Thwarting C2 Communication of DGA-Based Malware using Process-level DNS Traffic Tracking

Anjali Menon
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, United States
anjalim3@illinois.edu

***Abstract*— Many modern botnet malwares use Domain Generation Algorithms (DGAs) to dynamically generate the domain names that resolve to their command and control (C2) centers. This approach allows these malwares to subvert traditional detection systems which rely on blacklists of known domains associated with malicious activities to block malware communications. Since the advent of DGA-based malwares, the efforts to prevent the said malwares from contacting their command and control centers (C2) server have been centered around detecting Algorithmically Generated Domain Names through lexicographic analysis, isolating entire infected devices or both. Recent research has emerged, which more accurately identifies infected devices in a network, by monitoring the volumes of domain resolution failures. While effective, these techniques are slow to identify DGA generated domain names. Even after the delayed identification, the only preliminary mitigation known today is a complete shutdown of a device that is suspected to be infected.**

In this paper, we present a new method to counter DGA-based malwares by limiting the impact of mitigation. Instead of isolating the entire infected device from the network we limit network activity of the malicious process alone. Our objective is to prevent DGA-based malwares from communicating with their C2 centers while allowing an infected device to maintain its normal functionality. We achieve this by tracking Domain Name Service (DNS) responses of individual processes and blacklisting those processes for which DNS traffic have abnormally large numbers of domain resolution failures. The blacklisting at a process level ensures that non-malicious processes in the infected device can continue functioning.

Keywords—DGA, malware, domain generation, algorithms, software, security, DNS, domain lookup

I. INTRODUCTION

Botnets are networks of computers infected with malicious software that are controlled by a remote botmaster unbeknownst to the users. The botnet malwares that have infected a computer need to contact their command and control (C2) servers, which are owned by botmasters, for a variety of tasks. These tasks may include sending the compromised data of the infected systems, receiving commands from the botmasters, receiving upgrades for themselves, and downloading other malwares. The addresses of the C2 servers should be known to a botnet malware to make this communication. When they are statically embedded in the malware's code, computer security experts can analyze the code to obtain the C2 addresses. This enables the experts to thwart the malware's attempts to communicate to C2 by preemptively blocking all access to these addresses. Such thwarting attempts have prompted malware owners to avoid static embedding of C2 addresses and instead generate them on the fly. Domain Generation Algorithms (DGAs) are algorithms used to generate a large number of domain names called Algorithmically Generated Domains or AGDs. DGAs are unique to their malwares and employ techniques to ensure that the domains generated by them are not easily predictable. Certain botmasters only register the domains a few hours before the malware attempts to contact the C2. Therefore, traditional methods such as black-listing domains and creating sinkholes are not sufficient to prevent the infected device from contacting the C2 server.

Current research around DGA-based malware [19, 20, 22, 23] focus on predicting the AGDs ahead of time by lexically analyzing domain names using machine learning techniques. However, these methods have a training phase, and therefore if a new DGA is released in to the wild, it would take a while before its AGDs can be accurately identified. Additionally, the effectiveness of these techniques depends on the accuracy of the machine learning model used.

Malware DGAs generate over hundreds and thousands of domain names. However, only a subset of these domain names generated actually point to the command and control servers, since it is not feasible for the botmasters to register all possible domain names generated by the DGAs. DGA-based malwares attempt to contact each of the generated domains one at a time till they are able to successfully find the domain pointing to a C2 server. Antonakakis et al. [1] was the first to observe that most DNS queries by botnets result in Non-Existent (NXDomain) responses. This is a result of the fact that only a small number of AGDs successfully resolve to C2 servers. Reference [26] observed that 90% of the devices infected with the DGA-based malware, Murofet, generated more than 10 NXDomain responses in a day, whereas 92% of the devices created fewer than 10.

Reference [1] and other works [21] have made use of the observation that DGAs tend to generate queries that result in a high volume of NXDomain responses in order to aid lexicographic AGD detection.

In this paper we introduce the tool DGA-C2 Communication Thwarter, where we extend this concept of tracking the cardinality of DNS resolution failures to isolate potentially infected processes in a device using the tool DGA-C2 Communication Thwarter. This is in contrast to isolating the entire device from the network when a botnet is detected. We do this by tracking the DNS traffic of individual processes and blacklisting those processes that generate an abnormally large amount of NXDomain traffic. This is based on the reasoning that non-malicious processes are not likely to generate a significant number of domain resolution requests that result in failures. Therefore, any process for which the number of domain resolution failures is greater than a threshold can be identified with significant probability as a DGA-based malware that is attempting to contact its C2 server by cycling through a list of domains. For these blacklisted processes, we do not allow further DNS lookups to successfully resolve, but permit all other forms of network traffic. Processes that are not blacklisted will not have a large number of domain resolution failures and are therefore

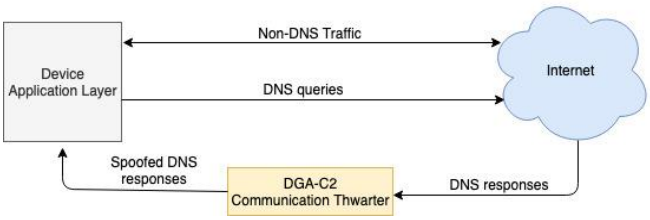

Fig. 1. Overview of DGA-C2 Communication Thwarter

allowed to perform all network actions including DNS lookups normally. The advantage of this approach is that the infected device can still continue functioning normally, or in the worst case partially, while ensuring that only communication of the device with C2 will be hampered. Such a partial shutdown is helpful when the infected systems are those performing business-critical functions.

Additionally, it can also be inferred that when a domain resolution finally succeeds for a malware process, the resolution would provide the IP address of a command and control server. The address obtained from this successful domain resolution can be blacklisted in the device firewall and all traffic from this address across the system can be blocked. This ensures that the malicious process will not be able to subvert other processes into contacting the C2 server.

This paper makes the following contributions:

- Provides a fast method to counter a DGA-based malware while limiting the impact of mitigation and keeping the infected device at least partly functional.
- Identifies potential C2 IP addresses and configure device firewall to permanently block all incoming traffic from them.

II. Related Work

Existing work on countering DGA-based malwares are restricted to identification of DGA-generated domain names, either by using machine learning techniques [1, 19, 20, 22, 23] or by manually reverse engineering the malware [27]. Domain identification and classification are intrinsically slow, while reverse engineering mechanisms are extremely time and effort consuming. The delay in detection creates a non-trivial window where the botnet is allowed to spread its infection.

Antonakakis et al. [1] introduced Pleiades which uses DNS traffic analysis to identify AGDs. Pleiades classifies domains of the NXDomain responses with labels of known DGA generated domain names. The hosts that generated the labeled domains are deemed to be infected by the corresponding DGA-based malware. Through the domain classification, Pleiades attempts to identify the C2 server of the botnet. Symantec [9] has also developed systems that employ analysis of failed DNS requests to detect malware infected devices. All of these approaches identify and blacklist the entire infected system. McAfee [10] similarly attempts to identify if a domain belongs to a DGA malware using lexical analysis.

Unlike previous works, our approach avoids rendering the entire infected host unfunctional by employing a more fine-grained tracking of network traffic. By limiting the isolation to infected processes, we ensure that a network-wide botnet infection would not imply a complete or significant network shutdown.

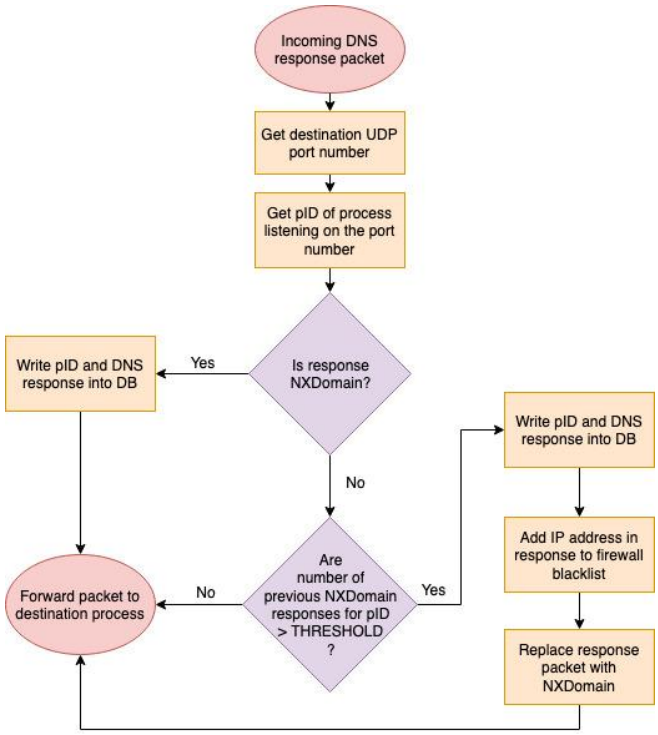

Fig. 2. Flow chart of DGA-C2 Communication Thwarter

III. Method

As observed by Antonakakis et al. [1], most of the domains generated by a DGA do not resolve successfully and their domain resolution requests result in NXDomain responses. The DGA-C2 Communication Thwarter is built on the inference from this observation that non-DGA processes are not likely to generate an abnormally large number of DNS queries that would result in domain resolution failures. Therefore, a process that receives domain resolution failures greater than some threshold is deemed to be running a Domain Generation Algorithm and attempting to receive data from such dynamically-generated domains. In addition to this, the successful domain resolutions by a process suspected to be a

Table I. Captured DNS Responses

Response	Process ID	Process Name	Count
NXDomain	11073	python	3993
91.189.91.26	16601	systemd-resol	1
208.100.26.251	16601	systemd-resol	4
52.15.158.54	16601	systemd-resol	1
172.31.40.210	16601	systemd-resol	9
91.189.92.38	16601	systemd-resol	1
91.189.92.20	16601	systemd-resol	1
91.189.92.19	16601	systemd-resol	1
52.18.210.215	16601	systemd-resol	1
NXDomain	16601	systemd-resol	7646
NXDomain	19812	http	1
NXDomain	19813	http	1

TABLE II. BLACKLISTED IP ADDRESSES

DNS Response	Is Blacklisted when THRESHOLD = 5?	Is Blacklisted when THRESHOLD = 20?	Is False Positive*?
91.189.91.26	No	Yes	Yes
208.100.26.251	No	Yes	No
52.15.158.54	No	Yes	Yes
172.31.40.210	Yes	Yes	Yes
91.189.92.38	No	Yes	Yes
91.189.92.20	Yes	Yes	Yes
91.189.92.19	Yes	Yes	Yes
52.18.210.215	No	Yes	Yes

*. IP Addresses that were manually verified and found to be non-malicious

malware would be resolutions to C2. The IP addresses in the A records of these successful resolutions are blacklisted at the device firewall.

Fig. 1 gives the system overview of the DGA-C2 Malware Thwarter.

A. Routing All Incoming DNS Responses to an Interceptor

We setup the firewall of the device to route all incoming DNS responses to our tool.

B. Process-level NXDomain Tracking

We perform the following actions for each DNS response packet that is routed to our interceptor:

a) Find the process to which the DNS response belongs: The initiator of the DNS request that belonged to the incoming response will be the process to which the latter belongs. The initiator process can be found by finding the process id of the open UDP connection whose local listening port is equal to the destination port in the DNS response packet. This provides an accurate method for finding the process id belonging to a DNS packet because the port on which a DNS request is sent out will be the same port that will be listening to the corresponding response.

b) Record pID of process and DNS response information: If the response is NXDomain, then we record the process id, process name and the DNS response information of the particular packet in our database.

c) Blacklisting suspicious processes: If the number of NXDomain responses previously received by a particular process id, as recorded in our database, is greater than a pre-defined threshold, then the process is added to a blacklist.

C. Spoofing Future Successful Domain Resolutions for Blacklisted Processes

We assume any successful domain resolution for a suspected malicious (blacklisted) process must give the IP address of an actual C2 server. Therefore, it is imperative that this process is not provided with the results of the successful DNS resolution. To achieve this, we spoof the DNS response packet by changing it into a NXDomain response and re-forwarding the modified response to the target process. We also record the IP address present in the original DNS response in our database.

All other packets and all packets of non-blacklisted processes are forwarded to the respective processes as is.

D. Block All Traffic from Suspicious IP Addresses

Since response IPs in the successful domain resolutions for blacklisted processes must likely be the addresses of the C2 servers, we mark these addresses as suspicious. We then add firewall rules to block all further traffic from these suspicious IP addresses.

E. Cleaning Stale Data

We should only be tracking DNS responses for active processes. If a process whose DNS traffic we were actively tracking has died, the operating system is free to recycle its process id for a future process. Therefore, we periodically run a job in parallel to clear our database off dead processes.

IV. EXPERIMENTAL SETUP

A. System Setup

We have setup our experimental system on an Amazon Web Services EC2 instance of Ubuntu 18.04. Despite the fact that most malware, particularly DGA-based malware are specific to Microsoft Windows operating system, we can also use Linux systems for our experiment, since the techniques we use in our software are universally applicable.

B. Murofet DGA

Since we were unable to obtain a live copy of a simple DGA-based malware [5], for the purpose of this experiment, we have used reverse-engineered DGA [4] of popular DGA-based malwares in place of the actual malware. This experimental setup should suffice to demonstrate the effectiveness of our tool since our aim is only to test the tool against the DGA component of the malware.

We have chosen to run our experiment against the DGA of version 1 of Murofet malware. Murofet, also known as LICAT, is a botnet targeting Windows systems that uses DGA to download more malicious files into the compromised system. Murofet takes the current date as the input to its DGA and generates 1020 domains a day [6].

C. Generating DNS Queries

For the experiment, we simulate Murofet's network traffic by attempting to fetch the website content for each of the domains generated by the malware's DGA using Python's urllib2 library's urlopen function. This function, internally, first performs domain lookup before proceeding to download the website contents of successfully resolved domains.

V. RESULTS

We ran the experiment with two different threshold values 5 and 20. The results of IP blacklisting for the different thresholds are presented in Table 2.

Lower threshold values were more successful in catching C2 IP addresses. This is because, for the Murofet DGA, the valid C2 domain name was one of the first AGDs generated by the domain generation algorithm. Here, DGA-C2 Communication Thwarter correctly identified malicious IP address 208.100.26.251. On manual verification it is found that this IP address is associated with multiple ransomware infrastructures [11].

However, we observe that lower threshold values also generated false positives by blacklisting a few non-malicious IP addresses. We know these IPs are non-malicious by performing reverse DNS lookups on them. In our experiment, these IP addresses belonged to AWS control servers [11, 12] and Snapcraft [13, 14, 15, 16] Linux package management. In Table 1 we see that these IP addresses were accidentally blacklisted because their DNS requests as well as DNS requests by Murofet DGA Python process were generated by the same third process, systemd-resolver. Systemd-resolved is a systemd service in Linux that performs network name resolutions on behalf of applications. These false positives in the blacklists are due to interferences from our experiment environment.

VI. Discussion and Limitations

DGA-C2 Communication Thwarter is able to detect potential DGA-based malwares and prevent them from contacting their C2 servers without interfering with other non-malicious processes. However, the accuracy of the tool is depended on the ability to make correct predictions of the threshold values. In this paper, we have not provided a method for making good threshold predictions.

Currently, DGA-C2 Communication Thwarter is likely to blacklist processes such as web-browsers, where a large number of DNS requests which result in resolution failures is a valid and normal behavior. This can be prevented by granting exceptions to such special processes and extending the tool to blindly allow all traffic for the exceptional processes.

For lower values of thresholds, the tool is likely to generate false positives for blacklisted IPs. For high threshold values, a large number of DNS responses need to be sampled before the Thwarter makes decisions for IP and process blacklisting. Hence, the tool might not blacklist the malicious process on time and therefore give false negatives.

Additionally, if the malware runs the DGA under multiple processes, the tool is unable to trace the DNS traffic of the various processes to the same malware.

Conversely, as observed in Table 1, if the same third process is servicing DNS traffic for malicious and non-malicious process, then the tool would actually block all valid DNS responses for the third process which in turn will affect DNS traffic of the non-malicious process also. For instance, in Linux, this can occur in system-resolved [7] which services all DNS requests for the OS. In this case it is difficult to identify the true source of the DNS request. We can prevent such cases of false positives by trapping the kernel calls for domain name resolutions and tracking the originating source of the DNS request or by disabling the local domain lookup services.

The tool is unable to find C2 IP addresses for malwares that use IP transformations. For instance, in the DGA-based malware Nymaim [8], the address of the C2 is obtained by performing transformation on the results of A resource records of the DNS response. However, it would still be able to blacklist the suspected malicious process.

VII. Conclusion

In this paper, we present a novel method to silo the impact of DGA-based malwares while keeping the infected device partially or completely functional, through the tool DGA-C2 Communication Thwarter. This tool, however, works best as a 'band-aid' solution or preliminary mitigation for preventing the malware from contacting its command and control server. It should be used in conjunction with detection mechanisms for Algorithmically Generated Domains to provide eventual correction of its false positives.

Acknowledgment

I would like to thank John Bambenek, President, Bambenek Consulting for his guidance and helpful feedback.

References

[1] 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 the Proceedings of 21st USENIX Security Symposium(USENIX Security '12), p 24

[2] D. -T. Truong, G Cheng, "Detecting domain - flux botnet based on DNS traffic features in managed network" in Security Comm. Networks, pp 2338-2347, 2016

[3] J. Bader Reverse Engineered Domain Generation Algorithms https://github.com/baderj.

[4] A. Menon DGA-C2 Malware Thwarter https://github.com/anjalim3/DGA-C2-Communication-Thwarter

[5] L. Zeltser Malware Samples https://zeltser.com/malware-sample-sources/

[6] J. Bader Three Variants of Murofets' DGA Blog post: https://johannesbader.ch/2015/09/three-variants-of-murofets-dga/

[7] Archlinux Documentation of systemd-resolved https://wiki.archlinux.org/index.php/Systemd-resolved

[8] J. Bader The New Domain Name Generation of Nymaim Blog post: https://johannesbader.ch/2018/04/the-new-domain-generation-algorithm-of-nymaim/

[9] W. E. Sobel [Symantec Corp] Google Patent for Systems and Methods for Detecting Malware Infections via Domain Name Service Traffic Analysis https://patents.google.com/patent/US20170155667A1/en

[10] N. Thakar, P. K. Amritaluru, V. Taneja [McAffee LLC] Google Patent for System and Method to Detect Domain Generation Algorithm Malware and Systems Infected By Such Malware https://patents.google.com/patent/US20160057165A1/en

[11] Ransomware Tracker IP Address https://ransomwaretracker.abuse.ch/ip/208.100.26.251/

[12] Amazon.com, Inc reverse DNS lookup https://ipinfo.io/52.18.210.215

[13] Amazon.com, Inc reverse DNS lookup https://ipinfo.io/52.18.210.54

[14] Canonical Group Limited reverse DNS lookup https://ipinfo.io/91.189.92.20

[15] Canonical Group Limited reverse DNS lookup https://ipinfo.io/91.189.92.19

[16] Canonical Group Limited reverse DNS lookup https://ipinfo.io/91.189.92.38

[17] Canonical Group Limited reverse DNS lookup https://ipinfo.io/91.189.92.26

[18] S. Shevchenko Domain Name Generator for Murofet Blog post: http://blog.threatexpert.com/2010/10/domain-name-generator-for-murofet.html

[19] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, E. Gerhards-Padilla "A Comprehensive Measurement Study of Domain Generating Malware" in the Proceedings of the 25th USENIX Security Symposium(USENIX Security '16), pp 263-278

[20] R. R. Curtin, A. B. Gardner, S. Grzonkowski, A. Kley-menov, and A. Mosquera, "Detecting DGA domainswith recurrent neural networks and side information" arXiv:1810.02023 [cs.CR], 2018.

[21] M. Periera, S. Coleman, B. Yu, M. De Cock, A. Nascimeto, "Dictionary Extraction and Detection of Algorithmically Generated Domain Names in Passive DNS Traffic" in the Proceedings of the 21st International Symposium on Research in Attacks, Intrusions and Defenses, Heraklion, Crete, Greece, pp. 295-314 September 2018

[22] J. J. Koh, B. Rhodes "Inline Detection of Domain Generation Algorithms with Context-Sensitive Word Embeddings" in 2018 IEEE International Conference on Big Data (Big Data), pp 2966-2971, 2018

[23] J. Woodbridge, H. S. Anderson, A. Ahuja, D. Grant "Predicting domain generation algorithms withlong short-term memory networks," arXiv:1611.00791[cs.CR], 2016.

[24] B. Yu, J. Pan, J. Hu, A. Nascimeto, M. De Cock "Character Level Based Detection of DGA Domain Names" in the Proceedings of IJCNN at WCCI2018 (2018 IEEE World Congress on Computational Intelligence), 2018

[25] R. R. Curtin, A. B. Gardner, S. Grzonkowski, A. Kleymenov, A. Mosquera "Detecting DGA domains with recurrent neural network and side information" arXiv:1810.02023v1 [cs.CR] 4 Oct 2018

[26] Damballa Inc. "DGA in the Hands of Cyber-Criminals - Examining the State of the Art In Malware Evasion Techniques" whitepaper https://web.archive.org/web/20160403200600/https://www.damballa.com/downloads/r_pubs/WP_DGAs-in-the-Hands-of-Cyber-Criminals.pdf

[27] L. I. Kuncheva Combining Pattern Classifiers : Methods and Algorithms.Wiley-Interscience, 2004.