with very high infection rate:

```
bhzlyxh.ru.
qsxxzni.ru.
gwjijru.ru.
fyxkmbh.ru.
qwoumzw.ru.
kulfxxy.ru.
nrxboty.ru.
…
```

instead of a DGA, the author has hardcoded them in the binary and deployed different lists in different binaries. These names are used in Ruskill/Dorkbot as reported at http://tech.cert-hungary.hu/vulnerabilities/CH-14106 and https://github.com/360netlab/DGA/issues/36.

Update: there is an ongoing discussion here https://github.com/360netlab/DGA/issues/36#issuecomment-350660012 about the DGA behind Dorkbot, where Johannes Bader has commented that Dorkbot generates these names every 10 seconds and uses as decoys.

Press enter or click to view image in full size

SUMMARY

DGA is one of the most effective and most popular tools in the attackers' toolbox. It is being used by a variety of malware families to hide the location of their C&C servers, and by that maintain the robustness of the botnet. At the same time, DGAs leave a substantial footprint in the DNS traffic.

In the deathmatch between DGA creators and security researchers, the attackers do their best to hide the C&C and to avoid collision with other DGAs and legitimate services, while researchers use both the traditional reverse engineering and modern machine learning to clearly identify and block these DGAs. This battle is far from over and will continue to emerge as both sides grow stronger. We will keep you updated, stay tuned…
