Investigation of Triangular Spamming: a Stealthy and Efficient Spamming Technique

Zhiyun Qian[1], Z. Morley Mao[1], Yinglian Xie[2], Fang Yu[2]
[1]University of Michigan and [2]Microsoft Research Silicon Valley

***Abstract*—Spam is increasingly accepted as a problem associated with compromised hosts or email accounts. This problem not only makes the tracking of spam sources difficult but also enables a massive amount of illegitimate or unwanted emails to be disseminated quickly. Various attempts have been made to analyze, backtrack, detect, and prevent spam using both network as well as content characteristics. However, relatively less attention has been given to understanding how spammers actually carry out their spamming activities from a network angle. Spammers' network behavior has significant impact on spammers' common goal, sending spam in a stealthy and efficient manner. Our work thoroughly investigates a fairly unknown spamming technique we name as triangular spamming that exploits routing irregularities of spoofed IP packets. It is highly stealthy and efficient in that triangular spamming enables 1) exploiting bandwidth diversity of botnet hosts to carry out spam campaigns effectively without divulging precious high-bandwidth hosts and 2) bypassing the current SMTP traffic blocking policies. Despite its relative obscurity, its use has been confirmed by the network operator community. Through carefully devised probing techniques and actual deployment of triangular spamming on Planetlab (a wide-area distributed testbed), we investigate the feasibility, impact of triangular spamming and propose practical detection and prevention methods. From our probing experiments, we found that 97% of the networks which block outbound SMTP traffic are vulnerable to triangular spamming and only 44% of them are listed on Spamhaus Policy Blocking List (PBL).**

I. Introduction

Spam constitutes an enormous waste of network resources. As reported, over 90% to 97% of all emails are spam [8]. Despite all the past efforts in spam mitigation, the problem still remains unsolved. There are two main categories of spam filtering techniques: content-based and blacklist-based. While content-based filtering is the canonical way, blacklist-based approach (*e.g.,* Spamhaus, Spamcop [19], [18]) is receiving much attention recently because it does not rely on email content and may be more efficient and less susceptible to evasion. While IP-based blacklist is simple and lightweight, compiling and maintaining such a list is challenging due to the changing landscape of compromised hosts: more hosts can become compromised; they could change IP addresses over time; and they may also be patched. As a result, it is not surprising that most IP blacklists provide a very limited coverage of malicious IPs involved in sending spam [36].

Further, as spam detection and prevention techniques evolve, so do spamming techniques. Spammers are increasingly more stealthy by restricting each IP or compromised host to only send very few spam messages to each target in order to stay under the radar [39]. In the meanwhile, ISPs are enforcing the outbound SMTP (port 25) blocking policy for their end-hosts in an effort to reduce spam originated from their networks [13], [14].

In this paper, we systematically study triangular spamming, a clever spamming technique that has been known for several years, but never systematically studied. Triangular spamming, as its name suggests, involves three main parties, target mail server, original spam sender (or high-bandwidth bot) and relay bot (or low-bandwidth bot). The key idea is that with relay bots' cooperation, the original sender (high-bandwidth bot) can send spam in high throughput while hiding its own IP address by spoofing the relay bots' IP addresses. In a recent NANOG survey [9], although the network operator community is already aware of such problems, our study shows that most ISPs still do not enforce the correct SMTP blocking policy to prevent triangular spamming.

We focus on three key questions:

1. What are the requirements of triangular spamming, and is today's network vulnerable to such spamming behavior?

2. What are the benefits of triangular spamming, and is it used in the wild today?

3. What are the possible solutions to prevent or mitigate such a spamming approach?

As triangular spamming essentially exploits network-level vulnerability, it requires a detailed understanding of network operational practices that are usually overlooked in security research domain. In this paper, we surveyed the network policy practices in addition to conducting large-scale experiments to verify and explore current network policies of various ISPs. More specifically, we are focusing on the port blocking policy employed by ISPs.

Our study makes the following contributions:

1. We designed an accurate and effective probing technique to discover the networks that attempt to block outgoing port 25 traffic but fail to enforce the correct port blocking policy, thus are vulnerable to triangular spamming.

2. We found that 97% of the blocking networks fall into the above category. Only 44% of such prefixes are listed on Spamhaus PBL [37].

3. We conducted experiments to ascertain the existence of triangular spamming at the mail server side.

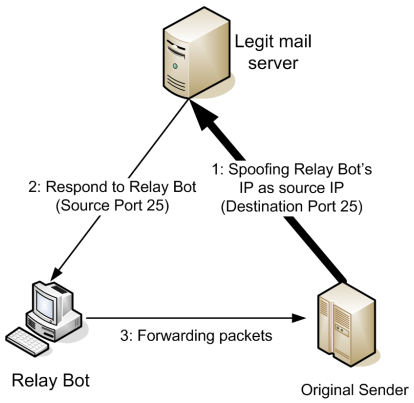

Figure 1. Triangular spam delivery example

4. We systematically evaluated the feasibility and benefits of triangular spamming via experiments of actually deployed setups on Planetlab. Based on the operational experience, we discuss promising prevention and detection approaches to triangular spamming.

The remainder of the paper is structured as follows: § II describes the basic requirements and implication of triangular spamming. § III studies the port blocking policy extensively for thousands of networks. § IV describes our experience and lessons learned from building triangular spamming and deploying it on Planetlab on our own. § V describes possible detection and prevention techniques and ascertain the existence of triangular spamming. § VI surveys the related work and § VII concludes our work.

II. Triangular Spamming Mechanism and Implication

As shown in Figure 1, triangular spamming exploits IP spoofing to route packets indirectly for the purpose of hiding the identity of actual sending hosts and increasing spam throughput. The spammer picks one or more high-bandwidth bots (or original sender) to send spam directly to target mail server while spoofing the source IP addresses of relay bots. These bots listen for any relevant packets, *e.g.,* those from port 25 from the mail server and forward them back to the original spammer.

A. Triangular spamming requirement

IP spoofing is allowed at the origin sender network. IP spoofing has long been studied for implications such as DoS attacks [28]. Although the problem has been studied extensively for two decades or so, it is still largely unsolved due to various deployment challenges, *e.g.,* the network policy for enforcing anti-spoofing such as unicast reverse path forwarding (uRPF) [26], [21] is limited by multi-homing, route asymmetry, complexity of managing and updating the filtering rules. Indeed, based on the Spoofer study [27], 31% of the IP addresses studied allow successful spoofing of an arbitrary, routable source address. We perform our own study to determine the degree IP spoofing is possible in order to ascertain the feasibility of triangular spamming on today's Internet.

Traffic from mail server to relay bots are not blocked. As we can observe from Figure 1, even though the relay bot does not have to contact the mail server directly, it must receive packets from the mail server in order to relay them back to the original sender. However, if such traffic is blocked, then triangular spamming will fail to operate. In §III, by conducting intelligent probing to infer port blocking policies, we show that most ISPs do not block such traffic today. On the other hand, traffic from the relay bot to the original sender can be easily tunneled and encrypted so that it can be hard to detect and filter.

Also, it is generally more difficult for NATed hosts to participate as relay bots given that they will have to be able to receive packets on a specific source port. However, with the development of NAT traversal techniques such as uPnP [22] (many home routers by default enable this feature), it is rather easy for compromised hosts to initiate requests to add or modify port mappings on their routers. In fact, previous attacks have demonstrated that a malicious Web site can use Flash to control the client's uPnP-enabled router [6]. Note also that the port value only needs to be larger than 1024 (which will unlikely be in conflict with other applications).

B. Implications of triangular spamming

Port blocking policy bypassing. Many ISPs nowadays enforce outbound SMTP traffic (port 25) blocking in an effort to prevent compromised hosts or bots inside their networks from sending spam targeting destinations outside their networks. In the following we denote from the perspective of a given network outgoing packets with destination port 25 as *OUT* traffic and incoming traffic with source port 25 as *IN* packets (which is usually the response packets sent from the mail server) for ease of exposition. The phrase "outbound SMTP traffic blocking" refers to an abstract policy that tries to prevent outbound spam by either blocking IN traffic or OUT traffic or both. The problem is that *only* blocking OUT traffic but not IN traffic (which is the second requirement of triangular spamming) by ISPs is insufficient to fully prevent their internal hosts from participating in spamming activities. Using triangular spamming, those IP addresses can still be "hijacked" to send spam. Note that for ISPs that do not try to prevent outbound spam, they will not block IN traffic either as it is necessary for outgoing SMTP connection to be established.

Higher spamming throughput compared to sending directly from botnets. Spammers can rent high bandwidth pipes to send spam with higher throughput due to the nature of triangular spamming — most of the traffic is uplink traffic directly flowing from the spammer to the mail server without going through bots (See Figure 1). Although response packets from the mail server have to inevitably traverse bot hosts, they may not be the bottleneck as the spammer can parallelize connections by leveraging many different bots they may already have access to today.

III. ISP Port Blocking Policy Inference and Policy Impact Analysis

How ISPs configure outbound SMTP traffic blocking determines whether triangular spamming can work. As we discussed, many ISPs now enforce the outbound SMTP traffic blocking policy, but it is unclear what the exact policy is. In this section, we present a systematic empirical analysis on the port blocking policy of various ISPs. More specifically, we intend to study 1) which ISPs currently enforce outbound SMTP traffic blocking, covering as many ISPs as possible, 2) under their current policies, how many are vulnerable to triangular spamming either as sending hosts or as relay hosts as described previously.

A. Port blocking model

We make several reasonable assumptions about the firewall blocking model in order to design tests to infer firewall policies. First, we assume that ISPs are not blocking port 25 traffic based on packet content (also known as Deep Packet Inspection or DPI) given DPI is more expensive and difficult to operate at line speed. Indeed, direct port 25 blocking is the most commonly enforced policy [13], [14]. We also assume that blocking is directional and can be configured based on TCP/IP header, *e.g.,* source/destination IP address, source/destination port, protocol types (*e.g.,* TCP or UDP) and TCP flags (*e.g.,* SYN, ACK). This model is commonly adopted in most modern firewalls ranging from heavy-weight devices (*e.g.,* Cisco PIX firewall [4]) to host firewall software on PCs (*e.g.,* iptables [10]). For example, a sample firewall rule that blocks outbound SMTP traffic would appear as:

SrcIP	DstIP	SrcPort	DstPort	Protocol	TCP-flags	Action
Any	Any	Any	25	TCP	ALL	Drop

There are two important observations to make here. First, suppose this rule is applied to outgoing traffic, *i.e.,* traffic from ISPs' internal hosts to external networks, it effectively blocks only unidirectional outgoing traffic, implying that packets from an external mail server destined to internal hosts with source port 25 will not be blocked. This motivates our study on inferring current port blocking policies of different ISPs to discover if they are vulnerable to triangular spamming. This problem is illustrated in Figure 2 — the ISP can either block OUT traffic, IN traffic, or both. It is known that many ISPs only block OUT traffic due to the simplicity of such policies and the additional complexity of configuring incoming port 25 traffic filtering as mentioned in previous work [27]. For instance, depending on where the firewall is located, there has to be many exceptions in the firewall rules specifically (sometimes separately) for outgoing mail servers and incoming mail servers. As the recent NANOG survey [9] shows, some real-world ISP operators do consider that blocking OUT is simpler and has less impact on the traffic (there is less outgoing traffic than incoming traffic).

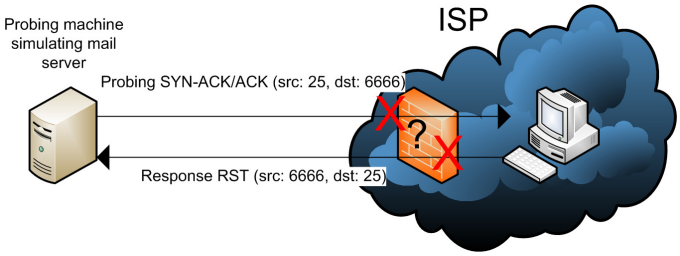

Figure 2. Possible outbound SMTP traffic blocking policy

Second, we note that a stateful firewall that tracks individual TCP connection states could block IN packets associated with triangular spamming simply because they are "out-of-state". For example, SYN-ACK packets without any prior associated SYN packets will be dropped by such a stateful firewall. However, it is difficult for ISPs to adopt this due to two reasons: 1) it is expensive to keep track of the state associated with a large number of flows, and 2) some out-of-state traffic, *e.g.,* probing, can be legitimate. Note that host firewalls can easily support stateful checking, but if the host is already compromised, such firewalls are easily disabled or bypassed. On the other hand, network firewalls are unlikely modified by spammers. Given this key difference, we attempt to distinguish host-based blocking from ISP-level port blocking as discussed later in §III-B2.

B. ISPs that block OUT traffic

To study whether most ISPs block OUT traffic instead of IN traffic, we first find a set of candidate ISPs or IP ranges that block outbound SMTP traffic and then use the probing methodology discussed in §III-C1 to distinguish OUT traffic blocking from IN traffic blocking.

1) Experiment design: There are several approaches to discover the outbound SMTP traffic blocking behavior.

Surveying ISPs would be the simplest approach. However, many ISPs treat such information as confidential and only reveal it to new or existing customers. Very few ISPs openly disclose such information (*e.g.,* Sonic.net [14]). ISPs' knowledge of their policies may lack sufficient detail and may also be inaccurate due to misconfigurations. Further, port blocking policy may change over time and may vary depending on location. For example, Comcast was known to enforce such policies [5]. However, our controlled experiment (via testing using Comcast service at home) indicates that Comcast is not blocking outbound SMTP traffic at the time we conducted the experiment.

The second approach is to obtain control at both endpoints by installing a probing program on end-hosts inside various ISPs, which communicate with a server managed by us. For instance, the ICSI Netalyzr [7] requires users to download a Java applet and likewise the Spoofer project [27] requests users to explicitly download a program to run. However, such an approach is more challenging to accomplish wide-scale adoption.

```
<object width='0' height='0'>
<param name='movie' value='http://OURSERVER-IP/flash.swf'>
<embed src=``http://OURSERVER-IP/flash.swf' width='0' height='0'>
</embed>
</object>
```

Figure 3. HTML code snippet

The third approach is to probe with single end control only. We can mimic a mail server sending TCP packets with source port 25 to the other end (on some well-known ports) with SYN-ACK or ACK flags. Depending on the OS and host firewall settings, probed end-hosts may respond with a RST packet (we verified this behavior for Windows XP SP2 and Linux Ubuntu 9.04). If all live end-hosts respond to the our source port 80 probing but never to our source port 25 probing, it is highly likely that this ISP is blocking outbound SMTP traffic instead of individual hosts doing so. This approach has the benefit of being easily carried out so that we can probe any host or network of interest. However, the choice of which IP address ranges to start with limit its use.

Considering tradeoffs of these various approaches, we adopt a hybrid one combining the second and third approach. In order to obtain control from ISP's side, we chose to develop a simple, invisible Flash [1] program that can be easily embedded in Web sites and transparently executed at the client side. Figure 3 shows the HTML code to be inserted into Web pages (Note that IP address of the server is used to avoid an additional lookup overhead). We inserted it at various university department and educational Web sites in the U.S. and China to obtain a variety of client IP addresses.

Note that simpler HTML code like can achieve the same goal. However, direct port 25 access in HTML is blocked by browsers like Firefox due to security reasons, *i.e.,* one can craft forged HTML Form Post formatted to send out spam emails. However, Flash is in a completely different domain from the browser and is allowed to initiate outgoing port 25 connection by default. If our Flash program indeed fails to establish the connection, then it is most likely blocked by firewalls at the host or at the network. To distinguish between these two, more data points from that network are needed.

The choice of Flash is supported by the observation that 99% of modern browsers deployed [23] have the Flash plugin installed. Thus almost every client can execute the program which simply tries to initiate an outgoing port 25 connection and terminates immediately upon success. Logging by our server will record this along with the initial download of the Flash script via HTTP. This allows us to distinguish IP addresses that succeeded in the test from those that failed to connect to the port 25 on our server.

2) Probing results: As shown in Table I, based on our two months of data collected, we gathered about 21,131 unique IP addresses (excluding 2,749 local IP addresses) in our Web log spanning across 7,016 BGP prefixes. Based on a simple DNS name heuristic, we classify the prefixes into educational institutions and ISPs, since our clients are mostly students who likely access through home or school networks. 341 of them are educational institutions, 2987 are ISPs, and 3691 are unknown (We randomly probed IP addresses within the prefix with some threshold, if none of them has a DNS name, then it is classified as unknown). Although 3,563 (51%) prefixes contain at least one IP address blocked for outbound SMTP traffic, only 2,600 prefixes (37%) have all IP addresses blocked. Interestingly, there are 622 IP addresses that connected to port 25 without connecting to our Web server. We suspect that these are spammers probing for open relays.

For many prefixes, we only have limited samples (IP addresses) and the blocking behavior may not represent the prefix-level policy, *i.e.,* it is possible the host firewall blocks the outbound port 25 traffic which is not easily determined by the Flash script. As a result, we conducted further probing to verify that the ISPs are indeed blocking outbound SMTP traffic in those IP ranges. Extensive probing (piggybacked in our IN/OUT blocking described in Section III-C) for every IP address in the prefix range is carried out to avoid incorrect conclusions caused by outliers, *i.e.,* host firewall rather than ISP firewall blocking traffic. Although we could also develop some randomized probing algorithm, the problem is that we do not know when is sufficient to stop and even if we stop at some threshold number of responses, we may still miss the remaining IP addresses with different behaviors.

The results show that about 688 prefixes have at least some /24 sub-ranges blocking outbound SMTP traffic, assuming the policy is configured at most at the granularity of /24, matching the finest routing granularity on the Internet. Out of these 688 prefixes, 25 are educational institutions, 483 are ISPs, and the remaining 180 are unknown.

To illustrate the diversity of our dataset, we look up the country for IP addresses from IP `whois` database [25]. They span across 127 different countries. Due to a lack of space, only countries with more than 100 IP addresses are shown in Table II. As expected we observe that most IPs are from the U.S. and China matching the locations of the hosting Web sites. At the prefix level, we analyze the percentage of blocking prefixes as verified using probing and show the diverse policy across countries. Since our instrumented Web sites are most likely visited by universities and home users, we expect that many prefixes should perform outbound SMTP traffic filtering at least at some sub-ranges. The results show that the top two countries for enforcing such port blocking policy are Turkey and Canada. Compared to the top two countries, the U.S. has a lower filtering enforcement rate. But it is still better compared to the remaining countries. China and Korea have the worst blocking percentages, implying that ISPs in those

countries visible in our data do not pay much attention to spam prevention through network-based filtering. This is consistent with previous findings that China and Korea are two big sources of spam emails.

Table I
SUMMARY OF IPS GATHERED FROM THE WEB FLASH EXPERIMENT

	# of IP addresses	# of prefixes
# of IP in web log (Baseline):	21131	7016
# of IP in port 25 log:	13576	4280
# of IP in web log but not port 25 log:	7555	3563
# of IP in port 25 log but no web log:	622	397

Table II
DISTRIBUTION OF IPS AND PREFIXES BASED ON COUNTRY

Country	# of IPs	# of prefixes	# of blocking prefixes	% of blocking prefixes
AU	638	162	13	8%
GB	198	120	8	6%
KR	341	145	2	1.3%
DE	118	81	6	7%
CN	6259	1006	4	0.3%
IR	270	89	3	3%
IN	1274	547	9	1.6%
US	10499	2714	252	9.3%
CA	274	151	53	35%
TR	150	87	36	41%

C. ISPs blocks OUT but not IN traffic

Based on the previous results, we obtain an estimate of how prevalent the outbound SMTP traffic blocking policy is on today's Internet. We delve deeper in the results to infer whether ISPs that block OUT traffic neglect to block IN traffic, where the latter is a necessary requirement for serving as a relay in triangular spamming.

1) Probing design: As shown in Figure 2 and discussed previously, it is easy to infer that the ISP is preventing outbound SMTP traffic, but non-trivial to discern at which direction blocking takes place. To summarize, we can first send a TCP SYN-ACK probe packet to some hosts in the IP range of interest with source port 80 and destination port 80 (or any other well-known ports). Depending on the OS and whether the port is open, the host may respond with a TCP reset (RST) packet. If we receive the corresponding RST packet, this shows that hosts will respond to probes to unused ports. We then immediately send another TCP probe packet but with source port set to 25. If we do not observe any response this time, assuming it is not the host firewall that blocks the traffic, it would be the ISP that blocks either IN traffic (which is our probe traffic) or OUT traffic (which is the RST response sent from the probed host). Note that it is also possible that the ISP spoofed the RST packets uniformly as their policy, and in this case, we will conservatively think that the ISP is not blocking port 25 while in reality spoofing RST can be a form of blocking. As a result, we may underestimate the port blocking prefixes.

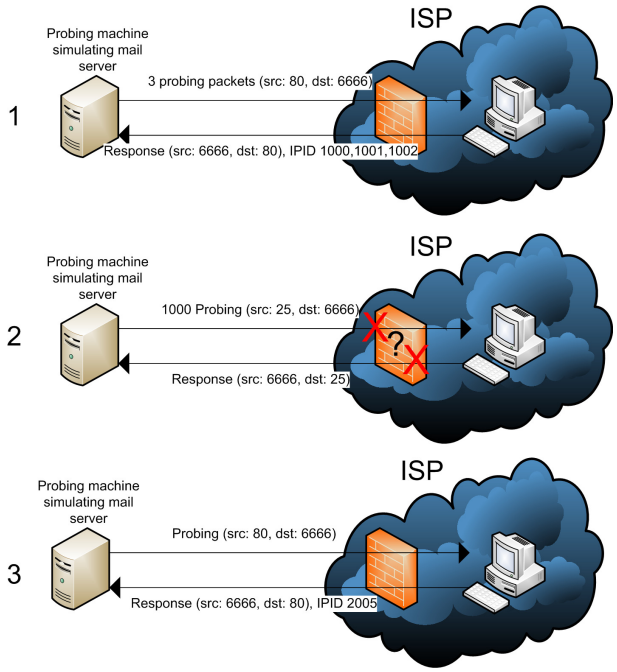

Figure 4. Outbound SMTP traffic blocking policy inference

However, since the latest large-scale study [43] did not report the exact same RST behavior (they discover the most popular RST injection is after SYN and SYN-ACK packet and in the same direction), we believe such behavior (RST after SYN-ACK in the opposite direction) is rare if deployed at all.

Making use of the properties of IPID values (ID field in IP header) generated by the end-host as many previous studies have done, we devise a simple approach to distinguish the IN or OUT traffic blocking. Figure 4 shows this probing methodology.

At step 1, suppose we already know that the ISP is blocking outbound SMTP traffic but have no idea whether it is IN or OUT blocking. We send several probing packet (*e.g.,* 5 packets) with source port 80 to some well-known ports. If we receive responses, we record the IPID values of the responses and detect the presence of a monotonically increasing pattern using a simple algorithm similar to nmap [20]. Let X be the last IPID value received. At step 2, we send a burst of packets (*e.g.,* 1000) with source port 25 to the same destination port and expect no response for these packets. At step 3, we send more probe packets again with source port 80 and examine the resulting IPID values in the response packets. If these values are roughly starting from $X + 1000 + E$ where E is the expected IPID value increase due to other packets sent by the host between Steps 2 and 3, then we can infer that the ISP performs OUT traffic blocking instead of IN traffic blocking. This is based on the conjecture that the increase in IPID values indicate that the host did receive our probe packets and responded to them, resulting in increase in IPID values. We did not receive the response packets due to ISP's firewall blocking

such OUT packets. On the other hand, if the IPID value has not increased by what is expected, we conclude that the ISP imposes IN traffic blocking and possibly also combined with OUT traffic filtering. Note that such a conclusion is unlikely to be incorrect due to the previously verified monotonically increasing IPID values.

Note that here we assume that the host probed has system-wide monotonically increasing IPID values which may not hold. For example, the IPID values can be random or always set to 0 for response packets that do not belong to the same TCP flow in recent Linux kernels. However, for Windows XP SP2 and SP3 that we tested (arguably still the most prevalent OS at the time we conducted probing), they all have such system-wide monotonically increasing IPID behavior. In fact, Windows 7 also has such property. Our probing results discussed next also verify this behavior for a large fraction of probed IP addresses. Hosts that do not have this property are not probed further. As long as we have sufficient number of samples from a prefix we can still infer the ISP-level policy.

Our probing test technique is summarized in Algorithm 1.

Algorithm 1 IN or OUT traffic blocking probing test algorithm

```
Input: Prefix p that has blocking behavior,
repeat {For each IP ip from the prefix p where except ip ended
with last octet 1 or 254 or 255}
   response1 = Probe(ip, 80, 80);
   response2 = Probe(ip, 25, 80);
   if(response2 == succ) notBlocking;
   else if(response1 == fail) unknown;
   else if(response1 == succ) {
       blocking;
       IPIDs = probeIPIDs(ip, 80, 80);
       if(increasing(IPIDs) == false) {
           IPIDNotIncreasing;
           continue;
       }
       burstProbe(ip, 25, 80);
       IPID = probeIPIDs(ip, 25, 80);
       if(IPID ≈ IPIDs[last] + E + 300)
       {   OUT-traffic-blocking;   }
       else IN-traffic-blocking;
   }

until [All ip in prefix p has been probed]
```

2) Results: We take candidate prefixes generated from our Web Flash experiment for the probing test algorithm to infer the ISP's policy. Some prefixes can be very large (*e.g.,*/11 or /12), requiring significant time to probe every single IP inside them. Instead, we probed only a subset of IPs in such prefixes due to time constraints and overhead. To prevent triggering any firewall alarms, for each prefix, we conservatively spawn only a single-threaded process to perform probing. As a result, on average, it takes 2 – 4 days to probe an /16 prefix. But since we are parallelizing the probing for different prefixes, we can still gather results in reasonable amount of time. We were able to probe most IPs for smaller prefixes, *e.g.,* prefixes smaller than /15. For larger prefixes, we covered about 25% (for some /11 prefixes) to 80% of their IPs.

Table III is an example of our probing result for prefix 24.247.80.0/20 which belongs to the Charter ISP [3]. We sub-divide the prefix into /24 prefixes based on the common assumption that finest policy granularity is at the /24 level. Each row represents the result for a particular /24. Each column shows the number of IP addresses within the /24 for a particular category. We can see that clearly most /24s are entirely OUT-traffic-blocking with only few exceptions. For instance, the fifth row has 7 IP addresses detected as OUT-traffic-blocking but only 1 IP was found to be not blocking outbound SMTP traffic, generating a potentially inconsistent configuration policy within the /24. However, we do know that it is common that ISPs allow customers to unblock port 25 for "power users" [16]. In this case, we believe that the few unblocked port 25 IPs are such exception cases. An anomaly is shown in the ninth row indicating that 24.247.88.0/24 has no IPs blocked for outgoing port 25. In fact, all 19 IPs are not blocked for either IN or OUT. Upon further investigation, we found that this /24 are static IP addresses (with DNS name such as 24-247-88-64.static.bycy.mi.charter.com) as opposed to dynamic IP addresses (with DNS names such as 24-247-80-0.dhcp.bycy.mi.charter.com). Usually static IPs are business-level customers who pay more for access to open ports [24]. Further, as expected we found that none of the IPs are identified to be IN-traffic-blocking.

Interestingly, we found that only 22 prefixes out of 688 blocking prefixes (3%) appear to deploy IN-traffic-blocking policy. We identify prefixes as IN-traffic-blocking if at least some /24 IP range contain the number of IN blocking IPs to be at least twice the number of both non-blocking ones and OUT blocking ones. In fact, when the number of IN blocking IPs is at least twice the number of non-blocking ones, it is almost always true that the OUT blocking IPs will be 0. However, it is still possible that some IPs are not blocked due to reasons such as customer's requests of sending outbound SMTP traffic. A quick analysis reveals that these IN blocking prefixes belong to US, Japan, and European countries like German, Sweden, UK, Belgium and Italy, mostly concentrated within European countries. There is only one educational instititution: umass.edu. Some ISPs in the U.S. also block IN traffic such as verizon.com. However, there are only 2 out of the 131 verizon prefixes that we probed perform IN blocking.

Note that some ISPs block outbound SMTP traffic (either IN or OUT blocking) on demand based on the SMTP traffic volume to prevent abuse. This can also reflect in our probing results that sometimes /24 prefixes have more non-blocked IPs than blocked ones. Typically, this is because the majority of the hosts are not sending spam, so outbound SMTP traffic from them is not blocked. Blocked hosts are likely the ones detected to be sending spam. In this case,

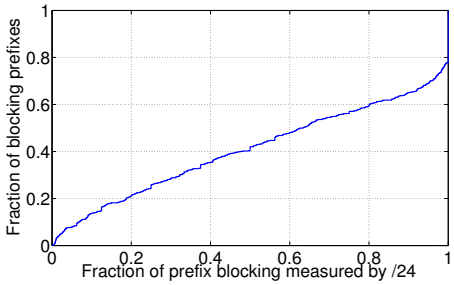

Figure 5. Distribution of blocking /24 subnet pctg (OUT SMTP traffic)

triangular spamming stealthily eliminates outgoing SMTP traffic from compromised bots, increasing the difficulties of detecting outgoing spam for ISPs.

Next, we analyze the results for OUT blocking prefixes. Besides the 22 IN blocking prefixes, the remaining 666 blocking prefixes are OUT blocking. Similarly, we consider a /24 as OUT blocking if the number of blocking IPs within the prefix is at least twice the number of non-blocking ones. Figure 5 plots the distribution of the percentage of blocked /24s within a prefix for all prefixes with at least some blocked /24s. We can observe that 20% of the prefixes are blocking all of their /24 sub-ranges and about 40% of the prefixes have about 50% /24 sub-ranges blocking outbound SMTP traffic. This shows the surprisingly non-uniform policy configuration at the ISP level and demonstrates room for improvement, as discussed later in §V. In subsequent discussions, we assume it is OUT blocking instead of IN blocking whenever we refer to blocking if not specified.

Table III
AN EXAMPLE OF IN/OUT TRAFFIC BLOCKING PROBING RESULTS FOR 24.247.80.0/20 (CHARTER.COM)

IP prefix	# of OUT blocking	# of non-blocking	# of unknown (*e.g.,* host down)	# of IN blocking
24.247.81.0/24	14	0	238	0
24.247.82.0/24	13	0	239	0
24.247.83.0/24	7	0	235	0
24.247.84.0/24	7	1	244	0
24.247.85.0/24	6	0	246	0
24.247.86.0/24	10	0	242	0
24.247.87.0/24	9	1	242	0
24.247.88.0/24	0	19	233	0
...	...	...	...	...

We analyze the inconsistent policy configuration behavior for prefixes with nonuniform outbound SMTP traffic filtering setting for its subnets. We found that the following three types are most common:

1. Dynamic IP vs. static ranges (or unblocked dynamic ranges), examples shown in Table III.

2. Sub-ranges delegated for other purposes. An example is shown in Table IV. Within the Stanford university's /14 prefix, some prefixes ranging from 171.66.120.0/24 to 171.66.127.0/24 have been assigned for other purposes. Many of corresponding reverse DNS names of IPs for such prefixes are changed to names like 'lcgsl.highwire.org' instead of '*.stanford.edu'. These IP blocks appear to be used for the Stanford University Press, which likely requires more admissive SMTP traffic policies.

Table IV
IN/OUT TRAFFIC BLOCKING PROBING RESULTS OF 171.64.0.0/14 (STANFORD.EDU)

IP prefix	# of OUT blocking	# of non-blocking	# of unknown (*e.g.,* host down)	# of IN blocking
...	...	...	...	...
171.66.120.0/24	0	232	20	0
171.66.121.0/24	0	199	53	0
171.66.122.0/24	0	228	24	0
...	...	...	...	...
171.66.128.0/24	4	0	248	0
171.66.129.0/24	0	0	252	0
171.66.130.0/24	5	0	247	0

3. Legitimate mail servers residing in the prefix, sometimes even co-located with a blocking /24 prefix. For example, we found that in a university prefix 128.118.1.0/24 contains several machines allowing outbound SMTP traffic, which are legitimate mail servers according to the MX records. Other than these machines, however, 28 other hosts are blocked for OUT traffic (The rest did not respond), indicating that the policy made exceptions for these mail server machines.

3) Correlation with Spamhaus PBL: Spamhaus PBL [37] is a popular blacklist widely used for identifying IP or IP ranges that should not deliver unauthenticated SMTP emails. The list includes both dynamic and static IP ranges and is gathered either from ISPs (ISP operators may volunteer to contribute to the list) or through other analysis. We are curious to know if the blocking prefixes we identified are already on PBL. If so, even if triangular spamming were used, they will still eventually be blocked at the mail servers (since the spoofed IP addresses fall into PBL). Surprisingly, out of 666 prefixes, there are only 296 (44%) of them are listed on PBL. It is possible that ISPs may think that since they already block the user IP ranges for OUT port 25, there is no need to report these IP ranges to PBL. However, with triangular spamming, this is not the case. It is still useful to report these IP ranges to PBL which can be considered as another layer of defense.

To understand what kind of prefixes is missed by PBL, we studied three prefix types as described previously. Out of 23 blocking educational institutions, PBL only listed one of them. We imagine the reason is that many universities have departmental mail servers which are difficult to be captured in a large prefix. The only one university prefix that gets on PBL is 130.18.78.0/23 where its DNS names are of the form ts3.dialup.msstate.edu. For 466 blocking ISPs, only 194 of them are listed. For the remaining 176 unknowns, 101 are listed. This shows that prefixes without DNS names have the highest listed ratio, indicating that they are more likely to be bad prefixes. For ISP prefixes, we found that PBL is still largely incomplete.

We summarize the findings from our extensive probing based analysis. We found that most ISPs today are not careful in blocking incoming SMTP traffic (with source port 25), despite some effort in reducing spam originated from their networks by blocking outgoing traffic destined to port 25. This opens many prefixes as relay nodes in triangular spamming scheme, resulting in these nodes participating in spamming in a quite stealthy way. Our designed probing methodology enables marking of specific prefixes vulnerable to triangular spamming for subsequent detection purposes.

IV. Experience and Analysis on Triangular Spamming

The previous section shows that today's network policies allow the possibility of carrying out triangular spamming. To better understand its operational model, as the next step, we build an actual triangular spamming infrastructure deployed in the Planetlab environment to explore various tradeoffs. In particular, we focus on the following questions:

1. Does triangular spamming require significant engineering effort and how easily can it be deployed (maliciously installed) on the relay bot and/or original sender?

2. Does triangular spamming really work in the real world (via Planetlab deployment)?

3. How much bandwidth utilization or throughput benefit can there be by using triangular spamming?

4. What property of the system can be leveraged for detection?

A. Implementation

Figure 1 shows that triangular spamming requires two separate components: one on the original sender and one on the relay bot. We build both components under Linux. Linux is chosen due to ease of development and deployment on Planetlab testbed. The development effort involves about 1700 lines of C code for the original sender and about 700 lines of C code for the relay bot. For the component on the original sender, it can either be deployed on a spammer-owned machine or some bot with good network connectivity. One can imagine that the number of such machines is likely smaller compared to common bots. For example, based on the Torpig study [38], about 22% of the infected hosts are in corporate networks that tend to have better bandwidth support than dial-up and cable networks.

Next we discuss in detail the implementation and design choice for each component.

1) Component on the original sender: To support IP spoofing, one can either modify the mail sending program directly or implement in a transparent fashion independent of the mail software. The latter is the preferred approach adopted by us because one can write mail sending program independently of the triangular spamming infrastructure. Our design of the component is shown in Figure 6. We intercept outgoing packets destined to port 25 and dynamically rewrite the source IP address to the relay bot's IP. We also modify the source port due to specific constraints of the Planetlab, as it only allows one to intercept incoming packets destined to certain reserved port ranges.

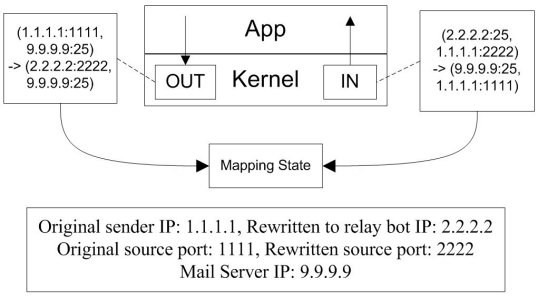

Figure 6. Component on the original sender

Note that the relayed packet has its source IP address set to the relay bot's IP (instead of the mail server's IP) because ISPs of relay hosts may prohibit IP spoofing. In fact, Planetlab does not allow IP spoofing [12]. As a result, we have to rewrite the source IP address to the mail server IP address upon receiving the packet at the original sender. We know which server IP address to rewrite to because we keep track of the mapping between spoofed source IP addresses and destination mail server IP addresses. Similarly, we rewrite the destination port to the actual port used by the original sender.

Since TCP is used, the sender and the receiver will take care of retransmission for lost packets. The relay bot is therefore stateless.

We use iptables' built-in support to intercept and deliver packets to the user-level program that can modify the packets and re-inject them. Since it involves additional kernel-to-user and user-to-kernel transition for every packet, it is not as efficient compared to a kernel-module-based approach. For ease of implementation, we chose to implement our prototype in user space. The associated overhead is not an inherent limitation.

2) Component on the relay bot: It is rather simple to implement the component on the relay bot given its functionality is to simply relay packets to the original sender by rewriting the destination IP (the source IP is also rewritten to avoid unnecessary IP spoofing). However, it is important to understand under which condition the bot relays packets. For instance, it is obviously unnecessary to relay non-SMTP packets. Instead, One simple strategy is to only relay packets with source port 25. As long as the original packet is not dropped and the user's SMTP traffic is not impacted (assuming the user also uses port 25), it is safe to relay them. However, sometimes, additional care has to be taken. Consider the original packet is not dropped and successfully received by the kernel, the kernel will consider the packet as out-of-state. Depending on the operating system (OS), it may generate a TCP reset packet upon receiving such packets. Reset packets are undesirable because they can terminate the TCP connection at the mail server side even though the original sender does not intend

to.

Fortunately, many OSes do not generate such packets. For instance, we have tested and verified that Windows XP SP2 and Windows XP SP3 (arguably still the most prevalent OS in use as of writing) by default do not generate such reset packets for closed ports. We suspect that the "silent drop" behavior is the correct behavior from security and privacy point of view. It is also the case for newer Linux kernels. However, we do discover that Windows Vista SP1 and Windows 7 will generate the reset packets.

In any case, even if the OS generates reset packets, they still can be dropped via mechanisms such as iptables.

B. Real-world deployment on Planetlab

We successfully deployed triangular spamming on Planetlab. Using tcpdump trace, we are able to verify that the server believes it is talking to the relay bot (setup on Planetlab node) instead of the original sender.

1) Limitation of Planetlab environment: Unfortunately Planetlab nodes generate reset packets instead of "silently dropping" out-of-state TCP packets likely due to their customized kernel. As a result, we have to modify the server to drop such reset packets to prevent the mail server side connection from being reset. However, such limitations are unlikely present on compromised machines because the attacker would have full control over the machines and can easily alter their behavior.

2) IP spoofing blocking study using Planetlab: After setting up the basic infrastructure, we investigate whether triangular spamming works in reality, meaning that if we spoof IP addresses of Planetlab nodes across the world, will our spoofed traffic be detected or dropped at certain ISP along the path via uRPF [21]? More specifically, although our university does not block spoofed traffic, intermediate ISPs may still block our traffic. Similarly, one may argue that spammers can always find an ISP that does not filter spoofed traffic, but this is only true for their home ISP, may not necessarily hold for intermediate ISPs. We empirically study from our vantage point to check whether our spoofed traffic is dropped.

The Spoofer project [27] shows that 80% of filters are implemented at a single IP hop from sources and 95% of the blocked packets are observably filtered within the source's Autonomous System. Their result is promising for triangular spamming, *i.e.,* as long as spammers can locate an ISP allowing IP spoofing (which is quite likely given that 33% of the tested IPs from the Spoofer study can spoof arbitrary IP addresses), it is unlikely that spoofed traffic is blocked in the middle of the path. This is because in practice implementation of such IP-spoof prevention techniques are often limited by multi-homing, route asymmetry, and other factors.

In our experiment, we attempt to spoof all available Planetlab nodes' IP addresses and use them to connect to multiple mail servers across various countries. Note that although the Planetlab node will generate a reset packet immediately after it receives the SYN-ACK packet from the mail server, it still relays the SYN-ACK packet back to the original sender so that we know the IP spoofing succeeded. We record each <spoofed IP, destination IP, isSucc> pair in our trace and the results are shown in Table V.

The result is mostly consistent with that of Spoofer [27]. Once the home ISP allows IP spoofing, it is likely to allow arbitrary routable IP address spoofing. For example, we can successfully spoof arbitrary Planetlab node IP addresses (except a node in Korea) to most popular U.S. mail servers except Yahoo. The result is shown in Table V. We have double verified by repeating the experiments several times for the failed IP addresses. Interestingly, not only India and Japan filter the spoofed Korea IP, Yahoo and Gmail in the US also filtered them. This is contradicting with Spoofer's results about arbitrary IP spoofing assertion that may be caused by insufficient diversity of spoofed nodes, although the Korea node seems to be an only exception here. For Yahoo mail server in China, the IP spoofing almost always failed. We verified that the result is consistent for other servers in China too, indicating that there is some specific filter along the path to China. However, despite these failed IP spoofing cases, the results are still very promising for triangular spamming as the filtering is sporadic and virtually non-existent for US destinations.

C. Bandwidth utilization analysis

1) Bandwidth utilization shifting: As we can see in Figure 1, the bandwidth utilization behavior completely changes from the perspective of relay bots. Without triangular spamming, a bot directly initiates a SMTP connection to the mail server and sends spam messages using its uplink bandwidth. With triangular spamming, all the uplink traffic is shifted to the original sender and such traffic is invisible to the relay bot's network, effectively lowering the bandwidth utilization of bots.

Further, since relay bots are most likely DSL or cable modem hosts, their downlink bandwidth is usually much higher than uplink bandwidth. For instance, a certain ADSL connection has a downlink bandwidth of 8 Mbit/s and a uplink bandwidth of 1.0 Mbit/s. Although the relay bot still needs to forward packets to the original sender consuming its uplink bandwidth, the forwarded packets are SMTP response packets and the SMTP-response traffic is much smaller than SMTP-send traffic. Analyzing randomly sampled spam from our local mail server log, we estimate that the size of SMTP-send traffic is about 5 to 10 times of the SMTP-response traffic, depending on what the spam message contains. As a result, we can conclude that triangular spamming allows spammers to associate many more concurrent connections with a relay bot without triggering alarms from bandwidth-usage-based anomaly detectors. We will show that in the next section, we can use selective forwarding to even reduce the bandwidth usage on relay bot even further.

Table V
IP SPOOFING RESULTS - SPOOFING PLANETLAB NODE IPS

Destination	Location	Spoofing-succ count	Failed Count	Failed IP & Location
Local Mail server	US	313	0	N/A
Hotmail	US	313	0	N/A
Yahoo	US	312	1	116.89.165.133 (Korea)
Gmail	US	312	1	116.89.165.133 (Korea)
Yahoo.com.cn	China	6	307	All except some servers in the US
University server	France	313	0	N/A
University server	India	312	1	116.89.165.133 (Korea)
University server	Japan	312	1	116.89.165.133 (Korea)
University server	Brazil	313	0	N/A
University server	Korea	313	0	N/A

For relay bots whose IPs are blocked for outgoing SMTP traffic, there is a clear advantage of using triangular spamming. It is primarily because IP addresses are scarce resources and blacklists nowadays can identify malicious IPs rather quickly so that the IP addresses may be rendered unusable. As we have shown in previous results, the IP ranges that block outgoing SMTP traffic are not necessarily listed on blacklists. This gives spammers strong incentives to use such IP addresses given they could still successfully deliver spam.

2) Spamming strategy and techniques: In this section, we show that triangular spamming offers an opportunity to improve spamming throughput (*i.e.,* the number of emails sent per second). Consider the following two spamming strategies:
Strategy 1: All bots directly send spam at their full speed.
Strategy 2: Triangular spamming is used where only high bandwidth bots send spam.

Strategy 1 is the baseline for comparison. This strategy provides a good overall throughput since it utilizes the distributed resources of the botnet. However, it has two noticeable disadvantages: first, it will expose the high bandwidth bot IPs; second, even the low bandwidth bots risk of being detected at its hosting ISP if they are sending spam at full speed. On the other hand, bandwidth-usage-based detector may not be able to catch high-bandwidth bot since it is spoofing different IP addresses. Moreover, spammers may also rent their own high-bandwidth machines in 'spammer-friendly' ISPs. For strategy 2, the high bandwidth bot can hide its own IP address while sending at full speed. For low bandwidth bot, given their bandwidth limitation, we think it might be a good idea to conserve their spamming activity. Instead, they can be mostly focusing on relaying server responses back to the sender.

We envision two spamming techniques under Strategy 2 that can help improve throughput for triangular spamming:

Technique 1. Selectively relaying packets at the relay bot — reducing unnecessary network bandwidth usage.

Given that the common case is that senders can successfully deliver emails. It is not really necessary for the sender to receive the response from the mail server. We have verified using our triangular spamming prototype that the relay bot needs to relay only the TCP SYN-ACK packet to the high bandwidth bot for spamming. This technique can significantly reduce both the uplink bandwidth usage of the relay bots and the total bandwidth usage of the high bandwidth bot. Depending on the email size, the SMTP-response (incoming traffic) at the high bandwidth bot is around 1/5 – 1/6 of the total traffic when the email body size is around 1700 bytes (this is relatively large spam email size likely in HTML format). It is possible that some messages are larger such as image spam and some spam messages are smaller — many spam messages only contain a few words then a link to a scam Web site or a messenger contact. For cases where spam messages are smaller, it is a clear benefit in bandwidth usage reduction.

Note that the above is somewhat an ideal case, there are some minor issues at the TCP layer that need to be fixed. First, from the mail server's perspective, it may not receive any TCP ACK messages for its response packets since the high-bandwidth bot never gets them in the first place. But in reality, the mail server's initial congestion window is large enough to hold all the outgoing packets without receiving any ACK (although it may cause the mail server to unnecessarily retransmit the response packets). One possibility is to let the relay bot to ACK mail server's response packets directly without burdening the high-bandwidth bot. Similarly, at the sender side, although the initial congestion window at the high-bandwidth bot is also typically large enough to hold all outgoing packets without getting any ACK. It would again cause unnecessary retransmissions that waste bandwidth resources. In order to mitigate this issue, we have two options: 1) let the relay bot relay the ACK packets from the mail server to the high-bandwidth bot or 2) spoof the ACK packets locally at the high-bandwidth bot. The second option has the potential problem of not able to detect packet loss (since we always spoof the ACK without knowing whether it is received by the mail server), although this may rarely happen. The first option will use some bandwidth to relay the ACK packets but the size of ACK packets should be relatively small and it is simpler. We have successfully implemented the first option and verified that the emails can be successfully received by mail servers.

Technique 2. Aggressive pipelining.

The SMTP protocol is interactive and I/O bound as each SMTP session typically involves many round trips limiting the aggregated throughput. Thus, SMTP has incorporated

the pipeline support as introduced in RFC2920 [15] in 2000 to pipeline the commands to reduce the overall session time. In the extreme case, one may send all commands in a single batch to the server. However, as specified in RFC, the EHLO, DATA, VRFY, EXPN, TURN, QUIT, and NOOP commands can only appear as the last command in a group since their success or failure produces a change of state that the SMTP client must accommodate. In order to test the pipelining support in today's mail server, we pick a set of popular mail servers (both open source and commercial) including: sendmail, Java Apache Mail Enterprise Server (JAMES), Gmail, Hotmail, Yahoo mail, and AOL mail. Interestingly, only two mail servers, Gmail and AOL mail, strictly enforce the RFC. All other mail servers allow full pipelining (sending all commands in a single batch). For Gmail and AOL mail, we have to wait after the server processes each 'critical' command such as EHLO before we can issue the next set of commands. Normally we know that the server has finished processing a command by observing its response. However, if RTT is large, spammers will have to wait for very long before they can move on to the next set of commands. But based on our experiments on a variety of mail servers that we tested, the next set of commands will be accepted as long as the server has finished processing the previous 'critical' command. This means that it is possible to aggressively pipeline the commands such that the next set of commands arrive just after the server finishes processing the previous 'critical' command. Typically, the processing time of the 'critical' command should be smaller than the wide area RTT which can be hundreds of milliseconds.

Algorithm 2 and 3 have the pseudo-code that illustrates different pipelining approaches. In Algorithm 3, when $t1 = t2 = 0$, it becomes full pipelining.

Algorithm 2 Normal pipelining

```
send("EHLO [hostname]");
recv_and_process(response);
send("MAIL FROM: <sender@aaa.com>\r\nRCPT TO:
<receiver@bbb.com>\r\nDATA\r\n");
recv_and_process(response);
send("[actual data]\r\nQUIT\r\n");
```

Algorithm 3 Aggressive pipelining

```
send("EHLO [hostname]");
sleep(t1);
send("MAIL FROM: <sender@aaa.com>\r\nRCPT TO:
<receiver@bbb.com>\r\nDATA\r\n");
sleep(t2);
send("[actual data]\r\nQUIT\r\n");
```

Here, since the EHLO command is relatively simple to process, the processing time is usually very small. However, for the next set of commands (MAIL FROM to DATA), there are three commands combined together, which may take the mail server longer to process. By carefully choosing delay $t1$ and $t2$ in Algorithm 3, one can potentially increase the throughput for every single connection and possibly the overall spamming throughput.

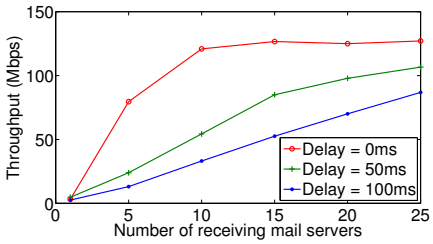

Figure 7. Impact of delays on the spamming throughput

Next, we try to quantitatively study the impact of delay $t1$ and $t2$ on the throughput. We conduct the throughput experiment on Emulab where 25 pc3000 machines with 1Gbps network interface are used. Each machine has a single 3GHz core. We pick one machine as the sender and the rest of the machines as potential receiving mail servers running the open source mail server JAMES. We spawn a large number of threads for concurrent connections for each mail server. Each thread continuously sends emails with a new TCP connection. As shown in Figure 7, the throughput increases as the number of mail servers increases, indicating that the initial bottleneck is at the mail server side. In the experiment, we choose the delay $t1$ and $t2$ to be 0ms, 50ms and 100ms respectively and draw the throughput curve accordingly. Figure 7 shows that it is difficult to gain higher throughput when the corresponding delay is large. Without aggressive pipelining, to achieve high throughput, spammers may need to pick a large number of concurrent mail servers (which could be possible). With aggressive pipelining, one may be able to achieve significant throughput improvement with the same number of mail servers by reducing the delay $t1$ and $t2$. For the case of 100ms and 50ms delay, the throughput improvement is about 1.5X – 2X with the same number of mail servers.

In reality, on the high-bandwidth bot, two of the following can happen: 1. Network bandwidth is the limiting factor (network bandwidth can be fully utilized). 2. CPU is the limiting factor (too many concurrent connections may cause context switches to occur too frequently such that the bandwidth may not be fully utilized). The throughput saturates or grows slowly as the number of concurrent connections increases.

Technique 1 applies to case 1 given that it can reduce unnecessary messages received at the high-bandwidth bot. At the high-bandwidth bot, it is likely that the uplink and the downlink are shared (Ethernet rather than ADSL). If the network bandwidth is the bottleneck, this technique can free up additional bandwidth to deliver spam messages.

Technique 2 applies to case 2 as shortening each individual RTT can help improve the overall throughput. Intuitively, $t1$ and $t2$ have to be at least greater than the server processing time for the corresponding commands. To get an idea of this value, we empirically vary $t1$ and $t2$ and target at one Gmail server which does not allow full pipelining. We perform the measurement on both peak hours (noon) and off-peak hours (mid-night). We found

that during peak hours, $t1$ = 400ms and $t2$ = 800ms are often large enough to ensure successful email delivery. For off-peak, $t1$ = 20ms and $t2$ = 40ms are large enough. The difficult question is what delay value for $t1$ and $t2$ to pick in practice. Without triangular spamming, since each bot can only send one or two spam messages (to avoid being blacklisted), there is no easy way for them to reuse the learned processing time. One possibility is to let bots coordinate the learned processing time, but this can be inefficient. Another possibility, offered by triangular spamming, is to use the measured processing time from one or more previous connections.

The reason that it can work under triangular spamming setting is that it is easy to share the measured processing time information across multiple connections (all with different spoofed IPs) given all connections happen on the same physical machine (the high-bandwidth bot). More specifically, when triangular spamming starts, we open multiple connections for each target server. There are some bots that relay packets back earlier than others. We can use the RTT values observed from quick bots as an approximation for the processing time. One potential problem to consider is that we should avoid making too many concurrent connections to the same server because it will likely overload the server and inflate the processing time. So it is a good idea to spread the connections across multiple mail servers. A simple way to do so is to spread the connections across multiple IP addresses/machines exposed by a single mail provider, or sometimes even a single IP address may also correspond to multiple servers internally.

To study the feasibility of the second technique, we again use the Planetlab to measure how diverse RTT values can be, *i.e.,* quick bots vs. slow bots, in a globally distributed environment simulating a botnet. We use a machine in a university to act as the original sender, as university networks are typically well-provisioned. The idea is that if there are indeed many slow bots, we can use the second technique to reduce RTT and increase throughput.

Figures 8 through 11 show the RTT distribution for different target mail servers. We can see that for Hotmail and Gmail servers, the RTT distribution is quite diverse ranging from 50ms to 300ms. If we assume that we only need a single connection to compute the approximate processing time, it can improve the throughput significantly.

For the local mail server experiment, we simulate the scenario where triangular spamming is carried out within the same ISP or organization as the victim mail server. In this case, although the direct RTT between our original sender and the local mail server is only 0.4ms, the RTTs observed are much larger due to triangular routing. However, the increased stealthiness achieved by triangular spamming has the cost of affecting throughput due to large RTTs. Aggressive pipelining could help to improve the throughput of each individual connection significantly.

For the Indian mail server experiment, we simulate the scenario where spammers are targeting a mail server far away from the original sender. We can see that the RTT is clustered at around 200 – 300ms, for 82% of IPs studied, which is mostly bounded by the RTT between the original sender and the target mail server. In fact, the smallest RTT is 227ms, indicating that it could be effective to use aggressive pipelining. But some initial measurement of the processing time has to be done rather than in parallel (where the processing time measured from quick bots can be used for slow bots).

D. Implication on detection

We observe that although the IP address can be spoofed, some properties exhibited by the original sender may not be easily imitated. For instance, they may run different operating system and resulting in different OS fingerprints. Also, the network delay between the target mail server and the original sender can be different from the delay between the target mail server and the spoofed host. If we can probe the spoofed host in real time to detect deviations in such properties, we may be able to discover triangular spamming. In this section, we briefly discuss several properties promising for detection. Detailed detection results will be shown in §V.

1) Round Trip Time difference: As we have shown in Figures 8 through 10, RTT can differ widely across relay bots. However, from the target mail server's perspective, it does not know the original sender's IP address and can only observe two other RTTs. One is active RTT between itself and the relay bot by direct probing. The second is passive RTT observed locally by observing the delay between sending SYN and receiving SYN-ACK. If no triangular spamming is involved, the two RTT values should be comparable.

However, in the presence of triangular spamming, the passive RTT is calculated as $t_1 + t_2 + t_3$ where t_1 to t_3 correspond to the network delays of the three steps shown in Figure 1. The active probed RTT can be calculated as $t_2 + t_2'$ where t_2' is the reverse path network delay of step 2. For simplicity, we assume t_2' to be roughly the same as t_2 (similarly for t_1 and t_3 as well) which allows us to calculate the likelihood of detecting RTT differences. If we compare the passive RTT with the active RTT, the difference is $(t_1+t_2+t_3)-2\times t_2 = (t_1+t_3-t_2)$. Although triangular inequality is shown to be invalid sometimes [40], we show that the chances that $t_1 + t_3 - t_2$ is close to 0 would still be small.

To understand how likely we can observe large values for $t_1 + t_3 - t_2$, we again conduct experiments on Planetlab. First, we measure $t_1 + t_2 + t_3$ as previously described. Second, we measure $2 \times t_1$ by probing from the original sender to the target mail server. Last, we measure $2 \times t_3$ by probing from the original sender to the Planetlab nodes. The distribution of the value $t_1 + t_3 - t_2$ is shown in Figures 12 through 15.

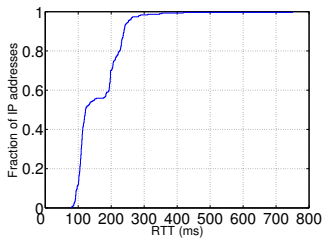

Figure 8. Hotmail RTT

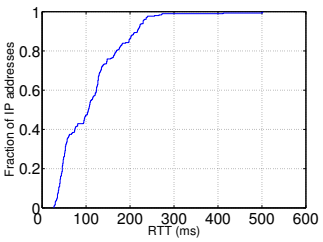

Figure 9. Gmail RTT

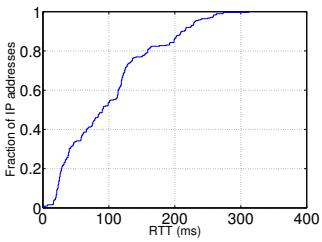

Figure 10. Local mail server RTT

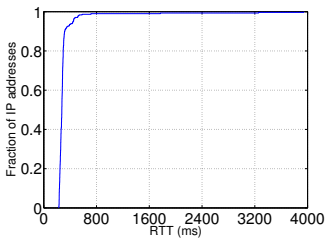

Figure 11. Indian server RTT

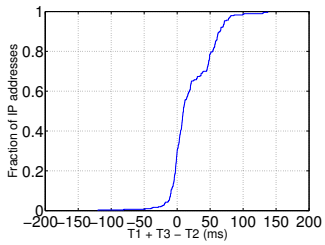

Figure 12. Hotmail passive/active RTT difference

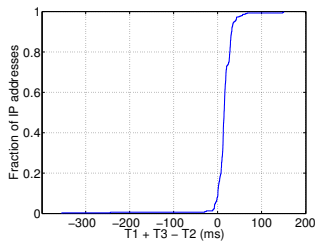

Figure 13. Gmail passive/active RTT difference

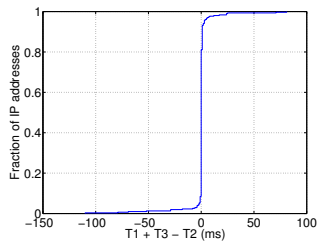

Figure 14. Local mail server passive/active RTT difference

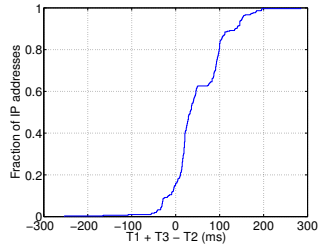

Figure 15. Indian server passive/active RTT difference

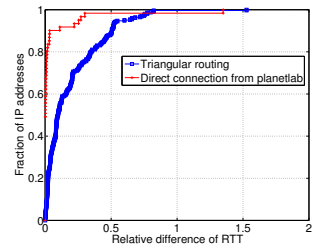

Figure 16. Hotmail relative deviation from passive RTT

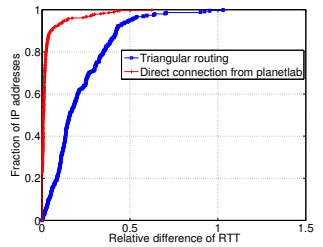

Figure 17. Gmail relative deviation from passive RTT

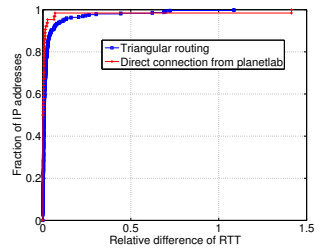

Figure 18. Local mail server relative deviation from passive RTT

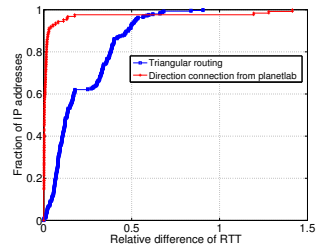

Figure 19. Indian server relative deviation from passive RTT

The results show that for Hotmail server, about 20% of the IP addresses exhibit a difference between passive RTT and active RTT of 50ms or larger. Depending on the relative deviation from the passive RTT, 50ms can be sufficiently large to be considered as an anomaly. For Gmail servers, the difference is much smaller, so is the absolute RTT value. For the local mail server, little difference is found between the passive RTT and the active RTT. This is expected because original sender and the target mail server are very close such that t_1 is close to zero and t_2 and t_3 are approximately the same.

From Figure 16 to Figure 19, we can see that the distribution of relative difference of the active and passive RTT computed by $\frac{t_1+t_3-t_2}{t_1+t_2+t_3}$. However, in order to know if the difference is large enough to be an anomaly, we need to know the baseline of normal RTT variation in a short period of time (within the order of seconds). Although it is known that the RTT variation can be very large over time, in triangular spamming detection, we can set up a real-time active probing infrastructure to probe the active RTT and compare it against the passive RTT. If the real-time RTT variation is small, it is still possible to detect such relatively large and stable RTT difference introduced by triangular routing. We measure the RTT variation by sending 3 consecutive probes with 1 sec interval from the Planetlab nodes. The results clearly show that 95% of the times the relative difference will be smaller than 0.1. This implies that the relative RTT difference can be a useful feature for detecting triangular spamming.

2) TTL value difference: Similarly, TTL values generated from the original sender can differ from the ones generated by the relay bot as observed via active probing. Previous work has studied the effectiveness of using TTL value to detect spoofed DDoS traffic [41]; thus, we do not repeat the study of measuring the difference in TTL values. Here we point out one key difference between DDoS attack and triangular spamming: spoofed DDoS attacks usually have no control over the hosts with spoofed IPs, but in triangular spamming, the relay bots coordinate with the original sender. It is thus harder to detect triangular spamming. More specifically, the original sender can craft a starting TTL such that the receiver cannot tell whether it is generated by a different host other than the relay bot.

However, this coordination has to be done on a per-destination and per-relay-bot basis which is likely high overhead and may severely degrade the spamming throughput. This is not a huge problem in DDoS attack since the attack is typically targeted and well prepared before the actual

attack. The difficulty lies in obtaining the correct initial TTL at the real sender.

In conclusion, we think TTL can be a useful feature for detecting triangular spamming despite the robustness problem outlined above.

3) OS fingerprint: It is also possible to collect lightweight passive OS fingerprints from the first SYN packet using tools such as p0f [11]. Clustering the IP addresses by fingerprints can help detect traffic associated with the same original sender. However, this can be evaded since the original sender can easily modify its kernel to mimic different types of OS fingerprints.

4) Port blocking behavior: If the OUT SMTP traffic is blocked at the ISP, there should not be any SMTP traffic generated from such IP addresses. As a result, if we can identify that an incoming IP is blocked for OUT traffic, then it is highly likely that triangular spamming is used. The limitation is false negatives. This can only detect cases where the relay bots are indeed blocked for OUT traffic which is not a requirement for triangular spamming.

V. Detection and Prevention

In this section, we discuss how to apply the previously discussed techniques using data on our departmental mail server to detect any evidence of triangular spamming and its prevalence. We also discuss possible prevention techniques.

A. Experiment setup

Data source. Monitoring a mail server at our university department mail server for 8 days from 2009.11.9 – 2009.11.16, we observe about 360,000 emails, of which 233,746 (87.6%) emails are spam emails (according to SpamAssassin) from 200,347 distinct IP addresses. Each log entry has four pieces of information: timestamp, sender IP, spam tag, and spam score output by SpamAssassin.

Spam filter - SpamAssassin. Our mail server runs SpamAssassin [2] as the spam filtering system. It employs several detectors including Spamhaus [19] (IP-based blacklist). Every email is labeled based on a computed score combining results from all detectors. If the score exceeds a fixed threshold (5.0 in our case), the corresponding email will be labeled as spam.

Real-time probing experiment. We probe in real-time to gather the TTL and RTT information to help detect triangular spamming. Two sets of probes per destination port per host are conducted, one with source port 25 and the other with source port 80. The destination port is chosen from the most popular ports potentially open on the hosts, *e.g.,* port 25, 80, 22. We limit probing to at most 4 different destination ports to limit the overhead. Probing is stopped if any destination port responds. A lack of response from source-port-25 probes in the presence of responses from other source ports (*e.g.,* port 80) indicates that the IP address is blocked for SMTP traffic.

B. Detection results

Given that triangular spamming can abuse port-25-blocked IP addresses for sending spam, we are interested in knowing the prevalence of such IP addresses seen by our mail server and the usefulness to correlate this with other features such as TTL discrepancy from real-time probing. For instance, if their port 25 is blocked, does that mean the actively probed TTL value is more likely to be deviate from the passively observed TTLs?

Our results show that on average about *4%* of IP addresses observed by our mail server are blocked for port 25 based on our real-time probing, indicating likely presence of triangular spamming. We then compute various correlations to verify whether such blocked IPs differ from other IPs in any particular properties. Note that if these IP addresses are involved in triangular spamming, they might also relay our probing packets such that we can still get response. However, the original sender may think our probing packet is invalid and thus do not care to respond.

Spam ratio. The spam ratio of these blocked IP addresses are 99.9%. Compared to the overall 87.6% spam ratio, it is evident that these IP addresses are much more likely to send spam. In fact, we found only 4 blocking IP addresses (out of 7246) sent legitimate emails. Upon a closer look, 1 of them is likely false negative by SpamAssassin based on its DNS name (18.151.195.200.static.copel.net) and it is listed on multiple blacklists. The other three do appear to be legitimate mail servers. However, we found that these three servers have a particular security policy that results in no response to our SYN-ACK packet if the packet has source port 25. They do however respond to SYN packets with source port 25. We plan to incorporate SYN probing in our real-time probing in the future to more reliably determine the port 25 blocking behavior.

TTL or hop count value difference. For these blocked IP addresses, we compare the actively probed TTL value using source port 80 with the passively observed TTL value. Using TTL value directly can result in inaccuracies due to differences in initial TTL value settings. Instead we infer the hop count values using previously established approach [31] to compute the absolute difference in hop count for the two TTL values. Besides triangular spamming, the discrepancy could also stem from firewall or gateway responding to our probing, which is not easy to discern from anomalous triangular routing. Figure 20 plots the distribution for hop count difference for blocking and non-blocking IPs. It indicates that blocked IP addresses are more likely to have a larger hop count difference, likely caused by triangular spamming.

RTT difference. Similar to TTL, we plot the relative difference of RTT for blocking and non-blocking IP addresses shown in Figure 21. We can see that clearly blocking IPs tend to have a larger relative difference. In fact, more than 50% of the blocking IPs have a relative difference greater than 0.1 which is already larger than the expected difference

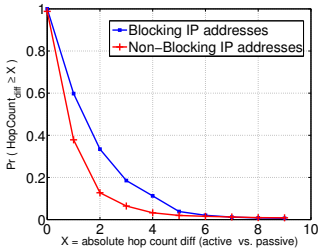

Figure 20. Absolute hop count for blocking and non-blocking IPs

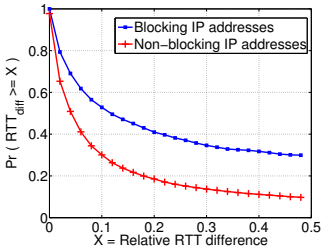

Figure 21. Relative RTT difference for blocking and non-blocking IPs

as shown in our planetlab RTT study.

OS fingerprints. We group a set of IP addresses into clusters based on the passively observed OS fingerprints. Here we include hop count in the OS fingerprint (which we call signatures) to consider IPs with same OS fingerprint but with different hop count as different senders. The results show that although there are 52,860 different signatures out of 233,746 IP addresses, the entropy of the signature is only 10.7 bits, indicating that it may not be diverse enough. We also found that the blocking ports are often with different signatures, likely caused by the fact there are multiple original senders involved in the current triangular spamming architecture. In fact, there can be a hierarchical structures where there are a set of high bandwidth bots acting as original senders and each of them peers with multiple relay bots.

Blocking prefix correlation with a popular mail service provider. By correlating with a month-long mail service provider's sampled log trace using the coarse-grained blocking prefix information, we gathered 93,359 IP addresses of which 97% are sending purely spam. While there are certain legitimate mail servers mixed in the blocking prefixes, it is possible to filter them based on their DNS MX records (reverse look up their hostname based on IP, and then look up the MX record for that domain name). Further, 95% of these IP addresses appear only in 5 or fewer days and 54% of them only appear in a single day in the month-long trace, indicating that they are more likely to be spam hosts. It is known that stable IP addresses are tend to be legitimate while appear-once IP addresses are likely to be spamming IP [39].

From the results above, we can conclude that despite the stealthy behavior of triangular spamming, it also exposes information that can be leveraged for detection. Features proposed above are lightweight enough that they can be easily collected during run-time to help classify spam.

C. Prevention

Two straightforward ways exist to prevent triangular spamming. The approach of disallowing IP spoofing at every ISP is not feasible given the scale of the Internet and a lack of unified configuration enforcement. The other more realistic way is to have ISPs that block OUT traffic also block IN traffic. However, it does put the management burden on ISPs to correctly configure the firewall rules. Another less obvious prevention method is to deploy stateful firewalls at ISPs to prevent relay bot from relaying out-of-state TCP traffic or just focus on port 25 related traffic to limit imposed overhead. For instance, it is possible to push such functionality into the modem so that it does not have to be deployed at some centralized point to cause performance problems. The question is whether stateful firewall is a desired feature for customers. We think arguably that this can be the right decision and most users will not likely be impacted, just as the case for outbound SMTP traffic blocking.

VI. Related Work

There has been a significant amount work on spam detection over the years. The techniques proposed can be broadly categorized into content-based, blacklist-based and behavior-based. Content-based approach has been mostly extensively studied [30], [32]. The blacklist-based approach was originally from third-party used to blacklist individual IP address. Popular ones include SpamHaus [19], SpamCop [18] and SORBS [17]. Behavior-based approach is complementary and is growing in popularity (*e.g.,* [34], [29]). The detection methodology we proposed focusing on spammer behavior also belongs to this category.

Our work is instrumental in highlighting the arms race caused by evolving spamming techniques in response to the improved detection methods. When the blacklist-based approach first came out, spammers try to avoid being listed by having each bot IP send very few spam to each target mail server domain to stay under the radar [33]. In an effort to prevent outgoing spam, many ISPs enforce outbound SMTP traffic blocking mostly on dynamic IP ranges. As more ISPs perform outgoing SMTP traffic blocking, fewer IP addresses are available to spammers. However, our work demonstrates that current ISPs' port blocking practice is insufficient and triangular spamming can still leverage those blocked IP addresses to send spam in a stealthy manner without triggering alarms monitoring outbound SMTP traffic. From the perspective of ISPs where relay bots reside, they will not notice too much traffic since the bots are only relaying response messages from the server side, at a much lower rate compared to original outgoing spam. Another side benefit is that the bot can still participate in other malicious activities such as DDoS attack.

In terms of detecting triangular spamming, we can borrow ideas from previous studies of general IP spoofing defense. For instance, TTL value is proposed to detect spoofed DDoS traffic in [41]. Cryptographic puzzles [42] can be introduced to slow down the spoofer. However, they are all different from the triangular spamming setting where botnets cooperate with the spoofer to carry out the attack.

Also related to our study is prior work on firewall policy inference such as FireCracker [35]. However, there are two key differences: 1). their goal differs from ours in that they try to infer the complicated policies that depend on IP/port

combinations and try to reduce the number of probes to infer such behavior, while our goal is to determine IN/OUT traffic blocking behavior. 2). We are inferring ISP-level policy which has certain characteristics that allow us to infer the IN/OUT blocking. More specifically, since policy is uniform on a range of IP addresses, we can probe the entire range and leverage the behavior in IPID value change for some hosts within that range to help our inference.

VII. Discussion and Conclusion

To conclude, our work is the first to highlight the practice of triangular spamming, a stealthy and fairly efficient spamming technique that can be relatively easily carried out on today's Internet by thoroughly investigating its feasibility in the presence of existing network policies and ease of operational deployment. We bring attention to the community that today's SMTP traffic blocking policies can be bypassed to carry out such spamming activities while protecting the identities of hosts actually sending out spam messages. Through extensive empirical data analysis and experimental evaluation, we propose detection and prevention schemes that are shown to be promising approaches to mitigate against this relatively new spamming practice.

Note that triangular spamming exploits the network level security vulnerabilities (IP spoofing and insufficient firewall port blocking policy). We believe this is just one instance where application protocols are misused due to the underlying network vulnerabilities. Other attacks remain possible.

References

[1] Adobe flash player. http://www.adobe.com/products/flashplayer/.

[2] The apache spamassassin project. http://spamassassin.apache.org/.

[3] Chater cable, internet, telephone. www.charter.net/.

[4] Cisco pix 515e firewall quick start guide, version 6.3. http://www.ciscosystems.org.ro/en/US/docs/security/pix/pix63/quick/guide/63_515qk.html.

[5] Comcast takes hard line against spam. http://news.zdnet.com/2100-3513_22-136518.html.

[6] Hacking the interwebs. http://www.gnucitizen.org/blog/hacking-the-interwebs/.

[7] The icsi netalyzr beta. http://netalyzr.icsi.berkeley.edu/.

[8] Microsoft: 3% of e-mail is stuff we want; the rest is spam. http://arstechnica.com/security/news/2009/04/microsoft-97-percent-of-all-e-mail-is-spam.ars.

[9] Nanog survey - isp port blocking practice. http://seclists.org/nanog/2009/Oct/727.

[10] netfilter/iptables project homepage. http://www.netfilter.org/.

[11] The new p0f 2.0.8 (2006-09-06). http://lcamtuf.coredump.cx/p0f.shtml.

[12] Planetlab acceptable use policy (aup). http://www.planet-lab.org/aup.

[13] Port 25 blocking. http://www.postcastserver.com/help/Port_25_Blocking.aspx.

[14] Port 25 (sonic.net). http://sonic.net/support/faq/advanced/port_25.shtml.

[15] Rfc2920, smtp service extension for command pipelining. http://tools.ietf.org/html/rfc2920.

[16] Sbc email problem. www.sfsu.edu/~helpdesk/sbc/index.htm/.

[17] Sorbs. http://www.au.sorbs.net/.

[18] Spamcop. http://www.spamcop.net/.

[19] Spamhaus. http://www.spamhaus.org/.

[20] Tcp/ip fingerprinting methods supported by nmap. http://nmap.org/book/osdetect-methods.html.

[21] Unicast reverse path forwarding. http://en.wikipedia.org/wiki/Reverse_path_forwarding.

[22] Universal plug and play. http://en.wikipedia.org/wiki/Universal_Plug_and_Play/.

[23] Useful statistics for web designers. http://www.tvidesign.co.uk/blog/useful-statistics-for-web-designers.aspx.

[24] Verizon - our attention needed: Re-configure your email settings to send email. http://www22.verizon.com/ResidentialHelp/HighSpeed/General+Support/Top+Questions/QuestionsOne/124274.htm.

[25] Whois ip address/domain name lookup. http://cqcounter.com/whois/.

[26] F. Baker and P. Savola. Ingress filtering for multihomed networks. In *RFC 3704*, 2004.

[27] R. Beverly, A. Berger, Y. Hyun, and k claffy. Understanding the efficacy of deployed internet source address validation filtering. In *In Proc. of IMC*, 2009.

[28] P. Ferguson and D. Senie. Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing. In *RFC 2827*, 2000.

[29] S. Hao, N. A. Syed, N. Feamster, A. Gray, and S. Krasser. Detecting Spammers with SNARE: Spatio-temporal Network-level Automatic Reputation Engine. In *Proceedings of Usenix Security Symposium*, March 2009.

[30] F. Li and M.-H. Hsieh. An empirical study of clustering behavior of spammers and group-based anti-spam strategies. In *In Proc. of CEAS*, 2006.

[31] Z. M. Mao, L. Qiu, J. Wang, and Y. Zhang. On as-level path inference. In *In Proc. of SIGMETRICS*, 2005.

[32] B. Medlock. An adaptive, semi-structured language model approach to spam filtering on a new corpus. In *CEAS 2006 - Third Conference on Email and Anti-Spam*, July 2006.

[33] Z. Qian, Z. M. Mao, Y. Xie, and F. Yu. On network-level clusters for spam detection. In *In Proc. of NDSS*, 2010.

[34] A. Ramachandran, N. Feamster, and S. Vempala. Filtering spam with behavioral blacklisting. In *In Proc. of CCS*, 2007.

[35] T. Samak, A. El-Atawy, and E. Al-Shaer. Firecracker: A framework for inferring firewall policy using smart probing. In *In the Proceedings of the fifteenth IEEE International Conference on Network Protocols (ICNP'07)*, 2007.

[36] S. Sinha, M. Bailey, and F. Jahanian. Shades of Grey: On the Effectiveness of Reputation-based "Blacklists". In *Malware 2008*, 2008.

[37] Spamhaus policy block list (PBL). http://www.spamhaus.org/pbl/, Jan 2007.

[38] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your botnet is my botnet: Analysis of a botnet takeover. In *In Proc. of CCS*, 2009.

[39] S. Venkataraman, S. Sen, O. Spatscheck, P. Haffner, and D. Song. Exploiting network structure for proactive spam mitigation. In *In Proc. of USENIX Security Symposium*, 2007.

[40] G. Wang, B. Zhang, and T. S. E. Ng. Towards network triangle inequality violation aware distributed systems. In *In Proc. of IMC*, 2007.

[41] H. Wang, C. Jin, and K. G. Shin. Defense against spoofed ip traffic using hop-count filtering. *IEEE/ACM Trans. Netw.*, 2007.

[42] X. Wang and M. K. Reiter. A multi-layer framework for puzzle-based denial-of-service defense. *Int. J. Inf. Secur.*, 2008.

[43] N. Weaver, R. Sommer, and V. Paxson. Detecting forged tcp reset packets. In *In Proc. of NDSS*, 2009.