The ISP Column
A monthly column on things Internet

February 2012
Geoff Huston

Bogon Filter Detection

Until recently IP network operators were encouraged to set up so-called "bogon address filters" at the edge of their networks. These filters were intended to discard all incoming traffic where the source address in the IP header was from a block of addresses that was known to be unallocated. The inference was that a matching packet was either an unintentional leak from some privately addressed network domain or was generated using source address spoofing. In either case there is no point in delivering the packet, since it comes from a demonstrably fictitious source.

The initial use of these bogon filters was to mark those /8 address blocks in the IPv4 address space that were held by the IANA, although some tools were subsequently developed (and deployed in more limited contexts) to also mark as bogons address blocks held as unallocated by the Regional Address Registries (RIRs), and other tools were developed to mark address blocks that were consistently used to originate SPAM or malicious traffic, based on collected statistics and other heuristics.

In the continuing escalating war with spam, abuse and malware, it appears that malware authors rapidly moved on from attempting to hijack such bogon addresses as a platform for launching attacks. The entire rationale for deploying bogon filters became somewhat dubious, and if my mailbox is any indicator, the volumes of spam on the Internet continued unabated. The continued efficacy in such filters in eliminating malware and abuse appeared to have little in the way of factual substantiation. But the ISP security industry apparently loves a good pantomime, where the superficial veneer of security replaces any substantive and potentially more intrusive and expense security response, and the use of bogon filters was certainly a widespread item of security pantomime in the ISP 's operational manual.

But not only were they largely ineffectual, these bogon filters started generating their own set of operational issues. A persistent issue with the use of these bogon filters is that they can lose currency. When IP address blocks changed from "unallocated" to "in-use" (such as an IANA allocation action, or a RIR assignment or allocation of addresses to a Local Internet Registry) then the rationale for maintaining the corresponding entry in a bogon filter ceased. But many network admins appear to be somewhat apathetic in their maintenance of these filters, and when they fall out of date they start to filter "real" network traffic and become an impediment to reachability. The innocent recipient of these recently allocated addresses finds that their view of the rest of the Internet is impaired to some extent.

But while individual cases of broken connectivity due to out of date bogon filters are relatively straightforward to identify and fix, this feels somewhat sub-optimal. Rather than responding after the user has already experienced the problem, is it possible to be a little more proactive here, and attempt to identify such poorly maintained bogon filters before they break end-to-end connectivity for users? To restate the question: when an address block is allocated, how can the new address holder work out where there are bogon filters that is listing the address block, and how can they get the filters changed?

Some of the Regional Internet Registries, most notably the RIPE NCC, have been operating a "Debogon Project" where addresses from selected address blocks are advertised for a period, and users can test reachability into these address blocks via conventional use of *ping* and *traceroute*. Additional active tests can be carried out from the announced blocks to known test networks and devices, performing a test in the opposite direction. But the experience with such efforts is not entirely satisfying. The haphazard nature of the reachability tests that are associated with these efforts means that many unmaintained bogon filters remain undetected, and recipients of address allocations from these address block are still left with the issue of trying to work out if there are further bogon address filters and where they may exist.

Is there a way of conducting a more comprehensive effort to detect both the presence and location of bogon filters that is largely automated, and at the same time avoids the considerable expense of deploying monitoring systems across the entire Internet? Indeed, can we go a little further and try and perform such a detection exercise in a very inexpensive manner?

The approach described here is somewhat different to the conventional debogon efforts, in that it does not rely on the use of cooperating users, nor on the deployment of probes or other network edge devices. Instead, it uses online advertisements to enrol end users to undertake part of the task of bogon filter detection through loading a simple reachability test in their browsers. If you can enrol a large number of users spread over the entire Internet through presentation of this ad to a large and diverse sample of end users, then the test can be comprehensive. So lets look at how to do this and what results we've been able to achieve so far.

Setting up a Reachability Test

The approach used here is to employ the Internet at large to perform the reachability tests through the normal behaviour of users' web browsers.

The basic method is to load the client's browser with instructions to fetch a number of a 1 pixel image files. The instructions, in the form of a script, is obtained from a server operating from known to be unfiltered IP address. All the test image files, except the final image, are sourced from domain names that map to IP addresses drawn from the address blocks being tested. The final image is sourced from the same address block as that from which the experiment was delivered from in the first place, so it acts as an end marker to indicate that the user has not interrupted the browser's actions mid way through the tests. The script also marks these images load to be performed as a background activity, and not to be handled by the browser's display object manager, so the entire test is both fast and invisible to the user.

This test methodology can be achieved in a number of ways, including in Javascript or in Flash code that has been embedded into a web page. However with such an approach of embedding the reachability test into the web page, then the clients who undertake the reachability test are restricted to the set of clients who visit a web page that has the test included in the source of the page. As we were wanting to have the test run by a very large collection and diverse of clients from across the entire Internet, we were interested in a way to present these tests in a broader fashion than could be possible with a small number of appropriately equipped web sites.

The approach we took was to to use Google's Image Advertisements as the vehicle for presenting the reachability tests to end client browsers. The test script, coded in Flash, was embedded into the ad material. The script is executed by the client's browser whenever the ad is delivered to the client, and the script will execute irrespective of whether the ad is "clicked" or not (At this point I should admit that in order to keep the costs down for this experiment we did try and devise an advertisement that was so bland and uninteresting that few users would be motivated to click through the ad!)

The test script fetches a set of control instructions from the experiment controller, using the server's control IP address. This set of control instructions lists number of image URLs, which form the core of the experiment. An example of the control URL and the generated file is shown in Figure 1.

```
http://www.a.rqa.rand.apnic.net/measureipv6.cgi

rqaa http://t2.u2980881285.s1326250419.i1110.v1010.a.rqa.rand.apnic.net/
           1x1.png?t2.u2980881285.s1326250419.i1110.v1010.rqaa
rqab http://t2.u2980881285.s1326250419.i1110.v1010.b.rqa.rand.apnic.net/
           1x1.png?t2.u2980881285.s1326250419.i1110.v1010.rqab
rqac http://t2.u2980881285.s1326250419.i1110.v1010.c.rqa.rand.apnic.net/
           1x1.png?t2.u2980881285.s1326250419.i1110.v1010.rqac
results http://results.c.rqa.rand.apnic.net/1x1.png?t2.u2980881285.
           s1326250419.i1110.v1010&r=
```

Figure 1. Example of an Experiment Control File

In this example there are two address reachability tests being conducted (*rqaa* and *rqab*), and a control test (*rqac*). The parameters to the test include the identification of the test version(*v1010*), the time the test was conducted (s*1326250419,* which maps to Wed Jan 11 13:53:39 2012 in this case) and a unique identifier that allows us to recombine the individual experiments back into a single test set when performing analysis of the web server's log files (*u2980881285*). Loading this information into the domain name part of the URL also implies that each user is delivered a different test, which means that any in-line web cache will treat these web objects as unique and not use any previously cached material (and thereby bypass the reachability test).

The images are 1x1 pixel images, and are fetched asynchronously from page rendering, and additionally are not included in the displayed page. This is to minimize any additional delay on the page view from the testing, and largely takes place without any visible impact on the users browser. The same backgrounding mechanism is employed by Google Analytics and other page view collection methods.

In this experiment three tests use three wildcard domain names, namely **.a.rqa.rand.apnic.net*, **.b.rqa.rand.apnic.net*, and **.c.rqa.rand.apnic.net*. These domain names map to the following IP addresses, from within the associated network address blocks:

Target	Test Address	Address Prefix
A	110.76.136.1	110.76.136.0/22
B	103.246.136.1	103.246.136.0/22
C	203.133.248.12	203.133.248.0/24 (control network)

The server environment was configured such that these test address prefixes were being advertised as reachable from this single server, and the server itself was configured with secondary IP addresses that match the three test addresses.

The Apache web server of the system was configured with a wildcard virtual server host of the form **.rqa.rand.apnic.net*, so that all the images are served from the same server

The Apache web server logs were used for reachability analysis.

If a client successfully performs the entire test then the web logs will contain a sequence of records, an example of which is shown in Figure 2.

```
# load the test set
110.67.xxx.xxx - - [02/Jan/2012:00:01:38 +0000]
      "GET /crossdomain.xml HTTP/1.1" 200 726 "-"
110.67.xxx.xxx - - [02/Jan/2012:00:01:38 +0000]
      "GET /measureipv6.cgi?&hash=3103003790 HTTP/1.1" 200 140
      "http://pagead2.googlesyndication.com/pagead/imgad?id=CICAgICQr"

# test b - 103.246.136.1
110.67.xxx.xxx - - [02/Jan/2012:00:01:39 +0000]
      "GET /crossdomain.xml HTTP/1.1" 200 726 "-"
```

```
110.67.xxx.xxx - - [02/Jan/2012:00:01:39 +0000]
      "GET /1x1.png?t2.u940792609.s1325462498.i1110.v1010.rqab
      HTTP/1.1" 200 157
      "http://pagead2.googlesyndication.com/pagead/imgad?id=CICAgICQr"

#test c - 203.133.248.12
110.67.xxx.xxx - - [02/Jan/2012:00:01:40 +0000]
      "GET /crossdomain.xml HTTP/1.1" 200 726 "-"
110.67.xxx.xxx - - [02/Jan/2012:00:01:41 +0000]
      "GET /1x1.png?t2.u940792609.s1325462498.i1110.v1010.rqac
      HTTP/1.1" 200 157
      "http://pagead2.googlesyndication.com/pagead/imgad?id=CICAgICQr"

#test a - 110.76.136.1
110.67.xxx.xxx - - [02/Jan/2012:00:01:41 +0000]
      "GET /crossdomain.xml HTTP/1.1" 200 726 "-"
110.67.xxx.xxx - - [02/Jan/2012:00:01:41 +0000]
      "GET /1x1.png?t2.u940792609.s1325462498.i1110.v1010.rqaa
      HTTP/1.1" 200 157
      "http://pagead2.googlesyndication.com/pagead/imgad?id=CICAgICQr"

#result - result 203.133.248.12
110.67.xxx.xxx - - [02/Jan/2012:00:01:44 +0000]
      "GET /crossdomain.xml HTTP/1.1" 200 726 "-"
110.67.xxx.xxx - - [02/Jan/2012:00:01:41 +0000]
      "GET /1x1.png?t2.u940792609.s1325462498.i1110.v1010.rqaa.za=32.
      zb=35.zc=30
      HTTP/1.1" 200 157
      "http://pagead2.googlesyndication.com/pagead/imgad?id=CICAgICQr"
```

Figure 2 - The Web Server's Log of a Reachability Test

Because the ad itself is served from a remote site, all fetches from the control server are in fact two fetches, the first for the crossdomain policy file to determine if fetches from this site are permitted (*crossdomain.xml*), and the second being the data file itself.

In the example shown here the experiment generated 10 fetches. The first pair of fetches retrieved the list of URIs to fetch (*measurev6.cgi*). The next 3 pairs of fetches performed the reachability test itself. The final fetch was the result fetch, indicating that the test has run to completion.

What we are looking for in the server's web logs are origin AS's where there is a systematic failure to retrieve the first, second or both of the first two images, yet a consistent ability to fetch from the control address.

Reachability Test Results

This test was run over 22 days in December 2011. During that time 513,141 individual client IP addresses performed the reachability test. These client IP addresses were sourced from 9,665 originating Autonomous Systems.

In order to eliminate potential problems with individual tests not completing (due to the client browsing over to a different webpage before the test was complete, for example), we have filtered out all tests where the final result fetch is missing from the log. In order to increase our level of assurance that what were are seeing has a strong correlation to AS level filters we also filtered out all originating AS's where we had only 1 or 2 clients performing the test. This filtering left us 507,121 tests, spanning 4,839 originating AS's.

All the clients in 3 AS's were unable to reach both the **A** and **B** target (indicating some form of filter on both the 110.76.136.0/22 and 103.246.136.0/22 address blocks) These AS's were:

AS	#Tests	CC	AS Description
23771	26	CN	SXBCTV-AP SXBCTV ,Internet Service Provider
3828	33	CN	CHINANET-SCIDC-AS-AP HINANET SiChuan Telecom Data Center
45143	50	SG	SINGTELMOBILE-AS-AP SINGTEL MOBILE INTERNET SERVICE

There were no origin AS where more than 2 individual experiments were able to reach the **B** target, but unable to reach the **A** target.

There were 16 origin AS's were able to reach the **A** target, but unable to reach the **B** target (i.e., the prefix 103.246.136.0./22 appears to be filtered). These AS's were:

AS	#Tests	CC	AS Description
4835	10	CN	CHINANET-IDC-SN China Telecom (Group)
6983	9	US	ITCDELTA - ITC Deltacom
9051	17	LB	IncoNet Data Management sal
14868	55	BR	Companhia Paranaense de Energia - COPEL
19332	7	MX	Marcatel Com, S.A. de C.V.
23252	3	CA	IK - WTC Communications
25248	77	CZ	BLUETONE-AS Ceske Radiokomunikace a.s.
28142	4	BR	DIGITAL DESIGN - SERVICOS DE INFORMATICA LTDA
28378	4	MX	TV Rey de Occidente, S.A. de C.V.
28678	3	PL	KOSMAN-EDU-AS Technical University of Koszalin
39566	12	PL	TRUSTNET-PL-AS trustnet.pl / smarthost.pl hosting datacenter
41088	13	CZ	CZNSYS N-SYS s.r.o.
43153	9	PL	SFERANET-AS SferaNET Sp. z o.o.
44651	12	HU	COMUNIQUE Com.unique Telekommunikacios Szolgaltato Kft.
49289	11	PL	NITKA-NET ELPRO - Elektronika Profesjonalna Waldemar Nitka
197100	4	ES	Esystel Servicios Multimedia, SL

If one were to assume that this collection is a representative sample of reachability by origin AS, then of the total population of 39,908 AS's some 66 AS's could be assumed to be filtering 103.246.136.0/22, and 12 AS's could be assumed to be filtering both 110.76.136.0/22 and 103.246.136.0/22.

Further Questions

The experiment raises a number of questions, which we will briefly discuss here.

The first question is, can these experimental results be verified by other means?

A simple means of validation of the result is by using *ping*, with the source address option. We were able to validate a small number of addresses chosen from random from the client set, that where the end systems were responsive to *ping* packets sources from the "control" address of 203.133.248.12, they were unresponsive to *pings* when using the tested addresses as a source address. It is noted that not all addresses are responsive to *ping* probes, due to firewalls and filters, so this by no means a precise form of validation.

```
$ ping 194.126.xx.xx
PING 194.126.xx.xx (194.126.xx.xx): 56 data bytes
64 bytes from 194.126.xx.xx: icmp_seq=0 ttl=41 time=372.199 ms
64 bytes from 194.126.xx.xx: icmp_seq=1 ttl=41 time=372.317 ms
64 bytes from 194.126.xx.xx: icmp_seq=2 ttl=41 time=372.270 ms
```

```
^C
--- 194.126.xx.xx ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 372.199/372.262/372.317/0.049 ms


$ ping -S 103.246.136.1 194.126.xx.xx
PING 194.126.xx.xx (194.126.xx.xx) from 103.246.136.1: 56 data bytes
^C
--- 194.126.xx.xx ping statistics ---
4 packets transmitted, 0 packets received, 100.0% packet loss
```

Figure 3. Example of ping probes

The second question relates to the question of where the client may be positioned in the inter-AS routing space. In this case we are interested to establish whether there is a common AS path segment for a number of AS's that show evidence of a bogon filter, or whether there is no common path segment.

```
    AS     AS Path from Server to Client
           Common Prefix      AS Transit Path

  23771   4608 1221 4637    4134   4837 18118 23771
  38283   4608 1221 4637    4134  38283
  45143   4608 1221 4637    7473   9506 45143
   4835   4608 1221 4637    4134   4835
   6983   4608 1221 4637    2828   6983
   9051   4608 1221 4637    3356  42020  9051
  14868   4608 1221 4637    3549  14868
  19332   4608 1221 4637     174  19332
  23252   4608 1221 4637    3356  23252
  25248   4608 1221 4637    3549   5588 25248
  28142   4608 1221 4637    3549  14868 28142
  28378   4608 1221 4637    3356  19332 28378
  28678   4608 1221 4637    3356   8501 28678
  39566   4608 1221 4637    6453  24724 15694 39566
  41088   4608 1221 4637     174  41088
  43153   4608 1221 4637    3320  20804 43153
  44651   4608 1221 4637    3356   8928 47169 44651
  49289   4608 1221 4637    3549   5588  8246 49289
 197100   4608 1221 4637     174 197100
```

There is no clear evidence that there is the systematic deployment of packet filters in the AS transit paths to these origin AS's.

Conclusions

There is no obvious evidence that either of the tested prefixes (110.76.136.0/22 and 103.246.136.0/22) are the subject of widespread bogon filtering in the Internet in December 2011.

There was some observed evidence of filtering of these prefixes, and of a total of 507,450 tests, some 329 tests, or 0.065% of the test population, showed some evidence of filtering of these prefixes.

There is also some evidence that the prefix 103.246.136.0/22 has a slightly greater level of filtering than the 110.76.136.0/22 prefix). This could be due to the fact that 110.0.0.0/8 was allocated to APNIC in November 2008, while 103.0.0.0/8 was allocated to APNIC in February 2011.

This evidence of filtering may the result of irregularly maintained bogon filters where the filter has not been updated to reflect the current IANA address allocation status.

The experimental technique of using advertisement placement to perform a large number of client-based reachability tests appears to be a relatively effective test methodology. It does not require investment in deployment of a large set of test units across the Internet, and because it uses a client-initiated test methodology it bypasses the limitations of NAT traversal which is associated with server-side initiated reachability tests. The ad placement approach also circumvents the common forms of packet filtering of ICMP *pings*, as all the test traffic sits within TCP port 80 transactions.

There is further work to be done to consider how representative this set of tested clients is in terms of the entire population of Internet clients, but as an initial exercise in showing how advertisement placement can be used as a useful mechanism to support some forms of client-based network measurement, this has been a highly informative and effective exercise.

Acknowledgements

This work was undertaken with my colleagues at APNIC; George Michealson, who configured the server for this experiment and managed the Google Ad account and Byron Ellacot who provided the Flash code for the test.

Disclaimer

About the Author

Geoff Huston B.Sc., M.Sc., has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for the initial build of the Internet within the Australian academic and research sector. He is author of a number of Internet-related books, and has been active in the Internet Engineering Task Force for many years.

www.potaroo.net