Back from the Dead: Simple Bash for complex DdoS

Mar 09, 2011 By Greg Bledsoe (/users/greg-bledsoe) in

If you work for a company with an online presence long enough, you'll deal with it eventually. Someone, out of malice, boredom, pathology, or some combination of all three, will target your company's online presence and resources for attack. If you are lucky, it will be a run of the mill Denial of Service (DoS) attack from a single or limited range of IP addresses that can be easily blocked at your outermost point, and the responsible parties will lack the necessary expertise to overcome this relatively simple countermeasure. Your usual script kiddie attack against a site with competent network and server administration is fairly short. If you are unlucky, you'll experience something worse: A small percentage of attacks is from a higher caliber of black hat, and while more difficult to deal with, the individual generally bores easily and moves on.

If you are very, very unlucky, someone highly skilled and just as determined will decide to have some fun with you. If this person decides they want to crack their way into your servers and explore your environment, eventually they will get in and their isn't too much you can do about it. As long as they don't do anything too obvious, like launch a huge dictionary crack attack against other sites from your servers, you may never know, even if you are pretty good and attentive. And if they decide they want to knock you off of the Internet, then down you go.

I had the misfortune to be on the receiving end of such an attack at a previous employer who shall remain nameless (but it was in 2007 and my linkedin is public: **http://www.linkedin.com/in/gregbledsoe** (http://www.linkedin.com/in/gregbledsoe)). Someone didn't seem to like us very much and decided to erase us from online existence. At first it was a standard DoS syn-flood that any script-kiddie could launch, a minor annoyance at best, easily mitigated by blocking the source IP at the point of Ingress. Then it got interesting.

The attacker adapted by engaging a substantial bot-net and it became a distributed denial of service (DdoS) attack. The targeted server address was down briefly until we engaged our carriers to block the inbound attack further out. Still, at that point, the crisis is over, right? Normally, yes. In this case? Not even close.

The attacker adapted the attack *again*, this time seeming to rotate through connections from real bot-net systems and also sending oodles of fake connection requests from random spoofed IP addresses. All told, the number of incoming connection requests was close to a million at a time. This took us down hard. Panic ensued, and after some quick brainstorming a number of mitigation techniques were attempted, all to no avail. The connections went through our firewall, through our load balancer, and hit one of three back-end systems, all of which were overwhelmed dealing with the load imposed by the attack. We tried using rate-limiting on the firewall, and while I'm not sure exactly what they implemented, this took everything behind the firewall down, not just the the targeted URL/server address. The rate limiting statements were taken back out of the configuration but everything stayed down. We discovered that the firewall equipment was out of memory, creating table space to keep track of all the connection attempts. It couldn't tell the difference between spoofed, real, and legitimate tcp SYN connection requests, so it tracked them all and let them through. Apparently the particular equipment we had did not allow more granular rate limiting. Options were discussed, including rejiggering our

DNS to send all our traffic through a (very expensive) company that promised to scrub the attack before it reached us. I was skeptical of this idea.

Being the Unix Guy, my domain was the backend servers and to a lesser extent, the load balancer. After watching the output of netstats, lsof -ni's, and tcpdumps for a while, I knew how to defeat this attack. I spent about 10 minutes crafting my counter measure and deployed it on all three back end servers and within seconds our environment was alive again. The red of nagios alarms cleared within a few minutes and our phones stopped ringing. Our total downtime was about an hour.

The thing that I noticed that made this counter measure work was that there was a clear threshold between the number of connections opened by legitimate users, and the high number of connections from both the real and spoofed IPs that were part of the attack. By identifying them on the back-end servers and sending TCP resets (with the RST flag on) back on all those connection requests over the threshold, we could clear out the connection information on the server, the load balancer, and the firewall and free up the memory that had been used to store that entry in the table - clear out enough of them quickly enough, faster than new attack IPs were coming in, and life became good again.

Here is the (very simple) script I ran on all three servers.

```
#! /bin/bash
while [ 1 ] ;
 do
 for ip in `lsof -ni | grep httpd | grep -iv listen | awk '{print $8
}' | cut -d : -f 2 | sort | uniq | sed s/"http->"//` ;
 # the line above gets the list of all connections and connection
attempts, and produces a list of uniq IPs
 # and iterates through the list
  do
    noconns=`lsof -ni | grep $ip | wc -l`;
    # This finds how many connections there are from this particular IP address
    echo $ip : $noconns ;
    if [ "$noconns" -gt "10" ] ;
    # if there are more than 10 connections established or connecting
from this IP
    then
      # echo More;
      # echo `date` "$ip has $noconns connections.  Total connections
to prod spider:  `lsof -ni | grep httpd | grep -iv listen | wc -l`" >>
/var/log/Ddos/Ddos.log
      # to keep track of the IPs uncomment the above two lines and
make sure you can write to the appropriate place
      iptables -I INPUT -s $ip -p tcp -j REJECT --reject-with tcp-reset
      # for these connections, add an iptables statement to send
resets on any packets recieved
    else
        # echo Less;
    fi;
  done
sleep 60
done
```

Our attacker made a number of attempts to adapt to this solution, trying for instance to have sections of the bot-nets start at some IP, like 1.1.1.1, and send one connection apiece rotating through IPs as quickly as possible to avoid tripping the threshold, but couldn't rotate quickly enough to wreak the same level of havoc as before. This script proved very robust against the rest of his attacks. Some fine-tuning was done, for instance to remove lines after they aged a particular amount, but the essense of the script remained the same.

What I really liked about this solution was the simplicity. I have found that the best solutions are usually the simplest. If you really understand the underlying technology and protocols, then you can often see right through to what underlies a problem, and avoid adding layer after layer of expense and complexity (and corresponding break points) to your environment.

I'm more than willing to release this under the GPL v2. If anyone is interested in incorporating this snippit or concepts into a larger solution for distribution let me know via the email address below.

_____

--
I was cloud before cloud was cool. Not in the sense of being an amorphous collection of loosely related molecules with indeterminate borders -- or maybe I am.
Holla @geek_king, http://twitter.com/geek_king (http://twitter.com/geek_king)

Comments

Comment viewing options
Threaded list - expanded ▼ | Date - newest first ▼ | 50 comments per page ▼ | Save settings
Select your preferred way to display the comments and click "Save settings" to activate your changes.

: instead of [1] (/content/back-dead-simple-bash-complex-ddos#comment-364431)
Submitted by Anonymous (not verified) on Wed, 04/20/2011 - 06:57.

You could use

while :;

instead of

while [1];

I really believe the (/content/back-dead-simple-bash-complex-ddos#comment-363913)
Submitted by Panagiotis Papadomitsos (http://dnhost.gr) (not verified) on Thu, 03/31/2011 - 03:49.

I really believe the important lesson here is that rejecting packets instead of dropping them can help the surrounding network get a hint of what's going on and mitigate the situation, even though dropping the packets may superficially seem more effective (because it does not create any more traffic on an already heavily burdened network, REJECT does).

However, one must pay attention to very large DDoS attacks, where this simple method can fill up the iptables table with lots and lots of rules, adversely affecting CPU and memory usage to the point where the mitigation itself becomes an autoimmune disease. I have yet to see such a case on physical servers whereas OpenVZ containers can easily die because of this, courtesy of the numiptent limit that UBC-based containers heed to.

Another simple script (/content/back-dead-simple-bash-complex-ddos#comment-363396)
Submitted by John T Sharpe (not verified) on Wed, 03/16/2011 - 13:16.

Very good idea.

Here is something similar that uses netstat to count connections, has configurable white list, and configurable number of connections.

http://deflate.medialayer.com/ (http://deflate.medialayer.com/)

Good ideas (/content/back-dead-simple-bash-complex-ddos#comment-363399)
Submitted by Greg Bledsoe (/users/greg-bledsoe) on Wed, 03/16/2011 - 16:29.

Wow. Remarkably similar. Any idea when this was written? (I really want mine to have been first! Come on 2008 or later!)

(/users/greg-bledsoe)

Reality is most good ideas occur to more than one person at different times... like the debate on who invented the lightbulb first, or calculus...

Bottom line is it works, simple, effective, fairly light-weight. I'll have to take a closer look at deflate, see if I can contribute at all... looks pretty complete though.

--
I was cloud before cloud was cool. Not in the sense of being an amorphous collection of loosely related molecules with indeterminate borders -- or maybe I am. Holla @geek_king, http://twitter.com/geek_king (http://twitter.com/geek_king)

Different approaches to the same problem (/content/back-dead-simple-bash-complex-ddos#comment-363406)
Submitted by John Sharpe (not verified) on Thu, 03/17/2011 - 00:01.

Greg, I can see an advantages to your approach also. Sometimes you would not want to go through the setup and the extra complexity when in an emergency situation. It's good and varied tools that help us be efficient at our jobs.

The only real difference (/content/back-dead-simple-bash-complex-ddos#comment-363446)
Submitted by Greg Bledsoe (/users/greg-bledsoe) on Thu, 03/17/2011 - 21:14.


(/users/greg-bledsoe)

The only real difference is that I switched to --reject-with-tcp-reset while he uses --DROP. --DROP leaves the record and memory usage in place on stateful network gear for the connections - I would say that is a slight "slight" advantage to my solution in some circumstances, but easy to change in Ddos.sh.

--
I was cloud before cloud was cool. Not in the sense of being an amorphous collection of loosely related molecules with indeterminate borders -- or maybe I am. Holla @geek_king, http://twitter.com/geek_king (http://twitter.com/geek_king)

He did it earlier... :) (/content/back-dead-simple-bash-complex-ddos#comment-363403)
Submitted by ztank (not verified) on Wed, 03/16/2011 - 18:30.

This is reported the ddos.sh file:

```
echo "DDoS-Deflate version 0.6"
echo "Copyright (C) 2005, Zaf "
```

Ciao.

Well darn (/content/back-dead-simple-bash-complex-ddos#comment-363404)
Submitted by Greg Bledsoe (/users/greg-bledsoe) on Wed, 03/16/2011 - 18:44.


(/users/greg-bledsoe)

I didn't see that in my first cursory look over it, since it isn't at the top of the file.

Then I guess the kudo's are yours Zak. You did it first. Darn you. :-D

--
I was cloud before cloud was cool. Not in the sense of being an amorphous collection of loosely related molecules with indeterminate borders -- or maybe I am. Holla @geek_king, http://twitter.com/geek_king (http://twitter.com/geek_king)

Very Clever, Understanding What Endpoints are being attacked (/content/back-dead-simple-bash-complex-ddos#comment-363370)
Submitted by Bruce Cramer (not verified) on Sun, 03/13/2011 - 08:07.

Hi Greg:

I must say it's rare to find the correct solution to defend against Web DdoS Attacks.

Your Solution is Simple, Effective and Very Clever, KUDO's !!!

To defend against Web DdoS in a Panic/Crisis Mode, Most Folks ultimately get their ISP's involved upstream. ISP's can/will cause blockage to legitimate Business Services, causing hundreds of help desk phone calls, exactly what the Attacker wanted to accomplish.

Preventing the Ddos Attack at the correct Endpoint, Web/Application Servers exposed on Public Internet, is by far the best solution to issue.

Can You Please Share Your Nightly Cron Job that You ran to remove stale Reject Rules ?

I am a Linux/Bash Newbie learning more things everyday. Doe's Your Bash Script run every 60 seconds continuously, until Servers are rebooted ?