SbuSocks: Break the Great Firewall
CSE534 Project

Abstract

SbuSocks is a secure SOCKS5 proxy service designed to circumvent censorship firewalls such as the Great Firewall (GFW) of China. To achieve the circumvention, hosts inside the firewall first find some proxy hosts outside the firewall. The proxy hosts act as relaying nodes, which relays the encrypted traffic. We implemented the SOCKS5 protocol to let application connect to client both at the same local host, and local client connect with to remote proxy server. A host's traffic does not travel to the real destination directly, but travel to the proxy host first and then the proxy host forwards the traffic to the real destination. The traffic between local client and remote server is encrypted, so that GFW cannot easily inspect the content of the traffic. Our experiments let 8 friends in 5 cities across China successfully broke the GFW and our SbuSocks server remained unblocked.

1 Introduction

Problem statement: Our project is motivated by serious problems that exist in the real world, several countries censor their people's access to Internet by building powerful firewalls. The most typical one is the Great Fire Wall of China (GFW) established by Chinese government. Thus Chinese have no access to Google, Facebook, twitter etc. which are taken for granted in all western countries. Besides these popular sites, GFW also blocks all political sensitive contents for the sake of regime stability and porn websites for the sake of people's mental health.

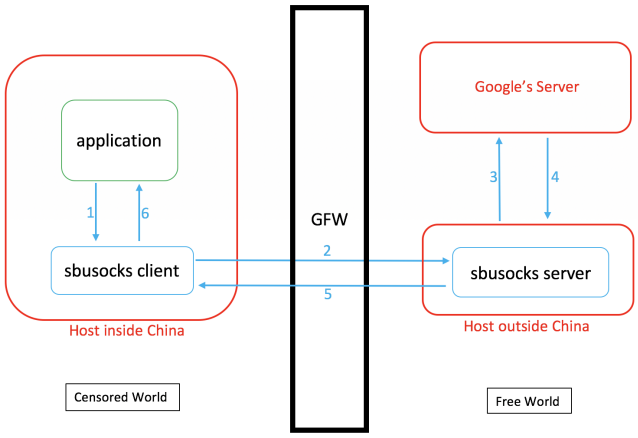


Figure 1: SbuSocks' architecture.

Since the beginning of GFW in around 2000, people fight back by building circumvention tools, such as Freegate, Tor, VPN, and shadowsocks. The arms race continues and GFW keeps evolving as more and more new circumvention tools emerge. **Approach: The basic methodology of circumvention tools is to find some proxy nodes and encrypt the traffic**. To understand why this basic methodology works, we need to know how GFW works.

GFW now has many weapons, three of which are the most classic: IP blacklist, DNS injection and TCP RST with deep packet inspection. GFW maintains an IP blacklist and it peers with all China's international gateways. GFW use BGP

routing to broadcast the blacklist, and all China's international gateway routers will have the blacklist. Then international gateway routers execute null routing which simply drop packets that contain an IP matches to one in the blacklist. DNS injection is forging a faked DNS reply, and this faked DNS reply will reach the local DNS resolver before the real DNS reply reaches local DNS resolver. So the resolver will receive the faked DNS reply first. Therefore, the DNS requester will not access the real destination. The above two methods only use the headers of the packet, while deep packet inspection (DPI) look deep into the payload of the packets. If GFW find some sensitive contents inside the payload then it will launch a TCP RST attack, which breaks the TCP connection.

We implement *SbuSocks*[6] which defeats the three classic weapons above, absorbing ideas from Shadowsocks[1]. SbuSocks's architecture is illustrated in Figure 1. SbuSocks has a server side and a client side. The server side runs outside the firewall and has a IP not in the GFW's blacklist. The client side runs on a machine inside the firewall that attempt to break the firewall. The server side acts as a relaying node, which receives traffic from the machine inside the firewall and forward to the real destination. By implementing SOCKS5 protocol, local application completes a handshake with the local sbusocks-client, then sbusocks-client completes a handshake with the remote sbusocks-server. The handshake phase build a connection between local application and sbusocks-client, and a connection between sbusocks-client and remote sbusocks-server. When an application on a machine inside the firewall attempts to connect Google, the traffic is first redirected to the local sbusocks-client(step 1 in Figure 1) on the same machine. Then sbusocks-client forward the traffic to sbusocks-server(step 2), and finally sbusocks-server forward traffic to the real destination(step 3). The reply from the real destination(Google) first goes to sbusocks-server(step 4), then sbusocks-server forward traffic to sbusocks-client(step 5), and finally application receives traffic from sbusocks-client(step 6).

This relaying system elegantly defeats the IP blacklist weapon, since sbusocks-server has an IP not in the GFW's blacklist. It also defeats DNS injector weapon because DNS resolver is a part of sbusocks-server, which is in the free world and thus not affected by GFW. To defeat the TCP RST attack with DPI, we encrypt the traffic between sbusocks-client and sbusocks-server, so GFW cannot easily inspect traffic's content. We launched SbuSocks on March 15, 2018. We started testing on May 4th, 2018. By May 10, 2018, our experiment let 8 friends in 5 cities across China breaks the GFW and access to the free world. During the period, our sbusocks server remained unblocked.

contributions of this report are two-fold:

1. We built SbuSocks that successfully breaks the GFW by defeating GFW's three classic weapons mentions above without being blocked by GFW.

2. We helped some friends in China break the GFW and found out that GFW will let very small scale suspicious traffic go, due to economic concerns.

2 Related Works

Virtual Private Network(VPN). VPN is usually used to do two things: connect two machines as LAN or to avoid government controlled networks. VPN works at lower level, IP level. A VPN is created by establishing a virtual point-to-point connection through the use of dedicated connections, virtual tunneling protocols, or traffic encryption. To avoid detection, first step is to implement VPN server on VPS. All the data from client will go though virtual network connection and encrypted to get to targeted server. For example, if a client want to connect to Google his/her packet will first go though VPN protocol encrypted as IP packet with source IP of virtual IP. Then packet will be send to VPS, and connection will be established by VPS and Google. Then VPS establish connection with client to output the result.

Shadowsocks is an open-source program based on SOCKS5 protocol. Shadowsocks uses encrypted proxy. And every Shadowsocks users can configure their proxy. So it makes hard to detect by GFW. VPN is more like freight forwarding, while Shadowsocks is more like shipping to a friend then friend re-ship to real target destiny.

Tor is another example. The goal of Tor is anonymous communication. Tor stands for "the onion router", the original software program. Tor achieve its goal by transmitting multiple time. First the packet of Tor client is transmitted by sending it to volunteer A. Then volunteer A peels off the header find the next volunteer B is address and send it to volunteer B. Then volunteer B peels off the header and find volunteer C is address. And so on. The process is like an onion, hence "the onion router". The Tor works perfectly to stay anonymous. But Tor doesn't not hide the fact that the user is using Tor. So GFW just block all Tor connection.

In recent years, GFW developed three more advanced weapons: active probing[3], anti-VPN machine learning, and great cannon [7]. The three classic weapons is not enough anymore since circumvention tools are evolving.

3 Background

3.1 SOCKS5 Protocol

Socket Secure (SOCKS) is an Internet protocol that exchanges network packets between a client and server through a proxy server. A SOCKS server can proxy TCP connections to an arbitrary IP address[8]. SOCKS5[5] add an authority mechanism, which only allows authorized clients using proxies. The pipeline of Socks5 is as follows:

1. The client sends real destination to the server.
2. The client sends real traffics to the server.
3. The server forwards traffics to the real destination and get responses.

VER	NMETHODS	METHODS
1	1	1-255

Table 1: Connection request. The second row are payload sizes in byte.

VER	METHOD
1	1

Table 2: Initiation response. The second row are payload sizes in byte.

4. The server forwards the responses to the client.

This procedure can give the client access to all hosts that the server can connect to. And the concealment of real destinations makes this protocol extremely suitable for circumventing censorship.

3.1.1 Packet Format

Initiation Phase. In the initiation phase, the client sends a connection request to the server as Table 1. For SOCKS5, the VER segment is $X'05'$. And in this project, we set METHODS segment as $X'00$, which means NO AUTHENTICATION REQUIRED.

The server needs to choose a method that the client supports, and then response with a packet as Table 2.

Connection Phase. In the connection phase, the client send the real destination to the server, the request packet is as Table 3.

For TCP connections, the CMD field is $X'01'$, which means CONNECT. The address can be an IP address or a domain name, which is differentiate in the ATYP field: IP V4 address - $X'01'$, DOMAINNAME - $X'03'$ and IP V6 address - $X'04'$.

VER	CMD	RSV	ATYP	DST.ADDR	DST.PORT
1	1	$X'00'$	1	Variable	2

Table 3: SOCKS request. The second row are payload sizes in byte.

VER	REP	RSV	ATYP	BND.ADDR	BND.PORT
1	1	$X'00'$	1	Variable	2

Table 4: SOCKS response. The second row are payload sizes in byte.

After the server receive this message, it will try to build a TCP connection to the real destination, and then return a response to the client as Table 4. The REP field indicates the connection status: succeeded - $X'00'$, host unreachable - $X'04'$, etc.

Streaming Phase. In the streaming phase, the server will forward whatever TCP packets from the server to the real destination and forward whatever TCP packets from the real destination to the client, until the connections close.

3.1.2 Application Supports

Nowadays, there are a lot of application support SOCKS5 protocol. For example, with SwitchyOmega[1] chromium extension, we can use chrome to connect a socks5 proxy server. Proxifier[2] can even take over all local network traffics and let them stream under SOCKS5 protocol.

3.2 Traffic Obfuscation

Firewalls may use DPIs to resolve contents of packets. So it is important to hide real contents to bypass censorships. A nature way is to use encryption protocol, such as SSL. But DPI can detect which protocol is used and intercept certificates. Although obfuscation is some kind of encryption, the real purpose is to make firewalls have difficulties in distinguishing obfuscated traffics from normal traffics.

There are mainly three kinds of obfuscation methods [2]:

- **Randomization**: randomize every byte in the packet payload.
- **Mimicry**: masquerade as a whitelisted protocol.
- **Tunneling**: Use a special protocol, such as VPN.

4 Our Solution: SbuSocks

Our codes can be found on GitHub[6].

4.1 Architecture

The SbuSocks' architecture is already depicted in Figure 1. The client is running on the local PC, and the server is running on a remote host whose IP is not in the blacklist of the GFW. The traffics between the client and the server are obfuscated.

4.2 Implementation

4.2.1 Skeleton

The SOCKS5 protocol of servers are separated into two parts, and deployed in the SbuSocks client and the SbuSocks server respectively. In the view of the local application, it cannot see this architecture, only treats the SbuSocks client and the SbuSocks server as a whole part, as a virtual server in SOCKS protocol. The client open a listening to receive traffics from the local application and the server open a listening port to receive traffics from the client.

Initiation Phase. In the Initiation phase, the local application supported SOCKS5 initiates an connection request to the SbuSocks client. This request is automatically generated by the local application. Then the SbuSocks client sends a response response to the local applications.

Connection Phase. In the connection phase, the local application sends a SOCKS5 request to the SbuSocks client, and the client forwards this request directly to the SbuSocks server. The SbuSocks server parses the packet and gets the real destination. The server tries to build a TCP connection to the real destination and return a

[1] `https://github.com/FelisCatus/SwitchyOmega`
[2] `https://www.proxifier.com`

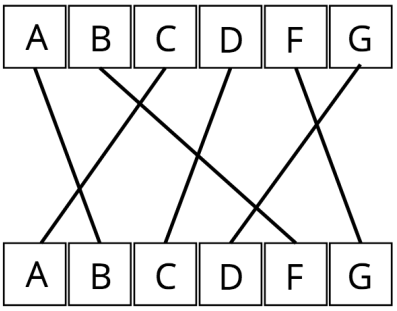


Figure 2: Symmetric randomization.

SOCKS5 response according to the connection status to the client. The client then forward the response to the local application.

Streaming Phase. In the streaming phase, the SbuSocks client forwards whatever packets from the local application to the SbuSocks server and whatever packets from the SbuSocks server to the local application. The SbuSocks server forwards whatever packets from the SbuSocks client to the real destination and whatever packets from the real destination to the SbuSocks client.

4.2.2 Enable Obfuscation

The traffics between the client and the server are obfuscated. We use the simplest way to encrypt messages: symmetric randomization.

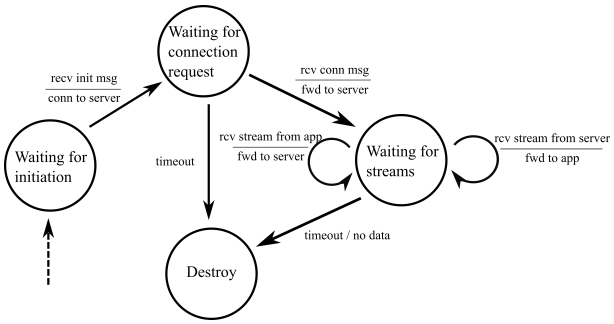
The server and the client have the same key, which is set manually and is fixed during communications. We first map the key to an integer, then we set a random seed using this integer and then we use this random state to shuffle $0-255$. The result is a bijective map from $0-255$ to $0-255$, we can use this key map to encrypt and decrypt data. This method is illustrated in Figure 2.

There are 256! kinds of different bijective maps from $0-255$ to $0-255$, so if the censor doesn't know the key, it is difficult to get access to real contents of data.

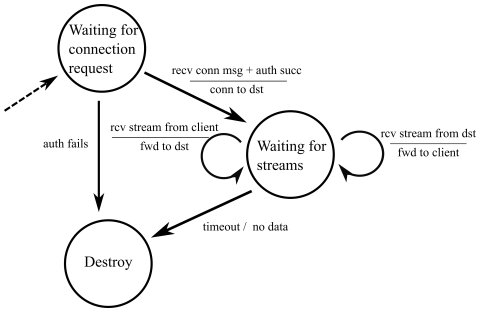
4.2.3 Enable Identity Authentication

If someone send a packet which is not encrypted by the key, we should drop the packet. So we can add some identity authentication procedure.

We propose a simple authentication method: In the connection phase, the client inserts the hashing of the key ahead of the SOCKS request before encryption. After the server decrypts this encrypted message, it compares the hashing of the key with the head of the decrypted message. If the authentication fails, the server will close the connection and reply nothing.



(a)



(b)

Figure 3: (a) Finite-state machine for the client; (b) Finite-state machine for the server

4.2.4 Put All Together

The whole procedure can be illustrated as two finite-state machines: one for the client and one for the server, as shown in Figure 3.

Note that the figure just illustrates one loop, that is, each new connection will create a new finite-state machine.

5 Experiments

5.1 Experiment Setup

SbuSocks project is launched on March 15, 2018. We rent a VM by cloud provider namely vultr, and a VM by Google Cloud. We started testing on May 4th, 2018 by giving our GitHub repository links to friends in mainland China and guide them run the sbusocks-client program and configure SwitchyOmega. We connected 8 friends in 5 cities across mainland China to use our SbuSocks client program. The 5 cities are: Beijing, Hangzhou, Qingdao, Shenzheng, and Wuhan.

We have two sets of experiments: relaying through a proxy node with traffic obfuscation enabled, and relaying through a proxy node with traffic obfuscation disabled.

5.2 Obfuscation Enabled

This set of testing uses the full version of SbuSocks, with both relaying proxy and obfuscation enabled. Our hypothesis is that friends in China can successfully break the GFW with sbusocks server unblocked. The testing result shows that our hypothesis is true. All of our friends happily watches YouTube. And to our sbusocks server is safe and sound.

5.3 Obfuscation Disabled

This set of testing uses an incomplete version of SbuSocks, with merely relaying proxy and no obfuscation. Our hypothesis is that we fail to break the GFW, or we can break the GFW for only a few minutes and then blocked by GFW. To our great surprise, our hypothesis came out false. Friends in China can still break the GFW for a long period until they shut down the computer and go to sleep. And our sbusocks server remained unblocked during the period.

SbuSocks	Hypothesis	Result
Proxy Relay + obfuscation	Break GFW	Break GFW
Proxy Relay	Blocked	Break GFW

Table 5: Summary of experiment results.

5.4 Explanation of the Experiment Results

Table 5 summarizes the result of our experiments. In the first experiment, the hypothesis meets the results, so our hypothesis is true. This means that the full version of SbuSocks can successfully circumvent GFW and defeats all three classic weapons mentioned in section one. In the second experiment, the hypothesis does not meet the results, so the hypothesis is false. We went through many blogs and literature, and found an explanation[4] that makes sense. The idea is that the traffic at the international gateway is so large that GFW does not has enough machine computing resource to do deep packet inspection on all traffic. According the 80/20 principle, GFW will focus their machine and human resources on the most suspicious 20 percent traffic. We guess that our traffic is too small to draw GFW's attention. If there is a gray list, we might end up in their gray list, but never enter the blacklist.

6 Future Work

TCP connection will terminate when either server or client send a TCP RESET packet. TCP RST attack is when a forged TCP RST packet is send to both server and client to force them close the connection. One major way GFW perform blocking is using TCP RST Attack. TCP RST Attack is triggered by keywords,such as facebook, twitter, etc. Once GFW detect that the packet contain such keyword, it will send TCP RST to both end of the TCP connection. One way to solve this problem is to send an ACK to server/client after receive RST to ask of confirmation. If RST is send by client/server, then connection ends; if not, RST will be dropped. However, current detecting sys-

tem need both server and client implementation.

7 Conclusion

In this project, we implement SbuSocks[6], a proxy tool to circumvent censorship firewalls, absorbing ideas from shadowsocks[1]. Basically, it is an extended version of SOCKS5 proxy which supports encryption. But the purpose of encryption is to obfuscate real traffics, not encryption itself. This framework composed of the server and the client, they connect with each other and form a bridge for local applications to communicate with the outside world.

In our experiments, we invited 8 friends across 5 cities in China to test our tools, they all succeeded to watch Youtube videos, search in Google, etc. What we find interesting is that when we disable obfuscation, the GFW didn't block our server. We guess it is because our traffics are so small such that it cost too much to confirm our intentions.

References

[1] CLOWWINDY, CYFDECYF, MADEYE, LINUSYANG, AA65535, AND LIBREHAT. Shadowsocks - a secure socks5 proxy. https://www.shadowsocks.org/en/index.html.

[2] DIXON, L., RISTENPART, T., AND SHRIMPTON, T. Network traffic obfuscation and automated internet censorship. *IEEE Security Privacy 14*, 6 (Nov 2016), 43–53.

[3] ENSAFI, R., FIFIELD, D., WINTER, P., FEAMSTER, N., WEAVER, N., AND PAXSON, V. Examining how the great firewall discovers hidden circumvention servers. In *Proceedings of the 2015 Internet Measurement Conference* (New York, NY, USA, 2015), IMC '15, ACM, pp. 445–458.

[4] FQROUTER. Guesses about how gfw blocks proxy servers. http://fqrouter.tumblr.com/post/45969604783/%E5%85%B3%E4%BA%8Egfw%E6%98%AF%E5%A6%82%E4%BD%95%E5%B0%81%E7%BF%BB%E5%A2%99%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%9A%84%E7%8C%9C%E6%83%B3.

[5] LEECH, M., LEE, Y., KURIS, R., KOBLAS, D., AND JONES, L. Rfc1928. https://www.ietf.org/rfc/rfc1928.txt.

[6] LI, X., ZHAN, C., AND XIANG, H. Sbusocks. https://github.com/caitaozhan/CSE534-Project.

[7] MARCZAK, B., WEAVER, N., DALEK, J., ENSAFI, R., FIFIELD, D., MCKUNE, S., REY, A., SCOTT-RAILTON, J., DEIBERT, R., AND PAXSON, V. An analysis of china's "great cannon". In *5th USENIX Workshop on Free and Open Communications on the Internet (FOCI 15)* (Washington, D.C., 2015), USENIX Association.

[8] WIKIPEDIA CONTRIBUTORS. Socks. https://en.wikipedia.org/wiki/SOCKS.