RTBH (University of California Computing Services Conference 2016)

Forest Monsen, Senior Information Security Analyst, University of California, Santa Cruz

1) What it is

a) RTBH stands for "remotely triggered black hole" routing or filtering.

2) Why you should do it

a) Reduce pointless noise in your environment, logs, and tools, from 24/7 script kiddies and brute forcers.
b) Discourage easy reconnaissance. Recon is the first step in the kill chain. The earlier you stop an attacker, the better.
c) Spend your time and attention wisely. You should spend your attention on attacks that matter. By making it slightly harder, you're dealing with a slightly different class of attacker.

3) Our story

a) When we started, attackers and scanners had unfettered access. Aside from some pockets of our address space that had been firewalled.
b) So folks from anywhere in the world could scan every system on our campus, search for juicy vulnerabilities at their leisure, and exploit them.
c) We did have netflow data, so we could see occasional blips in border traffic. Then we installed Bro and started to look more deeply. At that point we still were just watching the scans and brute force attacks stream on by.
d) We think of scanning as someone going around and rattling every single door and window in a neighborhood, to see which are loose. Someone could very well stop at a scan, and not go any further than that. But when attacking, a scan of some sort is usually the first step.
e) We also saw a lot of brute force attempts. Online brute force attacks involve iterating through passwords or usernames, making many, many attempts to get in, in any hole you can find.
f) We decided we needed some kind of a bouncer, to stop malicious folks.

4) How ours works, at a high level

a) Bro at the border watches traffic and characterizes it. Our SIEM retrieves logs and netflow, and looks for bad behavior there.
b) Block requests are sent from Bro and the SIEM to a queue (persistent storage, fronted by Django and its REST framework).
c) Blocking agents periodically check in with the queue, implement blocks as requested, and update the queue with what they've done.

5) More detail

a) Blocks by default expire in one hour. The goal is to interrupt the least skilled attackers, and make ourselves a less desirable target. We want to be dealing with folks who are more skilled.
b) Blocks are auto-scaled; if you come back repeatedly, you are blocked for more time.
c) A modular architecture: additional sources can request blocks; additional block implementers can be added. Currently two block requestors, and one implementer.
d) Django offers an admin interface out-of-the-box. The BHR site code allows for manual management of blocks through a simple Web interface.
e) The implementer is written in Python. Checks in with the Django queue. Implements blocks by talking to Quagga. Quagga advertises BGP routes to the borders. The border routers are configured to talk to Quagga as an iBGP peer.
f) On the border routers, the next hop for individual IPs is set to an IP address that dumps what it receives into the Null0 interface. See the Cisco whitepaper.

6) Benefits we've seen

a) We blocked, on average, 26k unique IP addresses per day, now down to 14k.
b) Over 5M total blocks after 7 months.
c) After correlating with public and private lists of "known bad" malicious addresses, we saw a 94% reduction in alerts regarding communications with these known bad malicious IPs.
d) So we did get a "bouncer." S/he works 24/7.

7) What you'll actually need, to implement the solution

a) Someone who understands BGP
b) Someone who can script against a REST API, understands how to install things like Django and can write some Python
c) Border routers configured for BGP
d) The Cisco whitepaper
e) Bro and its BHR integration
f) Quagga
g) NCSA's BHR site code
h) NCSA's BHR client code