Good Guys vs. Bot Guise: Mimicry Attacks Against Fast-Flux Detection Systems

Matthew Knysz, Xin Hu, Kang G. Shin

University of Michigan, Ann Arbor

Abstract. Fast-Flux (FF) service networks are botnet-based hosting or redirection/proxy services for hosting malicious and illegal content while affording botmasters a high level of misdirection and protection. With their use as service networks among criminals on the rise, researchers and security experts have designed fast and accurate detection systems based on their intrinsic behavior patterns. However, botmasters have responded, adopting a plethora of countermeasures to evade detection. In this paper, we explore the escalating "arms race" between FF botnet detectors and the botmasters' effort to subvert them, presenting several novel mimicry attack techniques that allow botmaster to avoid detection. We first analyze the state-of-art FF detectors and their effectiveness against the current botnet threat, demonstrating how botmasters can—with their current resources—thwart detection strategies. Based on the realistic assumptions inferred from empirically-observed trends, we create formal models for bot decay, online availability, DNS-advertisement strategies and performance, allowing us to compare how different mimicry attacks affect the overall online availability and capacity of botnets.

1 Introduction

A botnet is a vast collection of compromised computers under the control of a botmaster utilizing a Command-and-Control (C&C) infrastructure. Among the numerous criminal uses of botnets, one of the more advantageous is the botnet-based hosting service, which proxies or redirects unsuspecting users to illegal or nefarious content. This strategy grants criminals a high level of anonymity while enabling easy and centralized management of the malicious content. However, because botnets are composed of thousands of disparate compromised systems from around the globe, all with varying resources and network connectivity, it is not uncommon for them to unpredictably go offline. To prevent bots' unreliable connectivity from disrupting the availability of their nefarious service/content, botmasters must continuously replace offline IPs in their malicious domain's DNS records with those of online bots. As a result, they adopt fast-flux (FF) DNS techniques, which frequently change the domain-name mappings to different bots' IP addresses. By recruiting a large pool of IPs and supplying a large number of IPs per query, botmasters can ensure, with high probability, that the malicious domain resolves to an online bot's IP. Using this FF technique, botmasters effectively turned their botnets into a global Content Delivery Network (CDN), providing highly available and reliable content-hosting services despite frequent node failures/disconnectivity.

While their tremendous success as service networks has spurred researchers and security experts to develop novel ways for detection, FF botnets remain a persistent threat.

The advent of fast, reliable FF detection systems has not yet eradicated FF botnets; rather, it coaxed them to evolve, developing more robust, efficient, and stealthy mechanisms for subverting detection. This cycle continues, with defenders and botmasters caught in an ever-escalating "arms race", each side temporarily one-upping the other in a constant game of give-and-take. Unfortunately for the good guys, bots are free, easy to come by, capable of giving significant amounts of coordinated processing power, and incredibly effective sources of revenue. They have, in many ways, transformed the malware landscape from a place to showcase "1337 h4x0r" skills to a bustling underground economy.

It has come to our attention during our global monitoring of FF botnet domains, that despite the detection mechanism or mitigation strategy imposed, botnets constantly evolve methods for subverting them, often at a startling pace. In spite of newly-emerging detection strategies uniquely targeting them, they are still around and continue to grow into ever more formidable systems—in many ways resembling enterprise-level CDNs. This occurs because, while efficient when first introduced, many FF detection systems quickly become outdated; they are designed to detect the current advertising strategies of FF botnets, which are all too easily and quickly adapted to avoid detection. It is not sufficient to base FF detection on the current class of differentiating features. Instead, improvements could be made if the botnets' limitations were also taken into consideration. Thus, while previous research has focused on identifying behavioral features uniquely intrinsic to FF botnets for detection, we have decided to take an alternate approach; to better know our enemy, we will become our enemy. Analyzing the resources currently available to FF botnets, we developed models for their bot decay, online availability, DNS-management strategies, and performance. Using these models, we examined the potential success of novel mimicry attacks against state-of-art FF detection systems, demonstrating that such attacks are easily within current botnets' means. We do this, not to help FF botnets circumvent detection systems (they know how to do this already, quite skillfully), but to hopefully provide insight into the current arsenal they have at their disposal and how it can be—and is being—applied to defeat current detection strategies. We hope this will foster improvements to existing systems, as well as provide new insight into the adaptive limitations of FF botnets.

2 Background

In this section, we investigate the DNS IP-advertisement patterns of different domain types, including both malicious FF domains and two benign domain types. First, we will describe how we set up a globally distributed DNS monitoring system and then discuss unique features discovered through several months' monitoring of thousands of domains. This provides us with a unique, global perspective of how the different types of domains advertise their IP addresses to DNS servers; allowing us to better understand the current state of FF domains and design effective mimicry attacks.

2.1 Global DNS-Monitoring System

We created a distributed DNS-query engine called *DIGGER*, deployed on 312 geographically disparate nodes in the PlanetLab testbed [9]. The nodes were chosen based on the

location of the DNS servers they queried, such that DIGGER would issue queries to DNS servers in different geographic locations around the world. Table. 1 shows the continental distribution of DIGGER nodes, which is reflective of the overall distribution of available PlanetLab nodes.

Continent	N. America	Europe	Asia	S. America	Oceania	TOTAL
DIGGER Nodes	143	94	46	19	10	312
% of TOTAL	45.83%	30.13%	14.74%	6.09%	3.21%	

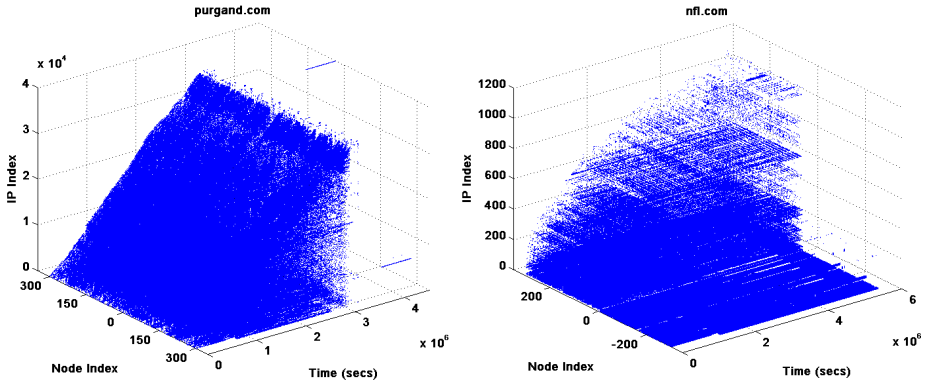
Table 1: Global distribution of DIGGER nodes by continent

On each node, for malicious and benign domains, DIGGER performs DNS queries on their A (address) records, NS (authoritative name server) records, NA records (A records on name servers) and the *reverse DNS* (rDNS) lookup (i.e., PTR records) for the A and NA record IPs. Based on a domain's most recently returned DNS-query results, DIGGER classifies the domain as either active or offline.[1] DIGGER continues to dig active domains periodically based on their observed TTL, eliminating wasteful DNS-queries while ensuring fresh DNS-query results. Domains that have been determined to be offline are intermittently dug every 24 hours, so that DIGGER can discover if they come back online. Meanwhile, for each domain, DIGGER also collects connectivity information on both A and NA record IPs by attempting to establish TCP connections on ports 80 and 53. Notice that although DNS primarily uses UDP on port 53 to serve requests, DNS servers also accept TCP connections in order to support response data exceeding 512 bytes or for tasks such as zone transfer [5]. Based on the connection results, DIGGER classifies each IP as either online or offline, which is used to derive an accurate bot online-decay model. We aggregate the global DNS-query results for domains compiled from multiple sources, including online repositories of phishing and malware websites as well as the top 1000 most popular domains. DIGGER has been deployed and gathering global DNS data for over 4 months. By applying simple heuristics on the data, we manually identified and verified 35 CDN domains and 45 FF domains by looking for IP addresses with rDNS names indicative of popular CDN companies (e.g., Akamai) or compromised computers. While FF domains may not always use IPs that return rDNS results, over our 4-month monitoring period, it becomes highly likely we will observe at least one bot IP with a rDNS result indicating a compromised computer.

2.2 Domain Types

Fast-Flux Domain FF domains are malicious domains utilizing a FF DNS-advertisement strategy, typically built atop botnets. These domains are often used for phishing scams or hosting malicious contents. Thus, the profits botmasters can gain from their botnets depend directly on the availability of the hosted services/content. However, because botnets are composed primarily of compromised home computers with unreliable connectivity, it is not uncommon for them to unpredictably go offline (e.g., the computer is turned off or the installed malware is discovered and removed). To ensure the availability and stability of the hosted service/content, botmasters adopt FF DNS techniques and advertise numerous IPs in their DNS-query results with frequently changing mapping between the domain name and different bots' IP addresses.

[1] a domain is offline if its DNS query returns no A record.



(a) FF domain (b) CDN domain

Fig. 1: Global DNS-query results for Fast-Flux and CDN domains

Figure 1-a illustrates the global IP usage—across all DIGGER nodes—for an example FF domain. In the figure, the *Time* axis represents the time (in seconds) since DIGGER started monitoring the domain; *Node Index* represents the PlanetLab node that the IP was observed on, with positive values indicating an A record IP and negative values an NA record IP; *IP Index* is a unique index incrementally assigned to each newly-observed IP. From the figure, we can see that FF domains slowly and nearly continuously accrue unique IPs (in its A, NA or both records) over its entire online lifetime. Over the 4-month monitoring period, we have observed that a typical FF domain usually advertises thousands of unique IP addresses, with the most aggressive botnets advertising over 35,000. As we will demonstrate later, this huge IP pool affords the botmaster great flexibility and abundant resources to mimic a wide range of benign DNS behaviors for evading detection.

Benign Domains CDN domains are benign domains that use a Content Delivery Network (CDN), such as Akamai, to improve the delivery of their content. CDNs—consisting of a system of computers networked together for the purpose of improving the performance and scalability of content distribution—produce DNS-query results resembling those of malicious FF domains: numerous, changing IPs per query with short TTL values. For instance, *nfl.com*, a CDN domain shown in Fig. 1-b, has very short TTL (20 seconds) and constantly changes its A record IPs, resulting in the accumulation of almost 1200 IP addresses during our monitoring period. This affinity between CDN and FF domains is a consequence of their similar goal to provide reliable content delivery despite node failures, as well as their shared assumption that any node can temporarily or permanently fail at any time. This affinity can allow botmasters to cloak their malicious DNS advertise strategy as normal behavior.

Non-CDN domains are benign domains that do not use a CDN for delivery of their content. Typically, non-CDN domains use a few stable content servers and a modest number of name servers (NSes). Some popular non-CDN domains may advertise more than 18 IPs in a single DNS query, using the same set of IPs in each query and rotating the order across queries for load-balancing purposes. This type of DNS strategy is often referred to as *round robin DNS* (RRDNS).

3 Fast-Detection Systems (2 queries)

3.1 Good Guys

The original FF detection system proposed by Holz *et al.* [1] (i.e., Holz detector) and RB-Seeker's first-tier detector [2] are considered fast-detection systems, as they are capable of detecting FF domains with high accuracy from only 2 DNS queries. This is achieved through the use of a linear decision function containing weighted terms derived from the DNS queries and a bias term. The functions for the Holz detector and RB-Seeker's first-tier detector can be found in Eq. (1) and Eq. (2), respectively.

$$f(x) = 1.32 \cdot n_A + 18.54 \cdot n_{ASN} - 142.38 \tag{1}$$

$$f(x) = -1.257 \cdot N_{unique\_IPs} - 26.401 \cdot N_{ASN} - 13.024 \cdot N_{DNS\_bad\_words} + 162.851 \tag{2}$$

In Eq. (1), the number of unique A records and Autonomous System Numbers (ASNs) are represented by n_A and n_{ASN}, respectively. In Eq. (2), N_{unique_IPs} represents the number of unique IPs seen in the A records, N_{ASN} the unique ASNs, and $N_{DNS\_bad\_words}$ the number of *reverse DNS* (rDNS) lookups (i.e., PTR records) containing "bad words" indicative of compromised home computers, such as comcast, charter, dynamic, dialup, etc. In both equations, the magnitude of $f(x)$ represents the degree of confidence when classifying domain x as FF or benign, with positive values indicating a FF domain for Eq. (1) and a benign domain for Eq. (2).

3.2 Bot Guise

ASN-Mimicry Attack From Eqs. (1) and (2), we can see that the dominant factor in identifying FF domains is the number of unique ASNs; for RB-Seeker, it is twice as influential as $N_{DNS\_bad\_words}$, and for both detectors, it is an order-of-magnitude more significant than the number of unique A records (i.e., unique IPs). Clearly, an effective mimicry attack against these fast-detection systems should reduce the number of ASNs to levels seen for benign domains.

Since DNS queries on benign domains (e.g., *www.avast.com*) often contain A record IPs from 2 ASNs, let us assume that a fast-detection system adopts the following overly strict policy: *over 2 DNS-queries, any domain containing IPs from more than 2 ASNs will be flagged as malicious.* This policy will result in false-positives for benign domains with IPs from more than 2 ASNs, such as some CDN domains. However, if this Draconian approach can be effectively subverted, so can more lenient constraints.

To discover if this is feasible with current botnet resources, we aggregated the IPs for each FF domain globally monitored by DIGGER, determined their ASNs, and then analyzed their IP distribution across ASNs. We found that, despite the size of the botnet, the distribution was long-tailed, with at least one ASN containing a disproportionably large number of IPs. This trend is demonstrated in Fig. 2 for 3 representative FF domains of varying sizes; to keep the graph readable, we have only plotted the distribution for the top 20 ASNs from which the botnets have the most IPs. While there are many ASNs from which the botnets control moderate to few IPs, there invariably exists at least one

ASN with a large number of IPs. This is possibly due to certain ASNs containing a large proportion of vulnerable computers, such as Internet cable providers, or from botmasters targeting certain institutions. In any case, assuming botnets contain a suitable number of IPs from at least a single ASN (as our data indicates), there is a simple IP-advertisement strategy for mimicking the ASN behavior of benign domains.

This mimicry strategy is demonstrated in Fig. 3, with each TTL (i.e., fresh DNS query) showing the distribution of IPs from various ASNs. For example, at TTL1, the majority of advertised IPs are from AS1 with a smaller subset from AS2. In this case, the botnet controls a large number of IPs from AS1 and a moderate to small number of IPs from AS2. During the next TTL, some of the IPs that have gone offline are replaced with new IPs from either AS1 or AS2. While there exists a sufficiently large pool of online IPs from AS1, this is not the case with AS2, eventually requiring the introduction of IPs from a different ASN. However, because the detection window is 2 DNS-queries, the botmaster must ensure that all the IPs seen over 2 consecutive queries belong to no more than 2 ASNs. Thus, before IPs from a new ASN can be introduced, she must first advertise only IPs belonging to one of the ASNs present in the previous TTL, as shown in TTL3. Then, at TTL4, she is free to utilize IPs from the new ASN, AS3. If she happens to control a large number of IPs in AS3, she can slowly replace AS1 as the dominant ASN, as shown in TTL4–TTL7. In this way, botmasters can successfully mask their use of numerous ASNs from fast-detection systems.

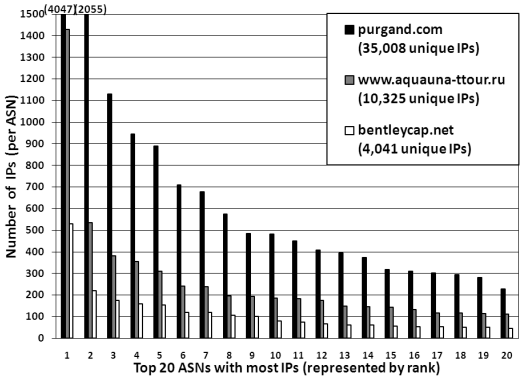


Fig. 2: IP distribution for top 20 ASNs

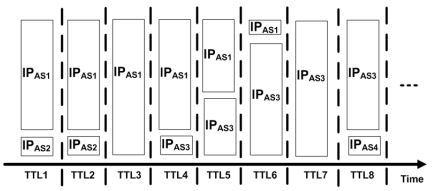


Fig. 3: ASN-mimicry strategy (2 DNS queries)

rDNS-Mimicry Attack From Eq. (2), we see that the second most influential term when identifying FF domains is $N_{DNS\_bad\_words}$. However, RB-Seeker asserts that a rDNS lookup on an IP will not always return a result, although when it does, it can be useful. Despite its inconsistency, the term is still an order-of-magnitude more important than the number of unique IPs. Therefore, an effective mimicry attack should include a mechanism for subverting this detection metric.

Let us assume the following aggressive detection policy: *over 2 DNS-queries, any domain with more than 2 "bad words" in its rDNS results will be flagged as malicious.* Certainly, this policy is overkill, as many legitimate domains (e.g., *www.comcast.com*) will have rDNS results that contain "bad words". However, if botnets can defeat this harsh

limitation, more realistic thresholds can also be subverted. If current FF botnets contain enough IPs without rDNS results (i.e., rDNS=NONE IPs), then a mimicry strategy similar to that proposed for ASNs in Section 3.2 could be applied. To determine the feasibility of this approach, we aggregated the IPs for each FF domain monitored globally by DIGGER and determined the percentage of rDNS=NONE IPs. We discovered that for all the FF domains, at least 15% of their total IPs lacked a rDNS result. Furthermore, for ≈24% of the domains, over 50% lacked a rDNS result. Considering the large proportion of rDNS=NONE IPs and the fact that rDNS results for bots that aren't compromised home computers will be free of "bad words," the mimicry strategy proposed earlier for ASNs can easily be applied: IPs without rDNS results (or without "bad words") can be used in conjunction with IPs containing "bad words", such that only 2 "bad words" are observed over 2 queries.

However, to be truly effective, we need to ensure that this strategy can be combined with the previous ASN-mimicry attack. Thus, for each FF domain, we analyzed the distribution of rDNS=NONE IPs across ASNs, once again observing the long-tailed distribution. This phenomenon is shown in Fig. 4 for 3 representative domains of varying sizes. When only looking at the rDNS=NONE IPs, some of the smaller, shorter-lived botnets would have a hard time achieving the combined mimicry attack without diligent maintenance. However, this was not the case for the majority of botnets we observed, which possessed enough IP-dense ASNs to sufficiently mount the dual mimicry attacks.

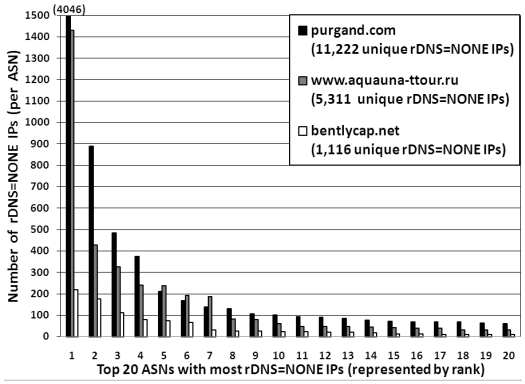


Fig. 4: rDNS=NONE IP distribution for top 20 ASNs

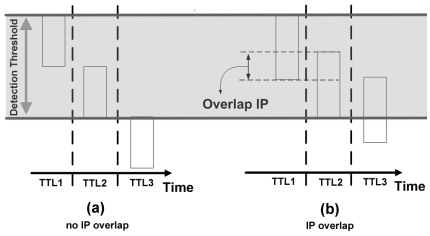


Fig. 5: DNS IP-advertising strategies

IP Mimicry Having determined that current botnet resources are capable of instigating ASN- and rDNS-mimicry attacks, we turn our attention to the final attribute utilized by the fast-detection systems in Eqs. (1) and (2), the number unique IPs. It stands to reason that the more IPs a FF domain advertises per query, the more likely some of the bots will be online. Furthermore, because most DNS servers perform round-robin scheduling within a given TTL, advertising more IPs per query decreases the load imparted on each bot, thereby increasing the botnet's total service capacity. Since benign non-CDN domains often advertise a large number of stable IPs (e.g., *hostingprod.com* uses 18 IPs per DNS query), FF domains are afforded a fair amount of freedom in the number of bots they can advertise; this is supported by Eqs. (1) and (2), where the number of unique IPs

is the least influential classification feature. However, benign non-CDN domains advertise the same set of IPs for every DNS-query, causing their total unique IPs to remain bounded and facilitating the use of a maximum IP threshold for detection.

Regardless of the chosen detection threshold, there are two basic strategies available to FF domains performing an IP-mimicry attack. The first, shown in Fig. 5-a, has no IP overlap, with the botnet advertising a completely new set of IPs every TTL. The alternate strategy, shown in Fig. 5-b, has IP overlap, with some of the IPs being advertised for multiple TTLs. Each strategy has certain pros and cons. Having no IP overlap allows for the rapid replacement of offline IPs, increasing the availability of advertised bots. However, as can be seen from Fig. 5, this reduces the number of IPs that can be used for any given TTL, which, in turn, increases the load per bot and decreases the botnet's service capacity. On the other hand, with an increase in IP overlap, more IPs can be advertised per TTL, decreasing the load per bot; however, this reduces the rate at which offline IPs can be replaced, resulting in a greater proportion of dead bots and failed victim connections. Considering bots' unreliable connectivity, finding the optimal IP-advertisement strategy for FF domains requires a better understanding of the underlying bots' online availability.

Bot Online-Decay Model We developed an accurate bot online-decay model, $P_{online}(t)$, to predict the probability a bot will be online after time t. In building the model, we first aggregated all the bot IPs seen for FF domains globally monitored by DIGGER, recording the time they were observed and if they were online and reachable at that time (i.e., a connection could be established). Unfortunately, DIGGER only observes the IPs of bots that have been advertised to the queried DNS server, resulting in a partial view of the botnet. Furthermore, to be efficient, DIGGER only performs queries when the domains' TTLs have expired, ensuring fresh results. It is at this time that DIGGER performs an online test by attempting to connect to the bots seen in the query. Thus, our view of the bots' online time is at the granularity of the FF domain's TTL. However, this shortcoming can be overcome due to the observation that many botnets are used for multiple online scams; thus, many of the same bot IPs will be observed in queries on different FF domains. Additionally, DIGGER is a globally distributed system on a shared resource (i.e., PlanetLab). As such, DIGGER nodes will perform queries for the same domain at slightly different times, depending on the other PlanetLab workloads vying for process time. By combining all available data points for each bot IP—regardless of the DIGGER node's location or the FF domain it was observed for—we can build a fairly complete picture of the online times of bots currently used by FF domains. When analyzing the data, if an IP is not seen by any DIGGER node for over 12 hours, we assume that it has gone offline during that time. The resulting bot online-decay model has a long-tailed distribution, with a non-zero probability that some bots will remain continuously online for over 2 months. In Fig. 6, which plots the first 72 hours of this model, the y-axis represents the probability that a bot is continuously online for more than some time t, represented by the x-axis. From the plot, it is clear that the probability of a bot being online decays exponentially with time, such that, after a day, there is less than a 10% chance it's still online. These findings reassert the notion that a bot's connectivity is highly unreliable, resulting from the varied usage patterns of the compromised computers' owners.

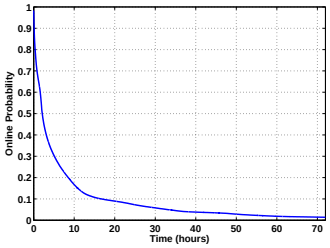


Fig. 6: Bot online-decay model (first 72 hours)

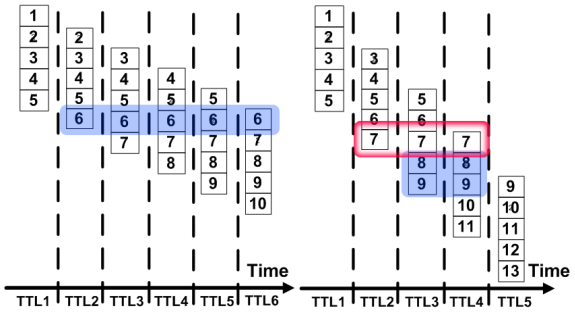


Fig. 7: Persistence of overlapped IPs

Performance Model Using our online-decay model, we can determine the optimal IP-mimicry strategy in terms of performance, which we evaluate based on the number of victim connections per unit time the botnet can handle. If the mimicry attack drastically reduces this amount, then the bots will become overwhelmed, resulting in dropped connections and decreased revenue for the botmaster. Table 2 defines the terms we will be using throughout the evaluation process. We assume both the inter-arrival time of victim connections and the bots' service time are Poisson processes with Markovian (i.e., exponential and memoryless) distributions; therefore, they have a cumulative distribution function of $A(t) = 1 - e^{-xt}$ and a probability density function of $a(t) = xe^{-xt}$, where t is time and $x = \lambda$ or μ for the inter-arrival and service times, respectively. Within a given TTL, most DNS servers perform round-robin scheduling when responding to DNS-queries. As a result, incoming victim connections will be evenly dispersed among the *online* bots advertised for that TTL. While each bot's processing and network resources will vary, the distribution will be essentially random and we can treat the online IPs advertised in a given TTL as N_{online} parallel and identical servers. We can then calculate an *individual* online bot's incoming connection rate λ^i as the ratio of the total rate λ to the number of online bots for that TTL. Using λ^i, we can model the online bots as N_{online} identical M/M/1/K queues, where K is the online bots' queue length, that is, the maximum connections each can queue before dropping additional connections. Applying queuing theory to this model, we can calculate the *connection loss probability*, i.e., the probability that an online bot will drop connections due to a full queue as:

$$P_{loss} = \begin{cases} \frac{\rho^K - \rho^{K+1}}{1-\rho^{K+1}} & : \rho \neq 1 \\ \frac{1}{K+1} & : \rho = 1 \end{cases} \qquad \text{where } \rho = \frac{\lambda}{N_{online} \cdot \mu}$$

Because we must assume that each online bot is identical, an individual bot's P_{loss} is equivalent to that of the entire botnet, allowing us to compare the various IP-mimicry attacks' performance; a higher probability of dropped connections results in fewer exploitable victims and decreased revenues.

DNS-Strategy Model Before we can successfully use this model for performance comparisons, we must be able to estimate the potential number of online IPs during a given TTL. This requires a formal relationship between an IP-mimicry attack's DNS-advertisement strategy and our online-decay model, $P_{online}(t)$, which predicts the probability a bot will still be online after time t. This relationship is straightforward when there is no IP overlap, as in Fig. 5-a. Since each TTL contains a fresh set of IPs under this strategy (i.e.,

T_{ttl}	The DNS A record's max *Time to Live* value (TTL) in seconds. IPs don't change within a TTL, so they are subject to decay.
$P_{online}(t)$	Probability an IP is online after it has been used for t seconds. Using our *bot online-decay model* , it captures the effect that IPs go offline as time passes due to the unstable nature of bots.
N	Number of IPs advertised at each TTL.
$N_{overlap}$	Number of overlapped IPs between 2 consecutive TTLs. Thus, it's the number of IPs inherited from previous TTL.
$N_{new} = N - N_{overlap}$	Number of new IPs added at the beginning of a new TTL.
$N_{online}(t)$	Number of A record IPs that are online at time $0 \le t < T_{ttl}$

N_{thresh}	The IP detection threshold. During the detection window, if more than N_{thresh} unique IPs are seen, the domain is considered malicious.
λ	Overall incoming rate of victim connections to botnet. Represented as the number of connections per second.
μ	Average bot capacity. Number of victim connections per second each bot IP can
$\lambda^{i}(t) = \frac{\lambda}{N_{online}(t)}$	Incoming victim connection rate for *individual* IP at time $0 \le t < T_{ttl}$ (overall incoming rate λ divided by the number of online IPs at time t)
P_{loss}	The probability the bots' connection queues are full and additional incoming connections will be dropped.

Table 2: Performance modeling terms

$N = N_{new}$), they can only decay for the time, t, that has elapsed in the current TTL; thus, $N_{online}(t) = N \cdot P_{online}(t)$, where $0 \le t < T_{ttl}$.

Determining $N_{online}(t)$ becomes more complicated for a strategy utilizing IP overlap, as in Fig. 5-b. Because IPs are persistent for multiple TTLs, they suffer an increased probability of going offline. For the modeling purpose, we can't query the bots' online state to aid in our replacement decisions, as an actual botmaster might. Instead, we must rely on reasonable assumptions, in this case, that older IPs—being more likely to be offline—will always be replaced before newer IPs. Additionally, to best distribute load among their bots, we can assume that botmasters will choose to advertise as many IPs as possible without exceeding the detection threshold N_{thresh}. These two assumptions imply an optimal replacement strategy from which we can deduce the following intrinsic properties: in any given TTL, (1) there exist a total of N_{new} IPs also present in the previous $0, 1, 2, \ldots, \lfloor \frac{N}{N_{new}} \rfloor - 1$ TTLs, and (2) there exist a total of $(N \bmod N_{new})$ IPs also present in the previous $\lfloor \frac{N}{N_{new}} \rfloor$ TTLs. The effect of these properties can be seen in Fig. 7 for two examples. Thus, for any given DNS-query, we can determine the number of previous queries for which the IPs were used, allowing us to formulate $N_{online}(t)$ in terms of $P_{online}(t)$ as:

$$N_{online}(t) = (N \bmod N_{new}) \cdot P_{online}(t + \lfloor \frac{N}{N_{new}} \rfloor \cdot T_{ttl}) + \sum_{n=0}^{\lfloor \frac{N}{N_{new}} \rfloor - 1} N_{new} \cdot P_{online}(t + n \cdot T_{ttl}) \quad (3)$$

where T_{ttl} is the max A record TTL in seconds and t is the number of seconds elapsed in the current TTL (i.e., $0 \le t < T_{ttl}$). Having defined $N_{online}(t)$ in terms of the IP-advertisement strategy, we can use it in our definition of $P_{loss}(t)$:

$$P_{loss}(t) = \frac{\rho(t)^K - \rho(t)^{K+1}}{1 - \rho(t)^{K+1}} \qquad \text{where } \rho(t) = \frac{\lambda}{N_{online}(t) \cdot \mu} \quad (4)$$

Empirical Evaluation Using our online-decay model and Eq. (4), we can now compare the performance of various IP-advertisement strategies in terms of both $\overline{N}_{online}$ and $\overline{P}_{loss}$. To establish a basis for current FF botnet performance, let us examine the 3 FF domains shown in Table 3. As can be seen from the table, the domains utilize very different DNS strategies. With both a large $\overline{N}$ and $\overline{N}_{overlap}$, it seems that *mountainready.com* is attempting to capitalize on the load-balancing benefits provided by a large number of advertised IPs, while also keeping the total number of unique IPs over 2 queries relatively low to avoid detection; additionally, its use of a fairly small T_{ttl} indicates a proactive approach to countering the bot decay phenomena, which will be accentuated due to its large $\overline{N}_{overlap}$.

Conversely, *old-and-girl.net* makes use of a far different strategy. With its $\overline{N}_{overlap}$ constituting only a small fraction of its $\overline{N}$, the effect of bot decay due to IP overlap is less severe, permitting less diligent IP replacement and allowing for a longer T_{ttl}. Interestingly, its decision to use a small $\overline{N}$ appears to be a double-edged sword; while keeping the total unique IPs over 2 queries low despite its small IP overlap, it also results in fewer IPs per TTL for load-balancing purposes, reducing the botnet's overall capacity. Lastly, *bentlycap.net* seems to have found some middle ground between the other techniques, with a T_{ttl} and $\overline{N}$ almost exactly between the those of others. However, like *old-and-girl.net*, it has chosen a small ratio of $\frac{\overline{N}_{overlap}}{\overline{N}}$, reducing the amount of bot decay and the need for more rapid IP replacement.

Fast-Flux Domain	T_{ttl} (Max TTL) seconds	N (# of IPs per query)			$N_{overlap}$ (# of IP overlap)			$\overline{N}_{online}$ (# online IPs)	$\overline{P}_{loss}$ λ = 100 conn/s μ = 10 conn/s K = 10
		min	avg	max	min	avg	max		(Connection Loss Prob)
old-and-girl.net	600	3	5	5	0	1	5	4.53	54.00%
bentleycap.net	300	10	10	10	0	3	10	9.41	12.10%
mountainready.com	120	4	19	20	0	14	20	17.8	0.14%

Table 3: Current FF DNS strategies and performance

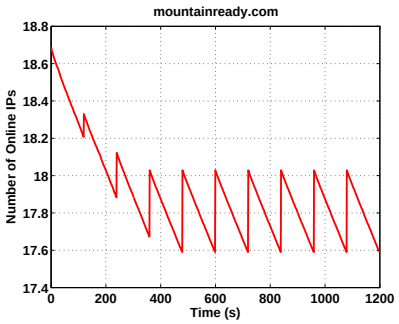


Fig. 8: Number of online IPs based on online-decay model

To compare the performance of these various strategies, we first apply $\overline{N}$ and $\overline{N}_{new}$ to Eq. (3), finding $\overline{N}_{online}$ when the system reaches a steady state, as shown in Fig. 8 for *mountainready.com*. The resulting $\overline{N}_{online}$ values for each domain can be found in Table 3. From the results, botmasters appear quite adept at configuring their DNS strategies to minimize the effect of bot decay. Through the skillful manipulation of their T_{ttl}, $\overline{N}$ and $\overline{N}_{overlap}$, these remarkably different strategies were all able to achieve greater than 90% online availability (i.e., $\frac{\overline{N}_{online}}{\overline{N}}$).

Next, we wished to examine the influence each type of strategy has on the botnet's overall capacity. Because botnet capacity translates to potential victims and revenue, maximizing it should be of great importance to botmasters. However, before we could apply their DNS strategies to Eq. (4), we needed to determine values for λ and μ. For comparison purposes, the actual choice for these values is trivial, so long as we are consistent and use the same values when evaluating each strategy. Based on the spam click-through rate reported in [3], which actually managed to control a small portion of a botnet, we estimate an incoming connection rate of $\approx$ 100 per minute. Since flash crowds could cause the entire 100 connections to occur in a short period of time, and botmasters would want to support such onslaughts of victims for the potential earnings, we assume (for comparison purposes) an overall incoming connection rate, λ, of 100 connections per second. We then chose $\mu = 10$, as its reasonable to assume the entire botnet can handle an order-of-magnitude more connections per second than an individual bot. Using these incoming rates and a bot queue length of $K = 10$, we applied Eq. (4) to each of the 3 botnets' DNS strategies. We then determined $\overline{P}_{loss}$ once the system achieved a steady

state. The domains' results are shown in Table 3. Interestingly, while the varying DNS strategies offered comparable performance in terms of $\frac{\overline{N}_{online}}{\overline{N}}$, they clearly differ in the total capacity each botnet can support. This is a direct consequence of the number of bots available during a given TTL. While the ratio of online IPs is roughly the same for each FF domain, the magnitude is not, with *mountainready.com* having approximately twice as many as *bentlycap.net* and 4*x* as many as *old-and-girl.net*. From the table, it is apparent that *mountainready.com*, with $\overline{P}_{loss} < 0.15\%$, can easily support our assumed connection rates; it is possible that this is even its expected victim load, necessitating its choice of DNS strategy. However, while *bentleycap.net* performs modestly under these conditions, *old-and-girl.net* does not. It's likely that neither botnets' DNS strategy was designed with this sort of load in mind.

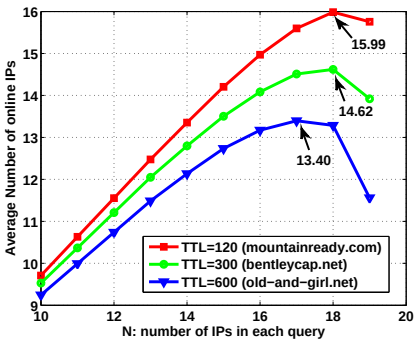


Fig. 9: $\overline{N}_{online}$ optimization: IP-mimicry attack

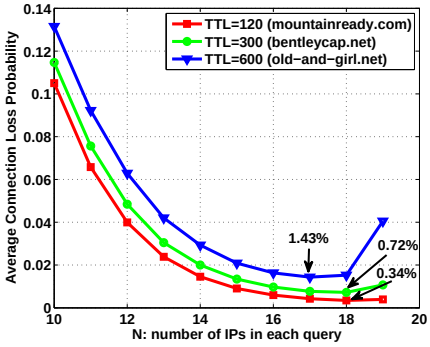


Fig. 10: $\overline{P}_{loss}$ optimization: IP-mimicry attack

IP-Mimicry Attack Having modeled the performance for various DNS strategies currently employed by FF domains in the wild, we can now determine if adapting them to our proposed IP-mimicry attack achieves comparable online connectivity and capacity. Since our mimicry attack manipulates $\overline{N}$ and $\overline{N}_{overlap}$ to evade detection while maximizing online time and capacity, we cannot restrict these to the current values imposed by each FF domain. Instead, we will retain the domains' T_{ttl} values, assuming that they were chosen by the botmasters in response to how diligently they were willing to monitor and replace IPs. In order to reduce false-positives from benign, non-CDN domains advertising a large number of stable IPs, such as *hostingprod.com*, let us assume—for the purposes of this mimicry attack—a detection threshold of $N_{thresh} = 20$ IPs, resulting in the following policy: *over 2 DNS-queries, any domain with more than 20 unique A record IPs will be flagged as malicious.*

It is clear from the results in Section 3.2, that the more online IPs available during a given TTL, the greater the botnet's overall capacity. Therefore, an optimal DNS strategy will necessarily advertise the maximum IPs allowed by the detector's threshold, N_{thresh}. This reduces the problem to determining what $N_{overlap}$ results in the most online IPs, which can be found by either maximizing Eq. (5) or minimizing Eq. (6), such that $2 \cdot N - N_{overlap} = N_{thresh}$ and $\sum_{t=1}^{T_{ttl}}$ is performed when a steady state is achieved.

$$\overline{N}_{online} = \frac{\sum_{t=1}^{T_{ttl}} N_{online}(t)}{T_{ttl}} \qquad (5) \qquad\qquad \overline{P}_{loss} = \frac{\sum_{t=1}^{T_{ttl}} P_{loss}(t)}{T_{ttl}} \qquad (6)$$

For the FF domains in Table 3, Figs. 9 and 10 show the results of Eqs. (5) and (6) across the search space $N \in [\lceil \frac{N_{thresh}}{2} \rceil, N_{thresh} - 1]$. From the figures, we can see that while *mountainready.com* and *bentlycap.net* achieve optimal performance with $N = 18$, for *old-and-girl.net*, $N = 17$. Apparently, its longer T_{ttl} of 600 seconds results in additional bot decay, causing $N = 17$—with its 2 fewer overlapped IPs—to provide better performance. We also find that for *bentlycap.net* and *old-and-girl.net*, their $\overline{N}_{online}$ has increased to 14.62 and 13.4, while their $\overline{P}_{loss}$ has decreased to 0.72% and 1.43%, respectively. While neither of these FF domains would have been detected by the imposed N_{thresh} under their original DNS strategies, utilizing the IP-mimicry attack has kept them from being detected while also greatly increasing their performance and capacity. On the other hand, the mimicry attack caused *mountainready.com* to suffer a reduction in $\overline{N}_{online}$, dropping from 17.8 to 15.99. The attack also caused its $\overline{P}_{loss}$ to more than double, increasing from 0.14% to 0.34%. However, *mountainready.com*'s original DNS strategy advertised 24 unique IPs over 2 queries, exceeding the detection threshold. Thus, the IP-mimicry attack has allowed it to successfully evade detection with only a minor decrease in performance—its average probability of a connection loss remains under 1% and its average online IPs has been reduced by less than 2.

3.3 Empirical Observations

Curious how the proposed Holz and RB-Seeker detectors would fare against today's FF botnet threat, we implemented the detectors and applied them to our set of 45 FF domains (which we manually verified as FF). Both detectors identified the same set of 6 FF domains, with the RB-Seeker detecting an additional 6 that were missed by the Holz detector. The resulting false-negative rates are 86.7% and 73.3% for the Holz and RB-Seeker detectors respectively. Both papers realize their weights and thresholds used for detection must be periodically retrained to counter future mimicry attacks. However, the poor results of the original detectors on current FF domains demonstrates the extent to which botnets have evolved since they were proposed, strengthening the need to better understand the extent of FF botnets' mimicry capabilities and limitations.

4 Increased Detection Window (more queries)

4.1 Good Guys

A logical extension to the fast-detection systems of the previous section is to increase their monitoring window to analyze more queries. Examining multiple TTLs when making a decision exploits a commonly known property of FF domains: they need to continuously advertise fresh IPs to account for their unstable constituent bots. Thus, while non-CDN domains may advertise a large number of IPs in their queries, they will be stable IPs and will not change over time. FF domains, previously able to hide behind non-CDN domains' numerous IPs to subvert fast-detection systems, will quickly be exposed once additional queries are examined. Furthermore, while CDN domains often demonstrate the fluxy behavior characteristically attributed to FF botnets, for many CDNs, a longer detection window can allow their more stable nature to emerge from the chaos.

Current detectors, such as FluXOR [7] and RB-Seeker's second-tier detector, make use of longer detection windows to increase accuracy and support the detection of stealthy FF domains, which use slower DNS advertisement strategies aimed at fooling fast-detection systems. For example, RB-Seeker monitors suspected stealthy FF domains for up to a week. Besides using an increased monitoring window, FluXOR also incorporates additional metrics in its detection decision in an effort to make its classification more accurate and harder to subvert. Unfortunately, how these features are actually used in detection is omitted from the paper. In any case, like the Holz and RB-Seeker detectors, FluXOR examines the number of unique A records and ASNs. These are augmented with additional features such as TTL and the number of return qualified domain names, or top-level domains (TLDs), to try and capture the quickly changing and dispersed nature of FF domains.

4.2 Bot Guise

ASN Mimicry Attack Unfortunately, extending the detection window in time does little to weaken the ASN mimicry attack described in Section 3.2. Because botnets seem to invariably control a sizable number of bots from within at least one ASN, the same essential attack can be performed by simply accommodating the larger detection window as shown in Fig. 11.

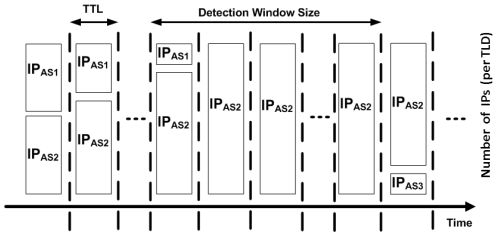


Fig. 11: ASN mimicry strategy (multiple queries)

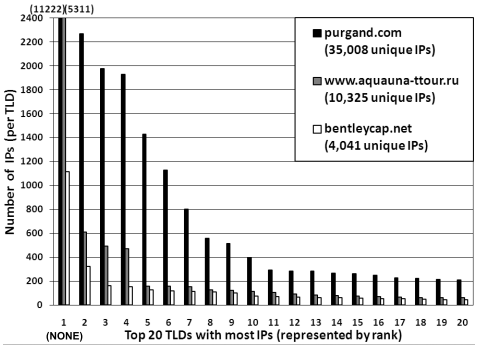


Fig. 12: IP disribution for top 20 TLDs

rDNS Mimicry Attack While the specifics of FluXOR's returned qualified domain metric are not revealed, we can assume it operates as any TLD metric would. Essentially, for any rDNS results returned, the number of unique TLDs are calculated—the insight being that FF botets, consisting of bots scattered across many networks, will return numerous TLDs. However, while botnet IPs do belong to many different TLDs, ultimately, this feature cannot be reliably used for detection. Like the rDNS metric in Section 3.2, it suffers from the inherent shortcoming of the rDNS lookup process, which doesn't always return a result. This results in a sufficient quantity of rDNS=NONE IPs (adequately distributed across ASNs) to perform a similar dual-mimicry attack. Additionally, we analyzed the distribution of bot IPs across TLDs and found a similar distribution as across ASNs, in that there exist some TLDs from which a large number of bots belong. In Fig. 12, we have plotted this distribution for representative FF domains of varying sizes. Like the

ASN distribution, it is long-tailed. While rDNS=NONE IPs dominate, there are clearly other TLDs with a sufficient number of IPs to similarly be used in the aforementioned mimicry attack, providing botmasters additional freedom in DNS advertisement strategies. Consequently, we find rDNS results to be an inadequate metric for detecting FF botnets, based on their current resources.

Improved DNS-Strategy Model Before examining IP-mimicry attacks, we must first extend the DNS-strategy model developed in Section 3.2 to accommodate the larger detection window. First, let us assume a detection window of D_{ttl} fresh DNS queries of length T_{ttl}. Let us further assume the detector applies a threshold, N_{thresh}, on the number IPs seen during this detection window. Under these constraints, the botnet could add at least one new IP every T_{ttl}, so long as $D_{ttl} \leq N_{thresh}$; However, if $D_{ttl} > N_{thresh}$, the botmaster can no longer introduce new IPs each T_{ttl} without exceeding N_{thresh} and triggering detection. Nevertheless, botnets can still keep their total IPs below the threshold by repeating the same set of IPs over multiple T_{ttl}. We term this the botnet's *repetition window* and define it as R_{ttl} DNS queries (of length T_{ttl}) for which the botnet repeats the same set of IPs. This effectively extends the duration of T_{ttl} while taking up more of the detection window, meaning we can determine $N_{online}(t)$ by substituting $R_{ttl} \cdot T_{ttl}$ for T_{ttl} in Eq. (3). If a botnet introduces N_{new} IPs every R_{ttl}, a detection window D_{ttl} will *at most* observe A_{ttl} DNS queries with new IPs, where A_{ttl} is definiens in Eq. (7). This relationship is shown for $D_{ttl} = 4$ and $R_{ttl} = 2$ in Fig. 13, where we see that $A_{ttl} = 2$. Thus, botnets can add N_{new} IPs every A_{ttl}, so long as Eq. (8) is satisfied.

$$A_{ttl} = \lfloor \frac{D_{ttl} - 2}{R_{ttl}} \rfloor + 1 \qquad (7) \qquad\qquad N + A_{ttl} \cdot N_{new} \leq N_{thresh} \qquad (8)$$

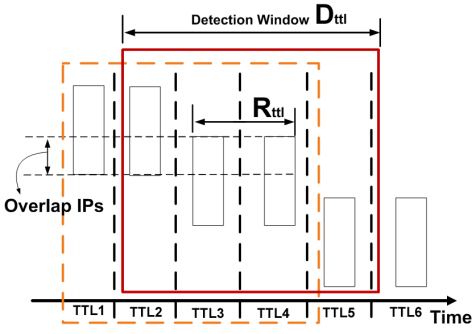


Fig. 13: Relationship between A_{ttl}, R_{ttl} and D_{ttl}

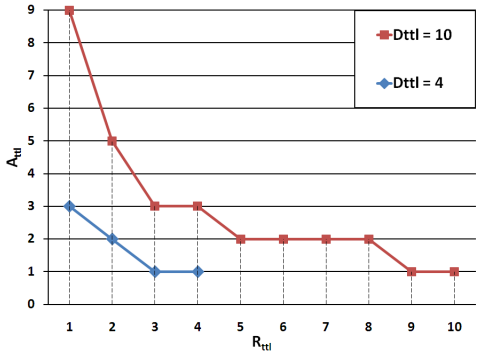


Fig. 14: A_{ttl} when $R_{ttl} \in [1, D_{ttl}]$ and $D_{ttl} = 4, 10$

IP-Mimicry Attack: TTL-based Detection Window By applying the improved DNS strategy using R_{ttl} to our previous performance model, we can determine how the proposed IP-mimicry attack against a larger detection window influences botnets' overall online availability and capacity. For this purpose, we examine the same real-world FF domains as in Section 3.2, again, fixing their T_{ttl} to the values originally used by each domain. Modifying Eqs. (5) and (6) to incorporate the increased detection window produces:

$$\overline{N}_{online} = \frac{\sum_{t=1}^{R_{ttl}\cdot T_{ttl}} N_{online}(t)}{R_{ttl}\cdot T_{ttl}} \quad (9) \qquad \overline{P}_{loss} = \frac{\sum_{t=1}^{R_{ttl}\cdot T_{ttl}} P_{loss}(t)}{R_{ttl}\cdot T_{ttl}} \quad (10)$$

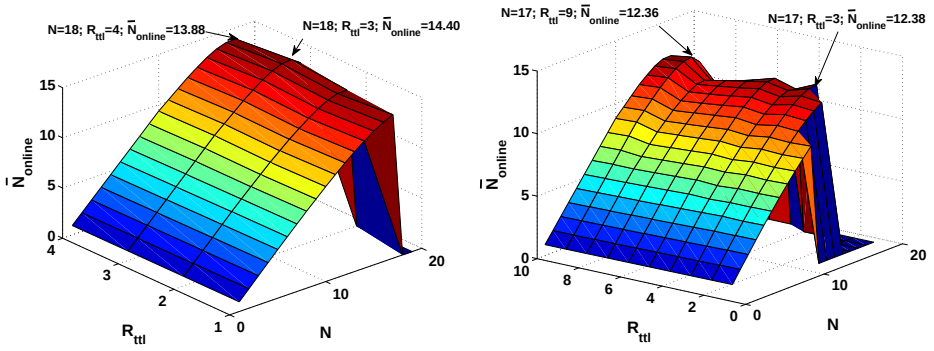
To find the optimal values for R_{ttl} and N, we maximize Eq. (9), or minimize Eq. (10), over the search space $R_{ttl} \in [1, D_{ttl}]$ and $N \in [\lceil \frac{N_{thresh}}{A_{ttl}+1} \rceil, N_{thresh} - A_{ttl}]$. While R_{ttl}'s search space is self evident, N's is found from Eq. (8) and the observation that $1 \leq N_{new} \leq N$. We set $N_{thresh} = 20$ and optimized the equations when they reached a steady state for detection windows, $D_{ttl} = 4, 10$; the results of these optimizations are shown in Table 4. The first thing we notice is that $\overline{N}_{online}$ and $\overline{P}_{loss}$ don't suffer much degradation. In fact, for *bentlycap.net* and *old-and-girl.net*, they achieve better results under the mimicry attack than their original settings. While *mountainready.com*'s performance goes down marginally, its $\overline{P}_{loss}$ is still less than 1% for $D_{ttl} = 4$ and less than 3% when $D_{ttl} = 10$. These slight decreases in performance are easily justified, considering that its original DNS strategy would have resulted in detection, with 34 unique IPs for $D_{ttl} = 4$ and 64 for $D_{ttl} = 10$. Furthermore, the use of an extended detection window would have caught all the FF domains under their previous DNS strategies, with the exception of *old-and-girl.net* when $D_{ttl} = 4$. Thus, the proposed IP-mimicry attack has only minimally degraded—and in many cases improved—the performance of the FF domains, while also preventing their detection against both fast-detection systems and those with an extended detection window.

Parameter Setup			Optimization Results				
D_{ttl} # of TTLs in detection window	Fast-Flux Domain	T_{ttl} (Max TTL) seconds	N (# of IPs per query)	$N_{overlap}$ (# of overlap IPs)	R_{ttl} # of TTLs botnet repeats same IPs	$\overline{N}_{online}$ (# of online IPs)	$\overline{P}_{loss}$ (Connection Loss Prob) λ = 100 conn/s μ = 10 conn/s K = 10
4	old-and-girl.net	600	16	15	1	11.20	4.92%
	bentlycap.net	300	17	16	1	12.71	3.11%
	mountainready.com	120	18	16	3	14.40	0.841%
		1	19	18	3	18.56	0.095%
10	old-and-girl.net	600	14	12	3	8.31	19.8%
	bentlycap.net	300	15	13	3	10.18	8.42%
	mountainready.com	120	17	16	3	12.38	2.58%
		1	19	18	9	18.39	0.103%

Table 4: Optimization Results: IP-mimicry attack against D_{ttl}

In Fig. 15, we show an example of the $\overline{N}_{online}$ optimization plots for *mountainready.com*'s $T_{ttl} = 120$. To better understand these plots, recall the relationship between D_{ttl}, R_{ttl} and A_{ttl} defined in Eq. (7) and shown for for $D_{ttl} = 4, 10$ in Fig. 14. From the figures, we find that for values of R_{ttl} resulting in the same A_{ttl}, the lowest R_{ttl} is optimal, with higher values resulting in a steady degradation of $\overline{N}_{online}$. This is best exemplified when $D_{ttl} = 10$ in Figs. 14 and 15-b, with local maxima at $R_{ttl} = 1, 2, 3, 5,$ and 9. This occurs because an increase in R_{ttl} results in additional bot decay, due to repeating the same set of IPs over an extended duration. Thus, when increasing R_{ttl}, if A_{ttl} remains the same, the number of IPs advertised per query, N, cannot be increased to offset the decay without exceeding the threshold, resulting in the observed trend.

As an additional experiment, we determined the optimal strategy for a FF domain using a $T_{ttl} = 1$, as it provides the finest granularity for adjusting the IP replacement strategy in terms of R_{ttl}. This strategy, shown in Table 4 in gray, achieves better results than *mountainready.com*'s original configuration. With such a short T_{ttl}, a detection window of 10 will only monitor the domain for 10 seconds, resulting in little bot decay and allowing a larger N. Additionally, we see the optimal R_{ttl} in this case is $D_{ttl} - 1$, that is,



(a) $D_{ttl} = 4$ (b) $D_{ttl} = 10$

Fig. 15: $\overline{N}_{online}$ optimization: IP-mimicry attack, $T_{ttl} = 120$ and $N_{thresh} = 20$

the minimal R_{ttl} for which the detection window sees only a single IP change ($A_{ttl} = 1$). An $R_{ttl} < D_{ttl} - 1$ results in more IP changes, further limiting the maximum N achievable without going over the threshold. With an $R_{ttl} \geq D_{ttl}$, IPs are subjected to additional bot decay while A_{ttl} remains at its minimum value of 1; regardless of R_{ttl}, eventually the detection window always observes a single IP change. Thus, at the cost of more diligent IP management, this technique maximizes the number of online IPs possible per query while minimizing the effect of bot decay.

IP-Mimicry Attack: Time-based Detection Window Thus far, we have defined the detection window in terms in terms of D_{ttl} fresh DNS queries, showing that it can be subverted through the use of a repetition window, R_{ttl}. However, a detection window can also be defined in terms of absolute time (i.e., D_t seconds). In this case, a repetition window doesn't help mask the addition of new IPs, requiring the FF domain to adhere to the IP threshold imposed over the duration, D_t. Thus, the longer the duration, the more the FF domain's IPs are subjected to bot decay, worsening performance. We modeled this detection technique to evaluate its susceptibility to IP-mimicry attacks under current botnet resources. For the purposes of this evaluation, we adopt a D_t equal to 1 week and an IP threshold of N_{week}. Certainly, requiring longer than a week to arrive at a detection decision grants botnets sufficient time to perpetrate their scams under a given domain. To find a suitable value for N_{week} that will provide minimal false-positives, we analyzed the number of unique IPs accrued by benign CDN domains over 1 week. Not surprisingly, due to load-balancing techniques, CDN domains can advertise a large number of unique IPs. For example, we observed 171 IPs used by *nfl.com*. The amount was even greater for *www.myspace.com*, with many DIGGER nodes witnessing the use of over 400 unique IPs, and in one case, over 700. We analyzed the model for varying values of $N_{week} \in [100, 800]$, to see how increasing the threshold—to reduce false-positives—will affect a botnet's performance. To ensure that the mimicry attack would also continue to subvert fast-detection systems, we imposed the additional constraint of N_{thresh} IPs over 2 DNS queries as before. Then, for each value of N_{week}, we calculate the maximum queries for which D_t can observe new IPs without violating N_{week} as $A'_{ttl} = \lfloor \frac{N_{week} - N}{N_{new}} \rfloor$, such that $N + N_{new} \leq N_{thresh}$. If we assume IPs are changed every TTL, then we can calculate

the optimal T_{ttl} as $T_{opt} = \frac{T_{week}}{A_{ttl}}$, where T_{week} is the number of seconds in a week. Under these constraints, the FF domain won't exceed the threshold of N_{week} unique IPs over the detection window $D_t = 1$ week. Furthermore, for any 2 queries, the number of unique IPs will satisfy the threshold N_{thresh}. Finally, notice that a repetition window, R_{ttl}, can be applied to T_{opt} to defeat a D_{ttl} detection window.

Table 5 shows our optimized results for N, $N_{overlap}$, and T_{opt} with $N_{thresh} = 20$ and varying thresholds of N_{week}. For all values of N_{week}, we achieved the same optimal values of $N = 19$, $N_{overlap} = 18$, and thus $N_{new} = 1$. This is because it is necessary to provide as many IPs per query as is possible to account for the enhanced botnet decay present under the longer T_{opt}. From the table, it is apparent that even for $N_{week} = 100$, which is well below the number of IPs seen for *nfl.com* and *www.myspace.com*, the botnet will continue to have online IPs. Despite the high probability of lost connections, the botnet is still reachable and thus can continue to generate revenue. In addition, we find that for $N_{week} = 200$, the botnet capacity is nearly the same as that of *old-and-girl.net* under its original configuration. Because benign CDN domains legitimately advertise such large amounts of unique IPs over time, current botnet resources can sufficiently mount IP-mimicry attacks despite an increased detection window, D_t.

N_{week} max # unique IPs / week	T_{opt} (optimal TTL) seconds	$\bar{N}_{online}$ (# online IPs)	$\bar{P}_{loss}$ (Connection Loss Prob) λ = 100 conn/s μ = 10 conn/s K = 10
100	7,466	2.89	71.10%
200	3,341	4.97	50.30%
250	2,618	5.79	42.20%
400	1,587	7.7	24.40%
700	888	9.95	9.37%
800	774	10.5	7.11%

Table 5: Optimization results: IP-mimicry attack against $D_t = 1$ week ($N_{thresh} = 20$: $N = 19$, $N_{overlap} = 18$)

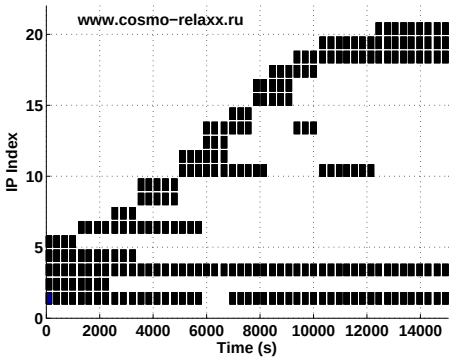


Fig. 16: Empirical observation of FF domain adopting certain evasion techniques ($T_{ttl} = 10$ seconds)

4.3 Empirical Observations

We discovered several FF domains in the wild adopting some of the mimicry attacks we have presented, many of which were able to defeat the Holz and RB-Seeker detectors in Section 3.3. While the strategies employed by FF domains in the wild aren't as regular as those in our models, they are very close, only deviating from their average values rarely and in small amounts. An example FF domain is shown in Fig. 16, with each box in the plot representing a unique IP seen in its DNS results. Observe that it adds 1 or 2 IPs every $\approx$ 1,000 seconds, replacing older IPs to keep the total number equal to 5 per query; thus, it uses an $N = 5$ and an $N_{overlap} = 3, 4$. Since it has a $T_{ttl} = 10$, it's essentially using a repetition window of $R_{ttl} = 100$. Under this DNS strategy, the FF domain can defeat a fast-detection system with an $N_{thresh} = 7$, as it occasionally introduces 2 new IPs per query. Furthermore, it will also defeat an extended detection window with $D_{ttl} = 100$ and $N_{thresh} = 7$. By taking an average $N_{overlap} = 4$, we can use our model to predict that the

domain will achieve an $\overline{N}_{online} = 3.73$, resulting in an average of 75% of its advertised IPs being online. Clearly, FF domains are beginning to incorporate some of the mimicry techniques we have examined—a trend we expect will continue as detection systems improve.

5 Related Work

Recently, a number of techniques have been proposed to effectively detect FF domain names [1, 7, 2, 8]. They all studied a number of unique features—the detailed set of which varies across different techniques—that can be used to characterize FF domains. To extract the values for these features, they collected DNS queries for a large number of suspicious domains through either active or passive monitoring, over time periods ranging from 2 TTLs to weeks. Classification algorithms, such as support vector machines (SVM) and decision trees, were then applied to the features extracted from the DNS traces, determining if each domain under consideration is FF or not. Active probing approaches like Holz's detector [1], RB-seeker [2] and FluXOR [7] have been detailed in Sections 3 and 4. Compared to these active methods, the passive monitoring approach proposed by Perdisc *et al.* [8] provides a different vantage point and offers the unique advantage of detecting FF domains via stealthy means, without drawing attackers' attention. However, for good coverage of FF domains, it requires monitoring recursive DNS traces on multiple large networks, which can only be done by large ISPs. Additionally, as a passive approach, a decision can't be reached until multiple victims have visited the suspicious site and exposed themselves to the potentially malicious content.

The concept of a mimicry attack was first proposed for host-based intrusion detection systems (IDSes) which typically monitor application behavior in terms of system-call sequences. To slip under the radar, mimicry attackers attempt to cloak malicious system calls with innocuous-looking system-call sequences. Wagner and Soto [10] proposed a method that embeds nullified pre-existing system-call sequences (i.e., "semantic no-ops") between malicious system calls. Kruegel *et al.* [4] devised techniques that allow an attacker to regain control after a system call by corrupting the memory locations and manipulating code pointers. This allows attackers to extend transitional mimicry attacks on more sophisticated IDSes. More recently, Parampalli *et al.* [6] proposed the persistent control-flow interposition techniques that make mimicry attacks simpler, more reliable and stealthy. Similar to previous work, the goal of our work is to design and evaluate, using current FF botnet resources, potential mimicry attacks against FF detection systems. By anticipating the attackers' next moves, defensive systems can be better instrumented and remain effective for a longer period of time.

6 Conclusions and Future Work

In this paper, we have examined the current state-of-art FF detectors, analyzing their effectiveness in detection. In doing so, we developed accurate models for bot decay, online availability, DNS advertisement, and performance, which we used to evaluate novel mimicry attacks against FF detection systems. Based on these models, empirical evidence, and logical assumptions, we have demonstrated that current botnet resources are

sufficiently capable of subverting state-of-art FF detection mechanisms. We have discovered evidence of current FF domains adopting aspects of our proposed mimicry attacks. While botmasters don't appear to be monitoring their botnets as assiduously as our models assume, they are clearly incorporating similar—though less optimal—DNS strategies for detection evasion. As detection systems continue to improve, we expect to see an increased diligence in IP management as botmasters move toward these optimal strategies in an effort to extract the most from their botnets' resources. We hope showing the mimicry potential currently attainable by FF domains will foster improvements to existing detection systems, as well as provide new insight into the adaptive limitations of FF botnets. Our future work includes extending these models to handle a spatial dimension, allowing us to evaluate FF domains' current ability to mimic the location-aware advertisement strategies of CDNs. By extending detectors in space, we hope to impose additional constraints, straining current botnet resources beyond their capability to perform successful mimicry attacks.

References

1. Holz, T., Gorecki, C., Rieck, K., Freiling, F.C.: Measuring and detectin fast-flux service networks. In: Proc. of network and Distributed System Security (NDSS) Symposium (2008)
2. Hu, X., Knysz, M., Shin, K.G.: Rb-seeker: Auto-detection of redirection botnets. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2009 (2009)
3. Kanich, C., Kreibich, C., Levchenko, K., Enright, B., Voelker, G.M., Paxson, V., Savage, S.: Spamalytics: an empirical analysis of spam marketing conversion. In: Proceedings of the 16th ACM conference on Computer and communications security (2009)
4. Kruegel, C., Kirda, E., Mutz, D., Robertson, W., Vigna, G.: Automating mimicry attacks using static binary analysis. In: In USENIX Security Symposium (2005)
5. Mockapetris, P.: Domain names - implementation and specification. RFC 1035 (1987)
6. Parampalli, C., Sekar, R., Johnson, R.: A practical mimicry attack against powerful system-call monitors. In: ASIACCS '08: Proceedings of the 2008 ACM symposium on Information, computer and communications security (2008)
7. Passerini, E., Paleari, R., Martignoni, L., Bruschi, D.: Fluxor: detecting and monitoring fast-flux service networks. In: Proceedings of DIMVA'08 (2008)
8. Perdisci, R., Corona, I., Dagon, D., Lee, W.: Detecting malicious flux service networks through passive analysis of recursive dns traces. In: Proceedings of ACSAC '09 (2009)
9. Planetlab: An open platform for developing, deploying and accessing planetary-scale services. http://www.planet-lab.org/
10. Wagner, D., Soto, P.: Mimicry attacks on host-based intrusion detection systems. In: Proceedings of the 9th ACM conference on Computer and communications security (2002)