

originally had, so neighbors that don't filter your prefixes or limit the number of prefixes they accept from you (filtering on AS path doesn't help here) will start to send you the traffic for nearly all the affected destinations. As a result, your AS becomes a huge black hole that sucks traffic in but doesn't let anything escape. This is about the worst thing that can happen to the global interdomain routing system.

These risks are far from academic: it has happened several times in the past. A study done at the University of Washington shows that nearly 75% of all new announcements that show up in the global routing table are the result of some kind of mistake or problem. One of the reasons for this is the practice of first removing an access list or set of route maps with *no access-list ...* or *no route-map ...* before entering a new version of the filter. The few moments before the new access list or route maps become active are enough for the router to start announcing random routes in the BGP table. But even changing filters the "proper way" is no guarantee against unwanted routers slipping through. Just recently, I replaced one filter with an identical one under another name. I didn't remove anything first, so there was a valid filter configured at all times. This still triggered the maximum prefix limit on the neighboring router, so routers that weren't allowed by either the old or the new filter must have slipped through during the change. You should never depend on a single filter.

The "Understanding BGP Misconfiguration" paper can be found at <http://www.cs.washington.edu/homes/ratul/bgp/index.html>.

Redistributing static and connected routes into BGP

Increasing numbers of internal routes mean OSPF has to do more and more work. For really big ISP networks, the number of static routes redistributed into OSPF may become a burden to the protocol. To avoid this, some ISPs don't redistribute these routes in OSPF (or their IGP of choice), but rather in BGP. Because BGP doesn't require exact synchronization between all routers within the AS unlike link state protocols such as OSPF, it can tolerate large numbers of internal routes much better. Also, when there is a topology change in the internal network, there are no changes in the BGP or routing tables as long as the router announcing the subnet remains reachable. Unlike an IGP, BGP doesn't concern itself with how, exactly, the route is reachable: it's sufficient that the router that is listed as the next hop is somehow reachable, even if this requires several levels of recursion. If 7,000 iBGP customer routes point to 500 iBGP connected routes that point to 80 OSPF routes for loop-back addresses, OSPF has a lot less to do when a link goes down than if all those 7,000 customer routes had been present in OSPF. The router still has to recompute the routing table and route caches to adjust for the new IGP path towards the BGP next hop. However, this is done after OSPF has converged, so the time during which the network as a whole is in an inconsistent state (because OSPF is reconverging) is much shorter. Example 10-10 shows how to redistribute static and connected routes into BGP.

Example 10-10. Redistributing static and connected routes into BGP

```

!
router ospf 1
 network 192.0.2.64 0.0.0.63 area 0
 network 192.0.2.128 0.0.0.31 area 0
!
router bgp 60055
 no synchronization
 redistribute connected
 redistribute static
 neighbor 192.0.2.147 remote-as 60055
 neighbor 192.0.2.147 update-source Loopback0
 neighbor 192.0.2.148 remote-as 60055
 neighbor 192.0.2.148 update-source Loopback0
 neighbor 192.0.254.17 remote-as 40077
 neighbor 192.0.254.17 prefix-list no-internal out
 no auto-summary
!
ip route 192.0.2.0 255.255.255.0 Null0
!
ip prefix-list no-internal seq 5 permit 199.208.0.0/16
ip prefix-list no-internal seq 10 permit 192.0.2.0/24
!

```

Most of these configuration commands will be familiar by now. The address ranges 192.0.2.64/26 and 192.0.2.128/27 are put into OSPF area 0 because these are the address ranges for internal router-to-router links. Redistribution of static and connected routes into BGP is enabled with the *redistribute connected* and *redistribute static* commands. The *no auto-summary* command makes sure subnets are redistributed “as is” and aren’t summarized into classful routes in BGP. For instance, the route 192.0.2.64/26 would show up as 192.0.2.0/24 with *auto-summary* enabled, as is the default.

When redistributing static and connected routes into BGP, it’s more important than ever to carefully filter announcements toward external ASes: if you don’t, your transit ISPs, peers, and customers will receive all your internal routes. These routes are of no interest to them, use up resources, and obscure what you really want to announce, so troubleshooting gets harder than it needs to be. In this case, only 199.208.0.0/16 and 192.0.2.0/24 are allowed. Note that there are no *network* statements for these networks in the BGP configuration: the regular static routes to the Null0 interfaces for these prefixes (that normally serve to activate the *network* statements) are now directly redistributed into BGP.

This is how the 199.208.0.0/16 network, redistributed into BGP by another router in the local network, shows up in the routing table:

```

BR1#show ip route 199.208.0.0
Routing entry for 199.208.0.0/16
  Known via "bgp 60055", distance 200, metric 0, type internal
  Last update from 192.0.2.5 00:04:06 ago

```

```
Routing Descriptor Blocks:
* 192.0.2.5, from 192.0.2.147, 00:04:06 ago
  Route metric is 0, traffic share count is 1
  AS Hops 0
```

And in the BGP table:

```
BR1#show ip bgp 199.208.0.0/16
BGP routing table entry for 199.208.0.0/16, version 341
Paths: (1 available, best #1)
  Advertised to non peer-group peers:
  192.0.254.17
  Local
    192.0.2.5 (metric 128) from 192.0.2.5 (192.0.2.147)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
```

The next hop address for 199.208.0.0/16 is 192.0.2.5, so to forward a packet, we have to go back to the routing table and look up this address:

```
BR1#show ip route 192.0.2.5
Routing entry for 192.0.2.4/30
  Known via "bgp 60055", distance 200, metric 0, type internal
  Last update from 192.0.2.147 00:06:40 ago
  Routing Descriptor Blocks:
  * 192.0.2.147, from 192.0.2.147, 00:06:40 ago
    Route metric is 0, traffic share count is 1
    AS Hops 0
```

But the next hop 192.0.2.147 still isn't a directly connected address, so we have continue looking up routes:

```
BR1#sh ip route 192.0.2.147
Routing entry for 192.0.2.147/32
  Known via "ospf 1", distance 110, metric 65, type intra area
  Redistributing via ospf 1
  Last update from 192.0.2.1 on Serial2, 00:06:57 ago
  Routing Descriptor Blocks:
  * 192.0.2.129, from 192.0.2.147, 00:06:57 ago, via Serial2
    Route metric is 65, traffic share count is 1
```

Fortunately, recursively looking up routes in the main routing table isn't something that has to be done for each individual packet. The Cisco Express Forwarding (CEF) table immediately leads to the right information:

```
BR1#show ip cef 199.208.0.0
199.208.0.0/16, version 764
0 packets, 0 bytes
  via 192.0.2.5, 0 dependencies, recursive
    next hop 192.0.2.129, Serial2 via 192.0.2.4/30
    valid adjacency
```

Now it's time to see if all this additional complexity pays off, by shutting down the Serial2 interface that is used for this route. The effect on the routing table is:

```
BR1#show ip route 199.208.0.0
Routing entry for 199.208.0.0/16
```

```
Known via "bgp 60055", distance 200, metric 0, type internal
Last update from 192.0.2.5 00:04:46 ago
Routing Descriptor Blocks:
* 192.0.2.5, from 192.0.2.147, 00:04:46 ago
  Route metric is 0, traffic share count is 1
  AS Hops 0
```

In other words, no change whatsoever. But traffic is rerouted all the same:

```
BR1#show ip cef 199.208.0.0
199.208.0.0/16, version 764
0 packets, 0 bytes
via 192.0.2.5, 0 dependencies, recursive
next hop 192.0.2.133, Serial3 via 192.0.2.4/30
valid adjacency
```

This is exactly the intended effect of this configuration.

Redistributing static and connected routes into BGP has one major drawback: there must be stringent outbound filtering on all eBGP sessions, and these filters must be modified every time a new network is announced over BGP. The use of communities offers a way around this. Rather than having outbound filters that separate routes that should be visible externally from internal routes, external routes are tagged with a community at the point of origin:

```
!
router bgp 60055
 network 192.0.2.0 mask 255.255.255.0 route-map setexternal
!
route-map setexternal permit 10
 set community 60055:1
!
Outbound filtering on EBGp sessions is then limited to permitting routes
with community 60055:1:
!
router bgp 60055
 neighbor 192.0.254.17 remote-as 40077
 neighbor 192.0.254.17 route-map external-only out
!
ip community-list 1 permit 60055:1
!
```

The filters for EBGp sessions never have to change in this scenario; whenever a new prefix is announced, only the router sourcing the route must be reconfigured. This minimizes the potential for mistakes.

Traffic Engineering in the Internal Network

Chapter 6 was all about traffic engineering for traffic to and from external networks. If you have a network spanning a significant geographic area (so that leased-line costs are considerable), you may need to do traffic engineering in the internal network. In one way, this is easier than interdomain traffic engineering: you have full