

Self-hosting, self-defense



Tom Ryder

tom@sanctum.geek.nz

<https://sanctum.geek.nz/>

Setting the scene—1/4

- I host a **cg**it server with my Git projects:
<https://dev.sanctum.geek.nz/cgit>
- I try not to use GitHub when I can avoid it:
<https://sanctum.geek.nz/why-not-github.html>
- Publishing Git repositories with free software is pretty easy, and it's **good free software juju**.



Tom Ryder—Code

Maybe we can bring back the light.

[index](#) [about](#)

[search](#)

Name	Description	Owner	Idle	Links
<i>Dotfiles</i>				
dotfiles.git	Personal scripts and configuration files	Tom Ryder	6 months	summary log tree
<i>Experiments</i>				
adt-perl-demo.git	Demonstrating abstract data types with Perl for a friend	Tom Ryder	7 years	summary log tree
btree.git	Tinkering with terse implementations of binary trees	Tom Ryder	5 years	summary log tree
cat.git	Toy cat(1) clone	Tom Ryder	9 years	summary log tree
funcptr.git	Tinkering with function pointers	Tom Ryder	5 years	summary log tree
perlobj-demo.git	Demonstrating Perl's object system to a friend	Tom Ryder	9 years	summary log tree
spsh.git	The shitposting shell	Tom Ryder	8 years	summary log tree
texad.git	Tinkering with ideas for a text adventure in C	Tom Ryder	6 years	summary log tree
tunics.git	Toy implementations of Unix tools	Tom Ryder	9 years	summary log tree
<i>Forks</i>				
POE-Component-Client-WebSocket.git	Perl WebSocket client for POE event loop	Tom Ryder	4 years	summary log tree
go-gemini.git	Gemini protocol library in Go	Tom Ryder	4 years	summary log tree
shavit.git	Gemini server implementation in Go	Tom Ryder	4 years	summary log tree
<i>Games</i>				
doomsh.git	Set very low ulimits in Bash	Tom Ryder	4 years	summary log tree



index : dotfiles.git



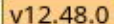

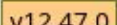
Personal scripts and configuration files

master

Tom Ryder

[about](#) [summary](#) [refs](#) **log** [tree](#) [commit](#) [diff](#)

log msg

	Commit message (Expand)	Author	Age	Files	Lines
* 	Merge branch 'release/v12.48.0'   	Tom Ryder	2024-11-06	28	-344/+32
\					
*	Bump VERSION	Tom Ryder	2024-11-06	1	-2/+2
*	Add X Compose sequences for check and cross marks	Tom Ryder	2024-11-06	2	-0/+5
*	Remove ix(1df)	Tom Ryder	2024-10-31	7	-30/+1
*	Stop trying to figure out what's a text filename	Tom Ryder	2024-10-31	16	-273/+0
*	Drop Windows support from Vim files	Tom Ryder	2024-10-31	2	-11/+4
*	Add an issue about Unicode character filenames	Tom Ryder	2024-10-28	1	-0/+2
*	Remove resolved lesskey issue	Tom Ryder	2024-10-28	1	-4/+0
*	Adjust comments to remove duplicates	Tom Ryder	2024-10-25	1	-4/+4
*	Update alternate_filetypes.vim plugin	Tom Ryder	2024-10-25	1	-0/+0
*	Update broken links in .vimrc	Tom Ryder	2024-10-24	1	-7/+7
*	Remove repeated logic in prompt	Tom Ryder	2024-10-24	1	-13/+7
*	Merge branch 'release/v12.47.0' into develop	Tom Ryder	2024-10-23	1	-2/+2
\					
* \	Merge branch 'release/v12.47.0' 	Tom Ryder	2024-10-23	2	-4/+7
\ \ \					
\					

Setting the scene—2/4

- Git repositories can be complex, and are inherently **reference-dense**.
- Repositories, commits, tags, branches, trees, objects, reflogs, submodules, diffs...
- When displayed on a web page, this amounts to *lots of links*.



Setting the scene—3/4

- All these links often mean a lot of attention from **bots**: programs acting like users to get data.
 - Usually poorly-written bots...
 - Just because there are a lot of links doesn't mean the content's worth indexing.
 - A bot would never care to haunt [my dotfiles](#)!
 - ...right?



Setting the scene—4/4

- Usually, the bots aren't a problem.
- Heck, usually, I don't even *notice*.
- Bandwidth is cheap, and cgit pages are only a few kibibytes.
- A bot requesting a few thousand pages over a few days isn't a big deal.



Business as usual—1/5

- If a bot *does* get bothersome, I first politely shoo away its **user agent** in [robots.txt](#):

```
User-Agent: ClassicBot/0.1
```

```
Disallow: /
```



Business as usual—2/5

- If, after a day or two, it's still bothering the server, I *block* its user agent in Apache HTTPD:

```
BrowserMatch ClassicBot bad_bot  
Require not env bad_bot
```



Business as usual—3/5

- And then...well, that's *usually* enough.
- Sometimes, a nasty bot will pretend to be a real person using a browser, so I *can't* block it reliably:

```
Mozilla/5.0 (Windows NT 10.0; Win64;  
x64) AppleWebKit/537.36 (KHTML, like  
Gecko) Chrome/135.0.0.0 Safari/537.36
```



Business as usual—4/5

- In that case, I have to *think* just a little bit (!)
- I identify the bot, usually by the number of requests per IP, its not asking for CSS or JavaScript, or images, etc...
- ...and block the offending IP addresses with **HTTP 403 Forbidden**.



Business as usual—5/5

- I've followed this pattern for *years*, both at home and at work, with only minor variations all that time.
- I could do it in my sleep...`awk`, `grep`, `ip route add blackhole...`
- Sometimes a botnet is bigger... maybe a few hundred hosts. Gather a list, block them all, and I'm done.



But things have been a bit...
different in 2025.

Attack begins—1/2

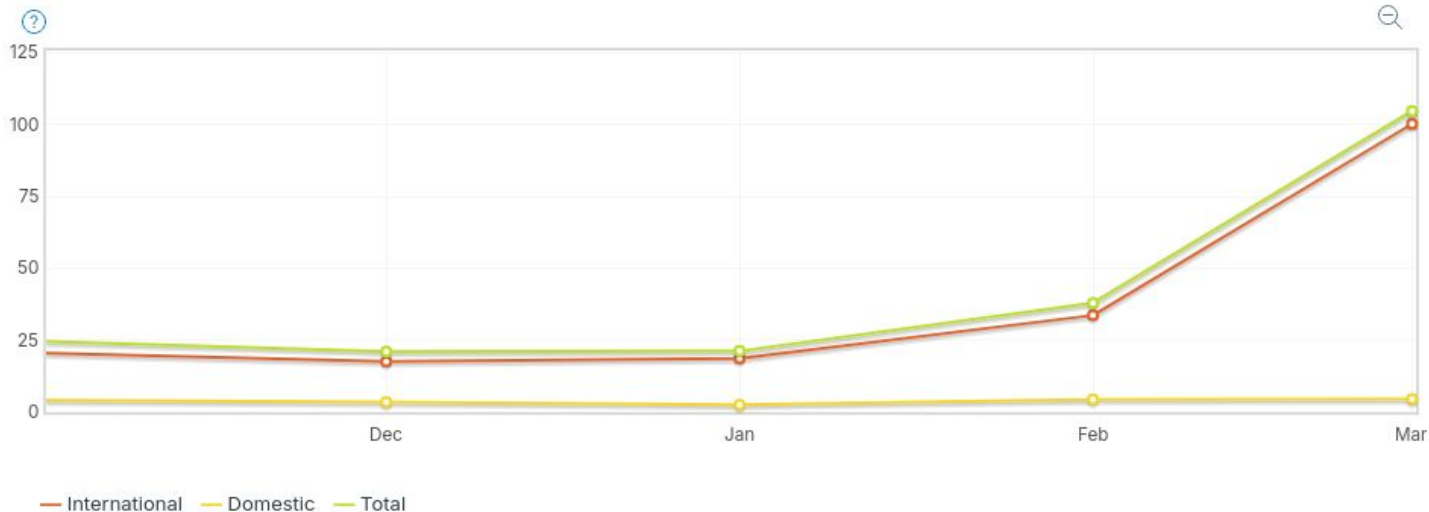
- Through late February and much of March, the sanctum.geek.nz webserver performed *really badly*.
 - Exhausted Apache HTTPD process slots
 - Very slow response times
 - Memory exhaustion (kernel killing processes!)
 - Eventually locking up completely...



Bandwidth

⊕ Create Alert

Monthly Bandwidth: sanctum.geek.nz



100 GiB of traffic in a month...

Attack begins—2/2

- On inspection of the logs, I was getting requests from a botnet...a big one.
- And more sophisticated than usual:
 - Presenting with user device browser agents...
 - With plausible-looking requests...
 - Coming from many different addresses...
 - Both standard IPv6 and legacy IPv4...
 - All asking for pages from [my dotfiles](#)!



Mitigation begins—1/3

- So, I knuckled down and got to work blocking it. Annoyed, but not really worried at this point.
- I randomly sampled a few of the IPs. They were all residential, from end-user ISPs, all around the world.
 - No, not all Brazil or Russia...
 - No server rooms, really. No big iron.
 - Normal people's *homes*.
 - Perhaps oddest: *not a single Amazon IP*.



Mitigation begins—2/3

- They were from many different networks (**autonomous systems**). Even blocking countries wasn't going to work.
- Literally *thousands of requests per minute*...
- And it was getting *worse*.



Mitigation begins—3/3

- A lot of the randomised fake user agents were...*implausible*.
- Internet Explorer 5.5 on Windows CE in Burmese, from a French IP? *Cool story, bro.*
- I made a list of the weirdest ones, and started collecting IP addresses.



Oh, the *naïveté*!—1/2

- Soon, I had a nice automatic loop going:
 1. Detect particularly implausible user agent.
 2. Add the IP address to an Apache HTTPD block list.
 3. Reload Apache HTTPD every few minutes.
 4. Watch as the 403s start rolling in!



Oh, the *naïveté*!—2/2

- Except...they *didn't*.
- And when I got above about 100,000 Require not ip directives, Apache HTTPD was getting very slow to reload...
- And the botnet was *still* getting faster. Coming in massive *waves* now...over 100,000 requests per hour.



Let's look at a couple of videos...

A new kind of botnet—1/2

- Almost every IP was making a single request, and then never turning up again.
- A few of them requested two or three pages.
- I churned through software firewall IP blocks, until I had blocked **3 million addresses**, with no improvement...
- ...and accepted this *wasn't working*.



A new kind of botnet—2/2

- This was so *weird* to me that for a while I was wondering if someone was tricking me or had hacked me.
- How could it *possibly* be coming from so many IPs? No (classical) botnet is that large.
- But everything I checked showed the traffic was *real*.
- The TCP handshake worked both ways.



A new kind of botnet—3/3

- “Why not just take that one site down for a while?”
- I *did*, for two days—the requests *didn't stop*.
- Even just fielding them with 403 Forbidden over and over was straining Apache HTTPD.
- Not to mention my looming SiteHost bandwidth bill...



Slough

- I won't lie: by this point I was *pretty angry*.
- I'd spent hours over two weeks fighting off the botnet, with essentially *no* progress.
- My [Jitsi](#) server for tabletop games was unstable.
- I was consuming gibibytes of bandwidth a day just answering the botnet's ceaseless requests.
- I needed a new strategy.



Weight of souls—1/2

- I rolled out **Anubis**, a **proof-of-work** challenge proxy.
- It issues human-presenting browsers a JavaScript crypto challenge, and doesn't let it past until it's solved.
- For a real person on a modern computer, it's only a few seconds.



Making sure you're not a bot!



Calculating...
Difficulty: 4, Speed: 10.645kH/s



► Why am I seeing this?

Protected by [Anubis](#) from [Techaro](#). Made with ❤️ in 🇨🇦.

Mascot design by [CELPASE](#).

Success!



Done! Took 3761ms, 43774 iterations

► Why am I seeing this?

Protected by [Anubis](#) from [Techaro](#). Made with ❤️ in 🇨🇦.

Mascot design by [CELPASE](#).

Weight of souls—2/2

- The bots didn't try to solve the challenge, so they just kept loading the Anubis page from RAM.
- This helped a bit with resource exhaustion, as data wasn't being pulled from git and rendered as a web page.
- Any human with JavaScript enabled—and well-behaved bots—could still browse the repositories.



Early and often—1/8

- But what to do about the *massive* amount of traffic?
- My SiteHost plan charged extra for international bandwidth over 100 GiB a month.
- I needed to cut the bot requests off as *early as possible*...and then hope they gave up.



Early and often—2/8

- I did a bit of research into how a botnet could be so massive, and learned about **residential proxies**.
- These are formed by software like VPNs and browser extensions that offer features like watching Netflix as served in another country with a few clicks.
- In return, the company sells bandwidth on the user's connection to anyone who wants it.
- So: a **botnet**, with a veneer of legitimacy.



Early and often—3/8

- This seemed the most likely explanation for how *millions* of IPs were apparently interested in scraping my Git repository.
- So a relatively small set of computers was using a massive network of SOCKS proxies to send a phenomenal number of *unblockable* requests...
hmm.



Early and often—4/8

- So what else could *all* the requests have in common that might help me identify them?
- Any unusual headers? Nope, all looked pretty normal. Accept :, Accept-Language :, Content-Type :, all used correctly.



Early and often—5/8

- What about **TLS**? They're all connecting to my HTTPS endpoint.
- I wonder if they all negotiate the complexities of HTTP over TLS with SNI the exact same way?
- The same protocols, the same supported ciphers and algorithms...
- Is there a way to *fingerprint* that...?



Early and often—6/8

- I ran `tcpdump(8)` on my public interface for a bit to record all the traffic and get some raw data:

```
$ sudo tcpdump -w pcap -i enX0 \  
    tcp and dst port 443
```

- The vast majority of the traffic in the packet capture file was indeed the botnet.



Early and often—7/8

- I ran **JA3** over the packet capture to get **TLS client fingerprints**...
- And *every single botnet request* had the same hash fingerprint:

5cc600468c246704e1699c12f51eb3ab

- *None* of the legitimate requests had this signature.
- There were no search engine results for it.





Early and often—8/8

- I installed the [Suricata](#) firewall, and put it into NFQ mode.
- Suricata had one rule: **drop** any traffic matching that JA3 HTTPS signature.
- I put all my incoming HTTPS traffic through it...



Sweet, sweet
silence.

Victory—1/2

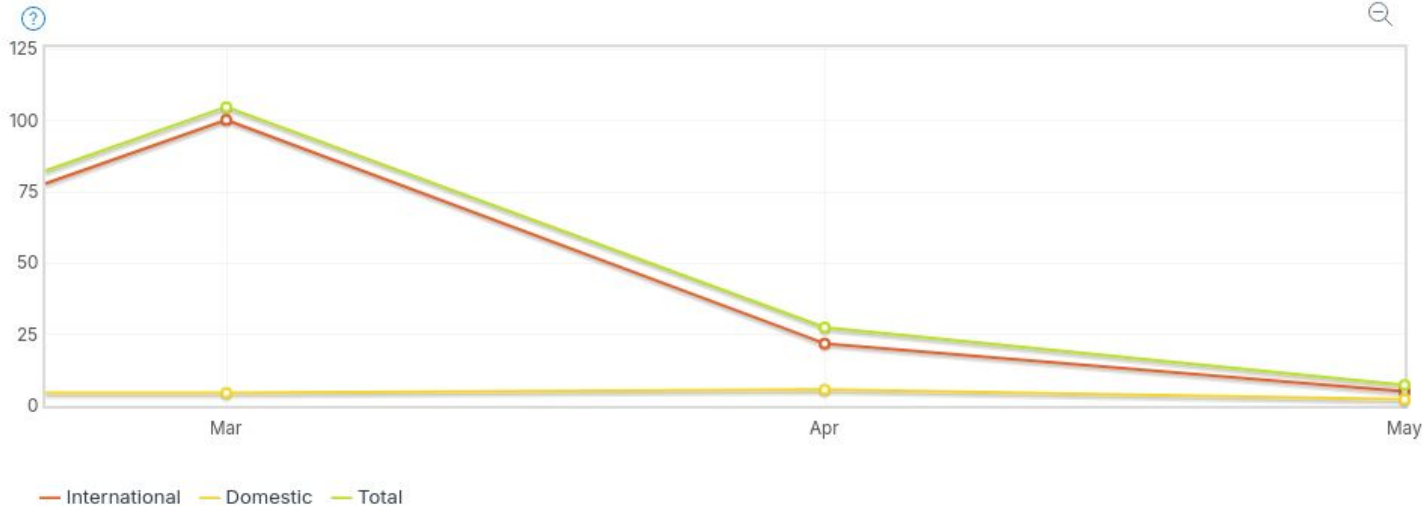
- The server was at once perfectly responsive.
- The load average plummeted.
- My Jitsi instance worked perfectly.
- The Apache HTTPD logs for the site hummed gently with *real traffic*.
- One little hash had fixed everything.



Bandwidth

⊕ Create Alert

Monthly Bandwidth: sanctum.geek.nz



Eventually, the botnet gave up and stopped.

Victory—2/2

- A few days later, I found a [blog post](#) (in French) by someone who seemed to be having very similar problems to me...
- I replied to their [Fediverse post](#), to see if I could help out with my new powers.





Dryusdan

@Dryusdan@social.dryu...

🌐 Apr 9



Nouvel article : "L'IA et Forgejo" <https://dryusdan.space/lia-et-forgejo>



L'IA et Forgejo

Dryusdan.space 🚀 Voyagez dans m...



1+





Tom Ryder

@tejr@mastodon.sdf.org

@Dryusdan

Hello; I have had very similar problems—maybe the same botnet. tcpdump with ja3.py showed the botnet requests all had the same JA3 TLS client sig. I blocked that with Suricata filtering inbound HTTPS in IPS mode (iptables NFQUEUE) and TCP RST, with a shorter timeout for Apache HTTPD requests (mod_reqtimeout). This has worked very well after what has been a maddening few weeks.

(Sorry, my French is good enough to read your post, but not to reply...)

Apr 14, 2025, 15:46 · 🌐 · Web · ↻ 1 · ★ 1



Tom Ryder

@tejr@mastodon.sdf.org

@Dryusdan If you try this, let me know
if the sig you find matches mine:
5cc600468c246704e1699c12f51eb3ab

Apr 14, 2025, 15:50 · 🌐 · Web · ↻ 0 · ★ 0



Dryusdan

@Dryusdan@social.dryusdan.fr

@tejr I find the same sig for this pattern !

Apr 14, 2025, 20:32 · 🌐 · ↻ 0 · ★ 1

But *why*?!

- People think this new generation of bots is scraping for Large Language Model (LLM) training content.
- The timing seems right, given LLM mania (May 2025).
- However, I have no proof. It's a *guess*.
- I don't know what the bots were after from me.
- I think they just got stuck on the many links.



Takeaways

- Botnets and sysadmins are still in an arms race.
- IP blocks don't help with residential proxies.
- Behavioral analysis in general is powerful.
- JA3 and TLS client signatures are great.
- Suricata and Anubis are also great.



Questions?

- Anubis
- JA3
- Suricata
- tcpdump

Email: tom@sanctum.geek.nz

Website: <https://sanctum.geek.nz/>

Fediverse: [@tejr@mastodon.sdf.org](https://mstdn.social/@tejr)

