Censored Planet LAB

Technical Analysis of the Geedge Networks Firewall Source Code Leak

Anna Ablove[1], J. Walker[2], B. Wolin[1], N. Niere[3], F. Lange[3], A. Ortwein[1], A. Huremagic[1], R. Priyanka[1], A. Zohaib[4], J. Sheffey[4], N. Heitmann[3], J. Alex Halderman[1], J. Somorovsky[3], A. Houmansadr[4], R. Ensafi[1], M. Wu[2], E. Wustrow[5]

University of Michigan[1], GFW Report[2], Paderborn University[3], UMass Amherst[4], CU Boulder[5]

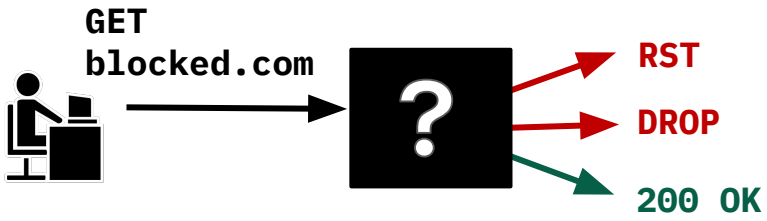


Censored Planet

Current State of Censorship Research

Reliance on Black Box Experiments

Researchers must **infer** censorship mechanisms **from the outside,** relying on **limited and noisy measurement** data.



Safety Risks at Scale

Running exhaustive measurements can be dangerous, exposing researchers and collaborators to **safety risks** .



WIRED SEP 8, 2025 11:00 PM NEWSLETTERS SUBSCRIBE

Massive Leak Shows How a Chinese Company Is Exporting the Great Firewall to the World

Geedge Networks, a company with ties to the founder of China's mass censorship infrastructure, is selling its censorship and surveillance systems to at least four other countries in Asia and Africa.

PBS NEWS WEEKEND

Massive leak exposes how China's 'Great Firewall' is being exported to other countries

Oct 18, 2025 5:40 PM EDT

Massive "Great Firewall of China" data leak reveals surveillance tech Silk Road

Published: 15 September 2025 · Last updated: 16 September 2025

Leaked files show a Chinese company is exporting the Great Firewall's censorship technology

Leak of Geedge Networks internal documents (100,000+ from Jira, Confluence, GitLab) #519

Censored Planet

The Geedge Networks Data Leak

LEAK COMPOSITION

572 GiB total · commits thru Nov 2024



■ Repo Mirror 463 GiB ■ Git 59 GiB ■ Confluence 46 GiB ■ Jira 2.5 GiB

Category	File	Key Contents
Source code	`mirror/repo.tar`	RPM bundles: Firewall, libcbd, QDPI & Glimpse detectors
	`mesalab_git.tar.zst`	Git repos: SAPP, Protocol Plugins, Stellar-on-SAPP, Stellar, Maat, tsg-os-buildimage
Confluence	*2 archives*	VPN signature logs, Psiphon collateral-damage analysis
Jira	`geedge_jira.tar.zst`	Deployment tickets (Myanmar, Ethiopia), bug fixes
Other	*14 files + filelist*	—

Going Beyond Inference

01 How does this system parse and analyze traffic?

02 What processes does it implement to block VPNs and circumvention tools?

03 Can we find empirical evidence of real world deployment?

Background on the Data Leak

Geedge Networks

Founded in 2018, in part by Fang Binxing, the “Father of the Great Firewall”

Ties to the Massive and Effective Stream Analysis (MESA) Lab and Chinese Academy of Sciences

Flagship product: **Tiangou Secure Gateway (TSG)**

GEEDGE networks

Civil Society Reporting

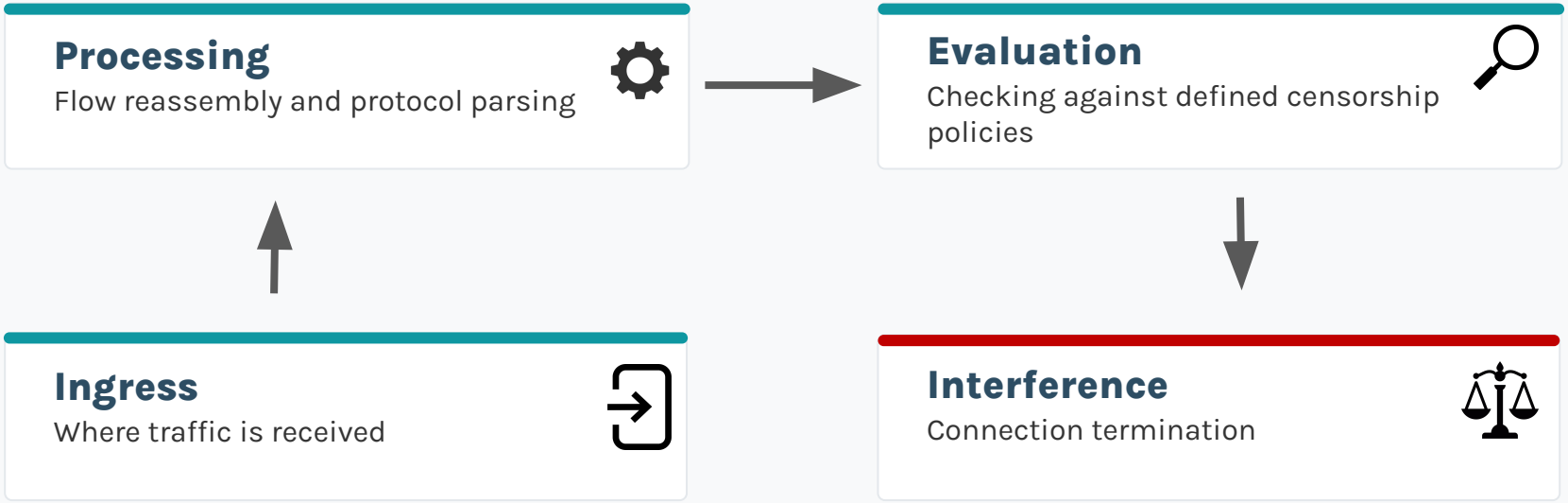
Groups such as **Amnesty International, IntersecLab, and Justice for Myanmar** had privileged access to the leak and reported on non-code files and documents.

Documented Exports to Countries

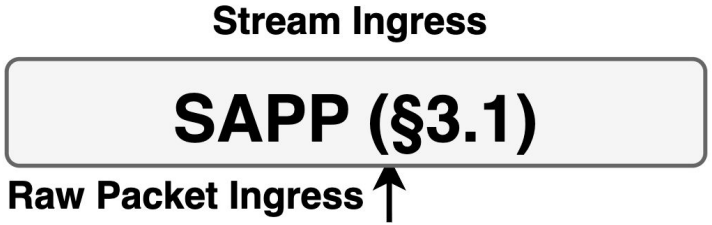
Reported deployments in **Pakistan, Myanmar, Ethiopia, and Kazakhstan**



DPI Architecture



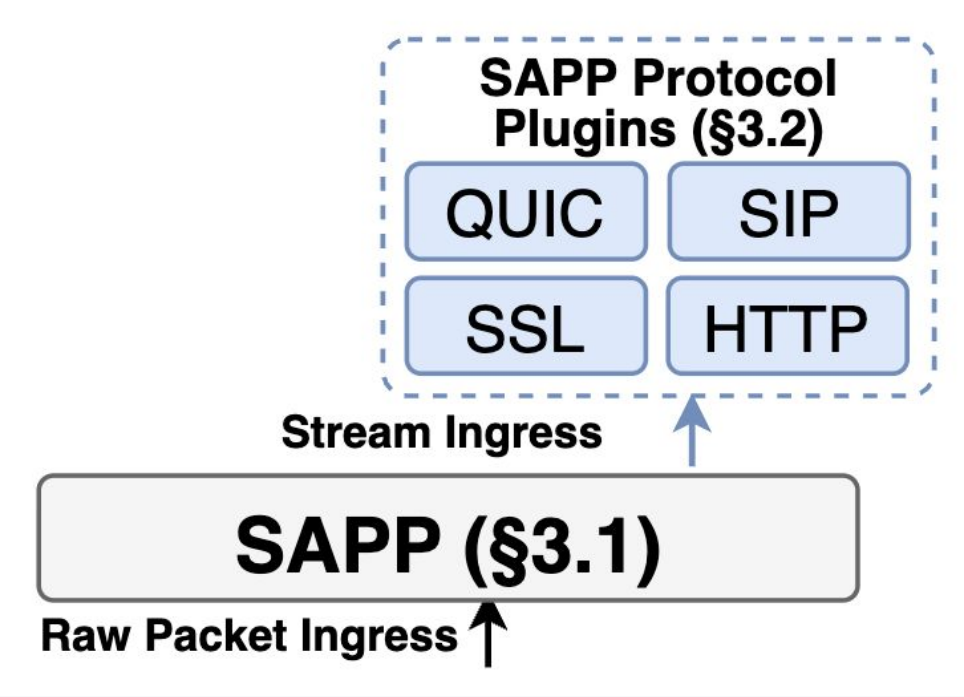
TSG Architecture – SAPP



SAPP: Raw Packet Ingress and APIs

- In development since 2005
- Layers 2–4: Ethernet, IP, TCP/UDP
- Stream reassembly → hands off to plugins
- Exposes APIs for packet drops and injections

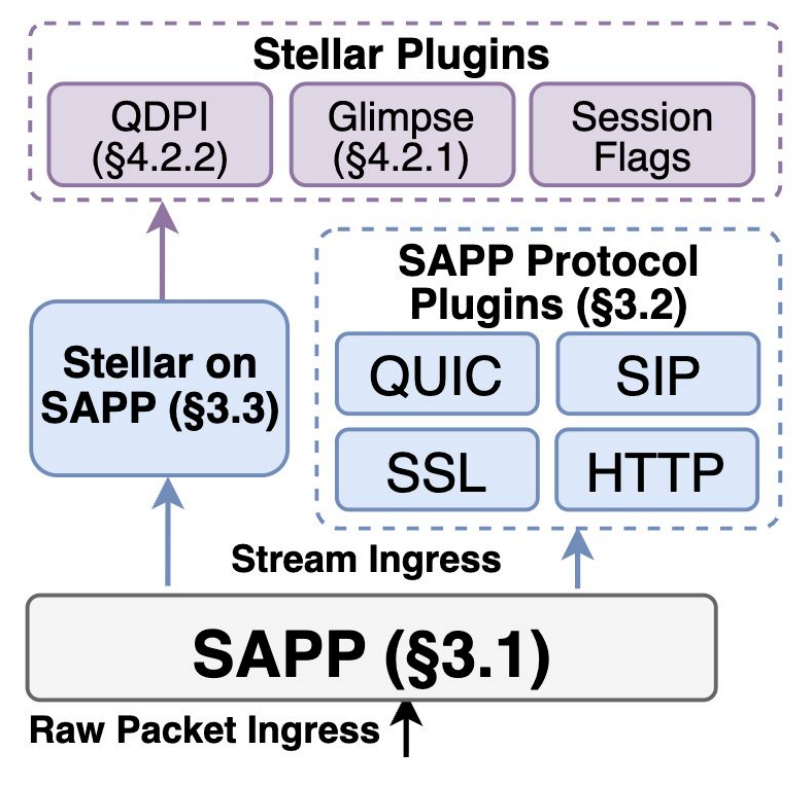
TSG Architecture – SAPP Protocol Plugins



SAPP Protocol Plugins: Stream Ingress

- Perform protocol detection and attribute extraction
- Ex. SSL plugin extracts:
  - Server Name Indication
  - Cipher Suites
  - Extension Lengths
  - Certificate fields
  - Computes JA3/JA4 hashes

TSG Architecture – Stellar & Stellar-on-SAPP

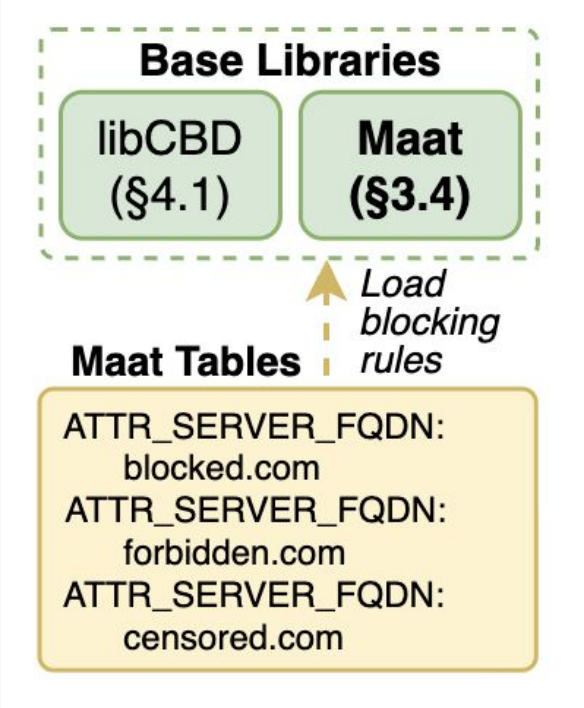


Stellar-on-SAPP: Stream Ingress

- SAPP plugin that converts SAPP streams → Stellar sessions
- Stellar: new, separate platform
- Consumes stream data directly from SAPP
- README self-describes as "transition solution"

Censored Planet

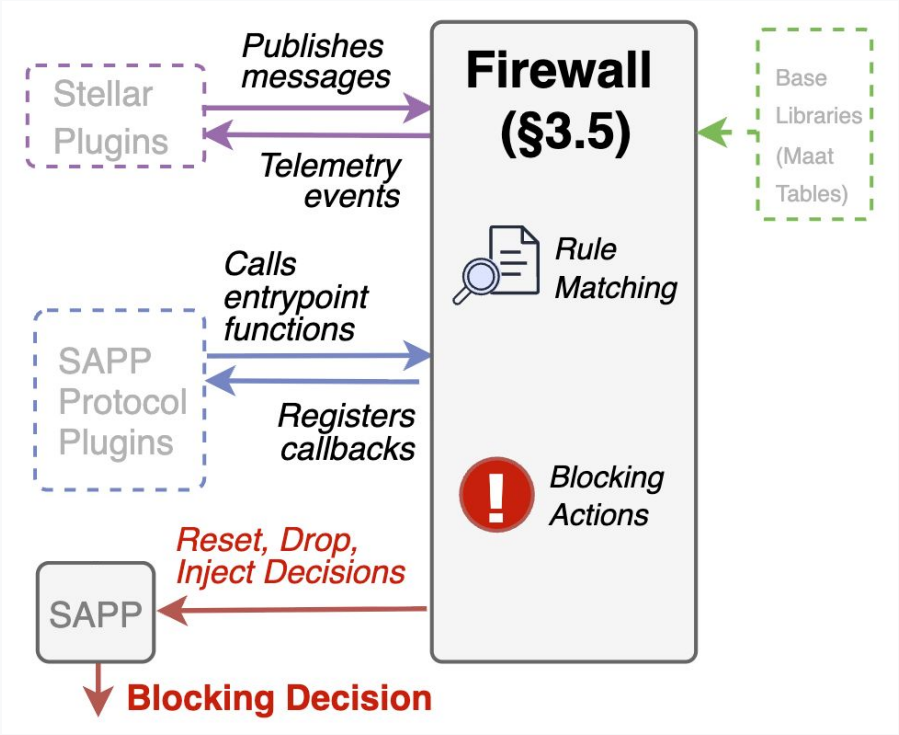
TSG Architecture – Maat



Maat: Rule Setting Engine

- Internal description: "unified description framework for network flow processing configuration"
- Plugins invoke Maat to scan attributes → receive blocking decision
- Rule format: action + sub-action + condition
- Wide attribute vocabulary: addresses, ports, ASN, country, SNI, tunnel metadata, raw payloads

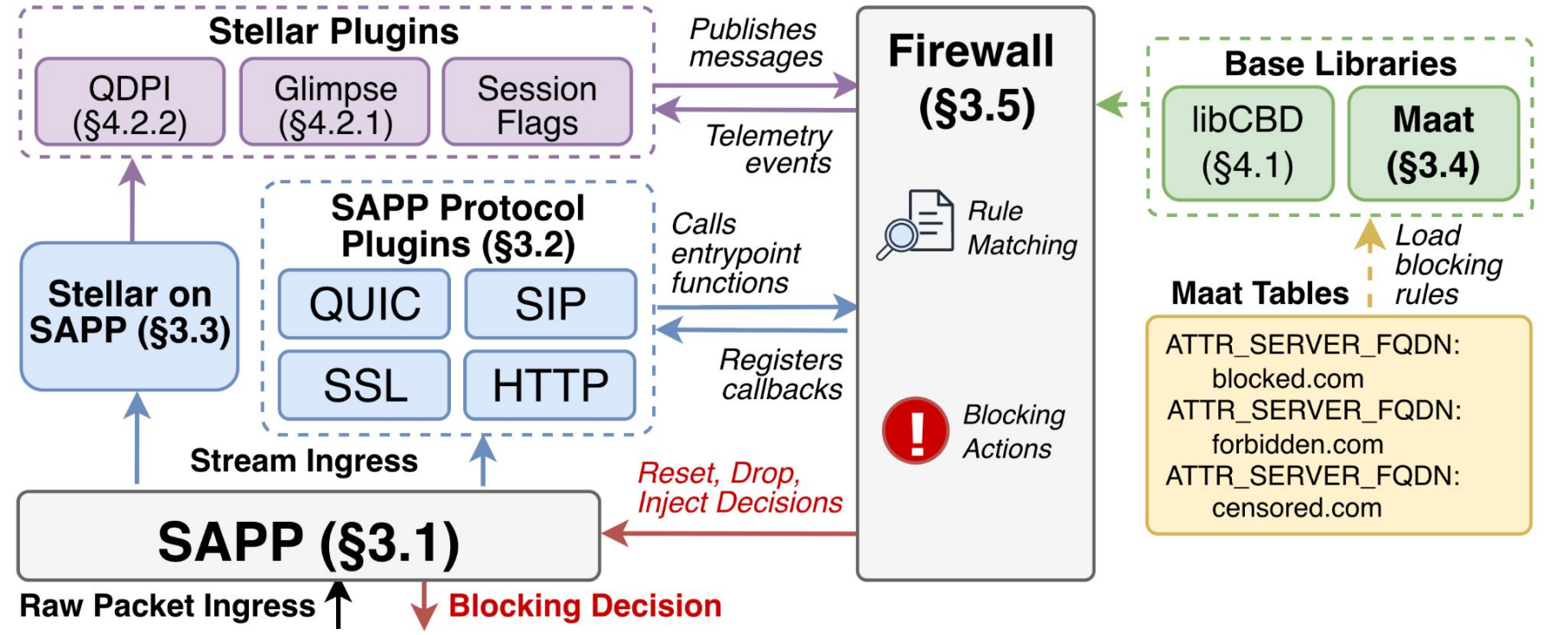
TSG Architecture – Firewall



Firewall: Administering Blocking

- Operates as both a SAPP and Stellar plugin simultaneously
- Receives parsed attributes from both platforms
- Scans attributes against Maat rule tables
- Applies blocking decisions via SAPP stream API

TSG Architecture – Full



Application Detection Systems

In-Hous

Context-Based

Base Library in Firewall

Geedge's own engine, formerly named AppSketch.

Full classification is costly to build from scratch.

Third-Party Code

Glimpse

FREE SOFTWARE

Stellar Plugin

Wraps libprotoident, a free software project from the University of Waikato

LibtraceTeam / libprotoident

nDPI

QDPI

COMMERCIAL SOFTWARE

Stellar Plugin

Calls into Qosmos ixEngine, a commercial DPI library

```
/* Qosmos ixEngine header */
#include <qmdpi.h>
```

ENEA Qosmos

In-House Signature Development: Risk Levels

VPN 特征文件更新记录 → VPN Signature File Update Log

Risk	Key Contents
5	Atlas VPN, Cyber Ghost VPN, Express VPN, Flash VPN, Hide Me VPN, Hotspot Shield VPN, Ivacy VPN, Nord VPN, Opera VPN, Psiphon-Server, TunnelBear VPN, Turbo VPN, Turbo VPN-Payload, Urban VPN, VPN Unlimited, WARP, Windscribe VPN
4	Gecko VPN, Psiphon-CDN, Psiphon-QUIC, Tor Browser, Ultrasurf VPN
3	Cyber Ghost VPN-Payload, Refraction Networking, WARP-Enhanced

In-House Signature Development: Fingerprints

VPN 特征文件更新记录 → VPN Signature File Update Log

Name	Fingerprint
Express VPN	• SSL JA3 fingerprint match
HideMe VPN	• UDP first client → server payload length == 54 bytes • UDP first client → server payload matches 0x000000016628
Psiphon	• `common.app_id` == 68 or 199 • Server FQDN matches `.com`, `.net`, or `.org` • Matches 1,023 CDN IP ranges • Excludes 56,396 known legitimate FQDNs

Overblocking Concerns: Ethiopia Jira Ticket

【E21 现场】业主在一个策略里对配置多个 VPN deny 包含 psiphon，针对策略验证效果时多个应用存在也被 deny 。业主不满意 VPN deny 策略效果 *[E21 site] Customer configured multiple VPN denys (including Psiphon) in one policy. When validating the policy, several other applications were also denied. Customer is not satisfied with the VPN deny policy behavior.*

In the internal lab env, T9K accessed: tiktok.com/about, bbc.com, edition.cnn.com, nytimes.com.

(1) Test IP not in Psiphon3 Client IP → access OK.

(2) Test IP in Psiphon3 Client IP, server IP not in Top Server IP, SNI not in Top SNI → access denied.

(3) Test IP in Psiphon3 Client IP, SNI in Top SNI → access OK.

Censored Planet

Customer–driven Development: Myanmar Jira Ticket

OMPUB-1372: 【M22 项目】Signal 应用封堵及 LetsVPN/LanternVPN 特征提取需求
[M22 Project] Signal app blocking and LetsVPN/LanternVPN signature extraction requirements

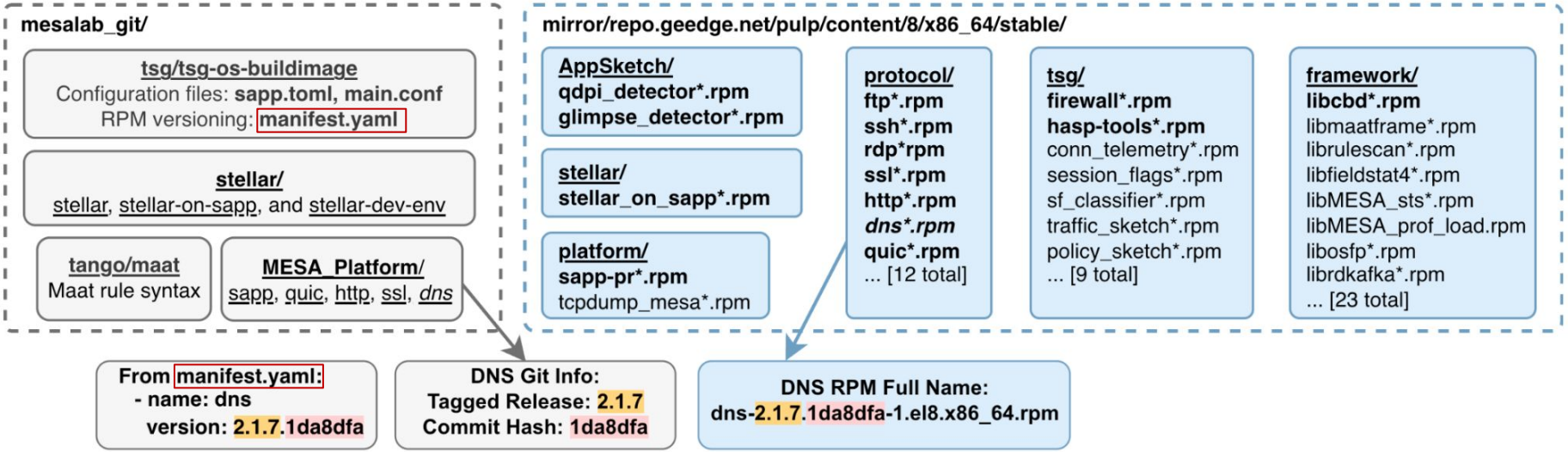
(1) Requirement for blocking the Signal application: when the app enables the "censorship circumvention" option, the blocking is ineffective. Based on experience from the previous Project K, please extract and merge the corresponding signatures. (Higher priority)

(2) LetsVPN signature extraction. (High priority)

+(3) LanternVPN signature extraction: discussed and confirmed with the M client; no specific completion time required (may take a long time). (Low)+

Censored Planet

Building and Running TSG



Censored Planet

Building and Running TSG

Configuring & Running TSG

We configure and run **TSG**, loading relevant modules (DNS, SSL, QUIC, etc...).

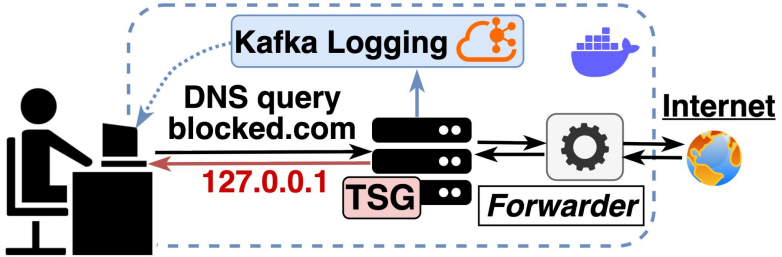
```
sapp use main config file: ./etc/sapp.toml
[MESA_handle_runtime_log_creation], INIT zlog finish,
Using (etc/sapp_log.conf).
Produced by
    __ _____________ ___
   /  |/  / ____/ ___//   |
  / /|_/ / __/  \__ \/ /| |
 / /  / / /___ ___/ / ___ |
/_/  /_/_____//____/_/  |_|
```

```
./plug/protocol/ssl/ssl.so init succ, using [72] us
load SSL success!
./plug/protocol/quic/quic.so init succ, using [855] us
load QUIC success!
./plug/protocol/mail/mail.so init succ, using [1091] us
load MAIL success!
./plug/protocol/http/http.so init succ, using [1329] us
load HTTP success!
./plug/protocol/ftp/ftp.so init succ, using [1840] us
load FTP success!
./plug/protocol/dns/dns.so init succ, using [1320] us
load DNS success!
```

Docker Testbed

We test the **DNS, HTTP, TLS, and QUIC** protocols and **trigger injections, drops,** and **deferred blocking.**





Where is TSG deployed?





Fingerprinting TSG

Looking at the IP, DNS, TLS, and QUIC protocols, we can find unique aspects of TSG's code.



Measurements to Find Real World Deployments

Leveraging the fingerprints, we can scan for real world deployments of TSG.

We can fingerprint idiosyncratic elements of TSG's codebase and scan live networks for deployments.

IP Fragmentation Boundary

Device	Observed Reassembly Boundary
TSG	101
GFW I	500*
GFW II	102
GFW III	0
Henan	0
Kazakhstan	0
Pakistan (RST)	0
Pakistan (DROP)	500*
Myanmar	500*

Note: 500* indicates reassembly was still successful through 500 fragments

Censored Planet

QUIC Permutations

Field	TSG	GFW	KZ	MM	PK
QUIC Version					
GQUIC: Q023–40, 42–43	✗	✗	✗	✗	✗
GQUIC: Q041, 44–45	●	●	✗	●	✗
GQUIC: Q046–49	●	●	✗	●	●
GQUIC: T050–51	✗	●	●	✗	✗
IETF: draft-22–28	✗	●	●	✗	✗
IETF: draft-29, v1	✗	✗	✗	✗	✗
IETF: v2	●	●	●	●	✗
QUIC Features					
Fragmentation	✗	✗	✗	✗	✗
Below 1200 Bytes	●	✗	✗	●	●
Forced Negotiation	●	●	●	●	●
Multi-Packet Initial	●	●	●	●	●
TLS Length Fields					
Message Length	✗	✗	✗	●	✗
Extensions Length	✗	✗	✗	✗	✗
SNI Extension Length	✗	✗	✗	●	●
SNI List Length	✗	✗	✗	●	●

● Bypasses Censor ✗ Blocked

Censored Planet

TLS Length Permutations

	TSG	GFW I	GFW II	GFW III	Henan	Myanmar	Kazakhstan	Pakistan (RST)	Pakistan (DROP)
Record Length	●	●	●	●	●	●	●	✗	✗
Sess. ID / CS Length	●	●	●	●	●	●	✗	●	✗
Comp. Methods Length	●	●	●	●	●	●	✗	✗	✗
SNI Name Length	●	●	●	✗	●	●	▼	●	✗
SNI Ext. Length	●	✗	●	●	▼	●	✗	✗	✗
Message Length	▼	▼	▼	●	✗	●	✗	✗	✗
Extensions Length	✗* / ▼	▼	✗	▼	✗	▼	✗	▼	✗
SNI List Length	✗	✗	✗	✗	✗	●	✗	✗	✗

● Bypasses Censor ✗ Blocked ▼ Varied

DNS Fingerprints

DNS Fingerprints

While RFC 1035 disallows forward jumps, TSG parses incoming DNS queries **up to 17 forward pointer jumps**.

TSG injects DNS responses that use **compression pointers** with the **flags 0x8180**.

```
int get_decompressed_name(char * msg, unsigned char**ptr,
        unsigned char *buf, int buflen, const char *end)

    /* too many pointers. */
    if(np ++ > 16)
        return -1;
```

Forward pointer boundary

```
int set_cheat_pkt_header(dns_hdr_t *dns_hdr)
{
    dns_hdr->qr = 1;        // 1bit: Response
    dns_hdr->opcode = 0;    // 4bits: Query
    dns_hdr->aa = 0;        // 1bit: authoritative answer
    dns_hdr->tc = 0;        // 1bit: Not truncated
    dns_hdr->rd = 1;        // 1bit: Recursion Desired
    dns_hdr->ra = 1;        // 1bit: Recursion Available
    dns_hdr->z = 0;         // 3bits: Reserved for future use
    dns_hdr->rcode = 0;     // 4bits: 0: No error condition
```

Injected DNS response header

DNS Fingerprints

DNS Fingerprints

While RFC 1035 disallows forward jumps, TSG parses incoming DNS queries **up to 17 forward pointer jumps**.

TSG injects DNS responses that use **compression pointers** with the **flags 0x8180**.

↪ **We find correspondence with the GFW's DNS Injector 2.**

Device	Flags	Compression	17-pointer	18-pointer
TSG	0x8180	✓	✓	✗
Iran	0x81a0	✗	✗	✗
CN Inj. 1	0x8580	✗	✗	✗
CN Inj. 2	0x8180	✓	✓	✗
CN Inj. 3	0x8400	✗	✗	✗

TCP RST Injector Fingerprints

RST Metadata Key

TSG's **RST injection** code supports **setting the IPID, TTL, and TCP window size** of the RST packet **according to a seed_key** parameter.

```
static void trick_algo_getrandval_with_seed(
        int thread_id, unsigned sip, unsigned dip,
        unsigned short *ipid, unsigned short *win,
        u_char *ttl, int seed_key, int seed_max_val) {
    int val = randgroup[thread_id];

    if (val * seed_key > seed_max_val) {
        randgroup[thread_id] = 128;
    } else {
        randgroup[thread_id]++;
    }

    *win = (unsigned short)(val + dip % seed_key);
    if (*win == 0)
        *win = 1;

    *ipid = (unsigned short)(
        seed_max_val - val * seed_key + sip % (*win));
    *ttl  = (u_char)(val % 200 + 48);
}
```

Given two packets, can solve for seed_key k

$$k = \frac{(\text{ipid}_i - \text{ipid}_j) + \left((sip \bmod \text{win}_j) - (sip \bmod \text{win}_i)\right)}{\text{win}_j - \text{win}_i}$$

TCP RST Injector Fingerprints

RST Metadata Key

TSG's **RST injection** code supports **setting the IPID, TTL, and TCP window size** of the RST packet **according to a seed_key** parameter.

↪ **We find correspondence with TCP Injectors GFW II and GFW III.**

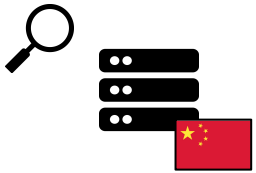
Device	IP DF	TCP Flags	Count	Stable k
TSG	✓	RST+ACK*	1-3	yes ($k = 13$)*
GFW I	✗	RST	1	no
GFW II	✓	RST+ACK	3	yes ($k = 13$)
GFW III	✓	RST+ACK	1	yes ($k = 13$)
Henan	✗	RST+ACK	1	no
Myanmar	✓	RST+ACK	1	no
Pakistan	✗	RST+AE+RE	1	no

For the first time in three decades of anti-censorship research, we have source code running in a national firewall.

Conclusion

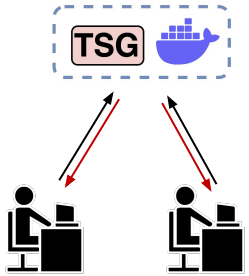
First Nation-State DPI Leak

Tying the leaked code to the GFW, we determine this is the **first leak of code** used in a national firewall.



Live Censorship Instance

We host our **live TSG instance** and make it conditionally available to researchers upon request.



Active Investigation

Visit our project website to learn more about **upcoming research projects** on the leak.

Censored Planet LAB

Technical Analysis of the Geedge Networks Firewall Source Code Leak

Anna Ablove[1], J. Walker[2], B. Wolin[1], N. Niere[3], F. Lange[3], A. Ortwein[1], A. Huremagic[1], R. Priyanka[1], A. Zohaib[4], J. Sheffey[4], N. Heitmann[3], J. Alex Halderman[1], J. Somorovsky[3], A. Houmansadr[4], R. Ensafi[1], M. Wu[2], E. Wustrow[5]

University of Michigan[1], GFW Report[2], Paderborn University[3], UMass Amherst[4], CU Boulder[5]

