Real-time Malicious Fast-flux Detection Using DNS and Bot Related Features

Sergi Martinez-Bea
Artificial Intelligence Research Institute, 08193 Bellaterra, Spain

Sergio Castillo-Perez
Autonomous University of Barcelona, 08193 Bellaterra, Spain

Joaquin Garcia-Alfaro
Telecom SudParis, CNRS Samovar UMR 5157, Evry, France

***Abstract*—Fast-flux is a protection technique used by botnets to protect their communication servers. We present a detection method for the real-time discovery of fast-flux services. We implemented our approach and conducted experiments that verify the superiority of our approach to previous efforts.**

***Index Terms*—Network Security, Botnets, Fast-Flux, Domain Name System, Malware**

I. Introduction

In the context of the botnets, fast-flux refers to the strategy of hiding the C&C (*Command and Control*) servers. Such servers are crucial for the life cycle of the botnet. The idea is to place the servers behind proxy nodes using the DNS (Domain Name System) protocol to map the hidden servers. This way, botnet operators often increase the robustness of their C&C services by deploying and enabling complete fast-flux service networks. The mapping of DNS names to IP addresses (e.g., via the *A* record of a DNS response) can change very quickly over the time, thereby making the botnet much more robust against countermeasures and failures of individual proxy nodes. However, and for the very same reason, the discovery of fast-flux services and their associated resources is a valuable way to discover botnet activities during the life cycle of a botnet.

The detection of fast-flux service networks is a hot research topic. There is a great number of approaches for malicious fast-flux detection, ranging from training classifiers, such as [5], [6], [11], [9], [12], to collaborative systems, such as [13], [14], [15]. To our knowledge, most relevant proposals in the literature are machine learning based. A relevant approach in this category is the work of McGrath *et al.* presented in [9]. The authors build a linear classifier grounded on Support Vector Machines (SVM) [10], and define a minimum set of features required to detect a fast-flux domain. Such features are the number of IP addresses associated to a given domain, the number of ASN (Autonomous System Numbers), the number of different prefixes, and the number of different countries that the associated IP addresses belong to. Hsu *et al.* presented in [6] an enhanced SVM classifier, whose detection features are completely different to those of McGrath *et al.* work. The new classifier bases their features in the intrinsic characteristics that bots have, such as the network delay, the request processing delay, and the document fetching delay.

In this paper, we extend these two above mentioned efforts, and present a novel approach. We show that our solution reduces the likelihood of erroneous detection, while providing better results than those two previous research efforts.

Paper organization — Section II elaborates further our motivation and provides the set of detection features. Section III presents the experimental evaluation and results. Section IV concludes the paper.

II. Our Detection Proposal

A fast-flux detector system must provide real-time decisions. This way, it is possible to warn potential victims before they connect to a malicious site. Most existing proposals in the literature rely on the number of IP addresses by querying a certain domain name, or by passively monitoring DNS queries, for a certain period of time. Thus, the time required to detect a fast flux domain of such strategies is counterproductive. At the same time, an efficient fast-flux detector system should minimize the number of erroneous detections (i.e., both false positive and negative rates). Erroneous detections are often caused by the similarities between illicit fast-flux network systems and similar (legitimate) services such as Round Robin DNS and Content Delivery Networks. Therefore, the goal of our proposal is twofold: (i) to provide a real-time detection strategy which does not require a long period of time for the detection, and (ii) to prevent erroneous recognition of legitimate DNS-based services that can be flagged as malicious fast-flux domains by mistake.

We propose the construction of a novel linear SVM classifier that extends those of McGrath *et al.* and Hsu *et al.* presented, respectively, in [9], [6]. These two approaches based their detection properties on the definition of certain fast-flux features. The main drawback of the McGrath *et al.* classifier is that it can be misled by botmasters, due to the nature of its set of detection features. For instance, the botmaster can assign less IP addresses to a domain, or use heuristics to select only those bots geolocated in the same country [7]. This would lead the McGrath *et al.* classifier to a great rate of false negatives. Contrarily, the set of features of the Hsu *et al.* classifier are intrinsic to malicious fast-flux networks and cannot be manipulated by the botmaster. Nevertheless, it also presents an important drawback. It can be misled by legitimate servers, e.g., Round Robin DNS servers or Content

Delivery Network servers, and end with a great number of false positives.

Our proposal addresses these two aforementioned limitations. First, it differentiates malicious fast-flux networks by their own features. It does so in an automatic way by using machine learning techniques to build a new SVM classifier trained via real features extracted from domains and bots. In the sequel, we list our proposed set of detection features.

A. Detection Features

Applying the features from [9] and [6] separately leads to false positives and false negatives. We propose to merge both kind of features with the aim of reducing false detection rates. The rationale is that those false positives and false negatives caused by the features of the first classifier shall be countered by the features from the second classifier, and vice versa. Based on this idea, we build a new set of features. The set can be divided in two different groups: (1) DNS-related features and (2) bot-intrinsic features. The former being features that are related to the DNS resolution process. The latter being features that are inherent to infected computers. In the sequel, we detail each of the feature sets.

1) DNS-related Features: Our proposed set of DNS-related features contains those characteristics that can be obtained by using DNS information. The information is extracted by using a DNS request issued to the authoritative name server for a given domain, and then is processed to obtain the features. The features that are contained in this group are the minimum set of DNS-related features needed to detect fast-flux [9]. We describe some sample DNS-related features next.

- **Number of IP addresses associated to the same domain:** Conventional legitimate domains usually have either one or two IP addresses associated to them. In fast-flux networks (legitimate or not) and similar technologies, such as CDNs (Content Delivery Networks) or RRDNS (Round Robin DNS), the number of associated IPs tends to be much higher. In fact, fast-flux, CDNs and RRDNS use multiple IP addresses for a given domain, with the goal of providing high availability and greater performance to the end user. This feature, then, tends to discriminate those conventional legitimate domains from fast-flux networks, CDNs, and RRDNS.
- **Number of associated Autonomous System Numbers:** The servers associated to a given conventional legitimate domain are usually located within the same autonomous system, as they are usually managed by the same company. In botnets, however, this is not the case. They are composed by infected domestic host computers that are spread across the world. Regarding CDNs or RDDNS, they exhibit the same behavior as legitimate domains, i.e., appear as a single vantage point within a unique autonomous system.
- **Number of associated prefixes:** IP address prefixes also give information about whether a domain is either legitimate or part of a fast-flux service. The IP addresses of hosts of legitimate networks usually belong to a few BGP prefixes per hostname, while in networks exhibiting fast-flux are usually associated to multiple BGP prefixes per hostname.
- **Number of associated countries:** As in the case of the Autonomous System Numbers, the servers associated to a legitimate domain are typically located within the same country. In fact, hosts belonging to a particular country code TLD (Top Level Domain) are typically located on IP addresses physically residing within that country. However, hosts of fast-flux domains are typically spread around the world. Therefore, a hostname associated to multiple countries is likely to be part of a fast-flux service.

2) Bot-intrinsic Features: The bot-intrinsic features are those strongly related to the characteristics of the compromised machines, that is, the bots. In this group of features we assume that botnet owners exploit the bots to execute web-based malicious services such as phishing pages and malware delivery sites. Therefore, the malicious software operating on each bot is assumed to provide an HTTP service and related flows.

Remember that botnets are typically formed by malware-infected home computers. Usually, there are big differences in hardware and software between home computers and dedicated hosting servers. Dedicated hosting servers are much more powerful, and connected to Internet via high bandwidth connections in order to obtain the best possible performance. Their running processes are those dedicated to provide web services. On the contrary, home computers have a more limited hardware, the bandwidth of their connection is also much more limited, and they run all kinds of software. These differences can be used to extract features to help discriminate legitimate domains from fast-flux domains. We describe some sample bot-intrinsic features next.

- **Network delay:** Refers to the time required to transmit packets back and forth over the Internet between a client and a server. It can be obtained by computing the difference between the time a client sends out the first `TCP SYN` packet to the server and the time the client receives the corresponding `TCP SYN+ACK` packet from the server.
- **Processing delay:** Refers to the time required for the server to process an erroneous HTTP request that does not incur any additional computation and I/O operations. Its measurement is done by sending out an HTTP request with an undefined method, such as a nonsense `BADMETHOD` method, and computing the difference between the sending time of the non valid request and the time when a 400 (Bad request) or 405 (Method Not Allowed) response is received. Then, the network delay has to be subtracted to this value, in order to obtain the processing delay.
- **Document fetch delay:** Refers to the time required from the server to fetch a web page, either from a hard disk or from a backend mothership. The fetch operation

TABLE I
DATA SET EXAMPLE

Domain	#IP	#ASN	#PREF	#C	ND	PD	DFD	Label
adultdatinghouse.info	3	3	3	3	0.1362	1.5847	1.5117	malicious
google.com	11	1	1	1	0.0481	0.0280	0.2593	legitimate
cosmodatelab.info	3	3	3	3	0.1160	0.2609	1.1189	malicious
cupidlocals.com	2	2	2	2	0.4668	0.1136	0.1718	malicious
youtube.com	11	1	1	1	0.0460	0.0314	0.3419	legitimate
yahoo.com	3	3	3	1	0.2443	0.3965	0.5745	legitimate

occurs at the server side and we cannot know exactly what happens, we compute it by doing the following. We compute the time difference between the send out of an `HTTP GET` request and the time the client receives the corresponding HTTP response (200 OK), and then subtracting the network delay. This way, we obtain an estimator of the document fetch delay.

B. Building the SVM Classifier

In order to detect fast-flux behavior we build a linear classifier by using the features presented in Section II-A. The linear classifier used is based on Support Vector Machines (SVM) [10]. The result is a non-probabilistic binary classifier that constructs a hyperplane in a very high-dimensional space. It achieves a good separation when the hyperplane have the largest possible distance to the nearest training data point.

More formally, given some training data $\mathcal{D}$ represented as a set of n points of the form described in Equation (1), where y_i indicates the class to which x_i belongs to. We want to find a hyperplane with the maximum margin that divides those points having $y_i = 1$ from those having $y_i = -1$.

$$\mathcal{D} = \{(x_i, y_i) \mid x_i \in \mathbb{R}^p,\ y \in \{-1, 1\}\}_{i=1}^{n} \tag{1}$$

When the data to be classified is linearly separable, as it is our case, two planes can be selected so they separate the data points without having any point between them, trying to maximize their distance. Those hyperplanes can be described by Equations (2) and (3).

$$w \cdot x - b = 1 \tag{2}$$

$$w \cdot x - b = -1 \tag{3}$$

To prevent data points from falling into the margin, the constraint described by Equation (4) is added.

$$y_i(w \cdot x_i - b) \geq 1, \text{ for all } 1 \leq i \leq n \tag{4}$$

Given that the distance between the two planes is $\frac{2}{||w||}$, the goal is to minimize the value of $||w||$ subject to the constraint described by Equation (4).

III. EXPERIMENTS

To validate our SVM classifier, we retrieved some sample lists of malicious fast-flux domains as well as legitimate domains. We used these lists to extract the features presented in Section II and label them accordingly (i.e., labeling each domain as malicious or legitimate domains). We obtained a list of 81 active malicious fast-flux domains from the *Atlas* Web site [2], and a list of 81 legitimate active benign domains from the *Alexa Top Sites* site [1]. These lists were processed in order to extract the features. For that purpose, we built a set of Python scripts. To obtain the different DNS related features, the script issues DNS requests in order to obtain the NS records as well as the A records. Then, using the IP addresses obtained with the requests and connecting to the *cymru* whois service [3], their respective autonomous systems, countries and prefixes are obtained. To obtain the bot-intrinsic features, the script sends a `TCP SYN` packet in order to measure the network delay, an `HTTP GET` request to measure the document fetch delay, and an invalid `HTTP BADMETHOD` request to measure the processing delay. The measurements of the delays are repeated 10 times and then the average is computed. An example of the data set can be found in Table I.

Once the data sets have been collected, the SVM classifier —written in Java and using the library Java-ML [4]— is validated by using the k-fold cross-validation method [8]. With this method, the data sets are split in k different groups, using one of them to train the classifier, and the other $k - 1$ to classify. This is repeated other $k - 1$ times, so that every time the training is done with a different group. The results obtained from these experiments are presented next.

Obtained Results: Table II outlines the corresponding number of true positives, false positives, true negatives, and false negatives associated to each of the three evaluated classifiers (i.e., McGrath [9], Hu [6], and ours). Our proposal obtains better results than just using the features from [9] and [6] separately. Therefore, and as expected, by combining the two kind of features we noticeably reduce the false positives and

TABLE II
COMPARATIVE STUDY BETWEEN [9], [6] AND OUR PROPOSAL

	McGrath [9]	Hsu [6]	Our proposal
True Positive	78	63	81
False Positive	4	9	1
True Negative	77	72	80
False Negative	3	18	0

the false negatives, at the same time that we slightly increase the true positives and true negatives. One can observe that our proposal provides only 1 misclassified results, which happens to be a false positive. Thus, all the malicious fast-flux domains were correctly detected, and only 1 legitimate domain is misclassified as a malicious fast-flux domain from our classifier. In order to clarify the weaknesses of the strategies proposed by McGrath and Hsu, we analyzed their corresponding false positives and negatives obtained in our experimental results. We discovered that such misclassifications were deeply tied with the features used by them, and not with the classifier itself.

On one hand, the McGrath's classifier exhibited only one false positive that was caused by a legitimate domain with a set of 7 IP addresses associated to 6 different autonomous systems, 7 different prefixes, and distributed along 5 different countries. This is not a common characterization of a legitimate domain, and this was the reason why the classifier considered it as a fast-flux domain. With respect to the false negatives, all the malicious domains classified erroneously had a reduced number of IP addresses, autonomous systems, prefixes and countries. This confirms our hypothesis that a malicious botmaster can create a fast-flux domain with a particular chosen set of features, and whose aim is to evade any detection based on using only DNS-related information.

On the other hand, the false positives obtained after using the Hsu proposal were caused by high values of network delays, processing delays and document fetch delays. Probably, such values were a consequence of a network problem (e.g. network congestion) or a high resource consumption from the server side. This corroborates that by using the bot-intrinsic features in an isolated manner cannot be sufficient for an efficient detection, since some random perturbation from the network or server standpoint can lead to undesirable misclassifications. From the false negatives point of view, we observed that the delay values associated to such fast-flux domains were low enough to consider them as benign domains. It is likely that the botmaster in charge of those fast-flux domains built them by using bots with a good network and server resources, leading to an evasion mechanism.

After analyzing how a malicious fast-flux domain could evade the proposals of McGrath and Hsu independently, one could think that the unification of both evasion strategies could also be applied to elude the detection process of our classifier. However, although theoretically possible, our experimental results show that none of the studied domains revealed yet such a behaviour. In fact, what our work shows is that the combination of features increases the difficulty of constructing a malicious fast-flux domain that is able to evade the detection. The resulting combination has proved to be much more robust against false positives introduced by legitimate values related to the DNS features, or by random noise in the delay measurements. This is possible since the DNS features counterbalance the bot-intrinsic features –or vice versa– in case of a false positive related with one of both scopes.

IV. Conclusion

Botnets use fast-flux as an evasion strategy to make difficult the trace-back and posterior take down. Despite that, there are also legitimate fast-flux networks, as well as similar technologies such as round robin DNS and content delivery networks. Detecting malicious fast-flux networks implies being able to discriminate them among those similar technologies. Most approaches in the literature use detection features that may mislead the discovery process and end with high rates of false positives and false negatives. We have extended two existing classifiers that suffer from such limitations, and conducted simulations that verify the feasibility and superiority of our approach.

Acknowledgments: This work was partially supported by a student grant of the *Master in Security of Information and Communication Technologies* (MISTIC), at the Autonomous University of Barcelona (UAB) through; and by the Spanish Government projects TSI2007-65406-C03-03 E-AEGIS, TIN2011-27076-C03-02 CO-PRIVACY, TIN2010-15764 N-KHRONOUS, and CONSOLIDER INGENIO 2010 CSD2007-0004 ARES grants.

References

[1] Alexa Top 500 Global Sites. http://www.alexa.com/topsites.

[2] Arbor Summary Report on Global Fast Flux. http://atlas.arbor.net/summary/fastflux.

[3] Team Cymru IP to ASN Lookup v1.0. http://whois.cymru.com.

[4] T. Abeel, Y. Van de Peer, and Y. Saeys. Java-ML: A Machine Learning Library. *J. Mach. Learn. Res.*, 10:931–934, June 2009.

[5] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling. Measuring and Detecting Fast-Flux Service Networks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2008.

[6] C.-H. Hsu, C.-Y. Huang, and K.-T. Chen. Fast-Flux Bot Detection in Real Time. In S. Jha, R. Sommer, and C. Kreibich, editors, *Recent Advances in Intrusion Detection*, volume 6307 of *Lecture Notes in Computer Science*, pages 464–483. Springer Berlin / Heidelberg, 2010.

[7] M. Knysz, X. Hu, and K. Shin. Good guys vs. Bot Guise: Mimicry attacks against fast-flux detection systems. In *INFOCOM, 2011 Proceedings IEEE*, pages 1844–1852, april 2011.

[8] R. Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *International Joint Conference on Artificial Intelligence*, pages 1137–1145, 1995.

[9] D. McGrath, A. Kalafut, and M. Gupta. Phishing Infrastructure Fluxes All the Way. *Security Privacy, IEEE*, 7(5):21–28, sept.-oct. 2009.

[10] I. Steinwart and A. Christmann. *Support Vector Machines*. Information Science and Statistics. Springer, 2008.

[11] Y. Xu, Y. Lu, and Z. Guo. The Availability of Fast-Flux Service Networks. In *Mobile and Wireless Networking (iCOST), 2011 International Conference on Selected Topics in Mobile & Wireless Networking*, pages 89–93, oct. 2011.

[12] X. Yu, B. Zhang, L. Kang, and J. Chen. Fast-Flux Botnet Detection Based on Weighted SVM. *Information Technology Journal*, 11(8):1048–1055, 2012.

[13] C. Zhou, C. Leckie, S. Karunasekera, and T. Peng. A Self-Healing, Self-Protecting Collaborative Intrusion Detection Architecture to Trace-Back Fast-Flux Phishing Domains. In *Network Operations and Management Symposium Workshops, 2008. NOMS Workshops 2008. IEEE*, pages 321–327, april 2008.

[14] C. V. Zhou, C. Leckie, and S. Karunasekera. Collaborative Detection of Fast Flux Phishing Domains. *Journal of Networks*, 4(1):75–84, Feb. 2009.

[15] C. V. Zhou, C. Leckie, and S. Karunasekera. A Survey of Coordinated Attacks and Collaborative Intrusion Detection. *Computers & Security*, 29(1):124–140, 2010.