

# A routing blackhole management tool

Ingvar Mattsson  
<ingvar@hexapodia.net>

June 5, 2009

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Thanks . . . . .	2
<b>2</b>	<b>Routing blackholes</b>	<b>2</b>
2.1	Routing blackholes in more detail . . . . .	2
2.2	Possible usage scenarios . . . . .	3
<b>3</b>	<b>The administrative tool</b>	<b>3</b>
3.1	Mid-layer . . . . .	4
3.2	Back-ends . . . . .	4
3.3	Front-ends . . . . .	4
3.4	Configuration file . . . . .	5
<b>4</b>	<b>Network layer</b>	<b>5</b>
4.1	Test network . . . . .	5
4.2	The blackhole router . . . . .	6
4.3	Other routers . . . . .	6

## 1 Introduction

This paper will present a routing blackhole administration tool and arguments for using the tool as opposed to more normal routing blackholing. The tool, aptly named “Blackhole Administrative Tool” (or “BAT” for short), presents an easy way to manage blackhole routing, without any blackholed route being forgotten about.

### 1.1 Thanks

Leigh Honeywell - for general encouragement  
Jim Prewett - for general encouragement  
Mark Lowes - for asking if it was possible in the first place  
Jacqueline Docherty - for reading early proofs

## 2 Routing blackholes

### 2.1 Routing blackholes in more detail

A routing blackhole is a route into a loopback interface (on a host) or a null interface (on a router), taking precedence over normal routing for a destination network N. This causes packets bound for N to end up in the blackhole instead of the normal destination.

Normally, a routing blackhole is seen as a disadvantage, as it effectively breaks the network, but in some circumstances this is exactly what you want. If, for example, a host on your network is busy sending spam, blackholing the machine’s IP address means that it cannot establish any TCP sessions as the returning SYN-ACKs will end up in the blackhole instead of at the machine, effectively stopping it from sending any more data.

With at least some border routers (I know specifically that it is possible with Cisco routers)<sup>1</sup>, a special form of reverse-path verification can also cause packets from outside your network to be dropped at the border. This allows slightly more efficient blackhole routing of external netblocks.

The model used in the example configurations are centred around using a dedicated blackhole router that only deal with blackhole routes and distribute them to the network using iBGP. Next-hop information and BGP communities to control the blackholing are all generated on the blackhole router and tagged static routes are used to distribute blackhole information from the administration tool to the blackhole router.

## 2.2 Possible usage scenarios

Routing blackholes are useful in any scenario when there's a wish to cut off traffic to a netblock (I'll be using "netblock" or "route" to mean one or more consecutive IP addresses), for any reason. This has multiple uses, ranging from dealing with abuse to cutting off service for non-paying customers.

## 3 The administrative tool

The blackhole administrative tool comes in several layers, with one mid-layer (deals with operator identification and the database layer), multiple back-ends and multiple front-ends, for increased flexibility.

The existing front-ends are a CLI tool, a web interface and a "reaper". The reaper is responsible for clearing out blackhole routes that have "expired" and the other front-ends are primarily intended to create new blackhole routes, but can also be used to list existing blackhole routes or delete them before they've expired.

The administrative information is kept in a MySQL database<sup>2</sup>. The database schema consists of one table for the blackhole information and one table for operator information. This provides a slight layer of abstraction between the blackholing tool and the actual blackholing back-end.

The database maintained by the administrative tool keeps a start and end date, the status of any specific blackhole, who created the blackhole and a reason for the blackholing, until such a time that the blackhole database is cleared out. This means that there's a history of all blackholed prefixes in the database, for audit purposes. However, as no hard security is in place, the audit trail is more informative than binding. It still tends to be better than what is available without the tool.

One peculiarity with the database schema used is that the end date column has a dual use. For an active blackhole, the end date is when we want that prefix to stop being blackholed, but for a non-active blackhole, it is instead when the blackhole was deactivated.

---

<sup>1</sup>by using the `ip verify unicast source reachable-via any` configuration, paired with routes into Null interfaces

<sup>2</sup>it would probably not be hard to re-write the database layer to use another database, as the tool mostly use SELECT and INSERT

### 3.1 Mid-layer

The mid-layer (`bhlib.py`) provides API calls for listing operator information, mapping an operator ID to an operator name, validating an operator and creating or deleting blackholes. The routes as specified to the API should be in CIDR notation and the mid-layer will normalise any netblock sent through it, before passing it on to the back-end (that does the actual work). The mid-layer is also responsible for loading the configuration file that specifies what back-end to use, authentication information and the like (the configuration is held in `bhconfig.py`).

To create a new blackhole route boils down to calling `bhlib.addpfx` with the relevant arguments. As an example, if we wished to blackhole `192.168.17.16/30` for one day, we could call it as follows.

```
bhlib.addpfx('192.168.17.16/30', 666, 'operator', 'password',
            86400, 'Blackholed as an example')
```

The tag used (666) is an example, but in the router configuration used in the test environment under development, this is used as the tag that means “make sure the route is blackholed on all routers in the network”. Note that only some back-ends actually honour the tag (specifically, the tag is not used by the unix backend).

### 3.2 Back-ends

The back-ends provide the code that actually installs and removes blackhole routes. The three existing back-ends are `nullbh.py`, `ciscobh.py` and `unixbh.py`, but writing new back-ends should not be that complicated. Any back-end needs to provide two API functions, plus setting the variable `bh_type` to a string indicating what back-end is being used<sup>3</sup>.

The two functions that must be provided are `add_blackhole(prefix, tag)` and `remove_blackhole(prefix)`. Remember that authentication and authorisation to create a blackhole is provided by the mid-layer and that no extra privileges are required (in the provided code) to remove a blackhole.

The Cisco back-end uses another module to actually communicate with the router and using that as a model, it should not be hard to write a module communicating with a router from another vendor (using it to communicate with a Zebra or Quagga router instance should border on trivial).

The null back-end was written to make it easier to test the mid-layer and front-ends, as it doesn't generate any blackhole routes, while allowing the mid-layer to exercise the database. The null back-end is thus not suitable for production use but may be worth using on a training instance or directly after installation.

### 3.3 Front-ends

At the moment, there are three front-ends for this tool, one command-line (`blackhole.py`) and one web-based (`blackholecgi.py`) interface for adding, listing and removing blackholes and one for the express purpose of removing

---

<sup>3</sup>This is currently not used anywhere, but if this is being relied on, it would make it possible to display what back-end is in use in the front-end tools.

blackholes that have expired (`reaper.py`). The reaper script should be run from cron with sufficient frequency to ensure that no blackhole stays active much longer than necessary, the exact frequency probably depends on the organisation deploying this tool, but a recommendation is to run it at least hourly.

None of the existing front-ends have any strict requirement on authentication for deletion of a route and only basic authentication for adding a blackhole route. It is therefore recommended that the blackhole management interface is deployed in such a fashion that it is hard to reach the management interface. Admittedly, both the command-line and the web interface require mapping a username to a password, but do not enforce HTTPS or strong session management, so any use of the web management interface across open networks puts the authentication credentials used at high risk. However, as the back-end does not inter-operate with LDAP, Kerberos, NIS or any other distributed authentication scheme, other systems are only “at risk” to the extent that operators share passwords between services.

There is, at the moment, no obvious way from either front-end, to change an operator’s password or create a new operator, there are however API functions in the mid-layer to do this and people deploying the Blackhole Administration Tool in production are encouraged to use these instead of raw SQL queries.

### 3.4 Configuration file

The BAT configuration is held in `bhconfig.py` and should be mostly self-explanatory. This file also contains the mapping between numeric tags and explanations and if any changes are made on the back-end, it is highly recommended to edit the tagmap mappings in `bhconfig.py`, as these are displayed in the web interface.

## 4 Network layer

### 4.1 Test network

The test network used under the development is depicted in 4.1. The four routers all have different roles in the network. The blackhole router (BHR) originates the blackhole routes and is the network device that BAT adds and removes static routes from.

The “packet capture router” (PCR) has a LAN interface configured on which a packet-capture machine is (notionally) installed. The intent here is to allow in-depth study of behaviour, the packet capture machine could, for example, be used to impersonate either a local or remote machine to capture examples of worms or spam.

The transit router (TR) provides a route out of the AS. While only one transit router is used in the example, a real network would probably have multiple paths in and out of the AS.

The out-of-AS peer router (PR) acts as a proxy for the rest of the Internet, originating a few select routes.

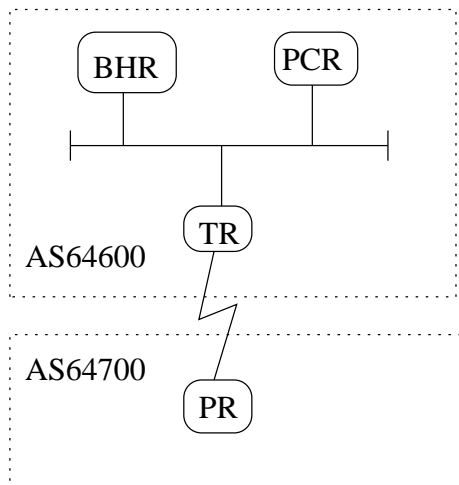


Table 1: 4.1 - Blackhole Admin Tool dev network, BHR - blackhole router, PCR - packet capture router, TR - transit router, PR - out-of-AS peer router

## 4.2 The blackhole router

When a given netblock is installed by the admin tool, it is always installed as a route to Null0 and tagged with the relevant tag from the tool. At that point, it's re-distributed from static routing to BGP and processed through a route-map that sets next-hop IP address and a few BGP communities.

In the development environment, I chose 10.0.0.0/8 as the network used by the "local" AS. I then chose 10.255.255.255 to be the AS-wide blackhole address and 10.255.255.254 to be the IP address of the blackhole router-internal sink. I then arbitrarily decided that the packet-capture machine was 10.200.0.3. As can be seen in fig. 4.2, these IP addresses show up in the route-map BLACKHOLE-STATIC and the intention is that these are then distributed into BGP through this route-map. Modifying this route-map to correspond to local circumstances should not be hard.

## 4.3 Other routers

On other routers in the network, there are two pieces of configuration necessary. One is to ensure that the router is speaking BGP, the other is to ensure routing for the local blackhole sink. In the development network, the address chosen as AS-wide blackhole sink is 10.255.255.253 and the only configuration needed on the routers that aren't the blackhole router can be seen in fig. 4.3.

```
ip classless ip route 10.255.255.253 255.255.255.255 Null0
ip route 10.255.255.254 255.255.255.255 Null0
ip bgp-community new-format route-map BLACKHOLE-STATIC permit 10
  match tag 666
  set ip next-hop 10.255.255.253
  set origin igp
  set community 64600:666 no-export
route-map BLACKHOLE-STATIC permit 20
  match tag 667
  set ip next-hop 10.255.255.254
  set origin igp
  set community 64600:666 no-export
route-map BLACKHOLE-STATIC permit 30
  match tag 668
  set ip next-hop 10.200.0.3
  set origin igp
  set community 64600:666
```

Table 2: 4.2 - Blackhole router configuration

```
ip route 10.255.255.253 255.255.255.255 Null0
```

Table 3: 4.3 - Configuration needed on the remaining routers