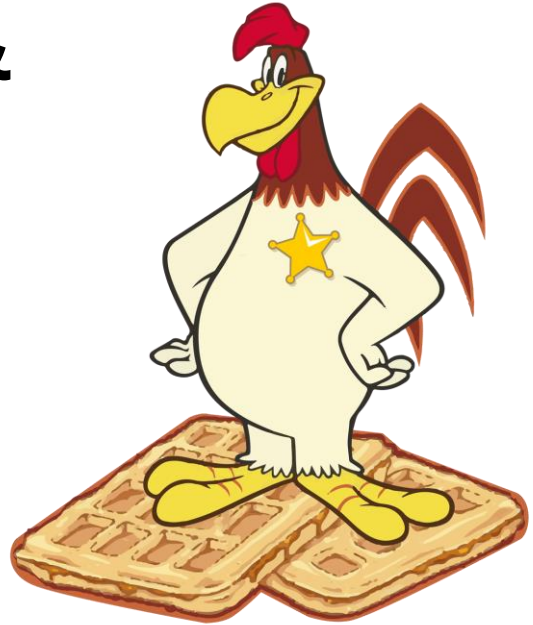
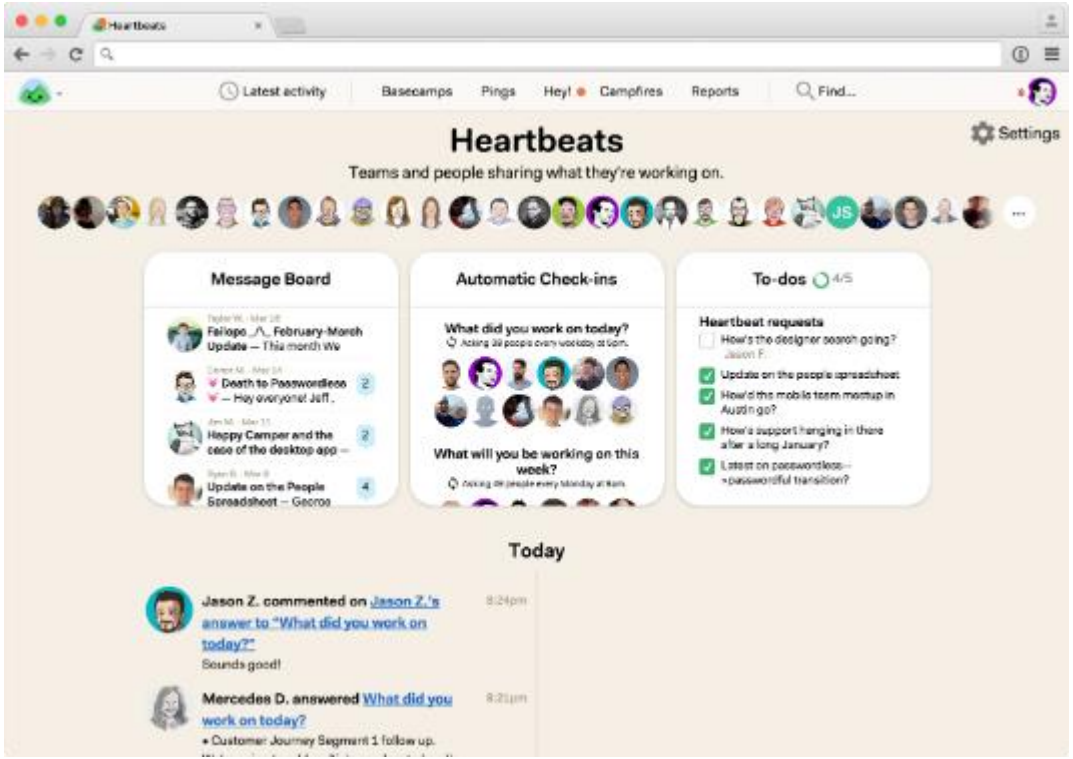
Identifying & Handling Malice with CHICKEN & WAFFLES

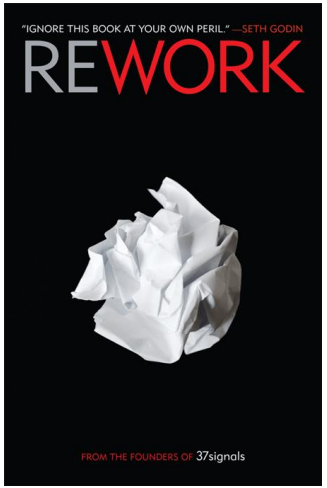
Eron Nicholson, Operations
Noah Lorang, Data

Basecamp



Basecamp makes Basecamp (and other things)





Basecamp is a little different…

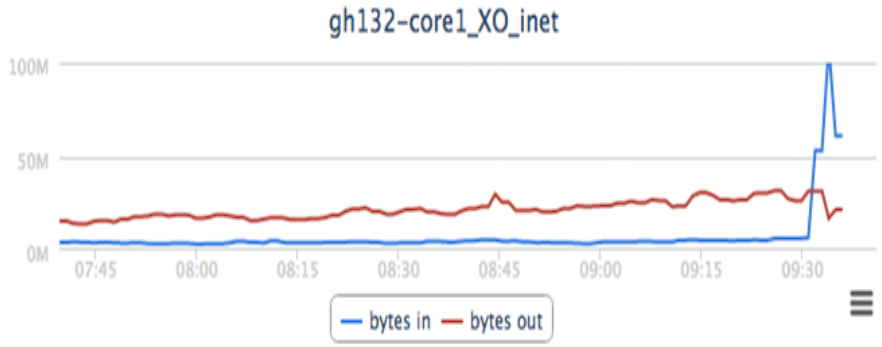
- **Not a startup:** been in business a long time, no outside funding, profitable since day one.
- **A remote company:** ~50 people spread around the world, very flat structure.
- **We own things:** operate out of leased datacenter space in three states using our own physical hardware.
- **We love Nagios.**
- **We actually do run email servers.**

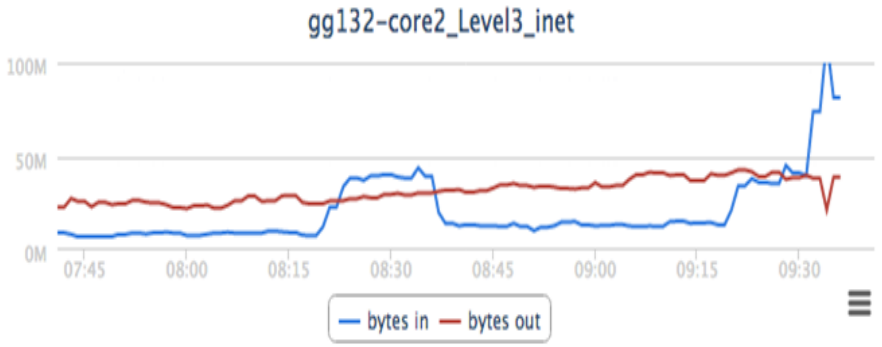

The first rule of DDoS club is that you don’t talk about getting attacked

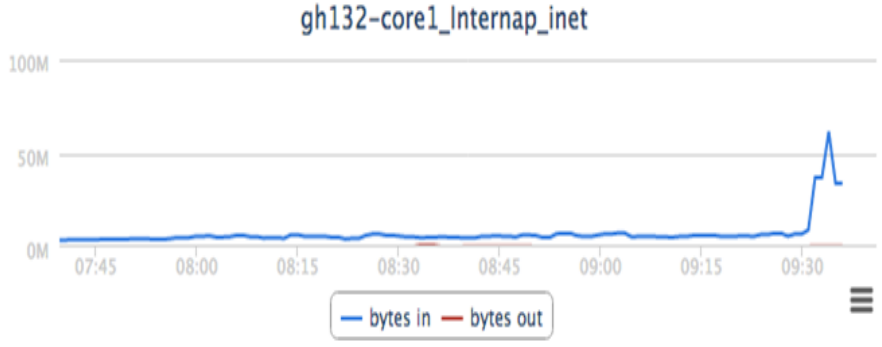
March 24, 2014: ‘twas a quiet morning in Ops when suddenly Nagios...

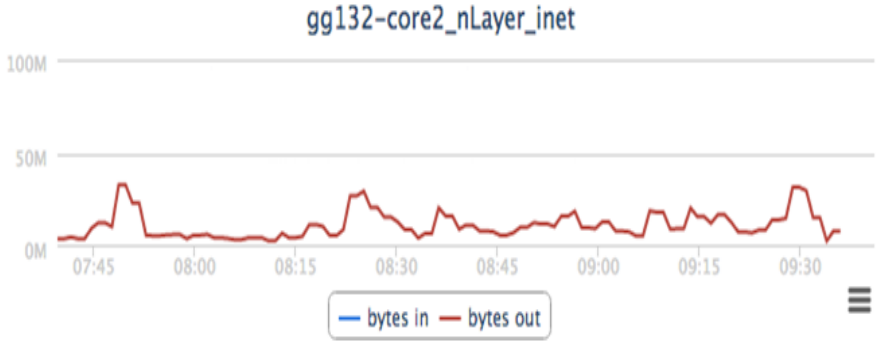
Nagios C.	sc-chi bcx-lb.rw-ash-external.37signals.com DOWN PROBLEM DOWN CRITICAL - Socket timeout after 10 seconds :
	rw-ash skybox-lb.sc-chi-external.37signals.com DOWN PROBLEM DOWN CRITICAL - Socket timeout after 10 seconds :
	rw-ash skybox-lb.sc-chi-external.37signals.com DOWN PROBLEM DOWN CRITICAL - Socket timeout after 10 seconds :
	rw-ash campfirenow-lb.sc-chi-external.37signals.com DOWN PROBLEM DOWN CRITICAL - Socket timeout after 10 seconds :
	sc-chi dash-lb.rw-ash-external.37signals.com DOWN PROBLEM DOWN CRITICAL - Socket timeout after 10 seconds :
	rw-ash campfirenow-lb.sc-chi-external.37signals.com DOWN PROBLEM DOWN CRITICAL - Socket timeout after 10 seconds :
	rw-ash Basecamp LB HTTPS Check CRITICAL basecamp-lb.sc-chi-external.37signals.com PROBLEM : CRITICAL - Socket timeout after 10 seconds - https://dash.37signals.com/health/nagios/show/B...
	rw-ash Echo LB HTTP Check CRITICAL echo-lb.sc-chi-external.37signals.com PROBLEM : CRITICAL - Socket timeout after 10 seconds - https://dash.37signals.com/health/nagios/show/E...
	rw-ash champagne-lb.sc-chi-external.37signals.com DOWN PROBLEM DOWN CRITICAL - Socket timeout after 10 seconds :
	rw-ash champagne-lb.sc-chi-external.37signals.com DOWN PROBLEM DOWN CRITICAL - Socket timeout after 10 seconds :
	rw-ash developer-lb.sc-chi-external.37signals.com DOWN PROBLEM DOWN CRITICAL - Socket timeout after 10 seconds :
	rw-ash developer-lb.sc-chi-external.37signals.com DOWN PROBLEM DOWN CRITICAL - Socket timeout after 10 seconds :
	rw-ash graceland-lb.sc-chi-external.37signals.com DOWN PROBLEM DOWN CRITICAL - Socket timeout after 10 seconds :
	rw-ash graceland-lb.sc-chi-external.37signals.com DOWN PROBLEM DOWN CRITICAL - Socket timeout after 10 seconds :
	rw-ash Portfolio LB HTTPS Check CRITICAL portfolio-lb.sc-chi-external.37signals.com PROBLEM : CRITICAL - Socket timeout after 10 seconds - https://dash.37signals.com/health/nagios/show/P...

Inbound bandwidth quickly maxed out









DDoS attack, warning

Inbox x

via 37signals.com

3/24/14

to hostmaster, admin, administrator, jason, admin, support

I don't have to explain myself. I will stop the attack for 1.9 Bitcoin (≈ $1,100).

Your network will be safe from further attacks coming from several botnets, think twice before making your decision, as even the best global DDoS mitigation won't be able to handle easily the incoming new Amp. methods.

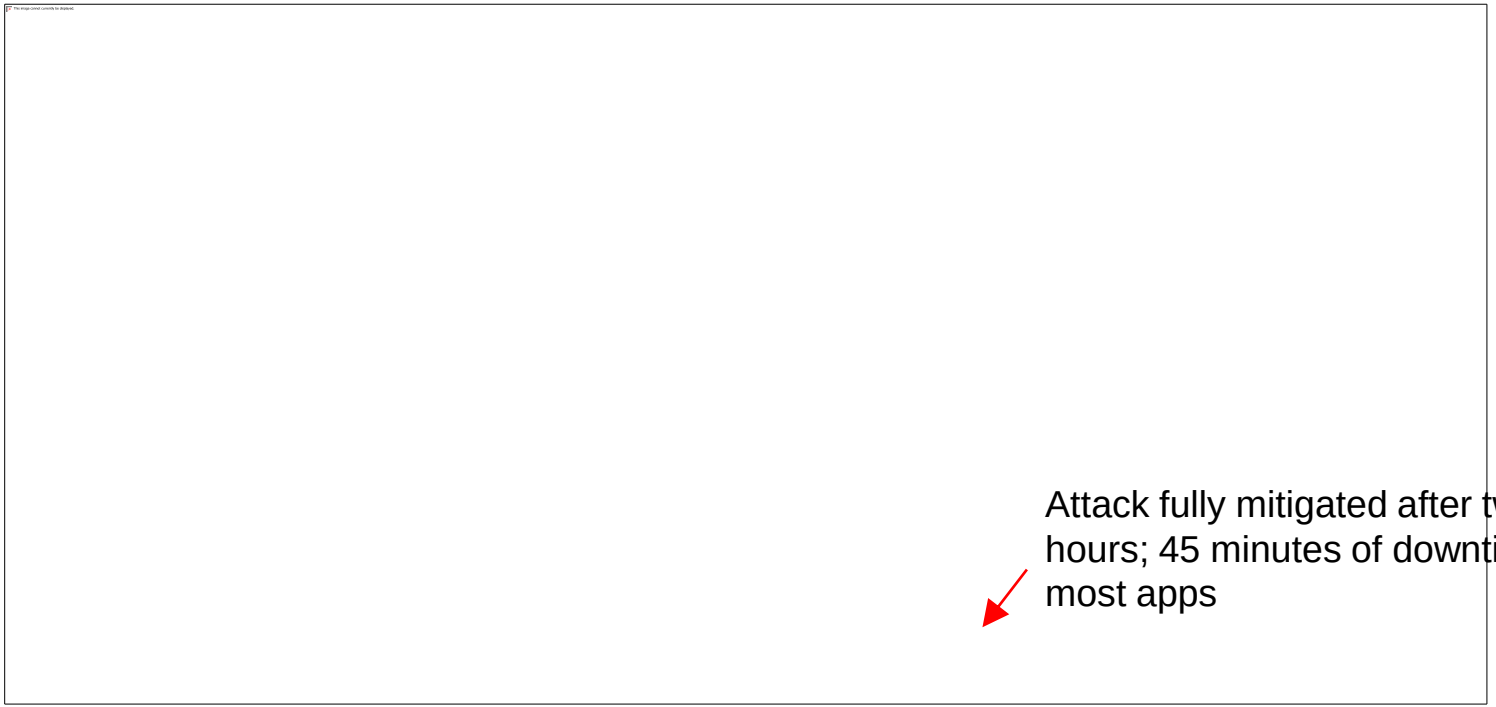
Let me know if you are interested in my offer.

A “small” attack: we measured about 80 gigabits/sec of attack traffic

- **DNS reflection** - Uses spoofed IP addresses to force open DNS servers to reply with a large payload. Amplifies attack bandwidth by ~50x.
- **NTP reflection** - Similar to DNS using open NTP servers; amplifies attack traffic ~200x.
- **SYN floods** - Large numbers of attempted TCP handshakes which are never completed.
- **ICMP flood** - Pings. Lots and lots of pings.

The attack shifted between attack types as we mitigated

Attack netflows observed on 2014-03-24



Attack fully mitigated after two hours; 45 minutes of downtime for most apps

We got serious about defense and mitigation; preparing for more volumetric attacks was the relatively easy part

- We’re fortunate to have a great datacenter partner with lots of transit that will filter for us instead of null routing our IPs.

- We went shopping:
  - Bought lots of new 10G circuits from multiple providers.
  - New routers to support the circuits and enhance our management tools.
  - Arrangements with vendors to provide emergency access to other mitigation tools.

- Felt reasonably good about approach to volumetric attacks by late 2014.

Also forced us to get serious about more subtle application level attacks

- Vulnerability scanners
- Nefarious crawlers
- SSL attacks
- Slow page requests
- Brute force password attempts
- ..etc

- We evaluated a handful of commercial hardware appliances, purchased a big name device, and ran with it in production in 2014-2015.

- Frustrated with performance and availability problems linked to that device.
  - Customers had mysterious SSL and connection issues.
  - “Fail-open” guarantees turned out to be mostly theoretical.
  - Didn’t actually get any measurable protection; signatures used were largely irrelevant to our applications.

What do we actually want in a defense system?

1. Protection against **application-level attacks** (rely on other defenses for purely volumetric attacks).

2. Allows our actual users to keep using applications uninterrupted.

3. Takes advantage of all of the data we have available, not just what you see of unterminated SSL requests.

4. Transparent in what gets blocked & why.

5. **Has a great name.**

What we need: CHICKEN & WAFFLES

WAFFLES (system that actually carries out blocking/filtering)

Web **A**pplication **F**irewall **F**rom **L**ess **E**xpensive **S**ervers

CHICKEN (identify who needs to be blocked/filtered)

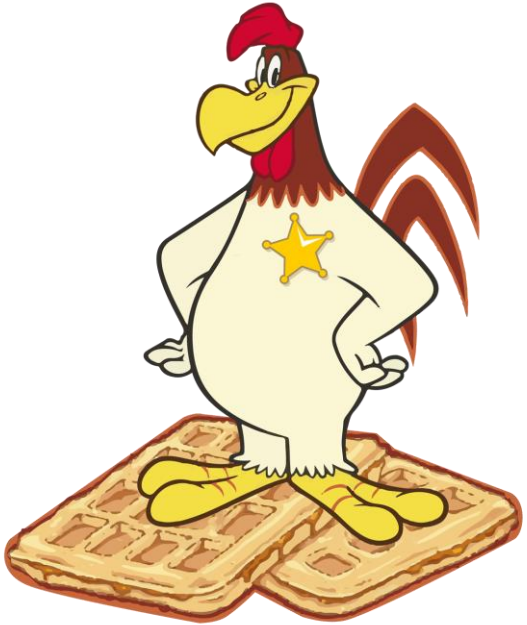
~~Customer Heuristics Influence Chances of Kicking Erroneous iNnocents~~

~~Checking How IPs Compute Kindness Errors Now~~

~~Clear Helpful Intelligent Correlations Keep Evil Nulled~~

~~Capturing Helpful Intelligence Characterizing Known Evil Nodes~~

Ok, it’s a meaningless backronym



CHICKEN: who is a real user and who is malicious?

165.29.32.239
25.74.44.231
181.68.178.93
162.52.47.79
232.165.88.236
196.123.188.6
191.79.169.245
122.34.254.243
112.102.59.146
148.84.43.216
36.156.250.165
40.104.23.20
23.188.19.48
132.167.148.125
111.58.94.243
128.104.141.60
181.44.204.226
71.145.152.83
246.24.35.254

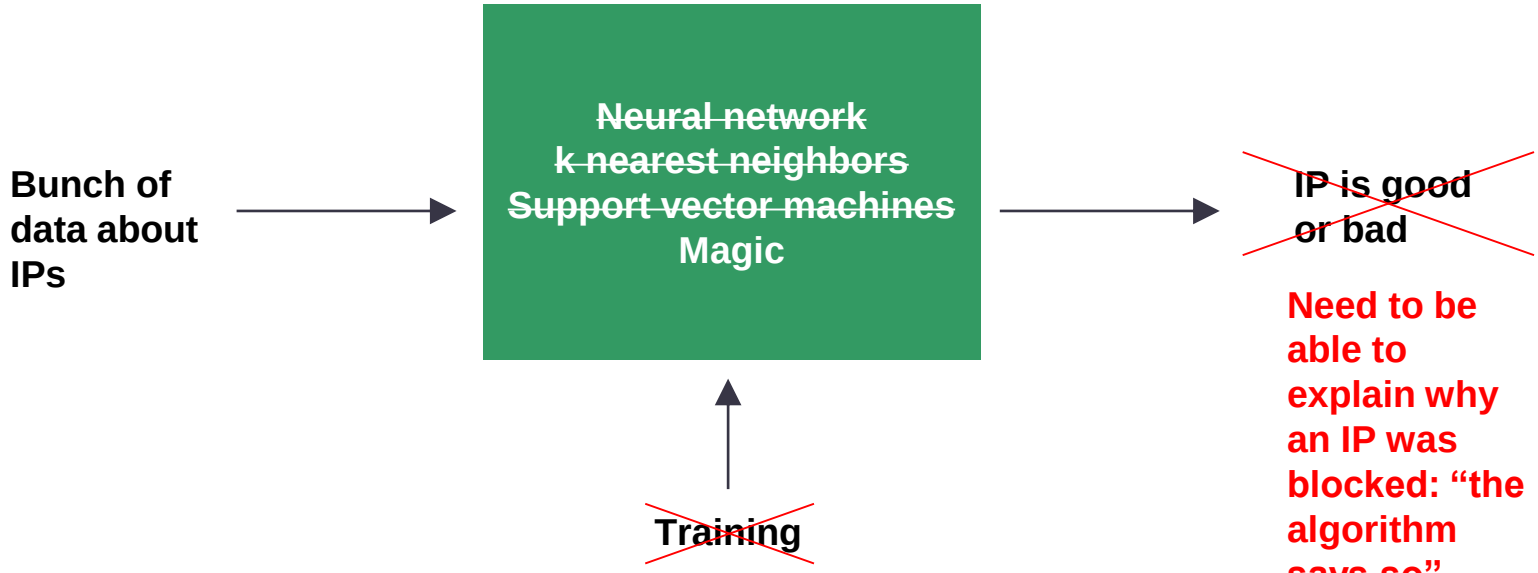
1.166.157.119
173.149.92.133
35.162.173.181
184.122.146.121
46.90.92.10
143.229.196.114
205.218.130.127
104.7.7.238
240.250.104.86
164.74.197.45
58.225.248.1
17.108.60.83
220.140.69.15
46.119.140.203
157.33.56.216
149.188.236.212
52.240.85.246
101.13.218.194
45.95.25.44

16.120.88.140
175.176.84.129
20.18.122.157
97.176.240.137
242.170.78.71
177.196.45.245
152.160.142.150
251.181.254.114
103.206.40.217
139.129.29.8
134.18.77.145
246.137.203.162
217.207.237.104
235.206.192.11
217.110.17.98
132.77.14.247
79.85.11.202
164.228.172.68
74.65.17.144

CHICKEN: who is a real user and who is malicious?

165.29.32.239	**1.166.157.119**	16.120.88.140
25.74.44.231	173.149.92.133	175.176.84.129
181.68.178.93	35.162.173.181	20.18.122.157
162.52.47.79	184.122.146.121	97.176.240.137
232.165.88.236	46.90.92.10	242.170.78.71
196.123.188.6	143.229.196.114	**177.196.45.245**
191.79.169.245	205.218.130.127	152.160.142.150
122.34.254.243	104.7.7.238	251.181.254.114
112.102.59.146	240.250.104.86	103.206.40.217
148.84.43.216	164.74.197.45	139.129.29.8
36.156.250.165	58.225.248.1	134.18.77.145
40.104.23.20	17.108.60.83	246.137.203.162
23.188.19.48	220.140.69.15	217.207.237.104
132.167.148.125	46.119.140.203	235.206.192.11
111.58.94.243	157.33.56.216	**217.110.17.98**
128.104.141.60	**149.188.236.212**	132.77.14.247
181.44.204.226	52.240.85.246	79.85.11.202
71.145.152.83	101.13.218.194	164.228.172.68
246.24.35.254	45.95.25.44	74.65.17.144

Just a simple classification problem, right?



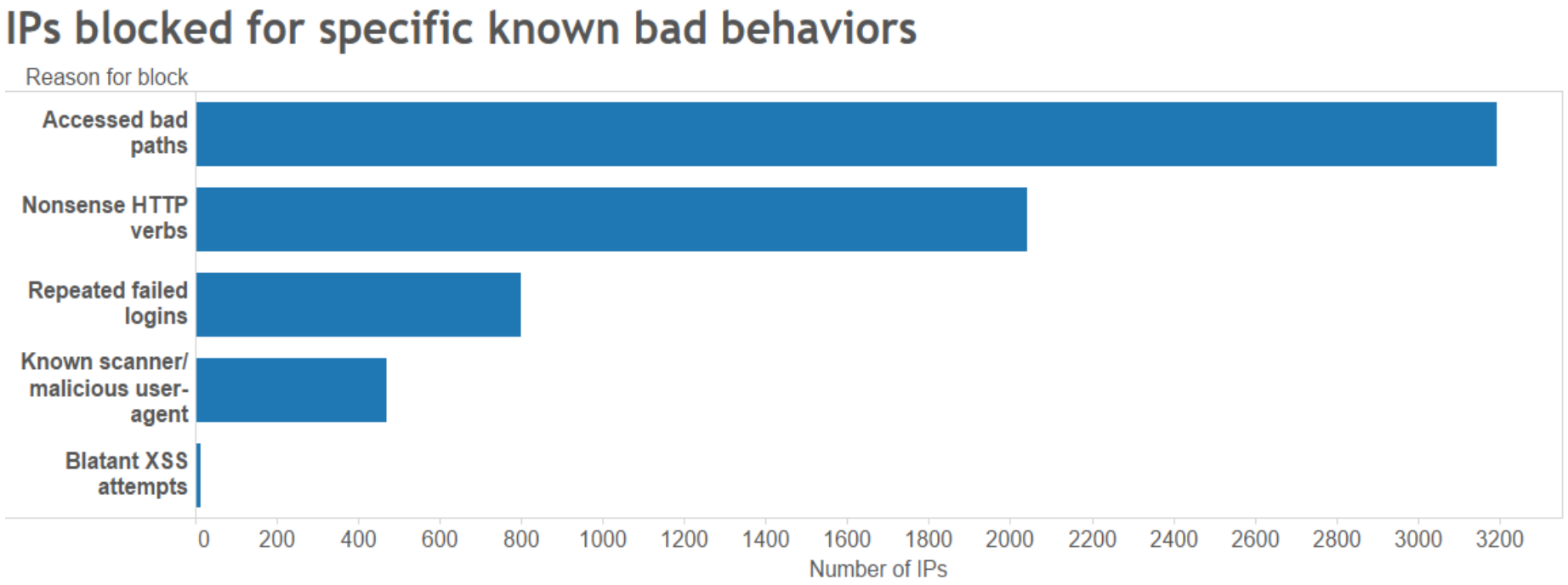
We prefer a simpler approach

1. **Some behaviors are known to be from people up to no good**
   - Crawling phpMyAdmin, CKEditor, or wp-admin paths
   - ../../../..
   - { :; }
   - Repeated failed login attempts

2. **Request history gives us a good idea of whether someone is a normal user, a broken script, or a malicious actor**
   - A history of successful, authenticated requests for a paying product is a pretty good indicator an IP address belongs to a legit user.
   - Normal users generally don't just make requests that return a 40x or 50x status code.
   - Normal users have pretty reasonable response times.

3. **External indicators provide us with other indicators about an IP**
   - IP databases provide location and ISP information; relative to our customers, malicious actors are more likely to be outside the US and use VPS providers
   - We participate in Facebook's Threat Exchange.

We've caught more than 6,000 IPs that failed to exercise the slightest amount of creativity in scanning us



Every incoming request is scored and a per-IP aggregate score calculated

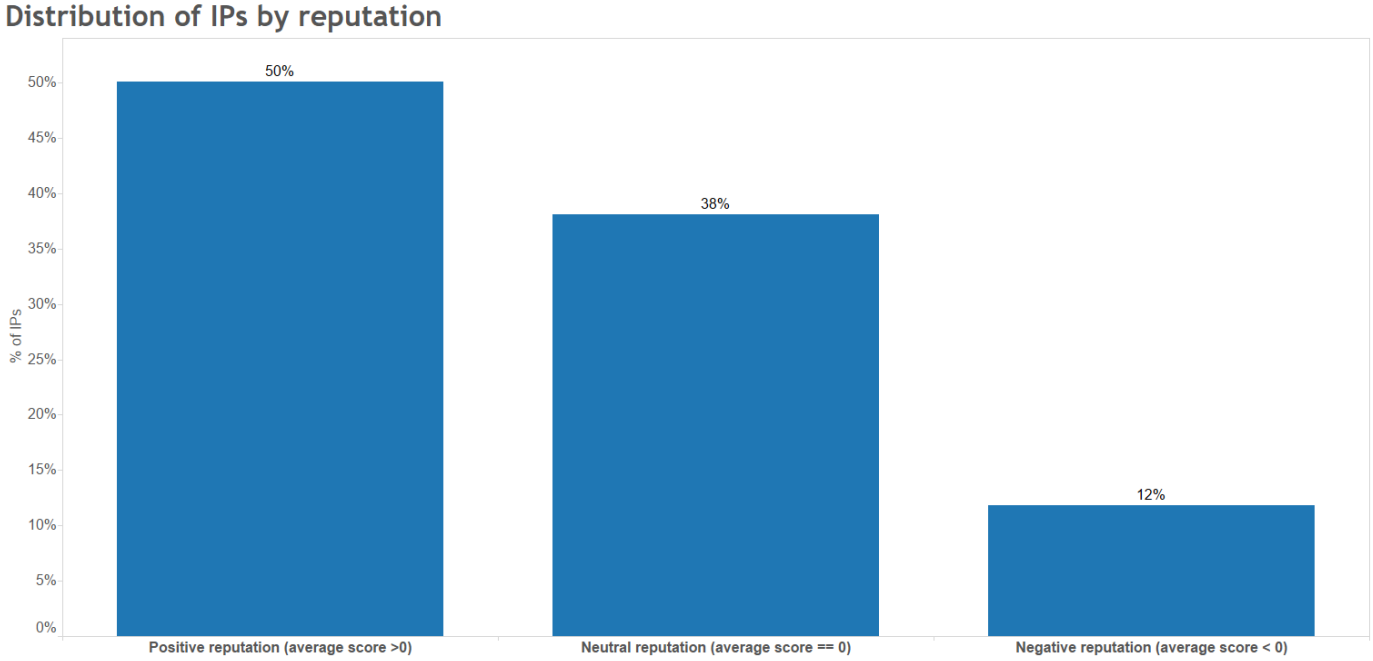
Completely arbitrary per-request scoring

- A successful (20x) request to an authenticated part of a paying application gets you up to one “point” per request.
- A 40x or 50x will lose you up to five points per request.
- Slow requests lose you points.

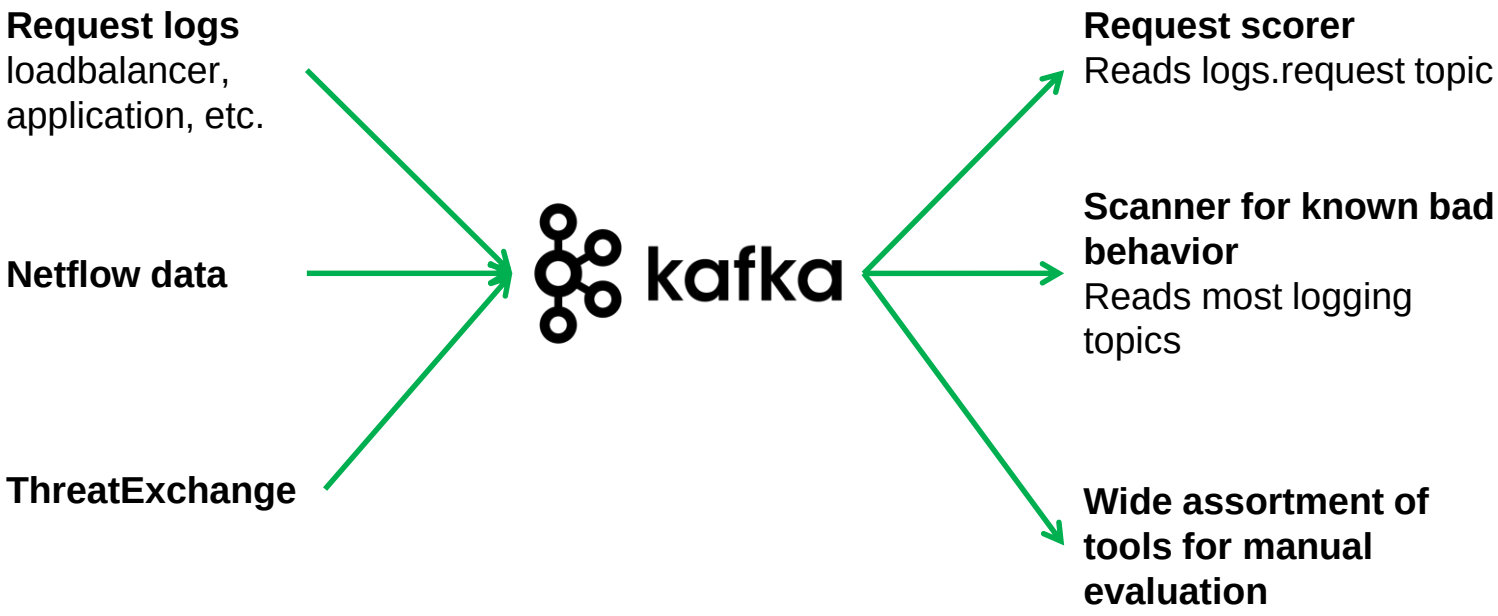
Aggregate IP reputation score

- An exponentially weighted moving average of your request scores over a given period of time results in an single per-IP reputation score
- Scores range from +1 to -Infinity
  - 1.0 = all of your requests are great
  - 0 = totally neutral, nothing positive or negative
  - <0 = perhaps this IP is not a normal user

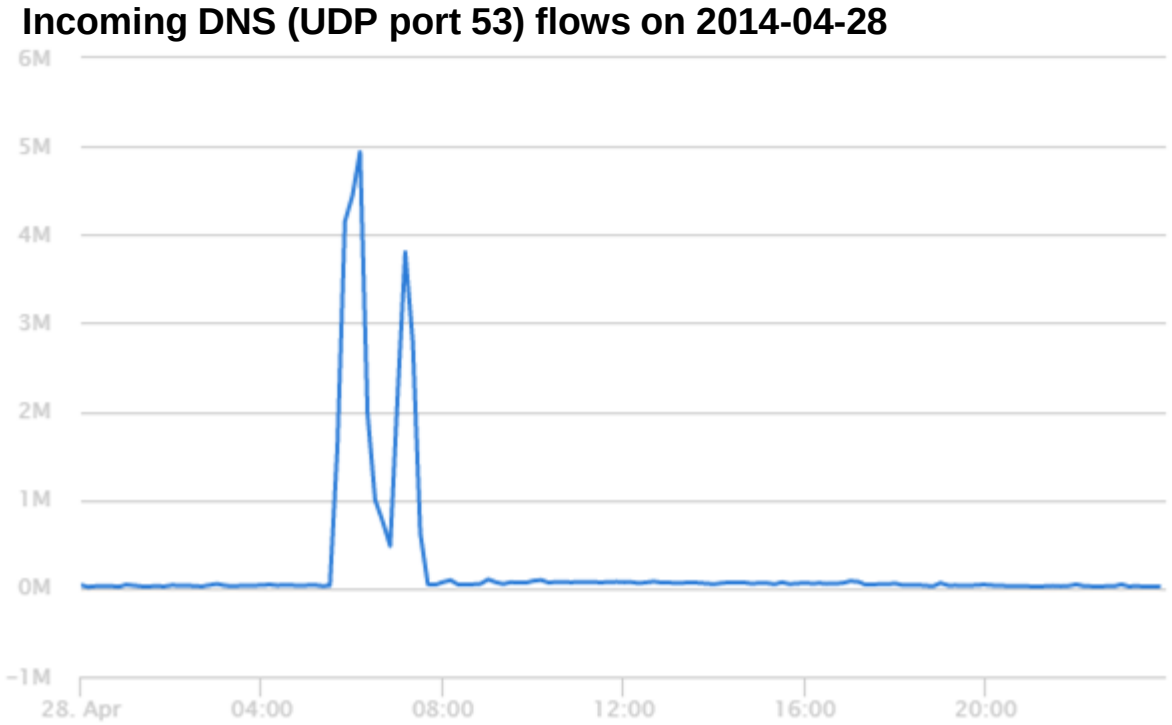
About one in ten IPs end up with a negative reputation score



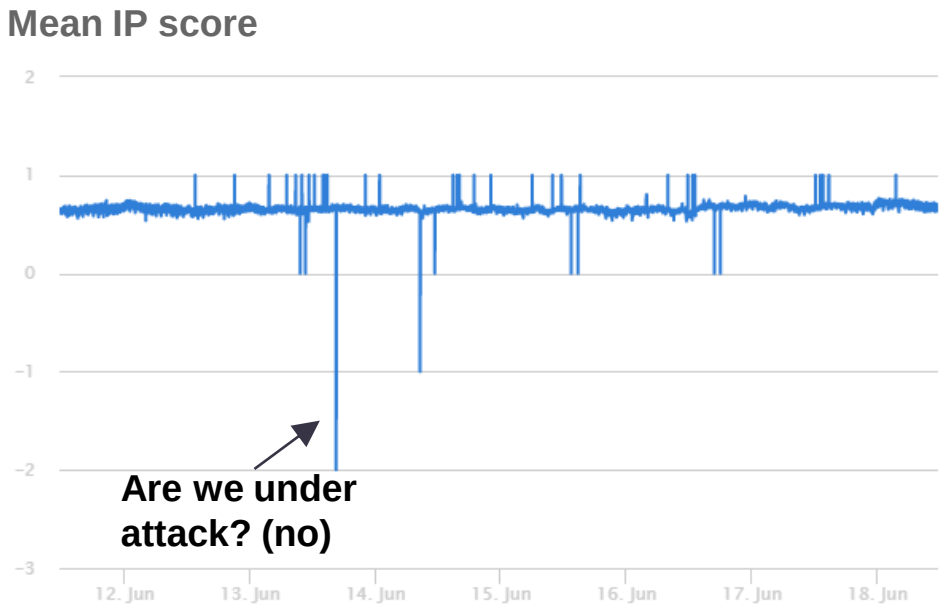
Scanning for blockable actions and scoring requests happens in near real-time using request byproducts



Monitoring for volumetric attack signatures is relatively straightforward



Average IP reputation gives us an early indicator to monitor for an application level attack

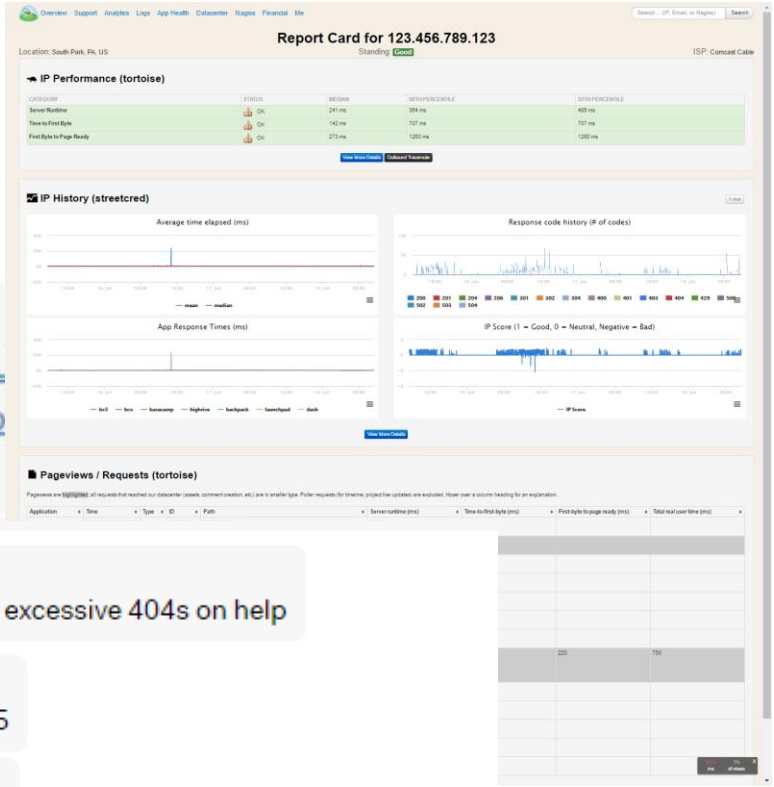


Tooling for incident response and everyday alerts

11:25am **Me**
!o chickenme
Delivered

o 11:25am
The riskiest IPs in the last half hour are as follows (score of 1 is perfectly clean, score of 0 is perfectly neutral, negative scores indicate risk):

IP	Score
65.255.146.28	-56.68
64.73.114.68	-52.98
54.162.168.126	-38.36
54.162.219.65	-35.8
203.111.224.65	-31.18
73.74.45.148	-30.97
66.215.176.56	-30.0
207.112.15.50	-28.0
192.254.250.18	-24.51
107.77.227.6	-22.52
213.162.121.140	-22.0
54.196.70.233	-20.86
174.129.135.232	-19.58
54.144.22.231	-19.42
54.81.42.200	-19.14
76.10.131.160	-18.0
70.194.70.143	-16.0
208.43.69.82	-15.97
107.178.195.134	-15.94
37.201.214.250	-15.71
185.35.186.236	-14.18

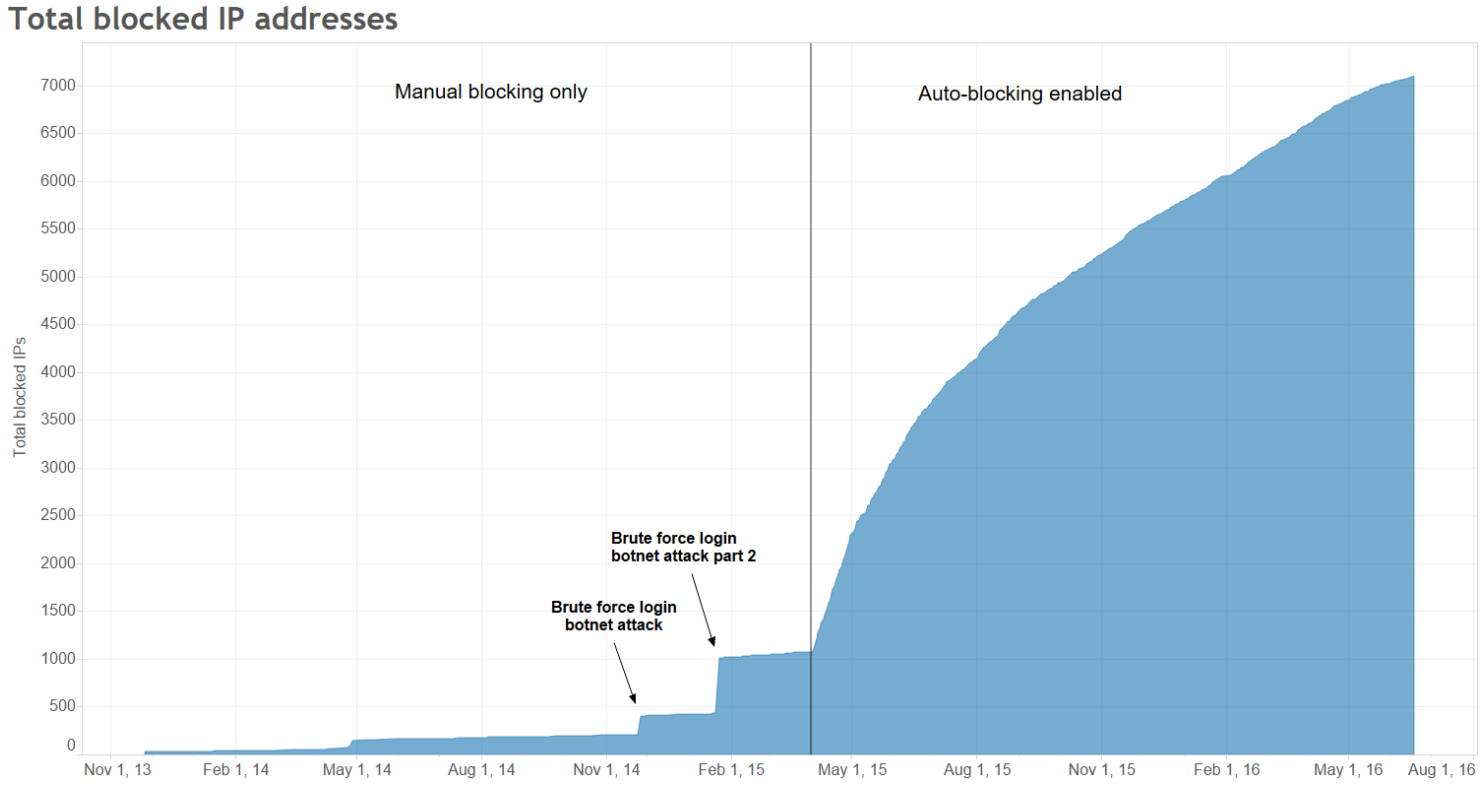


Matthew Kent 5:51pm
!o block 208.100.26.235 excessive 404s on help

o 5:51pm
Blocking 208.100.26.235

Blocked 208.100.26.235

CHICKEN has auto-blocked about 7,000 IPs with a <1% unblock rate



CHICKEN provides WAFFLES with four things

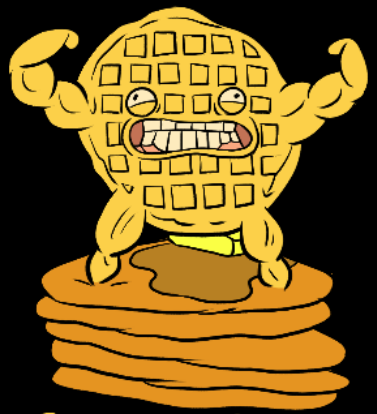
1. A list of IPs that should be blocked.

2. A list of IPs that should be trusted.

3. A list of IPs that we're unsure if we should trust or not.

4. **Roughly twice your recommended daily calorie intake.**

WAFFLES ARE JUST



PANCAKES WITH ABS

We've evolved our approach to enforcement over time to meet a number of requirements

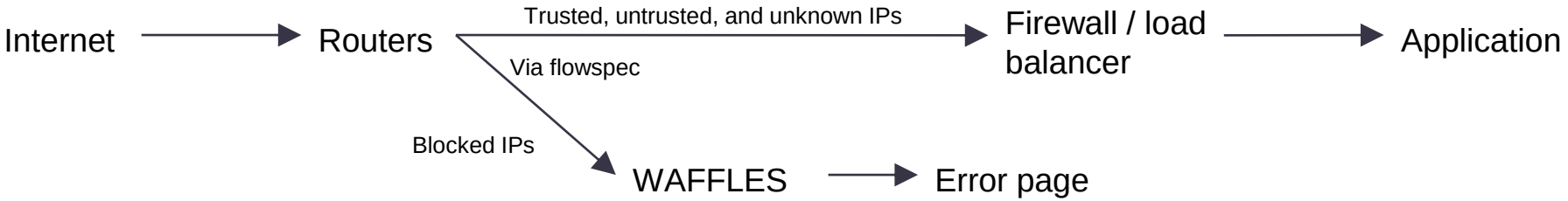
Requirement	iptables rules on HAproxy hosts	Rule on loadbalancer	Null route on routers	WAFFLES
Actions	Block	Block, rate limit	Block, bandwidth limit	Trust, block, challenge, rate limit
Automation	None	API	API, automatic blocks	API, automatic blocks
Scalability	CPU intensive	CPU intensive	-	Scale horizontally
Blocking unit	IP	IP, user-agent, path	IP	IP, user-agent, path
Customer experience	No response	No response	No response	Friendly error page

With WAFFLES off, traffic flows normally

Normal traffic flow

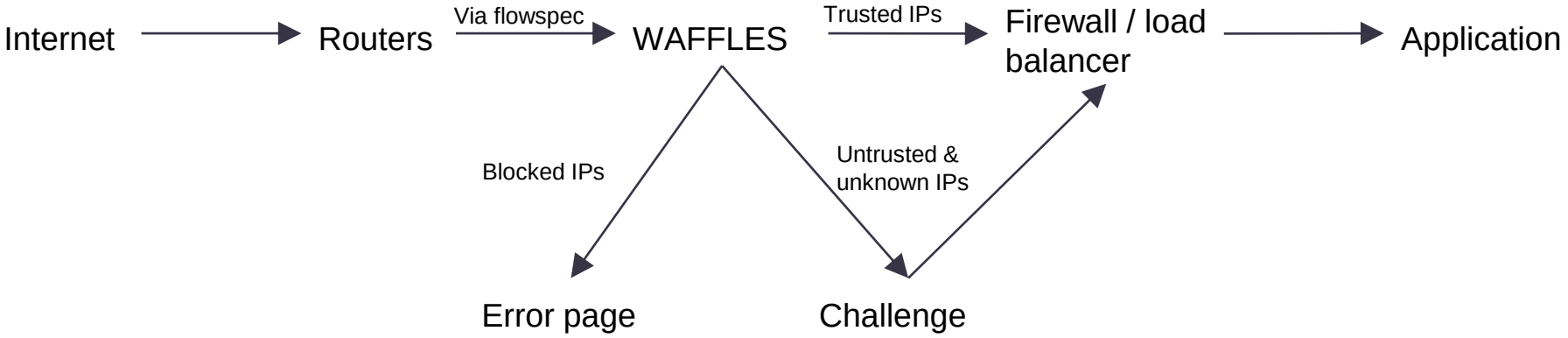
In penalty box mode, known bad traffic goes through WAFFLES and gets an error page

WAFFLES in penalty box mode



In proxy mode, all traffic goes through WAFFLES; some of it gets an error page or is challenged

WAFFLES in proxy mode



We use BGP flowspec rules to send traffic where we want it

- WAFFLES hosts live on a different network with limited access to our systems and advertise themselves using BGP to the routers.
- BGP flowspecs allow us to send specific flows of traffic to WAFFLES in realtime.
- These flows can be defined by source IPs of traffic or the destination IPs of our sites.

A WAFFLES host just runs Redis and Nginx

Redis

Blocked IP list

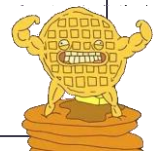
```
➜ redis-cli smembers blocked_ips
162.52.47.79
232.165.88.236
196.123.188.6
191.79.169.245
122.34.254.243
…
```

Trusted IP list

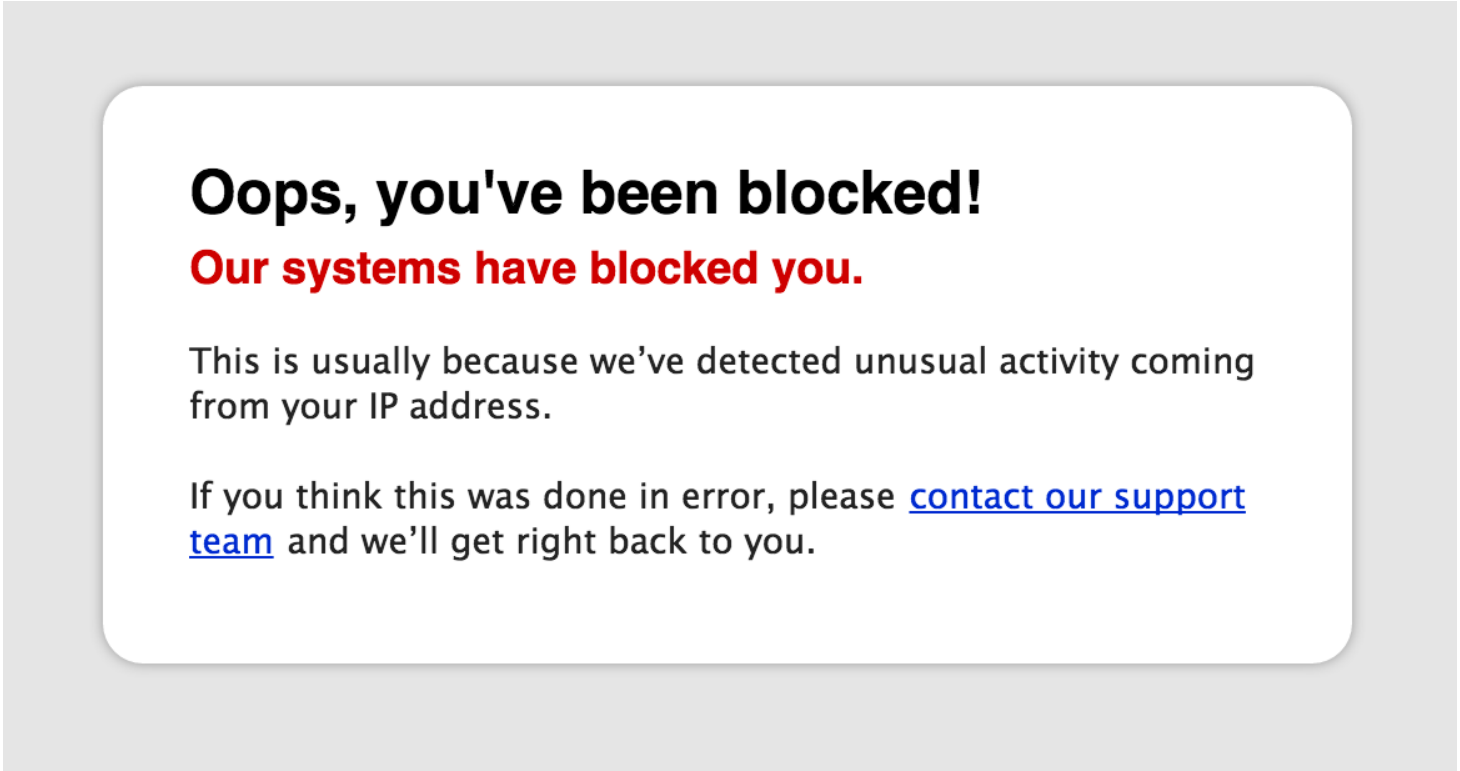
```
➜ redis-cli smembers trusted_ips
103.206.40.217
139.129.29.8
134.18.77.145
246.137.203.162
217.207.237.104
...
```

Nginx (openresty)

1. Terminates SSL
2. Checks for presence of IP in blocked or trusted lists using lua-resty-redis module
3. If blocked, serves error page
4. If trusted, reproxies to internal IPs on load balancer
5. If not blocked but not trusted, can use one of several “challenges” to determine risk of traffic (many open-source nginx modules available):
   - 302 w/ cookie set
   - Client-side crypto
   - CAPTCHA page
   - Etc.



Friendly error pages make blocks less painful



Simple challenges can be effective screeners of dumb attackers; infinitely more complicated challenges possible if you want to write them

```
curl -v https://waffles.37signals.com/fooooooo
< HTTP/1.1 302 Moved Temporarily
< Server: nginx
< Set-Cookie: WAFFLES173.240.252.102=sup3rs3kret; Domain=waffles.37signals.com;
Secure; HttpOnly
< Location: https://waffles.37signals.com/fooooooo

...

curl -v --cookie "WAFFLES173.240.252.102=sup3rs3kret" https://waffles...
< HTTP/1.1 200 OK
< Server: nginx

<h1>Hooray!</h1>
```

What we wish we knew on March 24th, 2014

1. If you don’t have good data and insight into who your users are, you can’t figure out who or what to mitigate.

2. Black box appliances don’t give you much mitigation, give you even less that you can monitor or debug.

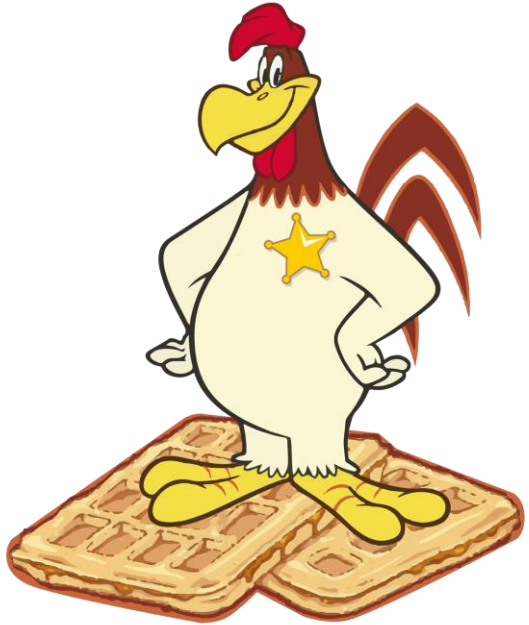
3. You can build on the shoulders of open-source giants to get a lot of bang for very little buck.

4. This isn’t “cheap insurance”, but it beats buying a hacker a new XBox.

5. Sometimes, you just need a lot of bandwidth.

Thanks!

eron@basecamp.com / @vinzclortho
noah@basecamp.com / @noahhlo



Basecamp