BUILDING A USEFUL NETWORK PROBE WHILE YOU WAIT

David Farrar / Exa Networks
UKNOF 40 - Manchester

The problem

- We were filtering customers on our new SurfProtect platform
  - HTTP and HTTPS proxy (we provide certificates to schools)
    in Golang - replacing ExaProxy, presented here a few years ago
  - Not yet battle tested at the time (still in early release)

- Things were going well .. until ...
  a handful of schools reported intermittent timeouts loading web pages

- Most customers were unaffected
  - Our monitoring showed no sign of the issue
  - Internal analytics showed no errors
  - No sign of latencies / packet loss (after some false alerts)

The REAL problem

- We couldn't replicate the issue ourselves

- The schools did not really want to cooperate
  - All of them were convinced our solution did not scale for them
  - And nobody wanted to risk disruption until we'd fixed the problem

- We knew that the problem was triggered with heavy traffic
  - But only with proxies explicitly configured in browsers
  - And only in a subset of locations
  - Which didn't include our testing network

- The assumption was that we'd hit some connection tracking limit on a firewall
  - But we had no way to collect the data to prove it

Quick ~~and Dirty~~ monitoring

- We’re used to building software that does exactly what we want
  - But that takes time
  - And needs testing (like we were doing now)

- Homemade monitoring solution
  - Glued together with BASH
  - CURL based monitoring solution
  - Run periodically with CRON
  - ICMP / TCP / HTTP / HTTPS and windows SSO out of the box

- We needed to track time-series data
  - Prometheus already used in internal monitoring
  - But I already knew we could write to InfluxDB via an HTTP POST
  - So we used InfluxDB

```
root@CustomerID:/home/pi# cat /usr/local/bin/check_surfprotect_adauth
#!/bin/sh

now=`python -c "import time; print(time.time())"`
probe=CustomerID

ad_auth_data=`curl --proxy ad.quantum.exa-networks.co.uk:3128 --user :
--proxy-negotiate
"http://monitor.surfprotect.co.uk/images/exa_logo.png?probe=$probe&ts=$now"
-o/dev/null -s -w"%{http_code} %{time_total}"`
ad_auth_status=$?
ad_auth_code=`echo $ad_auth_data | cut -d ' ' -f 1`
ad_auth_time=`echo $ad_auth_data | cut -d ' ' -f 2`


echo "latency,service=ad-auth,code=$ad_auth_code,status=$ad_auth_status
value=$ad_auth_time" | curl -i -XPOST 'http://localhost:8086/write?db=latency'
-o/dev/null -s --data-binary @-
```

Windows SSO

AD and Kerberos … close enough when you need quick testing.

Generate and Export a user

```
root@sp-kerberos:~# kadmin.local
kadmin.local:  addprinc -randkey quantumprobe
kadmin.local:  ktadd -norandkey -t /tmp/auth.keytab quantumprobe
```

To auto-login at boot

```
pi@pi100695:~ $ ps axf | grep k5 | ( grep -v grep )
20361 ?        Ss     0:04 /usr/bin/k5start -K 60 -U -f /srv/surfprotect/auth.keytab
```

Used ansible to deploy our monitoring

- We always use ansible
  - But now we had no direct access to the probes
  - And no idea how to get ansible to use teleport
  - .ssh/config to the rescue
  - Ansible can just directly connect to the probes

```
Host CustomerID.probe.exa.net.uk
HostName %h
Port 3022
User rpi
ProxyCommand \
    ssh -p 3023 power.user@bastion.exa.net.uk \
    -s proxy:%h:%p@CustomerID
```

No Problems found

- Still could not replicate the problem
  - Time for “PLAN B”

- We still had access to the probes
  - Decided to use AB (apache benchmark)
  - Finally saw the reported issue !

- PCAP to the rescue (on both client and server side)
  - Some connections froze (during TCP handshake)
  - Expected to see missing SYN (connection tracking limit reached)
  - But saw SYN with wrong SEQ number part of an established connection

Can you guess what is happening here ?

High performance TCP tuning

The answer: TCP TIME_WAIT …

- Time to look again at the TCP state machine …
- Great blog from Vincent Bernat https://vincent.bernat.im/en/blog/2014-tcp-time-wait-state-linux

Some vendors should read it …

- RFC default: 120 seconds
- Vendor default: 1 second
- Helps when passing MANY connections to unrelated IPs
- Value not modified when all the connections are to a single IP (the proxy)
- Causing our proxy to correlate unrelated connections

Change the vendor default to 60 … Everyone’s now happy (even if mismatched)

Other fun days included

- Google reCAPTCHA madness (traffic levels ???)
- Google directing IPv4 end-users to an IPv6 only host (ipv6.google.com)

- Chrome certificate pinning google.com … for dictation

- Do you see a pattern here ?
- See me at a break if you know someone at Google who has sympathy for NON-governmental filtering :-)

- But we should not ignore Facebook
  or anyone with an IOS app and using certificate pinning
- And everyone who thinks that 443 is the wild west for your homebrew protocol

Questions

- Happy to name and shame
- If you turn the video off :)

Otherwise it’s all Google’s fault for making the world secure !