A Comprehensive Measurement Study of Domain Generating Malware

Daniel Plohmann, *Fraunhofer FKIE;* Khaled Yakdan, *University of Bonn;*
Michael Klatt, *DomainTools;* Johannes Bader; Elmar Gerhards-Padilla, *Fraunhofer FKIE*

https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/plohmann

This paper is included in the Proceedings of the
25th USENIX Security Symposium
August 10–12, 2016 • Austin, TX
ISBN 978-1-931971-32-4

A Comprehensive Measurement Study of Domain Generating Malware

Daniel Plohmann
Fraunhofer FKIE

Khaled Yakdan
University of Bonn

Michael Klatt
DomainTools

Johannes Bader

Elmar Gerhards-Padilla
Fraunhofer FKIE

Abstract

Recent years have seen extensive adoption of domain generation algorithms (DGA) by modern botnets. The main goal is to generate a large number of domain names and then use a small subset for actual C&C communication. This makes DGAs very compelling for botmasters to harden the infrastructure of their botnets and make it resilient to blacklisting and attacks such as takedown efforts. While early DGAs were used as a backup communication mechanism, several new botnets use them as their primary communication method, making it extremely important to study DGAs in detail.

In this paper, we perform a comprehensive measurement study of the DGA landscape by analyzing 43 DGA-based malware families and variants. We also present a taxonomy for DGAs and use it to characterize and compare the properties of the studied families. By reimplementing the algorithms, we pre-compute all possible domains they generate, covering the majority of known and active DGAs. Then, we study the registration status of over 18 million DGA domains and show that corresponding malware families and related campaigns can be reliably identified by pre-computing future DGA domains. We also give insights into botmasters' strategies regarding domain registration and identify several pitfalls in previous takedown efforts of DGA-based botnets. We will share the dataset for future research and will also provide a web service to check domains for potential DGA identity.

1 Introduction

Botnets are networks of malware-affected machines (*bots*) that are remotely controlled by an adversary (*botmaster*) through a command and control (C&C) communication channel. Botnets have become the primary means for cyber-criminals to carry out their malicious activities, such as launching denial-of-service attacks, sending spam, and stealing personal data. Recent studies have shown that some botnets consist of more than a million bots [40], illustrating the magnitude of their threat.

Law enforcement and security researchers often try to disrupt active botnets by performing takedown attempts. The main target of these attacks is the C&C communication infrastructure of the botnet. A prominent example of these attacks is *sinkholing*, where all bots are redirected to an attacker-controlled machine called a *sinkhole*. In consequence, the bots will be prevented from communicating with the original C&C servers. As a response to these efforts, botmasters have started inventing new techniques to protect the infrastructure of their botnets. An important approach that has gained wide popularity in recent years is the use of domain generation algorithms.

A domain generation algorithm (DGA) is used to dynamically generate a large number of seemingly random domain names and then selecting a small subset of these domains for C&C communication. The generated domains are computed based on a given *seed*, which can consist of numeric constants, the current date/time, or even Twitter trends. The seed serves as a shared secret between botmasters and the bots to compute shared rendezvous points. By constantly changing the used domains, detection approaches that rely on static domain blacklists are rendered ineffective. Moreover, by dynamically generating domain names, botmasters do not have to include hard-coded domain names in their malware binaries, complicating the extraction of this information. Also, making the generated domains dependent on time lessens the value of domains extracted from dynamic malware analysis systems since different domains will be observed at different time points. Another advantage of using short-lived domains that are registered shortly before they become valid is evading domain reputation services.

The use of DGAs creates a highly asymmetric situation between attackers (botmasters) and defenders (security researchers and law enforcement). Botmasters need access to a single domain to control or migrate their bots while defenders need to control all of the domains to ensure a successful takedown. With more than 1000 top-level domains (TLDs) [9] to choose from, it is easy for an attacker to create a global spread of responsibility for domains, forcing the defenders into additional coordination and cooperation efforts. For example, the DGA of the infamous Conficker botnet (version C) generated 50,000 domain names per day, which spread out over 113 TLDs. This required global cooperative efforts of 30 different organizations including ICANN [2] to contain the threat.

In this work we perform a comprehensive measurement study of the DGA landscape by analyzing 43 DGA-based malware families and variants. Our analysis is based on reverse-engineering the DGAs of these families. We propose a taxonomy to characterize the main aspects of DGAs and use it to describe and compare the studied DGAs. We furthermore reimplemented all of these DGAs and computed all their possible outputs based on a set of 253 seeds used in previous and ongoing malicious campaigns. We then used this set to identify DGA-generated domains in a set of 9 billion WHOIS records collected over the last 14 years. We analyze the registration status of these domains and their ownership changes, which often indicates transitioning from a malicious into a sinkholed domain. This enables us to profile the registration behaviour of both botmasters and sinkhole operators and investigate in detail the lifetime of DGA domains.

To the best of our knowledge, our work reflects the first systematic study of the DGA landscape as employed by modern botnets. Security researchers have previously examined DGAs and proposed approaches to detect and cluster DGA-based malware [13, 18, 43, 54]. Our study is instead based on an in-depth analysis of the DGAs in a bottom-up manner, by reimplementing the algorithms and enumerating the complete set of domains they generate, enabling us to have a ground truth about DGA-generated domains with no false positives.

In summary, we make the following contributions:

- We propose a taxonomy for DGAs to characterize and compare their properties.
- We analyze the DGAs of 43 malware family and variants. Using 253 identified seeds, we enumerate all possible domains generated by those algorithms, covering the majority of known and active DGAs.
- We study the registration status of 18 million DGA-generated domains covering a period of 8 years and show that corresponding malware families and related campaigns can be reliably identified by pre-computing future DGA domains.
- We analyze the strategies of both botmasters and sinkholers with regard to domain registration and identify several pitfalls in previous takedown efforts of DGA-based botnets.
- We share the dataset for future research and provide a web service called DGArchive [38] to check whether a queried domain originates from a DGA.

2 A DGA Taxonomy

In this section, we propose a taxonomy for DGAs to characterize their properties and enable a comparison. To model the entire spectrum of different properties, we propose two features designed to capture the different aspects of a domain generation algorithm. Both features are then combined in a single taxonomy that classifies DGAs into classes.

2.1 Seed Source

The seed serves as a shared secret required for the calculation of generated domains, also referred to by the term Algorithmically-Generated Domains (AGD) [54]. It is the aggregated set of parameters required for the execution of a domain generation algorithm. Typical parameters include numerical constants (e.g., length of domains or seeds for pseudo random number generators) or strings (e.g., the alphabet or the set of possible TLDs).

Two properties of seeding have superior significance to characterize a DGA (cp. Barabosch et al.[16]):

Time dependence means that the DGA incorporates a time source (e.g. the system time of the compromised host or the date field in a HTTP response) for calculation of AGDs. In consequence, generated domains will have a validity period only during which these domains are queried by the compromised system.

Determinism addresses the observability and availability of parameters. For the majority of known DGAs, all parameters required for DGA execution are known to a degree that all possible domains can be calculated. Two DGAs use temporal non-determinism to disallow arbitrary prediction of future AGDs by using unpredictable but publicly accessible data for seeding. The malware family Bedep [44] makes use of foreign exchange reference rates published daily by the European Central Bank while a later variant of Torpig [50] used Twitter trends for seeding. In both cases, this only leads to attackers and defenders having to compete for the registration of domains in each active time window once the unpredictable data used for seeding becomes available. However, it does not prevent historic analysis, as the seeding data and thus generated domains can still be collected over time. Another kind of non-determinism has been observed by Symantec [45]. Their analysis of Jiripbot revealed that the malware exfiltrates a set of system properties including MAC address and hard drive volume ID to make it available to the attacker to be used in a DGA seed. In this case, the system information is considered non-deterministic to the attacker prior to compromise and also never publicly observable.

Time-dependence and determinism allow the following four combinations: time-independent and deterministic (TID), time-dependent and deterministic (TDD), time-dependent and non-deterministic (TDN), time-independent and non-deterministic (TIN). In our dataset, we have only observed DGAs using the first three classes of seeding properties.

2.2 Generation Schemes

Apart from the characteristics of seeding, 4 different generation schemes emerged during our analysis.

Arithmetic-based DGAs calculate a sequence of values that either have a direct ASCII representation usable for a domain name or designate an offset in one or more hard-coded arrays, constituting the alphabet of the DGA. They are the most common type of DGA.

Hash-based DGAs use the hexdigest representation of a hash to produce an AGD. We identified DGAs using MD5 and SHA256 to generate domains.

Wordlist-based DGAs will concatenate a sequence of words from one or more wordlists, resulting in less randomly appealing and thus more camouflaging domains. These wordlists are either directly embedded in the malware binary or obtained from a publicly accessible source.

Permutation-based DGAs derive all possible AGDs through permutation of an initial domain name.

We abbreviate the generation scheme with the respective starting letter: A, H, W, or P. As part of any of the above-mentioned generation schemes, some DGAs leverage pseudo-random number generators (PRNGs) to generate domains. These range from implementing own PRNGs to the use of well-known techniques such as linear congruential generators (LCGs) [36].

2.3 DGA Types

As DGA type, we consider the combination of seeding properties and generation scheme, denominated by the combined abbreviations, e.g. "TID-A" for a time-independent, deterministic DGA using a arithmetic-based domain generation scheme. Of 16 possible combinations, we have only observed 6 types being used by the 43 DGAs in our dataset: TDD-A (20), TID-A (16), TDD-W (3), TDD-H (2), TDN-A (1), TID-P (1).

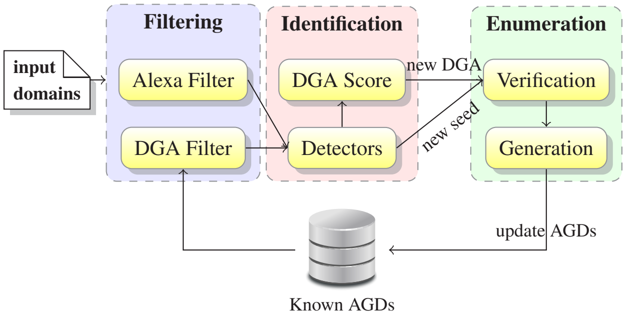

Figure 1: DGA collection approach.

3 DGA-Malware Dataset

In this section, we describe how we identified and collected malware samples that employ a DGA. Then, we describe our efforts to reverse-engineer the algorithms, reimplementing and evaluating their implementation.

3.1 Identifying DGA-based Malware

The first step of our study was to collect a representative set of DGA-based malware families. To this end, we developed a system to automatically identify potentially new DGAs by analyzing a set of domain names generated by a given malware sample. This helps us to minimize the set of malware samples we need to manually reverse-engineer, and thus focus our efforts on samples that are likely to implement new DGAs. A high-level overview of the system is presented in Figure 1. First, we filter out known AGDs and benign domains. Second, we identify domains that are generated by a previously known DGA but with a new seed, and domains potentially generated by new DGAs. Third, if a new DGA or a new seed is identified, we manually reverse-engineer the corresponding sample, extract the seed and the algorithm, and compute the set of domains it generates. In the following, we discuss these steps in detail.

Filtering. We first filter out known benign and popular domains by comparing them against the first 10,000 entries of the Alexa list of top-ranked domain names [10]. Second, we filter out known AGDs by comparing the input domains against our current collection of enumerated AGDs. A good starting point for collecting DGA-based malware samples are malware analysis reports and blogs. Using these as sources and based on our experience, we identified an initial set of 22 families using DGAs. In 9 of these cases, we already found a reimplementation of the DGA that we only had to verify. This enables us to compute an initial set of domains to use in the DGA filter.

Identification. In this step we identify samples that implement a previously known DGA but use new seeds, and potentially new DGAs. To that end, we use a set of *detectors* implemented as regular expressions to quickly decide for a given domain if it matches the expected output of one of these known DGAs. The regular expressions catch major characteristics such as minimum and maximum length of the generated part (L_{min} and L_{max}), DGA alphabet Σ, and the set of known TLDs. If a majority of input domains is matched by the same pattern, we mark the sample as using a known DGA with new seed.

If the detectors produce an inconsistent or insufficient result, we compute a *collective DGA score* for the input domains to measure the likelihood of these domains being generated by a DGA. This score is based on features such as n-gram frequency, entropy, and length analysis. If available, we further increase the score in case of an *NX domain* result. This approach relies on previous work on DGAs [13, 18, 34, 43, 54]. If the score exceeds a threshold derived through prior experiments and manual verification, we mark the sample as a candidate for using a new unknown DGA.

DGA enumeration. Whenever the system identifies a potentially new DGA, we manually verified this by reverse-engineering the corresponding sample. If present, we then extracted the domain generation logic from the binary, reimplemented it, identified the used seed, and finally computed all domains generated by the algorithm. Moreover, by analyzing the algorithm we updated the set of detectors to identify future samples that use the same algorithm with different seeds.

As input to our system, we used a special sandbox feed that was kindly provided by the Shadowserver Foundation, consisting of timestamp, malware sample hash, DNS query, and DNS response. This feed contains a mix of both newly observed malware samples and older samples undergoing re-processing, dating back to at least 2009. This re-processing increases the likelihood to encounter unknown seeds and families that may no longer be active. Taking data of 3 months starting in May 2015 from this feed, we examined a total of 1,235,443 sandbox runs, performing a total of 15,660,256 DNS queries towards 959,607 unique domain names. Based on the data of the Shadowserver feed, our system enabled us to identify, analyze, and re-implement another 21 DGAs and the majority of seeds listed in Table 3.

3.2 Reimplementing DGAs

To ensure our reimplementation is correct, we compared its output against the domain queries issued by the respective malware sample when executed in a sandbox. For each family, the reverse-engineering and reimplementation of the corresponding DGA took around one day. After analyzing the algorithm, we extracted the DGA seed from the binary. In many cases, we fully automated the process of seed extraction, which enabled us to easily extract seeds from new binaries of the same family. After unpacking the malware binaries, we rarely encountered further protection layers of the original unpacked code, which allowed considerable analysis speed. Nymaim and Suppobox samples are heavily obfuscated and employ family-specific code obfuscations. To handle these cases, we analyzed the obfuscation techniques and implemented custom deobfuscators to automatically deobfuscate these samples. Pykspa 2 was an exceptionally difficult case. Instead of reverse-engineering the seed derivation function, it was much easier to instrument it in order to generate all possible seed values that we later used in the reimplementation of the DGA.

The manual analysis of samples previously identified by the system helped us to eliminate several samples that would otherwise be false positives. For example, during the evaluations of the sandbox DNS feed, we came across many samples whose DNS queries appeared like typical AGDs but were in fact hardcoded domains as verified through our reverse-engineering. Furthermore, we only consider DGAs in which the AGDs are actually used as valid, potential C&C endpoints and not as distractions like in the following cases. For instance, we found an early variant of Sality [1] prepending dynamically generated third-level parts to hardcoded domains, which is the earliest hint to DGA-like behavior we were able to identify. Another example is the Zeus-derivate Citadel, which produces unused decoy AGDs when having detected a virtualized environment [21].

We are aware that more malware families than listed in this paper are potentially using DGAs. However, we believe that the given data set provides a sufficient basis to draw meaningful conclusions on a majority of effects caused by domain generation algorithms.

4 Insights into the DGA Landscape

In this section, we present a comprehensive overview of the 43 DGA implementations and their 253 seeds that we collected. We use our taxonomy to characterize and compare the studied DGAs. More specifically, we compare domain structure, validity periods, and generation schemes. We also study how random the generated domains are and the priority of the DGA as C&C communication mechanism. Table 1 presents an overview of the different DGA features. The table includes different variants of some families with different DGAs. This is the case for Gameover Zeus, Murofet, Pushdo, and Pykspa.

Name	Reference	DGA Type	$C2_{prio}$	Valid	$\lvert D_{cycle} \rvert$	L_{min}	L_{max}	TLDs	$\lvert\Sigma\rvert$	H_{rel}
Bamital	[37]	TDD-H (MD5)	1/1	1d	104	32	32	4	16	1.000
Banjori	[14]	TID-A	2/2*	∞	2,196-15,373	11	26	1	26	0.948
Bedep	[44]	TDN-A	1/1	7d	22-28	12	18	1	36	0.944
Conficker	[25]	TDD-A	2/2	1d	250-50,000	4	11	123	26	1.000
Corebot	[14]	TDD-A (NR LCG)	2/2	1d	40	12	23	1	34	1.000
CryptoLocker	[3]	TDD-A	1/1	1d	1,000	12	15	7	25	1.000
DirCrypt	[14]	TID-A (PM LCG)	1/1	∞	30	8	20	1	26	0.999
Dyre	[5]	TDD-H (SHA256)	3/3	1d	1,000	34	34	8	36	0.805
Feodo	-	TID-A (NR LCG)	1/1	∞	64	16	18	1	26	0.993
Fobber	[47]	TID-A (own LCG)	1/1	∞	1,000	10	17	2	26	1.000
Gameover DGA	[14]	TDD-A (MD5)	1/1	1d	1,000/10,000	20	28	4	36	0.983
Gameover P2P	[11]	TDD-A (MD5)	2/2	1-7d	1,000	11	32	6	26	1.000
Geodo	[30]	TDD-A	1/1	900s	time-based	16	16	1	25	1.000
Gootkit	[48]	TDD-A (PM LCG)	1/1	12h	1	16	16	1	26	0.997
Gozi	[4]	TDD-W (NR LCG)	1/1	1-3mo	5-80	12	24	12	26	0.883
Hesperbot	[27]	TID-A	2/2	∞	50-64	8	24	1	26	0.997
Kraken	[41]	TID-A	1/1	∞	300	6	11	4	26	0.998
Matsnu	[49]	TDD-W	2/2*	3d	3	12	24	1	27	0.895
Mewsei	[14]	TDD-A (MS LCG)	1/1	16d	64	8	15	1	23	0.939
Murofet 1	[14]	TDD-A (MD5)	1/1	1d	1,020	8	16	5	26	0.965
Murofet 2	[14]	TDD-A (MD5)	1/1	1-7d	1,000	32	47	6	36	0.994
Necurs	[14]	TDD-A	2/2	4d	2,048	7	21	43	25	1.000
Nymaim	[15]	TDD-A (Xorshift)	2/2*	1d	30	6	11	8	26	1.000
Pushdo	[6]	TDD-A (MD5)	2/2	60d	30	8	12	2	26	0.962
Pushdo TID	-	TID-A (NR LCG)	1/1	∞	6,000	10	10	5	26	1.000
Pykspa 1	[14]	TDD-A	1/1	2d	5,000	6	15	6	26	0.963
Pykspa 2	[14]	TDD-A (own LCG)	1/1	1-20d	1,000	6	12	4	26	0.998
QakBot	-	TDD-A (MT, CRC32)	1/1	8-11d	5,000	8	25	5	26	1.000
Ramdo	[29]	TID-A	1/1	∞	1,000	16	16	1	13	1.000
Ramnit	[46]	TID-A (PM LCG)	2/2	∞	1,000	8	19	1	25	1.000
Ranbyus	[14]	TDD-A	1/1	28-31d	40	14	14	8	25	1.000
Redyms	[32]	TID-A	1/1	∞	34	9	15	1	27	0.990
Rovnix	-	TID-A (MS LCG)	2/3	∞	10,000	18	18	5	34	0.999
Shifu	[24]	TID-A (own LCG)	2/2	∞	777	7	7	1	25	1.000
Simda	[14]	TID-A	1/1	∞	1,000	5	11	4	26	0.965
Suppobox	[22]	TDD-W	1/1	12h	168	8	26	1	26	0.889
Szribi	[52]	TDD-A	2/2	1-3d	4	8	8	1	15	0.949
Tempedreve	[7]	TID-A (own LCG)	1/1	∞	204	7	11	4	26	0.996
TinyBanker	[14]	TID-A	2/2*	∞	100-4,000	12	12	15	25	0.987
Torpig	[50]	TDD-A	1/1	1d	3	7	9	3	30	0.937
UrlZone	[14]	TID-A	2/2*	∞	2,000	9	15	2	32	1.000
Virut	[20]	TDD-A (Delpi LCG)	2/2	1d	100*100	6	6	1	26	0.984
VolatileCedar	[17]	TID-P	2/2	∞	170	14	14	1	9	0.959

Table 1: Overview of studied DGA implementation characteristics. *Type* according to our taxonomy. $C2_{prio}$ lists the priority of DGA among all C&C rendezvous mechanisms found (with * indicating hardcoded domains being redundant or used as part of seed). *Valid* describes the duration and D_{cycle} the number of domains generated per period. $L_{min,max}$ denotes extrema of domain length, *TLDs* is a combined value over all seeds of this DGA. Σ is the alphabet used in AGDs, H_{rel} normalized entropy.

4.1 Domain Structure

Alphabet. We first study the alphabet, denoted by Σ, used by the DGAs to generate domains. The alphabets contain between 9 and 36 characters. While some DGAs use intentionally short alphabets, the majority of DGAs with small alphabets seems to result from flawed implementations. An example of the first case is VolatileCedar, which generates domains by permuting the second-level part of a hard-coded domain (`dotnetexplorer.net`). Ramdo's DGA is an example of a buggy implementation. Only letters with odd indexes are chosen when building the domain name (i.e., `a, c, e, ...`), resulting in an alphabet with only 13 letters. Hash-based DGAs have alphabets of 16 characters consisting of digits and letters from `a` to `f`.

Many DGAs use a hard-coded array of characters from which is chosen by an index computed iteratively. A common bug in these DGAs are *off-by-one* errors, where one or more characters are never chosen. For example, Geodo omits the last character of its alphabet, thus generated domains never contain a `z`. Other DGAs with this bug include CryptoLocker, Necurs, Ramnit, Ranbyus, Shifu, TinyBanker, and Torpig. They all miss the last character in their alphabet. This type of error is worse for DGAs that use multiple separate arrays to choose characters from. For example, Mewsei uses two arrays, one for vowels and one for consonants (intentionally or not, missing the letter `j`). An off-by-one bug results in one character in each array to be missed, reducing the effective alphabet size by two. Rovnix and Corebot use two separate arrays of letters and digits and suffer from the same malfunction, causing their AGDs to never contain a `z` or `9`. We found this bug in 11 of the studied DGAs.

We also observed *truncation* errors, where the alphabet is truncated into a smaller array that is used for computing the domains. For example, Szribi truncates its 26 intended hard-coded letters to only 15. The same error occurs in Urlzone, reducing the possible 35 characters to only 32. Torpig suffers from both bugs: Torpig's randomness is flawed in a way that only 30 out of 34 possible (36 minus 2 resulting from the off-by-one error) are reachable. The largest alphabet of 36 symbols is used Dyre, Gameover DGA, and Murofet 2.

AGD length. The length of generated domains ranges from 4 to 47 with median minimum length 9 and median maximum length 16. 14 DGAs produce AGDs that all have identical length. In three cases (Banjori, Simda, and Torpig), we found multiple length values across seeds but all AGDs of one seed have the same length.

Domains levels. Only three families (Corebot, Kraken, and Mewsei) generate third-level domains, using one or more Dynamic DNS providers. All remaining DGAs only generate a second-level domain that is then concatenated with a top-level part. Note that we consider constructs as `co.uk` or `com.tw` as top-level part since they are managed by a single registry. Conficker and Necurs make extensive use of TLDs with 123 and 43 TLDs respectively. These cases strikingly illustrate the need of global cooperation to successfully sinkhole some botnets. This also holds for botnets with DGAs that use a smaller set of TLDs. Table 2 shows the most popular TLDs used by the studied DGAs and identified seeds. The 7 most common TLDs are the same in both listings and they only differ in their order. `com` and `net` are the two top TLDs. We observe a trend to use popular TLDs with no regional reference, as `com` and `net` together make up about 45% of all registered domains [8]. We assume that attackers want their generated domains to blend in well with benign traffic.

per DGAs		per Seeds	
TLD	Occurrences	TLD	Occurrences
com	28	com	178
net	21	net	82
org	16	ru	56
info	15	biz	40
biz	13	in	40
ru	10	info	32
in	6	org	29
cc	5	pw	26
su	5	su	21
eu	4	cc	18

Table 2: Popularity of TLDs, both on for DGAs and seeds.

4.2 Domain Validity Periods

More than half of the studied DGAs (24/43) are time-dependent, meaning that the generated domains are only valid for a certain period of time. Most of these DGAs (21/24) generate domains with disjunct validity periods. That is, only a single set of domains is valid at each point in time. The three exceptions are 1) Matsnu, which generates 3 domains which are each valid for 3 consecutive days, meaning that there are 9 potential C&C domains at a time; 2) Pushdo, which will generate domains relative to a given date, starting 45 days in the past and up to 15 days in the future. With 30 domains per day, this gives 900 domains valid at a time; and 3) Suppobox, producing one AGD per 512 seconds, which is then valid for the next 85 periods, totalling to 12 hours, 5 minutes, 20 seconds per AGD.

11 time-dependent DGAs generate domains that are valid for one day. Other families increase their domain validity by performing certain calculations on the date. For example, both Gameover P2P and Murofet 2 round the day of month down to the next lowest value of 1, 7, 14, 21, or 28, resulting in validity periods of variable length (1-7 days). Qakbot employs a similar scheme but uses different values for rounding (1, 11, and 21). The domains generated by Szribi's DGA [52] are valid for 1-3 days. This variable period length appears to be a bug since the DGA tries to round the day to the nearest third day (i.e., Julian days).

Both versions of Pykspa have unique characteristics with regard to the validity of generated domains. Pykspa 1 generates a list of 5000 domains every two days, and the validity of each domain depends on its position in this list. The first 20 domains are generated at random while the remaining 4,980 are chosen from two static sets of domains, alternating between these sets every two days. This is caused by the DGA raising the initial 32-bit seed to power of 2 for each new computed domain, thus drastically reducing the randomness of computed domains. In order to increase resilience to detection, Pykspa 2 generates *fake* domains with shorter validity periods than the real ones. While 200 real domains are generated that are valid for 20 days, the DGA generates 800 fake domains, each of which is only valid for one day.

Geodo has a unique generation strategy with regard to validity periods among all time-dependant DGAs. It queries the current date from the response of an HTTP request to a Microsoft website. Then, it will consequently generate one AGD every 900 seconds starting from a hard-coded start date until this current date is reached.

4.3 Generation Schemes

Arithmetic-based DGAs are by far the most common generation scheme (37/43). Among these, 26 DGAs directly compute the ASCII codes of the characters to be used in the domain, while 11 DGAs compute an index that is used to select characters from hard-coded arrays representing the used alphabet. Three DGAs are wordlist-based: Matsnu, Suppobox, and Gozi. Matsnu and Suppobox embed the list of words in their corresponding malware binary. On the other hand, Gozi extracts its wordlist from a publicly available text file, which is very unlikely to be changed in the future. For example, it uses the United States Declaration of Independence to generate domains such as `amongpeaceknownlife.com`. Matsnu and Gozi randomly combine words until a certain length is reached, while Suppobox only combines 2 of its 384 included words and adds `.net` as TLD.

The two families Bamital and Dyre are hash-based and use the hexdigest output of hashing functions MD5 and SHA256 over date and domain index as input parameters. VolatileCedar is the only permutation-based family, which can produce 170 possible permutations of the initial domain. It is worth mentioning that Banjori, TinyBanker, Urlzone and VolatileCedar use a domain mutation scheme where either a seed or a previously generated domain is used as input for the calculation of the subsequent domain.

4.4 Domain Randomness

Given that various DGAs make use of PRNGs in their domain generation schemes, we analyzed the randomness of generated domains. To this end, for each DGA, we compute the global string by concatenating all generated parts of the domains. Then we calculate the Shannon entropy of this string. This gives a global indicator for the randomness of the algorithm. Finally, we compute the relative entropy, denoted by H_{rel}, by dividing the calculated value with the maximum entropy. This enables us to compare the relative entropy of different DGAs. For simplicity, we use the term entropy to refer to H_{rel}.

We make the following observations. All wordlist-DGAs have a significantly lower entropy ($H_{rel} < 0.9$), which is expected since they combine complete words whose characters are not uniformly distributed. However, the lowest entropy is observed in Dyre, a hash-based DGA. While the SHA256 algorithm produces uniformly distributed characters, Dyre prepends an extra character in the range `[a-z]` to the calculated hexdigest, thus destroying the uniform distribution since the hexdigest does not contain any letters in the range `[g-z]`.

The low entropy ($H_{rel} < 0.99$) observed in other DGAs is a result of specific implementation choices. We discuss this in the following.

1) **Multiple sub-alphabets**. By splitting the alphabet into multiple distinct lists and randomly drawing characters from these lists for different positions in the computed domain name, some DGAs cause an imbalance on the overall distribution of characters. This is the case for Bedep, Gameover DGA, Mewsei, Pushdo, Simda, Torpig, and Virut. Redyms uses a hyphen in every AGD.

2) **Imperfect PRNG**. We observed several causes of imperfect PRNGs: first, some DGAs (Pykspa 1 and Szribi) use their own self-designed PRNGs, which have imperfect randomness. Second, other DGAs (Murofet 1 and TinyBanker) impose certain conditions on the output of the used PRNGs, and thus cause an imbalance in the distribution of derived characters.

3) **Partial modifications**. Some DGAs compute the next domain based on an initial domain name. This transformation may impact the DGA entropy if it does not introduce enough modifications in the initial domain seed. An example of this category is Banjori, which only modifies the first four positions of a given hard-coded seed domain.

It is worth mentioning that several DGAs use well-known PRNG algorithms. Notably, almost a third of the studied DGAs (14/43) use a linear congruential generator (LCG) [36] defined by the recurrence relation $X_{n+1} = (aX_n + c) \bmod m$. By matching the used parameters a, c, and m against those of common LCGs, we found that 10 DGAs use known LCG constants: 4 DGAs use the implementation described in the Numerical Recipes book [39] (NR LCG), 3 DGAs use the *minimal standard* implementation proposed by Park and Miller [36] (PM LCG), 2 DGAs use the Microsoft Visual C library (MS LCG), and one DGA uses the Delphi LCG. The remaining 4 LCGs use self-chosen constants. Other uses of well-known PNRGs we identified are Mersenne Twister [33] and Xorshift [31]. Five DGAs use MD5 for initializing their own PRNG and one uses the checksum CRC32.

4.5 Command & Control Priority

More than half of the studied botnets (23/43) use their DGA as the only C&C rendezvous mechanism. Moreover, although 5 other families (identified by a star in Table 1) first try hard-coded domains before turning to their DGAs, the DGAs can also be considered as the primary communication mechanism. Banjori, TinyBanker, and UrlZone first try a single hard-coded domain that is later used in computing the initial seed of the corresponding DGA. Matsnu contains multiple prioritized hard-coded domains. However, the same domains are also generated by its DGA at some point in time. Nymaim contains a *legacy* primary hard-coded domain that is long mitigated and probably disregarded by the botmasters, meaning that it mainly relies on its DGA instead.

Other families use their DGAs as a backup mechanism when their primary communication method (usually hard-coded domains) fails. For example, Rovnix tries to connect to one of its hard-coded domains. If this fails, it resorts to its DGA before finally trying a connection via Invisible Internet Project (I2P). The remaining families use their DGAs as a last resort to reach the C&C server.

This clearly shows the prevalence of DGAs in modern botnets. For this reason, we believe that DGAs should be no longer perceived mainly as a backup mechanism but instead as a primary C&C concept.

5 DGA Domain Usage

In this section, we present the results of our analysis on how DGAs are actually used in practice. We base our analysis on WHOIS data for the last 14 years, and give insights into the family activity periods, registration status of DGA domains, and botmaster registration strategies. We also analyze the mitigation response time.

5.1 WHOIS Dataset

We based our analysis on the DomainTools WHOIS dataset, which contains over 9 billion WHOIS records collected over the last 14 years [19]. Based on our reimplementations and the available seeds, we computed all possible domains generated by the studied 43 DGAs. We provided the computed domains to DomainTools, and they kindly provided us with WHOIS records for DGA domains they found in their dataset. The data provided by DomainTools for this study was compiled on the 22nd September of 2015. This enabled us to identify 303,165 DGA-related WHOIS records for 115,387 domains.

A special care had to be given to time-dependant DGAs, which use the date information for computing their domains. To ensure a good coverage and minimize the set of DGA domains we miss, we computed all domains that cover a time period of one year before any publicly known starting date until 31st December 2015 (three months after the last WHOIS record in the DomainTools dataset). This end date provides around 3 months of *lookahead*, detecting domains that are registered for some time before being actually used.

These WHOIS records contain the following fields

- Date of the WHOIS record
- Date of updates to the WHOIS record
- Dates for domain creation and expiration
- Registrar name
- Registrant name and e-mail address
- Nameservers registered in WHOIS

This enabled us to infer the following information:

1) *Domain first registration*. This is based on the date of the WHOIS record and date of domain creation.

2) *Start of botnet activity*. The first record for any domain implicitly indicates that the domain is registered and helps to estimate when the corresponding malware family started its operation.

3) *Changes in domain ownership*. The date of updates to a WHOIS record often corresponds to a change in responsibility for a domain. This can be verified by accompanying changes to the registrar, registrant, and registered nameserver fields. This allows us to derive when a potential C&C domain has been mitigated (e.g., sinkholed), by observing changes of the WHOIS record from a non-sinkhole operator to a sinkhole operator. We identify 29 different organizations running sinkholes, but we will not disclose further details about the features used to detect sinkhole organizations to protect their operations.

4) *Parked domains*. The registrar, registrant, and registered nameserver fields convey information that we use to identify parked domains in order to investigate how common AGDs are held for the purpose of picking up accidental traffic or reselling. In total, we identify 36 services offering domain parking or domain reselling.

Our dataset covers all but five of the studied families: Conficker and Virut generate an exceedingly large number of domains, and were therefore not included in the dataset provided to us. Corebot, Kraken, and Mewsei use DDNS-based DGAs, and are thus not available in the original DomainTools WHOIS dataset, which only contains primary domain names but no subdomains. It is noteworthy that non-deterministic seeding as used by Bedep does not prevent retroactive analysis, as we are still able to generate all domains by collecting the used exchange rates over time.

5.2 Family Activity Periods

In this section, we analyze the period of time in which the studied botnets were active and domains were registered for malicious purposes. To this end, we first identify the date of the first related record in our dataset, denoted by T_{first}. In many cases, this is the first observed registration overall. In cases where the data indicates collisions with benign domains (cp. Section 5.4), we analyze records from the first sinkhole event backwards. Then, we identify the time when the domains were either taken down or they were registered last, denoted by T_{last}. The detailed activity periods for the studied botnets are shown in Table 3. For families we did not have data for, their respective T_{first} is estimated from public sources as mentioned before.

In case of Bamital, CryptoLocker, and Gameover P2P, T_{last} is the date of their takedown, marked by a †. For Conficker, T_{last} identifies the date on which the last known Conficker variant deleted itself from compromised systems [26]. A special case is Rovnix, for which

Name	T_{first}	T_{last}	$\|S\|$	$\|D_{gen}\|$	$\|D_{uniq}\|$	$\|R_{all}\|$	$\frac{\|R_{all}\|}{\|D_{uniq}\|}$	$\|R_P\|$	$\|R_M\|$	$\|R_S\|$
Kraken	2007-07*	-	1	300	300	-	-	-	-	-
Torpig	2008-01	2011-06-22	2	17,610	17,610	139	0.79%	2	1	57
Szribi	2008-11	2011-06-22	1	4,396	2,949	54	1.83%	0	8	40
Conficker	2008-11*	2009-05-03†	3	129,807,750	125,118,625	-	-	-	-	-
Pushdo TID	2009-07	2012-04-06	1	6,000	6,000	245	4.08%	0	0	0
Pykspa 1	2009-10	2012-09-09	1	32,920	22,764	455	2.00%	12	0	49
Gozi	2010-01	2015-09-18	9	21,890	16,963	305	1.80%	48	4	143
Murofet 1	2010-08	2011-09-08	2	4,063,680	4,063,680	3,172	0.08%	0	50	369
Bamital	2010-11	2013-02-06†	1	197,600	197,600	8,340	4.22%	0	150	30 (7,891)
Nymaim	2011-06	2015-09-16	3	277,112	65,040	656	1.01%	70	17	388
Simda	2011-06	2014-11-06	12	13,000	11,528	379	3.29%	66	9	44
Ramnit	2011-06	2015-02-07	18	18,000	18,000	939	5.22%	0	126	372
Virut	2011-08*	-	1	16,140,000	15,355,008	-	-	-	-	-
Murofet 2	2011-09	2011-12-20	1	262,000	262,000	559	0.21%	0	4	261
Gameover P2P	2011-09	2014-05-28†	1	262,000	262,000	74,755	28.53%	0	23	391 (72,713)
Feodo	2012-02	2012-10-06	3	192	192	110	57.29%	0	1	9
Gootkit	2012-06	2013-11-08	1	2,190	730	198	27.12%	0	0	4
Redyms	2012-12	2014-02-10	1	34	34	11	32.35%	0	2	2
Necurs	2013-01	2015-06-12	6	3,551,232	3,551,232	295	0.01%	10	0	158
CryptoLocker	2013-01	2014-05-30†	1	1,108,000	1,108,000	3,820	0.34%	0	341	240 (2,899)
Suppobox	2013-02	2015-09-20	3	545,169	98,304	11,338	11.53%	8,434	19	792
Banjori	2013-03	2013-09-10	30	434,556	421,390	683	0.16%	0	3	33
Pushdo	2013-03	2015-08-05	4	124,080	124,021	453	0.37%	3	0	54
Pykspa 2	2013-04	2013-10-01	2	775,400	775,342	1,927	0.25%	757	5	101
VolatileCedar	2013-04	2015-03-30	1	170	170	13	7.65%	0	0	7
DirCrypt	2013-07	2014-06-15	14	420	420	86	20.48%	0	13	21
Hesperbot	2013-07	2015-01-07	3	178	178	15	8.43%	0	1	10
Ramdo	2013-10	2014-05-03	3	3,000	3,000	47	1.57%	0	5	23
UrlZone	2013-11	2015-09-20	6	12,006	10,009	127	1.27%	0	24	34
QakBot	2013-12	2015-09-20	1	385,000	385,000	1,088	0.28%	0	61	35
Matsnu	2014-01	2015-09-20	2	3,375	3,346	610	18.23%	244	33	61
Dyre	2014-06	2015-08-19	1	592,000	592,000	850	0.14%	0	1	273
Gameover DGA	2014-07	2014-11-21	2	6,182,000	6,182,000	1,081	0.02%	0	14	549
TinyBanker	2014-08	2015-09-21	90	84,291	81,930	1,733	2.12%	0	272	326
Geodo	2014-10	2014-11-16	2	90,240	90,232	107	0.12%	0	0	39
Tempedreve	2014-10	2015-04-19	1	204	204	20	9.80%	0	0	13
Mewsei	2014-10*	-	1	1,984	1,984	-	-	-	-	-
Fobber	2014-10	2015-07-01	2	2,000	2,000	13	0.65%	0	2	4
Ranbyus	2015-01	2015-08-10	7	105,840	64,400	98	0.15%	0	0	36
Rovnix	2015-01*	-	1	10,000	10,000	1	0.01%	0	0	1
Bedep	2015-02	2015-09-20	4	3,906	3,806	654	17.18%	0	10	201
Corebot	2015-06*	-	2	18,160	18,160	-	-	-	-	-
Shifu	2015-07	2015-09-10	2	1,554	1,554	11	0.71%	0	0	8
Aggregated	-	-	253	165,161,439	159,712,234	115,387	0.63%	9,646	1,199	5,177 (83,503)

Table 3: Overview of DGA usage characteristics. $|S|$: seeds known for this DGA. D: domains generated until 31.12.2015, R: registered domains ($|R_P|$: prior to T_{first}; $|R_M|$: turned into a sinkhole; $|R_S|$: directly registered as sinkhole; in brackets: related to botnet takedowns). Registration percentage based on D_{uniq^*}, thus considering only AGDs where registration data was available.

we only identified a single registration by a sinkhole operator, and thus decided to leave T_{last} open.

After identifying the starting date for each family (T_{first}), we removed the AGDs we computed for earlier points in time. This reduces the computed set to 165,161,439 AGDs in total, 159,712,234 of them being unique across all DGAs. Additionally, we remove the computed domains for the families not available in the DomainTools dataset, leaving us with a set of 18,446,125 unique AGDs. We denote this set by D_{uniq^*} and use it for all following examinations.

Having a closer look at T_{first} shows that although the concept has been first used in 2007, more than half of the analyzed DGAs (25/43) were introduced 2013 and later. It seems like malware authors waited for early adopters to gather experience with running DGA-based botnets before trying it out themselves. Cases with short activity times indicate that some authors seemingly also only experimented with DGAs. For example, while early versions of Geodo used hard-coded IP addresses for C&C communication, one version with a DGA was used for only about 6 weeks. After that, the botmasters returned back to the old communication method.

Murofet 2 remained active for three months only, before its widespread successor Gameover P2P, which used a very similar DGA, appeared. Gameover DGA, another successor that appeared shortly after the takedown of Gameover P2P, was active for only about 5 months. Although binary diffing implies that it relies on the same source code as Gameover P2P, it is uncertain whether it was operated by same original author Slavik [42] or if it was only used as a distracting maneuver.

We use T_{last} to identify active families as of this writing. We consider a family to be active if its DGA generated a domain that was used within the last month of the time period covered by the WHOIS dataset. Based on that, we identify 10 families: Gozi, Nymaim, Suppobox, UrlZone, QakBot, Matsnu, Dyre, TinyBanker, Bedep, and Shifu. This serves as a lower bound for estimating active families since some do not use DGA as their primary C&C rendezvous mechanism.

5.3 Domain Registration Status

Out of the 18,446,125 unique generated AGDs, 115,079 were actually registered (0.62%). 72,37% of these registered AGDs correspond to sinkhole operators, while the remaining 27,63% are non-takedown domains. We divide the set of registered AGDs into four categories:

1) **Pre-registered domains (30.25%)**. Domains that have been registered before T_{first}. These domains were usually registered long before the corresponding botnet appeared, and are very likely benign.

2) **Mitigated domains (3.76%)**. Domains that were originally held by a non-sinkhole registrant but then changed to a sinkhole operator, usually indicating that the domain has been mitigated (Section 5.6).

3) **Pure sinkhole domains (16.24%)**. Domains registered by a sinkhole operator from the start.

4) **Remaining domains (49.75%)**. This set contains domains that were registered between T_{first} and T_{last} but we were not able to reliably identify them as sinkholes, or otherwise tell whether they are benign or malicious (mostly due to the use of a WHOIS privacy service). However, the number of benign domains (at least for non-wordlist DGAs) is highly likely to be a minority, given that few DGAs have pre-registered domains and the common randomness and domain length produced by most DGAs.

5.4 Domain Collisions

In order to evaluate if AGDs are a good feature to reliably identify the corresponding botnet, we analyze collisions between domains generated by different DGAs. Out of the 43 DGAs, the two aggressive DGAs Virut and Conficker have collisions among each other and with Necurs, Nymaim, and Pykspa 2. More specifically, Virut has 1791 collisions with Pykspa 2, 1316 with Conficker, and 179 with Nymaim. Conficker has 11 collisions with Pykspa 2, 4 with Necurs, and 1 with Nymaim. All of the colliding domains are 5 to 6 characters long. Apart from that, only Nymaim and Pykspa 2 have a single collision since they both generate the AGD `wttttf.net`.

Next, we analyze collisions between DGA-generated domains and benign domains. To this end, we use the Alexa list [10] and compare it to the set of domains generated by each DGA. Again here, only a small number of collisions were found. Virut has 2507 collisions with Alexa top million domains, which corresponds to 0,016% of its unique AGDs. For all other DGAs, we only found 27 collisions with the Alexa list: 24 collisions with wordlist-based DGAs (Suppobox: 21, Matsnu: 2, Gozi: 1), and three collisions with Pykspa 2 domains that are 6 character long.

We then investigated the collisions between DGA-generated domains and domains that were already registered before the corresponding botnet appeared (before T_{first}). As one would expect, the highest number of collisions were observed with wordlist-based DGAs: 74.39% of Suppobox AGDs and 40.0% of Matsnu AGDs collide with already existing domains. In spite of being a wordlist DGA, Gozi has a low number of collisions with already registered domains. This is because the DGA truncates, with a probability of 33%, a chosen word before appending it to the computed AGD. This creates many domains that contain a *broken* word, making accidental registration far less likely. For the remaining families, 856 of 928 (92.24%) pre-registered AGDs have length 5 to 6, and everything longer is accidentally a word (e.g. `veterans.kz`) or otherwise very pronounceable (e.g. `kankanana.com`, `kandilmed.com`).

These results clearly show that DGAs not based on wordlists and generating domain names longer than 6 characters (which is the case for 34/43 DGAs) serve as a reliable source to detect the corresponding botnet family.

5.5 Domain Registration Lookahead

A special property of AGDs from time-dependent DGAs is, that they are only valid during certain periods of time. In this section, we give insights into the domain registration *lookahead* for time-dependant DGAs, i.e., the amount of time from registering the domain until the point in time where it becomes valid (produced by the DGA at that time). Here, we distinguish between sinkhole and non-sinkhole registrations (potentially by botmasters).

Figure 2 provides an overview of our results for the 22 time-dependant DGAs in our dataset. For each family, the results are summarized using a boxplot showing the registration times of DGA domains relative to the start of the corresponding validity periods (day zero). That is, a registration event at $x = -1$ means that the corresponding domain was registered one day before it became valid. For better visualization, we choose a symmetric mixed linear (for the first 10 days) and logarithmic (everything beyond) scale. This allows best to display data with a finer resolution around the majority of validity periods but also allows to include events up to a thousand days away. For better orientation, the red bars indicate the whole validity period of AGDs per family. In this Fig-

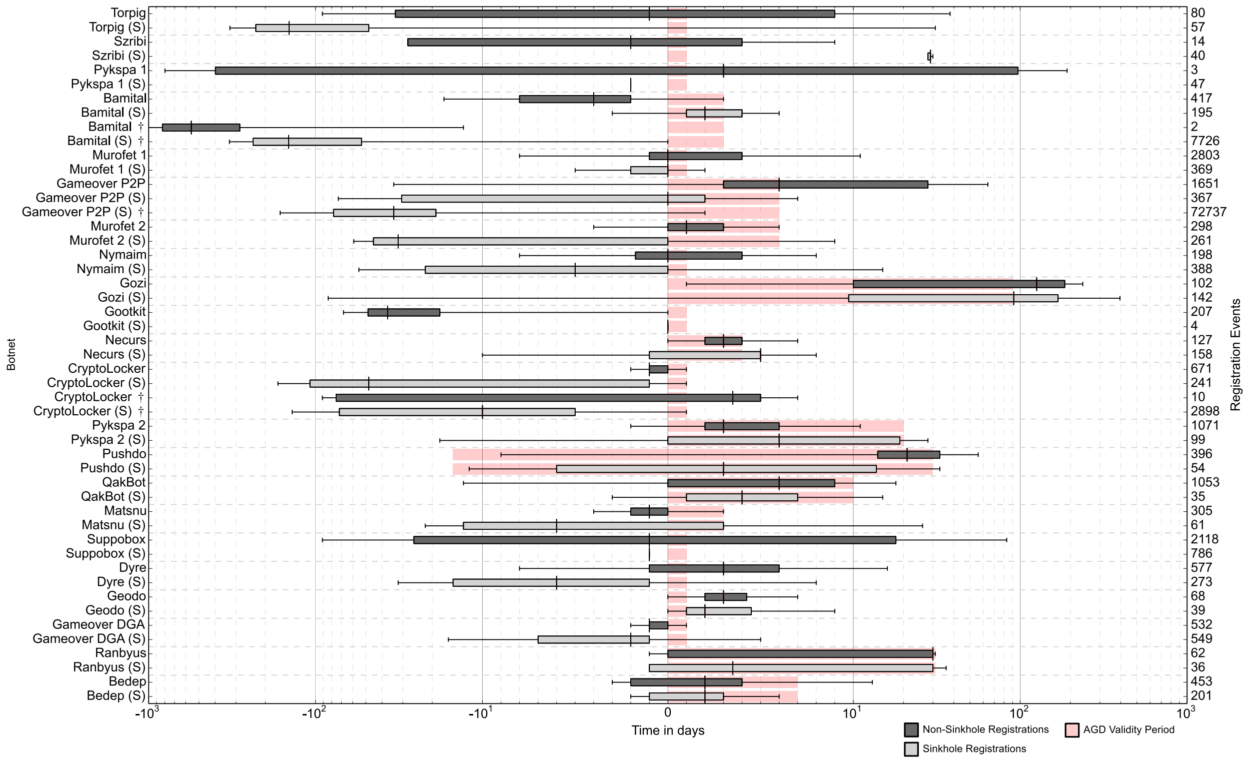

Figure 2: Lookahead of domain registrations in time-dependent DGAs, divided into identifiable sinkholes and remaining domains. The data in this boxplot is applied relative to the start the respective AGD's validity period (shown in light red for better orientation). Bamital, GameoverP2P, CryptoLocker data is further divided into pre and post takedown (indicated by †).

ure, we distinguish between four possible categories for each family: 1) none sinkhole registrations; 2) sinkhole (marked by S) registrations; 3) pre-takedown registrations; and 4) post-takedown registrations (marked by †). Note that the registration numbers may vary from those in Table 3. This is a result of plotting data after T_{first} with respect to validity start instead of the registration dates.

The main observation is that for 14 of 22 DGAs, sinkhole operators registered the domains earlier than non-sinkholers. Comparing the medians of these cases reveals that in 7 cases sinkhole registrations happen between 1 and 10 days earlier, while in the remaining 7 cases they happen between 18 and 143 days earlier. The 143 days are for Torpig, where the majority of sinkhole domains were registered 5 months in advance with close validity periods, which seems to be a takedown attempt. The second highest difference (48 days) was for CryptoLocker. For 3/22 DGAs, sinkhole and non-sinkhole registrations happened at the same time. In the remaining 5/22 DGAs, non-sinkhole registrations occurred between 2 and 37 days before sinkhole registrations.

In the case of Gootkit, a single sinkhole operator registered DGA domains on the same day when they became valid, indicating a reactive response to these domains being used by the malware but not being taken yet, potentially blocking the botmaster from registering them.

Sinkhole operators usually do not fear that their domains will be taken away by other entities, which explains the general trend of sinkholers registering domains before botmasters. Moreover, this enables them to evaluate incoming traffic for these domains even before they become valid. On the other hand, botmasters have to assume that their domains will eventually be mitigated. As a result, they register their domains shortly before or inside the validity period in order to ensure the availability of these domains when the bots are expected to try to contact them. This means that constant monitoring of AGDs with validity periods around the takedown date is a necessary condition for success.

For the three families with takedowns, sinkhole registrations happened at least weeks and often months prior to validity of the AGDs. This ensures that botmasters cannot quickly regain control of their botnet by registering further AGDs. While the takedown of Gameover P2P's AGDs was seemingly complete, AGDs for both Bamital and CryptoLocker were registered by non-sinkholers after the takedown. In case of CryptoLocker, these domains were registered shortly before their validity started, indicating that not all domains remain blocked through the same central authority.

We observed an interesting pattern in the case of Bamital. On January 1st, 2011, AGDs for January 4th, 2012, January 3rd, 2013, and January 4th, 2014 domains were registered by the same registrant. We believe that this was done by the operators of Bamital in an attempt to timely secure *insurance domains* to be used as backup in case of a later takedown, speculating that these domains do not get discovered. We identified similar strategies by the botmasters of Nymaim and Murofet. In case of Nymaim, on December 6th, 2012 and December 9th, 2012, two domains becoming valid in 1 and 2 years respectively were registered. For Murofet, on March 12th, 2011 and the 3 following days, 4 domains becoming valid in 1 year, 5 valid in 2 years, 3 valid in 3 years, and 3 valid in 4 years have been registered. All of them use different, apparently fake identities but the same registrar. This shows that potential registrations of AGDs should be checked considerably far ahead during the preparation of a takedown campaign.

The number for sinkhole registrations of Pykspa 1 and Suppobox is impressively small. For these families, most AGDs are registered with the same lookahead by single sinkhole operators over longer periods of time. Registering only a small fraction of available domains indicates botnet monitoring operations by sinkholers. On the other hand, we observed many different entities involved on non-sinkhole registrations of Suppobox domains. Given the comparatively high collision rate with benign domains (Section 5.4), we believe that many of these domains belong to benign registrants. This shows that the Suppobox DGA blends in very well with legitimate domain owners, allowing the botmasters to hide their C&Cs among benign domains.

We observed a few peculiarities with regard to domain registrations. Registrations for Gozi tend to be late for both sinkholers and others. We found no coherent explanation for this. Another oddity are the sinkhole registrations for Szribi. While 8 registrations happened in 2008 when the botnet was active, another 32 registrations happened in 2015, more than 4 years after the botnet disappeared. Moreover, all of these 32 domains were registered one month after they became invalid. We believe that the respective sinkholing organization used a wrong DGA reimplementation to calculate the AGDs.

5.6 Mitigation Response Time

By mitigation we mean a change of domain ownership from a non-sinkhole to a sinkhole operator. In this section, we analyze the time offset for these events in order to measure the effectiveness of these operations. Here, we only consider DGAs where 10 or more mitigations were identified that are not related to takedowns. Table 4 summarizes the results of this examination. In most cases, the first mitigations against new DGAs or seeds are carried out within a week.

One would expect that after initial identification of a DGA or a new seed for a DGA, all further mitigations would have a much lower response time as the potentially generated domains should be known. We observe relatively short median reaction times $\tilde{m}$ for Murofet 1, Suppobox, Gameover DGA, and TinyBanker. These are also the DGAs where most AGDs were mitigated within their respective validity periods. On the other hand, the countermeasures for Bamital and CryptoLocker have been very ineffective as they mostly targeted AGDs that were no longer valid.

Response times for all other DGAs increase after the first mitigation. Interestingly, multiple follow-up mitigations for these DGAs were carried out on the same day. This pattern corresponds to a common practice by defenders. After identifying initial malicious domains, they look at other domains that point to the same C&C server by performing a reverse IP lookup. This identifies additional AGDs which may have been registered even before the initially identified AGD. This was observed for Nymaim, Ramnit, and UrlZone.

5.7 DGAs and Domain Parking

Next, we examine how many AGDs are registered towards domain parking services. For identification of such domains, we used a procedure similar to Vissers et al. [51]. In total, we found 6,458 AGDs that contained a registration pointing to a parking service at some point in time. This corresponds to the common practice that after the registration period of an original owner ends, the expired domain is transferred to domain parking or picked up by a domain reseller.

To our surprise, for 3,852 of these AGDs, the parking registration was also the first and in many cases only registration entry. In particular, first parking registration events constitute a significant portion of registrations for the following DGAs: Banjori 620 (90.78%), QakBot 595 (54.69%), Pykspa 2 883 (45.82%), Necurs 122 (41.36%), Pushdo TID 91 (37.14%), Ramnit 286 (30.46%), Murofet 1 736 (23.20%). Except for Pykspa 2, none of these DGAs has a significant number of pre-registrations $|R_P|$, which makes unintended or accidental registration extremely unlikely. It is also noteworthy, that for these DGAs, 2,917 (87.52%) domains are registered with the same domain parking service. With regard to validity periods, 464 out of 2336 (19.86%) time-dependent AGDs are registered before the end of the respective validity periods. In this special case, together with the time-independent AGDs, at least 1461 AGDs were potentially

Name	Seeds	Mitigations	Validity Period		First Reponse (in days)				Further Reponses (in days)			
			within	after	m_{min}	$\tilde{m}$	$\bar{m}$	m_{max}	m_{min}	$\tilde{m}$	$\bar{m}$	m_{max}
Murofet 1	1	50	37	13	25	25	25.00	25	0	0	1.77	25
Bamital	1	148	16	132	6	6	6.00	6	3	49	47.37	92
Nymaim	2	16	4	12	1	16	16.00	31	0	5	31.46	212
Ramnit	16	126	-	-	2	19	34.60	153	0	35	54.28	363
CryptoLocker	1	216	25	191	9	9	9.00	9	0	8	14.69	130
Suppobox	2	19	13	6	3	117	117.50	232	0	0	25.12	194
DirCrypt	7	13	-	-	0	3	7.00	26	0	5	10.50	41
UrlZone	4	24	-	-	0	4	6.75	19	0	114	112.40	251
QakBot	1	33	15	18	0	0	0.00	0	0	21	28.41	66
Matsnu	2	33	11	22	2	2	2.50	3	2	7	9.16	72
Gameover DGA	2	14	9	5	0	1	1.50	3	0	1	54.17	161
TinyBanker	51	272	-	-	0	3	6.39	61	0	2	5.66	60
Bedep	2	10	4	6	2	5	5.00	8	1	12	13.12	28

Table 4: Mitigation Response Timings for selected DGAs. For time-dependent DGAs, *Validity Period* describes the identified mitigations for active and outdated AGDs. *First Response* is the time until the first mitigation occurred, with values for minimum, median, average, and maximum, aggregated over seeds. *Further Responses* describes the same measures for all following events.

abused to drive automatically generated traffic from compromised hosts towards a domain parking system.

5.8 Discussion: Countering DGAs

In this section, we summarize the observations made in previous sections and discuss how they influence countermeasures against DGAs.

Looking at the distribution of domain generation schemes (Section 4.3), only 3 of 43 DGAs are based on wordlists and therefore produce somewhat meaningful domain names. On the other hand, methods for the detection of domains that appear randomly generated are well-studied [13, 34, 54] and our data suggests that they remain relevant and applicable in the future.

As the study of activity periods has shown (Section 5.2), DGAs have become very relevant to malware authors, especially over the last 2 years, as 25 out of the 43 considered DGAs surfaced 2013 and later.

One of the core concepts of DGAs is the implied economic asymmetry: A single valid domain grants an attacker control while the defender needs to deny access to all potential domains. The overall low number of actually registered AGDs (Section 5.3) underlines the fact that attackers only need to make sparse use of all potential domains when operating their botnets. Additionally, since most attackers seem to register domains very shortly before their validity or use (Section 5.5), there is a high chance that they can be hit by surprise and the perceived backup utility of DGAs can be cancelled. However, having identified registration events up to 4 years upfront means future domains should be carefully checked to ensure successful takedowns.

Our data indicates that in past takedowns, domains were acquired on a large scale, imposing significant financial efforts connected to the takedown. Having domains registered as consequence of a takedown allows them to be used as sinkholes in order to gather telemetry on compromised systems calling in. However, single AGDs per validity period would be enough to achieve the same monitoring effect, similar to how an attacker only needs a single domain, as long as the remaining domains are not available. Therefore, we propose to raise awareness of the relevance and innerworkings of DGAs to ICANN and make extensive use of blocking AGDs on the level of registry operators. The expected impact of blocking would actually be negligable for the majority AGDs: 34/43 of the DGAs analyzed have AGDs with minimum length 7 or more, for which we observed basically no collisions with existing domains for their entire set of AGDs (Section 5.4), meaning that these domains seem unlikely to be registered for benign purposes anyway. In case of time-dependent DGAs, this blocking could even be lifted once the validity period of the respective AGDs has passed, which could be a compromise to address wordlist-based DGAs.

Furthermore, we identified only 3,302 collisions between 5 DGAs within 159,712,234 unique DGA domains (Section 5.4). This means that a lookup database of AGDs like our data set serves as a very reliable resource to aid the identification of malware families based on contacted domains with basically no false positives. As a service to the community, we continue to collect information on DGAs and provide this data for free through DGArchive [38].

6 Related Work

Among others, the following works have addressed Domain Generation Algorithms in detail. Stone-Gross et al. [50] have performed a botnet takeover for Torpig in early 2009. Based on reverse-engineering, they took advantage of the fact that this family was using a Domain Generation Algorithm without further protection mechanisms in the C&C protocol. Their publication was one of the first descriptions detailing the concept of DGAs in academic literature. Barabosch et al.[16] have defined a taxonomy of DGA types based on the 2 features time-dependency and causality. They also explained a method for the automated localization of DGA-related code using dynamic analysis for unpacking and API tracing in combination with data flow analysis for code extraction. Mowbray et al. [34] have used collected DNS data to identify potential DGA domains examining the query source IP address and length distribution of queried domain names. With this approach they identified 19 different schemes of AGDs and list a subset of characteristics of those given in Table 1.

Several works target the detection of DGAs and other maliciously used domains based on collected network traffic. In 2010, Yadav et al. [54] have evaluated several statistical measures over character distributions and n-grams in domain names in order to detect generated domain names through anomalies. Antonakakis et al. [13] have presented Pleiades, a DNS-based system that detects clusters of potential DGA domains by monitoring unsuccessful DNS requests. As confirmed by our work, they built their system on the assumption that only a fraction of AGDs is actually registered. They were able to find 6 new DGAs that were unknown at the time by evaluating traffic from a large ISP. Another approach using DNS data named Phoenix is proposed by Schiavoni et al. [43]. They have defined a model for pronounceable domains and afterwards detect those domains deviating from this model. Their system also groups identified domains and allows tracking the activity of DGA-based botnets. Bilge et al. [18] have introduced their system Exposure. It is used to detect malicious domain names based on a selection of 15 features observable for DNS traffic, including time-based, DNS answer-based, TTL, and domain name features.

There are also works that explore capabilities of dynamic and proactive blacklisting. In 2009, Ma et al. [28] have compared the usefulness of different information sources for blacklisting. They found out that WHOIS data, especially temporal information of registrations, are very valuable in this regard. Antonakakis et al. [12] have proposed Notos, a system to automatically assign scores to domain names that can be used to automatically generate blacklists. They focus on features of domain strings and TLDs. Xu et al. [53] have mentioned the idea to pre-generate AGDs in order to enable predictive blocking. However, they did not evaluate the effectiveness of this idea, as shown by us in this paper.

Recently, Vissers et al. [51] have investigated the relationship of domain parking services and malicious domains, using DNS and WHOIS data sets. Nadji et al. [35] have developed a concept for effective botnet takedowns in which they identify covering potential DGA presence as important step for effectiveness. With regard to the analysis of further innovations in C&C rendezvous mechanisms, Holz et al. [23] have investigated Fast-Flux Service Networks, while Rossow et al. [40] have performed a survey of botnets using P2P mechanisms.

7 Conclusion

In this work, we presented the first comprehensive measurement study of domain generation algorithms as used by modern botnets. Our study is based on reverse-engineering the DGAs of 43 malware families and variants. Using reimplementations of the algorithms, we generated a collection of 159,712,234 unique DGA domains. We then performed an analysis on domain registrations, utilizing historic WHOIS data provided by DomainTools. Our main findings are that our domain dataset can be used for both predictive blocking of attempted C&C accesses as well as the accurate determination of malware families and campaigns with basically no false positives. Additionally, we characterized the registration behavior of botmasters and sinkholers and examined the effectiveness of domain mitigations.

As a further contribution, we continuously collect further information on DGAs. The full domain data set which results from our work is published for free here: https://dgarchive.caad.fkie.fraunhofer.de. This web service called *DGArchive* offers reverse domain lookups to support malware analysis, as well as forward generation of domain lists, which can be particularly used as blocklists for network protection.

Acknowledgments

The authors would like to express eternal gratitude to the Shadowserver Foundation for continuously supporting malware research. We would also like to thank the anonymous reviewers of USENIX Security as well as Daniela Bennewitz, Arnold Sykosch, and Matthias Wübbeling for their valuable feedback.

References

[1] W32/sality.m, February 2006. Malware description by McAfee: http://www.mcafee.com/threat-intelligence/malware/default.aspx?id=138354.

[2] Conficker Working Group: Lessons Learned. Tech. rep., The Rendon Group, http://www.confickerworkinggroup.org, January 2011.

[3] Ransom Cryptolocker. Tech. rep., McAfee Labs Threat Advisory, November 2014.

[4] Tracking Rovnix, 2014. Blog post: http://labs.bitdefender.com/2014/11/tracking-rovnix-2/.

[5] Chasing cybercrime: network insights of Dyre and Dridex Trojan bankers. Tech. rep., Blueliv, 2015.

[6] Pushdo It To Me One More Time. Tech. rep., Fidelis Cybersecurity, April 2015.

[7] Tempedreve - Botnet overview and malware analysis. Tech. rep., Anubisnetworks, 2015.

[8] The Domain Name Industry Brief - Volume 12, Issue 3. Tech. rep., Verisign, 2015.

[9] TLD DNSSEC Report, 2015. Statistics page published by ICANN: http://stats.research.icann.org/dns/tld_report/.

[10] ALEXA. Top sites on the Web, 2015. Website: http://www.alexa.com/topsites.

[11] ANDRIESSE, D., ROSSOW, C., STONE-GROSS, B., PLOHMANN, D., AND BOS, H. Highly resilient peer-to-peer botnets are here: An analysis of Gameover Zeus. In *Proceedings of the 8th International Conference on Malicious and Unwanted Software (MALWARE)* (2013).

[12] ANTONAKAKIS, M., PERDISCI, R., DAGON, D., LEE, W., AND FEAMSTER, N. Building a Dynamic Reputation System for DNS. In *Proceedings of the 19th USENIX Conference on Security* (Berkeley, CA, USA, 2010), USENIX Security'10, USENIX Association.

[13] ANTONAKAKIS, M., PERDISCI, R., NADJI, Y., VASILOGLOU, N., ABU-NIMEH, S., LEE, W., AND DAGON, D. From Throw-away Traffic to Bots: Detecting the Rise of DGA-based Malware. In *Proceedings of the 21st USENIX Conference on Security Symposium* (2012).

[14] BADER, J. Domain Generation Algorithm analyses, 2015. Blog posts on various DGAs: http://www.johannesbader.ch/tag/dga/.

[15] BARABOSCH, T. Behavior-Driven Development in Malware Analysis: Can it Improve the Malware Analysis Process?, 2015. Presentation: https://itsec.cs.uni-bonn.de/spring2015/downloads/barabosch.pdf.

[16] BARABOSCH, T., WICHMANN, A., LEDER, F., AND GERHARDS-PADILLA, E. Automatic Extraction of Domain Name Generation Algorithms from Current Malware. In *Proceedings of the NATO Symposium IST-111 on Information Assurance and Cyber Defence* (2012).

[17] BAUMGARTNER, K., AND RAIU, C. Sinkholing Volatile Cedar DGA Infrastructure, 2015. Blog post: https://securelist.com/blog/research/69421/sinkholing-volatile-cedar-dga-infrastructure/.

[18] BILGE, L., SEN, S., BALZAROTTI, D., KIRDA, E., AND KRUEGEL, C. Exposure: A Passive DNS Analysis Service to Detect and Report Malicious Domains. *ACM Trans. Inf. Syst. Secur. 16*, 4 (Apr. 2014).

[19] DOMAINTOOLS. Company Profile, 2015. Website: https://www.domaintools.com/company/.

[20] FALLIERE, N. W32.Virut: Using Cryptography to Prevent Domain Hijacking, 2011. Blog post: http://www.symantec.com/connect/blogs/w32virut-using-cryptography-prevent-domain-hijacking.

[21] GASTESI, M., AND GEGENY, J. Citadel Updates: Anti-VM and Encryption change, June 2012. Blog post for S21sec: http://securityblog.s21sec.com/2012/06/citadel-updates-anti-vm-and-encryption.html.

[22] GEFFNER, J. End-To-End Analysis of a Domain Generating Algorithm Malware Family. In *Proceedings of the 2013 Blackhat Conference* (2013).

[23] HOLZ, T., GORECKI, C., RIECK, K., AND FREILING, F. Measuring and Detecting Fast-Flux Service Networks. In *Proceedings of the 15th Annual Network & Distributed System Security Conference (NDSS)* (2008).

[24] KESSEM, L. Shifu: 'Masterful' New Banking Trojan Is Attacking 14 Japanese Banks, 2015. Blog post: https://securityintelligence.com/shifu-masterful-new-banking-trojan-is-attacking-14-japanese-banks/.

[25] LEDER, F., AND WERNER, T. Know Your Enemy: Containing Conficker, To Tame a Malware. Tech. rep., The Honeynet Project, http://honeynet.org, 2009.

[26] LEUNG, K., LIU, Y., AND KIERNAN, S. W32.Downadup.E Technical Details. Tech. rep., Symantec, 2009.

[27] LIPOVSKY, R. Hesperbot - A new, Advanced Banking Trojan in the Wild. Tech. rep., ESET, 2013.

[28] MA, J., SAUL, L. K., SAVAGE, S., AND VOELKER, G. M. Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2009), KDD '09, ACM, pp. 1245–1254.

[29] MALWARE PROTECTION CENTER. MSRT April 2014 on Ramdo, 2014. Malware description by Microsoft: http://blogs.technet.com/b/mmpc/archive/2014/04/08/msrt-april-2014-ramdo.aspx.

[30] MALWARE PROTECTION CENTER. Trojan:Win32/Emotet.C, 2014. Malware description by Microsoft: https://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Trojan:Win32/Emotet.C.

[31] MARSAGLIA, G. Xorshift RNGs. *Journal of Statistical Software 8*, 1 (2003).

[32] MATROSOV, A. What do Win32/Redyms and TDL4 have in common, 2013. Blog post: http://www.welivesecurity.com/2013/02/04/what-do-win32redyms-and-tdl4-have-in-common/.

[33] MATSUMOTO, M., AND NISHIMURA, T. Mersenne Twister: A 623-dimensionally Equidistributed Uniform Pseudo-random Number Generator. *ACM Trans. Model. Comput. Simul. 8*, 1 (Jan. 1998).

[34] MOWBRAY, M., AND HAGEN, J. Finding Domain-Generation Algorithms by Looking at Length Distribution. In *Proceedings of the 25th IEEE International Symposium on Software Reliability Engineering Workshops, ISSRE Workshops, Naples, Italy* (2014).

[35] NADJI, Y., ANTONAKAKIS, M., PERDISCI, R., DAGON, D., AND LEE, W. Beheading Hydras: Performing Effective Botnet Takedowns. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2013), CCS '13, ACM.

[36] PARK, S. K., AND MILLER, K. W. Random Number Generators: Good Ones Are Hard to Find. *Commun. ACM 31*, 10 (Oct. 1988).

[37] PIOTR KRYSIUK, V. T. Trojan.Bamital. Tech. rep., Symantec, 2013.

[38] PLOHMANN, D. DGArchive. Fraunhofer FKIE: https://dgarchive.caad.fkie.fraunhofer.de.

[39] PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3 ed. Cambridge University Press, New York, NY, USA, 2007.

[40] ROSSOW, C., ANDRIESSE, D., WERNER, T., STONE-GROSS, B., PLOHMANN, D., DIETRICH, C. J., AND BOS, H. SoK: P2PWNED — Modeling and Evaluating the Resilience of Peer-to-Peer Botnets. In *Proceedings of the 34th IEEE Symposium on Security and Privacy (S&P)* (San Francisco, CA, May 2013).

[41] ROYAL, P. On the Kraken and Bobax Botnets. Tech. rep., Damballa, April 2008.

[42] SANDEE, M. GameOver ZeuS - Backgrounds on the Badguys and the Backends. Tech. rep., Fox IT, 2013.

[43] SCHIAVONI, S., MAGGI, F., CAVALLARO, L., AND ZANERO, S. Phoenix: DGA-Based Botnet Tracking and Intelligence. In *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)* (2014), vol. 8550 of *Lecture Notes in Computer Science*.

[44] SCHWARZ, D. Bedep's DGA: Trading Foreign Exchange for Malware Domains, 2015. Blog post: https://asert.arbornetworks.com/bedeps-dga-trading-foreign-exchange-for-malware-domains/.

[45] SECURITY RESPONSE. Butterfly: Corporate spies out for financial gain. Tech. rep., Symantec, July 2015.

[46] SECURITY RESPONSE. W32.Ramnit Analysis. Tech. rep., Symantec, February 2015.

[47] SEGURA, J. Elusive HanJuan EK Drops New Tinba Version, 2015. Blog post: https://blog.malwarebytes.org/intelligence/2015/06/elusive-hanjuan-ek-caught-in-new-malvertising-campaign/.

[48] SINEGUBKO, D. Runforestrun and Pseudo Random Domains, June 2012. Blog post for Unmask Parasites: http://blog.unmaskparasites.com/2012/06/22/runforestrun-and-pseudo-random-domains/.

[49] SKURATOVICH, S. Matsnu. Tech. rep., Check Point Software technologies Ltd., May 2015.

[50] STONE-GROSS, B., COVA, M., CAVALLARO, L., GILBERT, B., SZYDLOWSKI, M., KEMMERER, R., KRUEGEL, C., AND VIGNA, G. Your Botnet is My Botnet: Analysis of a Botnet Takeover. In *Proceedings of the 16th ACM Conference on Computer and Communications Security* (2009).

[51] VISSERS, T., JOOSEN, W., AND NIKIFORAKIS, N. Parking Sensors: Analyzing and Detecting Parked Domains. In *Proceedings of the 2015 Network and Distributed System Security (NDSS) Symposium* (2015).

[52] WOLF, J. Technical details of Srizbi's domain generation algorithm, November 2008. Blog post for FireEye: https://www.fireeye.com/blog/threat-research/2008/11/technical-details-of-srizbis-domain-generation-algorithm.html.

[53] XU, W., SANDERS, K., AND ZHANG, Y. We Know It Before You Do: Predicting Malicious Domains. In *Proceedings of the 24th Virus Bulletin Conference (VB2014)* (2014).

[54] YADAV, S., REDDY, A. K. K., REDDY, A. N., AND RANJAN, S. Detecting Algorithmically Generated Malicious Domain Names. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement* (2010), IMC '10.