Real-time behavioral DGA detection through machine learning

Federica Bisio, Salvatore Saeli, Pierangelo Lombardo, Davide Bernardi, Alan Perotti, Danilo Massa
aizoOn Technology Consulting
Strada del Lionetto 6
10146, Turin, Italy
Email: [name].[surname]@aizoongroup.com

***Abstract*—During the last years, the use of Domain Generation Algorithms (DGAs) has increased with the aim of improving the resiliency of communication between bots and Command and Control (C&C) infrastructure. In this paper, we report on an effective DGA-detection algorithm based on a single network monitoring. The first step of the proposed method is the detection of a bot looking for the C&C and thus querying many automatically generated domains. The second phase consists on the analysis of the resolved DNS requests in the same time interval. The linguistic and semantic features of the collected unresolved and resolved domains are then extracted in order to cluster them and identify the specific bot. Finally, clusters are analyzed in order to reduce false positives. The proposed solution has been evaluated over (1) an ad-hoc network where several known DGAs were injected and (2) the LAN of a company. In the first experiment, we deployed different families of malware employing several DGAs: all the malicious variants were detected by the proposed algorithm. In the real case scenario, the algorithm discovered an infected host in a 15-day-long experimental session, while producing a low false-positive rate during the same period.**

I. Introduction

Cybercrime constitutes one of the most serious threats to the current society, with heavy and sometimes dramatic consequences for many companies, organizations and single individuals [14, 18, 20, 26, 30]. During the last number of years, a key role in cybercrime has been played by botnets: these are networks of compromised computers (popularly referred to as *zombies* or *bots*), which are controlled by a remote attacker (popularly referred to as a *bot herder*) through specific Command and Control (C&C) channels. The strength of the botnet resides in the fact that it can be a highly distributed and highly changeable network, making the tracing and the recovery of all the infected components very difficult, and therefore allowing a secure and stable platform for the implementation of a wide range of malicious and illegal activities such as the spreading of ransomwares, exploit kits, banking trojans, etc. [4, 7, 11, 12, 17, 23, 25, 32].

In botnets, the bot herder and bots can exchange information using different protocols; P2P-based botnets have a more robust C&C structure that is difficult to detect and take down, but they are typically harder to implement and maintain.

In order to combine the simplicity of centralized C&Cs with the robustness of P2P-based structures, many attackers employ HTTP botnets that locate their C&C servers through the dynamic generation of domains using a Domain Generation Algorithm (DGA), also known as domain-flux. With this technique, each bot, using a precalculated seed value known to the bot herder (e.g., the current date), automatically generates hundreds or thousands of pseudo-random domain names that represent candidate C&C domains. The bot sends DNS queries until it connects to the IP address associated to a resolved domain. The key advantage of this strategy is that even if one or more C&C domain names or IP addresses are identified and recovered, the bots will query the next set of automatically generated domains and it will eventually get the IP address of a relocated C&C server. DGA provides therefore a remarkable level of agility and a very resilient communication channel between bots and C&C, making it one of the most used technique in botnet control [6, 7, 8, 12, 15, 24, 33].

For these reasons, DGA detection is of crucial importance in cyber security. A number of different approaches to DGA-detection have been implemented, but DNS-based analysis is one of the most appropriate to obtain quick responses, since it does not need file dumps and it only requires the analysis of a small part of the network traffic (in particular, it can ignore packets' payloads). For this reason, many recent works focused on automatically recognizing DGA within DNS traffic, whenever occurring. Many efforts have been made to use supervised or signature-based approaches [5] but these have obtained limited results in the highly dynamic DGA realm. Therefore, some works have applied unsupervised techniques on DNS traffic data provided by some Internet Service Provider [9, 22, 28, 29] or retrieved by collecting the DNS traffic of a single network [15, 21, 33].

In this paper, we report on an effective DGA-detection algorithm which analyzes the DNS traffic of a single network in *near-real-time*: if we consider a network as the one described in Tab. I, the average execution time of the algorithm is 2 seconds, while the average time between two subsequent runs of the algorithm is 3 minutes and 48 seconds (see Sec. III for more details). The ability to detect an attack in near-real-time is crucial, as it allows for a quick reaction and it is the only way to prevent a potentially severe damage to the company which is using the network under attack [14, 31].

The remainder of the paper is structured as follows. In Sec. II we briefly describe aramis, the monitoring platform

978-1-5386-1585- 0/17/$31.00

which contains the DGA detection method which is the focus of this paper and which is described thoroughly in Sec. III. Finally, we discuss the experimental results in details in Sec. IV and possible future developments in Sec. V.

II. ARAMIS

The proposed DGA-detection algorithm has been deployed in aramis (Aizoon Research for Advanced Malware Identification System), a network security monitoring platform able to automatically identify a wide range of malware and attacks in near-real-time. aramis software is bundled with dedicated hardware[1], and its structure can be summarized in four phases:

1) Collection: sensors are placed in various nodes of the network. Each sensor gathers the data from its segment of the network, pre-analyzes them in real-time and sends the results to a NoSQL database.
2) Enrichment: inside the NoSQL database, data is enriched with information coming from the aramis Cloud Service, which collects intelligence from various OSINT sources and from internally managed sources.
3) Analysis: two kinds of analyses are executed on the stored data: (i) *advanced cybersec analytics* to spot and highlight specific attacks, among which DGAs, and (ii) a *machine learning engine* which compares the behavior of each node with the usual one.
4) Visualization: the results are presented through cognitive dashboards, which are crucial to highlight anomalies.

The cycle of the four phases restarts after a period Δt which slightly depends on the quantity of analyzed traffic. On the network described in Tab. I $\Delta t = 228 \pm 92$ seconds, which represents the best trade-off between the need of many network data in order to have statistically significant results and the requirement of near-real-time analysis.

III. DGA Detection Method

The aim of the proposed DGA-detection method is the near-real-time identification of domain-flux attacks via the monitoring of a single network. To this purpose, the method is composed of several steps of analysis:

- Collection of unresolved DNS requests (UNRES): in order to detect a bot trying to connect with the C&C, all the UNRES in a suitable amount of time are collected. The sudden and huge increase of UNRES may in fact indicate the tentative of connection with several untrusted automatically generated domains.
- Filtering and preprocessing of UNRES: all the queries due to user errors (e.g., typos of popular domains) and system misconfigurations are removed.
- Outlier detection: the hosts producing the highest peaks of UNRES are identified.
- Extraction of resolved DNS requests (RES): since a bot stops querying when an existent domain is hit and a successful connection is established, RES near the peaks identified in the previous step are collected.
- Domain features extraction: all the collected RES and UNRES are mapped in a feature space able to embed the linguistic and semantic components (see Sec. III-E).
- Clustering: domains which are similar according to the features extracted in the previous step, are grouped together in order to spot common patterns of the specific bot.
- False positives removal: the level of homogeneity of the clusters is calculated. This allows the distinction between true DGAs (associated with highly homogeneous clusters) from the expected legit unresolved DNS peaks (associated with less homogeneous clusters).

In the following we describe the details of each step.

A. Collection of UNRES

aramis framework operates in near-real-time, therefore all the UNRES are continuously downloaded and analyzed. On the network described in Sec. IV-B, the complete DGA-detection algorithm takes an average time of 2 seconds to complete. Anyway, as discussed in Sec. II, the algorithm is integrated in aramis, which takes an average time Δt of about 3 minutes and 48 seconds to collect all network data and perform the analyses. After this process, aramis produces a detailed analysis of the network risk and restarts a new cycle of analysis.

B. Filtering and preprocessing of UNRES

The following filters are applied to the retrieved UNRES:

- Requests containing invalid or malformed Top Level Domains (TLDs) are removed. Typically, they are due to typos or user errors.
- Overloaded DNS: DNS queries are sometimes overloaded so to provide anti-spam or anti-malware techniques. In order to reduce noise, the overloaded DNS are removed.
- Local and private domains are removed.
- White list domains (i.e., domains that are known to be trusted) are removed.
- Popular domains are removed. More specifically, three popular domains sources are considered: the top 10000 domains in the world provided by Alexa [2], the web URLs of the 500 world biggest companies provided by Forbes [3] and the top 100 domains collected inside the network under analysis. In all these cases, the second and third level domains of an input domain are extracted and compared with the second and third level domains of the list of popular domains; if the Jaro-Winkler distance [27] is below 0.1, the input domain is considered as a misspelling of a popular domain and removed.
- Configuration words: domains containing certain substrings (e.g., words related to network system and structure) are filtered out, because they represent congenital network traffic.
- ARPA domains are filtered out, since they are only used for reverse DNS lookup.

[1]E5-2690 2.9GHz x 2 (2 sockets x 16 cores) 16 x 8GB RAM, 1.1TB HDD

- If a TLD is found in the third or higher levels, it is considered as a misconfiguration of the web browser or of the particular application and hence it is removed.
- If an IP address is found in the third or following levels, it is considered as an internal domain and it is removed.

The filtering phase removes the largest part of the initial UNRES; usually just a 5-10% of the queries are not filtered out and proceed through the other steps of the algorithm.

C. Outlier Detection

In order to recognize burst in the UNRES traffic, time is discretized and the number of UNRES for each machine in each time interval is considered part of a time series, which is described in terms of 6 different statistical methods:

- deviation from the expected distribution calculated via
  - Gaussian estimate
  - kernel density estimate
- arima model [10]
- deviation from the expected behavior calculated on a moving window via
  - mean and standard deviation
  - median and median absolute deviation
  - interquantile range

Each method can be considered as a binary classifier between ordinary points and outliers, and the results of all classifiers are combined with an *ensemble classifier* based on a weighted majority rule[2], which has been shown to perform typically better than any single classifier [13]. The identification of outliers in the distribution of the number of UNRES allows to detect potentially suspicious machines.

D. Extraction of resolved DNS requests

Once the suspicious machines are detected, the extraction of the related RES is performed. In particular, all the RES occurring in a time interval τ around the UNRES peaks are collected. The interval τ is set to 20 seconds; this choice represents a trade-off between the need of a large τ to compensate possible delays in the network data collection and the necessity of a small τ in order to avoid casual associations of RES with a cluster of UNRES.

E. Domain features extraction

The purpose of this phase is the creation of a common feature space for RES and UNRES, able to map into an array of numbers the linguistic peculiarities of the domains under analysis. Pseudo-random domains generated by the same algorithm typically share at least some common linguistic attributes, while legitimate domains are not generated by an algorithm and, hence, should not show similarities in the domain structure. It is known however [15] that some modern DGAs employ English dictionary words with little modifications; for this reason both linguistic and non-linguistic features have been considered [3]. Therefore, the main idea is to extract the most relevant features of both RES and UNRES in order to find common patterns able to characterize a specific C&C connection. In this way, we are able to perform the subsequent clustering phase and group together domains showing a similar pattern, therefore defining the behavior of a particular bot.

The extracted linguistic features are the following:

- Number of levels in the domain
- For the second and third levels: distance of the monograms probability distribution from the one of monograms in the English language
- For the second and third levels: distance of the bigrams probability distribution from the one of bigrams in the English language
- Entropy in characters distribution of the second and third levels
- Number of characters of the second and third levels

TABLE I
NETWORK DESCRIPTION

	Real Network	**Malware Lab**
N. of machines	288	269
N. of clients	209	185
Average N. of connections	136 k / hour	452 k / hour
Average N. of UNRES	791 / hour	14 k / hour
Average N. of RES	59 k / hour	184 k / hour

F. Clustering

Once the domain features are extracted, a k-means clustering [16] is performed on the feature space. The number of clusters N_c is set equal to a fifth of the number of input domains, because this was found as the best trade-off between the need of a large N_c in order to obtain highly homogeneous groups and the need of a small N_c to avoid the separation of domains belonging to the same DGA into many different clusters. Moreover, every cluster has an associated homogeneity value corresponding to the average proximity of the samples of the cluster with the related centroid.

After creating the clusters, malicious clusters have to be recognized; they are identified in the following way:

- Clusters formed by both RES and UNRES and where the number of UNRES is higher than the number of RES;
- Clusters which only contain UNRES.

In both cases, we assign an anomaly indicator A to each malicious cluster proportional to its value of homogeneity. Therefore, A has minimum value $A = 0$ (no anomaly detected) and maximum value $A = 1$ (maximum anomaly detected). The two kinds of clusters contain respectively DGAs which eventually contacted a C&C and DGA attempts which did not find a C&C. Thus, A for the second case is reduced by a corrective factor $\lambda_{\text{fail}} = 0.8$.

[2]The weight used for each method is proportional to the inverse of the mean number of outliers detected by that method: this means that an alarm reported by a method which often presents alarms has a smaller relevance compared to an alarm presented by a usually cautious method.

[3]This choice allowed for the identification of English-dictionary-based DGAs (see Sec. IV-A), e.g., Gozi ISFB, Rovnix, Matsnu, Suppobox.

TABLE II
MALWARE DESCRIPTION AND DETECTION RESULTS

Malware type	Domain layout	Domain length	Malware names	C&C / sinkhole alive	Clusters	Resolved domains	A
Banking Trojan	Alphabetic	Fixed	Fobber [Tinba v3]	Yes	40	2	0.9841
			Ranbyus	Yes	38	5	0.9842
			Tinba [TinyBanker, Zusy]	Yes	29	6	0.9937
		Variable	Qakbot	Yes	71	2	0.9864
			Ramnit	Yes	40	1	0.8885
			Vawtrak [Neverquest, Snifula]	No	82	2	0.9638
	Alphabetic + seed	Fixed	Banjori [MultiBanker 2, BankPatch(er)]	Yes	116	3	0.9955
	Alphanumeric	Fixed	Qadars v3	No	52	1	0.9850
		Variable	Newgoz [Gameover Zeus]	Yes	42	3	0.9926
			Shiotob	No	57	2	0.9615
			ZeusBot	Yes	84	1	0.9731
			Murofet v3 [Licat]	Yes	44	1	0.9859
	Alphanumeric + DDNS	Variable	Corebot	Yes	55	4	0.9847
	Dictionary	Variable	Gozi ISFB[a] [Ursnif, Snifula, Papras]	Yes	30	2	0.9766
			Gozi ISFB[b] [Ursnif, Snifula, Papras]	Yes	51	3	0.9776
			Rovnix	No	174	3	0.9875
Botnet	Alphabetic	Fixed	PushDO [Pandex, Cutwail]	No	63	2	0.9995
	Alphabetic + DDNS	Variable	Kraken v1 [Bobax, Oderoor]	Yes	70	5	0.9834
	Alphabetic	Variable	Necurs	Yes	39	2	0.9664
Exploit Kit	Alphabetic	Variable	Blackhole	No	63	3	0.9924
Ransomware	Alphabetic	Fixed	Cryptolocker	No	24	2	0.9984
			Padcrypt	Yes	84	4	0.9908
		Variable	DirCrypt	No	16	2	0.9784
			Locky v3	Yes	30	3	0.9738
Trojan Horse	Alphabetic	Fixed	Dnschanger [Alureon]	No	57	1	0.9959
			Ramdo	No	131	3	0.9894
			Simda	Yes	34	3	0.9984
			Sisron [TOMB, Trojan.Scar]	Yes	10	1	0.9807
			Srizbi	No	14	1	0.9964
		Variable	Bamital	Yes	21	1	0.9888
			Nymaim	No	48	3	0.9643
	Alphabetic + DDNS	Variable	Vidro	Yes	130	4	0.9866
			Symmi	Yes	49	2	0.9816
	Alphanumeric	Fixed	Chinad	Yes	17	1	0.9861
		Variable	Beped	No	35	2	0.9822
	Dictionary	Variable	Matsnu	No	74	3	0.9897
			Suppobox	Yes	52	1	0.9267
Worm	Alphabetic	Fixed	Tempedreve	No	38	2	0.9877
		Variable	Proslikefan	Yes	98	5	0.9957
			Pykspa [Pykse,Skyper, SkypeBot]	Yes	58	5	0.9835

[a]Employing "luther" dictionary
[b]Employing "nasa" dictionary

G. False positive removal

The anomaly indicator of each cluster is rescaled in order to reduce false positives. The effect of this rescaling is to further decrease low values of A (usually associated with false positives), to highlight large values of A and to enhance the differences in the interval $[0.3, 0.75]$, which has been recognized in the training phase as the overlapping region between the most uncertain false positives and true positives.

IV. EXPERIMENTAL EVALUATION

The DGA-detection algorithm described above was evaluated within two different experimental designs:

- 40 DGA snippets belonging to different malware families were used to inject real DGA network traffic into an ad-hoc network (*malware lab*, see Tab. I). The malware families of the DGA snippets include banker trojans, ransomwares, worms (see Tab. II for a complete list) and cover all the most relevant DGA-attack scenarios.
- The LAN of a real company (described in Tab. I) was observed for a 15-day-long experimental session.

A. Network traffic injection

The first round of experiments consisted in 40 DGA snippets belonging to different malware families used to simulate real DGA traffic inside the *malware lab*, which is described in Tab. I. In order to simulate the successful connection to the C&C, a technique similar to *sinkholing* [19, 8] was used: before the injection of the traffic generated by each snippet, a couple of the domains produced by the snippet were registered in the FakeDns of the *malware lab*. Each registered domain was associated to an IP address of a honeypot running a web

server [4].

Tab. II provides a synthetic description of the malware used in the experiments and the related detection results. For each malware, it contains the following information:

- Malware type
- Domain layout, i.e., elementary components of the generated domains [24]
- Domain length (fixed or variable)
- Specific names of the malware; aliases of the malware names are reported in square brackets
- C&C / sinkhole alive, i.e., a field indicating if at least one domain (in addition to the ones registered as described in Sec. IV-A) was resolved during the experiments
- Number of clusters, i.e., number of groups of similar domains found by the algorithm described in Sec. III-F
- Number of clusters containing resolved DNS requests
- Anomaly indicator A, as described in Sec. III-F

From Tab. II it is possible to notice that the proposed DGA-detection framework successfully detected all the malware variants with a high anomaly indicator. In fact, the value of A averaged on all the samples is 0.9806. Moreover, all the malicious RES have been identified, thus giving the possibility to detect all the active C&Cs, which were reported to the appropriate OSINT repositories.

B. Real company scenario

The second round of experiments was performed through the monitoring of the LAN of a real company, in order to provide a real case test of the proposed solution. In this way, we were able to reliably estimate the false positive rate.

aramis was run for a 15-day-long experimental session. In order to evaluate the performances, we distinguished between RES and UNRES requests. The RES case represents the riskiest situation, since the complete domain-flux attack took place; in this case, therefore, the first concern is the avoidance of false negatives, while some false positives might be tolerated.

On the contrary, the UNRES situation is less risky since it indicates that the potential malware unsuccessfully tried to connect to the C&C. Therefore, in the latter case, a higher false negative rate might be tolerated.

Out of the 285 k unresolved domains of the whole network collected during the observation, the algorithm detected 37 and 1720 domains respectively for the RES and the UNRES case.

We collected results with two different granularities:

- $A > 0$: in this case all the alarms are considered.
- $A > 0.7$: in this case only the alarms with Anomaly Indicator greater than 0.7 (high risk) are considered.

Tab. III reports the obtained results. In particular, it is possible to notice that a real domain-flux attack, including the final contact with the C&C (RES case), has been completely detected. In fact, the alarms associated with this detection were investigated and led to the discovery of the activity of a banking trojan (VawTrak [1]). In this case, as it can be noticed in Tab III, all malicious clusters were detected with $A > 0.7$. Moreover, it is possible to observe that the false positive rate is equal to zero for the RES case, hence allowing to completely distinguish the real attacks from the normal traffic.

[4]Besides the DNS registered in the experiment, other domains were resolved, revealing the presence of active C&Cs or *sinkholes*.

TABLE III
REAL NETWORK RESULTS

UNRES case	$A > 0$		$A > 0.7$	
	DGA	**Not DGA**	**DGA**	**Not DGA**
Detected	1.65 k	70	1.65 k	42
Undetected	0	285 k	0	285 k
RES case	$A > 0$		$A > 0.7$	
	DGA	**Not DGA**	**DGA**	**Not DGA**
Detected	37	0	37	0
Undetected	0	21.3 M	0	21.3 M

Besides, we also analyzed the UNRES case. It is possible to notice that all the 1.65 k malicious UNRES were detected with $A > 0.7$, while the false positive rate is very low: the 0.02% of the 285 k non-malicious UNRES are presented in output by the algorithm, and only the 0.01% has $A > 0.7$.

Therefore, we can conclude that the proposed method is able to detect potentially infected machines in near-real-time and with high anomaly indicators, while limiting the false positives at the same time.

V. CONCLUSIONS

In this paper, we proposed a DGA-detection method based on the analysis of the DNS traffic of a single network; the analysis requires aramis security monitoring system.

The proposed solution has been evaluated over two networks: (i) an ad-hoc network, where traffic generated with DGA snippets belonging to different DGA attacks was injected and (ii) the LAN of a real company. In the first experiment, all the malicious variants were detected, while in the real case scenario the algorithm discovered an infected host. Thus, the experimental evaluation has confirmed the effectiveness of the proposed approach.

As a future development, we plan to refine the clustering algorithm by adding weights to domain levels in the linguistic features extraction phase and by recognizing the layout (alphanumeric or alphabetic). Another aspect that we plan to improve is the filtering phase, with the introduction of a filter for Content Delivery Networks (CDN) and Round Robin DNS (RRDNS), with the development of a CDN/RRDNS identification algorithm. Other developments include the introduction of a specific dictionary for the country where the network is located, the filtering of queries used by some browser (e.g., Chrome and Chromium) to determine if the user is on a network that intercepts and redirects requests for nonexistent hostnames, and the refinement of local domains identification.

REFERENCES

[1] https://www.blueliv.com/downloads/network-insights-into-vawtrak-v2.pdf.

[2] http://www.alexa.com.

[3] http://www.forbes.com.

[4] K. Alieyan, A. ALmomani, A. Manasrah, and M. M. Kadhum. A survey of botnet detection based on dns. *Neural Computing and Applications*, pages 1–18, 2015.

[5] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster. Building a dynamic reputation system for dns. In *USENIX security symposium*, pages 273–290, 2010.

[6] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon. From throw-away traffic to bots: Detecting the rise of dga-based malware. In *USENIX security symposium*, volume 12, 2012.

[7] A. H. R. A. Awadi and B. Belaton. Multi-phase irc botnet and botnet behavior detection model. *arXiv preprint arXiv:1501.03241*, 2015.

[8] T. Barabosch, A. Wichmann, F. Leder, and E. Gerhards-Padilla. Automatic extraction of domain name generation algorithms from current malware. In *Proc. NATO Symposium IST-111 on Information Assurance and Cyber Defense, Koblenz, Germany*, 2012.

[9] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. Exposure: Finding malicious domains using passive dns analysis. In *Ndss*, 2011.

[10] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[11] D. Dagon, G. Gu, C. P. Lee, and W. Lee. A taxonomy of botnet structures. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, pages 325–339. IEEE, 2007.

[12] C. J. Dietrich, C. Rossow, F. C. Freiling, H. Bos, M. Van Steen, and N. Pohlmann. On botnets that use dns for command and control. In *Computer Network Defense (EC2ND), 2011 Seventh European Conference on*, pages 9–16. IEEE, 2011.

[13] T. G. Dietterich et al. Ensemble methods in machine learning. *Multiple classifier systems*, 1857:1–15, 2000.

[14] T. Grance, K. Kent, and B. Kim. Computer security incident handling guide. *NIST Special Publication*, 800:61, 2004.

[15] M. Grill, I. Nikolaev, V. Valeros, and M. Rehak. Detecting dga malware using netflow. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 1304–1309. IEEE, 2015.

[16] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[17] G. Hogben, D. Plohmann, E. Gerhards-Padilla, and F. Leder. Botnets: Detection, measurement, disinfection and defence. *European Network and Information Security Agency*, 2011.

[18] M. Korolov. Cyber security review. *Treasury & Risk*, 2012.

[19] F. Leder, T. Werner, and P. Martini. Proactive botnet countermeasures: an offensive approach. *The Virtual Battlefield: Perspectives on Cyber Warfare*, 3:211–225, 2009.

[20] F. Lemieux. Investigating cyber security threats: Exploring national security and law enforcement perspectives. *2011 Developing Cyber Security Synergy*, page 63, 2011.

[21] M. Mowbray and J. Hagen. Finding domain-generation algorithms by looking at length distribution. In *Software Reliability Engineering Workshops (ISSREW), 2014 IEEE International Symposium on*, pages 395–400. IEEE, 2014.

[22] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero. Phoenix: Dga-based botnet tracking and intelligence. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 192–211. Springer, 2014.

[23] E. Soltanaghaei and M. Kharrazi. Detection of fast-flux botnets through dns traffic analysis. *Scientia Iranica. Transaction D, Computer Science & Engineering, Electrical*, 22(6):2389, 2015.

[24] A. K. Sood and S. Zeadally. A taxonomy of domain-generation algorithms. *IEEE Security & Privacy*, 14(4):46–53, 2016.

[25] M. Stevanovic and J. M. Pedersen. On the use of machine learning for identifying botnet network traffic. *Journal of Cyber Security and Mobility*, 4(3):1–32, 2016.

[26] R. W. Taylor, E. J. Fritsch, and J. Liederbach. *Digital crime and digital terrorism*. Prentice Hall Press, 2014.

[27] W. E. Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. 1990.

[28] S. Yadav, A. K. K. Reddy, A. Reddy, and S. Ranjan. Detecting algorithmically generated malicious domain names. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 48–61. ACM, 2010.

[29] S. Yadav and A. N. Reddy. Winning with dns failures: Strategies for faster botnet detection. *Security and privacy in communication networks*, pages 446–459, 2012.

[30] T. Yadav and R. A. Mallari. Technical aspects of cyber kill chain. *arXiv preprint arXiv:1606.03184*, 2016.

[31] X. Yin, W. Yurcik, Y. Li, K. Lakkaraju, and C. Abad. Visflowconnect: Providing security situational awareness by visualizing network traffic flows. In *Performance, Computing, and Communications, 2004 IEEE International Conference on*, pages 601–607. IEEE, 2004.

[32] H. R. Zeidanloo, M. J. Z. Shooshtari, P. V. Amoli, M. Safari, and M. Zamani. A taxonomy of botnet detection techniques. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 2, pages 158–162. IEEE, 2010.

[33] H. Zhang, M. Gharaibeh, S. Thanasoulas, and C. Papadopoulos. Botdigger: Detecting dga bots in a single network. In *Proceedings of the IEEE International Workshop on Traffic Monitoring and Analaysis*, 2016.