$bad news

you’re in a DEFENSE talk
$whoami

ryan huber
risk.io
orbitz.com
ebookers.com
small local ISP
$ contest

ssid: DoS2own
key: DoS2own!
target: http://feeblechat.com/ or http://10.0.0.2
rules: network DoS doesn’t count, MITM meh..
goal: polling must fail for > 30 seconds
prize: MY WEBSERVER!
topics

app DoS intro/attack demo
mitigation strategies
bouncer
recap
$99\%?$

service 99\% of users (limit false positives)

at 99\% utilization (know your capacity)
$\text{denial of service}$

network DoS
application DoS
topics

app DoS primer/attack demo
mitigation strategies
bouncer
recap
application DoS

its goal is
goal is to exceed your capacity to handle requests

it is
effective with few attack resources
targets or creates slow operations

more difficult to detect

no easy off-the-shelf mitigation
app DoS categories

webserver

your application
the number we care about is

MaxClients
slow headers

(slow loris)

connect

send a header every X seconds

wash, rinse, repeat
slow POST

(R U Dead Yet)

find a form

POST 2gb of data @ .0000001KB/s
$slow\ read$

GET /a_page

read it @ .000000001KB/s
Welcome to feeble-chat v0.01

s
s
df
asdf
asdf
asdf
asdf
asdf
this chat server is terrible
a/s/l?
yuno?
hello feeblechat
blah blah
test
chat works, hurray!

protip: (just hit Enter to reload)

Please don’t click this link, it hogs server resources and takes forever.
It will reload the main page after it is done processing lots of important things.
I implore you, please do not click this link.
If too many people click this link I’m surely done for.

Sincerely,
The Server
app DoS (your app)

(targeted attack)
repeatedly execute expensive queries
large downloads
exceed a backend connection pool
create many sessions
$topics$

app DoS primer/attack demo

mitigation strategies

bouncer

recap
$ apache

mpm

mod_security (Sec{Read, Write}StateLimit)

mod_reqtimeout/mod_qos

varnish/reverse proxy
strategies

keep slow pages behind login

limit POSTs

don’t generate sessions on GET /

leverage a CDN for large/static content

change webserver software

¡optimize your code!
Identification

- user-agent
- same url
- no referer
- hidden link(s)
- geoip
- ignores cookies/session

- missing common headers
- request timing
- first seen time
- proof of work
- If-Modified-Since?
- last resort: captcha
a real example

- German website
- 100,000 hosts
- 3 req/min
- Random valid URLs
topics

app DoS primer/attack demo
mitigation strategies
bouncer
recap
$bouncer

(written in node)

(inspired by netflow)
$why node.js?

because it’s @hipsterhacker approved!
$seriously why node.js?$

asynchronous
fast when your task is not CPU bound
great lib node-http-proxy
well known language
JSON is native
goals

- minimal code
- small memory footprint
- fail open
- works in cloud
- JSON messaging
- decisions made outside of proxy code
architecture

![Diagram](attachment:architecture.png)
$message format$

{"time":1379603264938,"type":"connect","host":"10.0.0.150"}

{"time":1379603264940,"type":"request","host":"10.0.0.150","url":"/changelog/","method":"GET","headers": (...),"uuid":"f42095a1-3a4b-41fc-b005-46f504cde2a0"}

{"time":1379603263662,"type":"end","uuid":"f42095a1-3a4b-41fc-b005-46f504cde2a0"}
$proxy.js$

reverse proxy

has own blacklist

has own greylist

has own disabled URL list

236 lines of code

dynamic header and request timeouts
proxy.js (does):

closes blacklisted sockets immediately
monitors total time to send headers (mitigate slow loris)
monitors total time from request to end of response
assigns UUID (v4) to every request
forwards request request records to aggregator
$aggregator.js (does)

links proxies and consumers
multiplex events to consumers
multiplex commands to proxies
64 lines of code
you write these
“drink from the (JSON) firehose”
make independent decisions
can be sized to the problem
send commands upstream via aggregator
consumer commands

block/unblock
grey
durl/eurl
htimeout
rtimeout
flush
example commands

BLOCK 10.0.0.1|10000
DURL /puppies_in_santa_costume.jpg
GREY 192.168.1.1|60000
HTIMEOUT 2000
RTIMEOUT 10000
track TCP connection attempts over a time period
cross the threshold, block
Welcome to feeble-chat v0.01

df
asdf
asdf
asdf
asdf
asdf
this chat server is terrible
a/s/i?
yuno?
hello feeblechat
blah blah
test
chat works, hurray!
hi
asdf

protip: (just hit Enter to reload)

Please don't click this link, it hogs server resources and takes forever.
It will reload the main page after it is done processing lots of important things.
I implore you, please do not click this link.
If too many people click this link I'm surely done for.

Sincerely,
The Server
track pages using the most time
track hosts using those pages
disable the url for hosts that cross threshold
demo consumer 2

Welcome to feeble-chat v0.01

df
asdf
asdf
asdf
asdf
asdf
asdf
asdf
this chat server is terrible
a/s/i?
yuno?
hello feeblechat
blah blah
test
chat works, hurray!
hi
asdf

protip: (just hit Enter to reload)

Please don't click this link, it hogs server resources and takes forever.
It will reload the main page after it is done processing lots of important things.
I implore you, please do not click this link.
If too many people click this link I'm surely done for.

Sincerely,
The Server

rhuber@EVILHOST:$
```bash
$ demo consumer 2a

rhuber@ubuntu:~/bouncer$ echo -e "S\n" | ncat localhost 5555

Welcome to feeble-chat v0.01

df
asdf
asdf
asdf
asdf
this chat server is terrible
a/s/i?
yuno?
hello feeblechat
blah blah
test
chat works, hurray!
hi
asdf

protoip: (just hit Enter to reload)

Please don't click this link, it hogs server resources and takes forever.
It will reload the main page after it is done processing lots of important things.
I implore you, please do not click this link.
If too many people click this link I'm surely done for.

Sincerely,
The Server
```
$example consumer 3$

Python + Redis

Sorted sets are AWESOME

times stored for each ip with specified granularity
$ logs

echo -e 'C\n' | ncat (aggregator) 5555 | gzip > /tmp/meh.log
$ running

proxy MUST be run with ulimit -n increase node 'forever' for daemonizing clock sync VERY IMPORTANT
AWS c1.medium example 2 -> 62k requests/s (datatest.py generated data)

network saturated before CPU
other uses

weathering a popularity storm

scraper-pocalypse
$the future, Conan?

gzip

operationalize
document

amazon amis

library of consumers

¿multicast?

¿log destroyed connections?
DoS mitigation is easier with a complete picture

suggestions are VERY welcome

contribution much appreciated
$further\ reading$

http://goo.gl/c2vyEe
$ \text{thanks!}$

https://www.github.com/rawdigits/Bouncer

@ryanhuber

rhuber@gmail.com
$ DEMO!!11!1!$