SoK: Active Probing and Fingerprinting Techniques

Gaukas Wang
Gaukas.Wang@colorado.edu
University of Colorado Boulder
Boulder, Colorado, USA

ABSTRACT

Strong censoring parties not only block banned websites and servers, but also tools used to bypass censorship. Evidence shows that censors, including the Great Firewall of China, have been using advanced techniques such as active probing and deep packet inspection to detect and block the use of circumvention tools, focusing on finding and exploiting discrepancies between hidden circumvention services and benign/allowed ones.

In this paper, we study well-known active probing and fingerprinting techniques against popular circumvention solutions, analyzing each technique for its efficacy and potential collateral damage. We also review popular countermeasures against such techniques and attacks, discussing the strengths and weaknesses of each.

Based on our analysis, we draw conclusions on the key criteria that a censorship circumvention solution needs to meet in order to be invulnerable against active probing and fingerprinting attacks. Lastly, we comment on possible future research directions on the general fingerprintability of circumvention tools.

CCS CONCEPTS

• **Security and privacy** → **Pseudonymity, anonymity and untraceability**; • **Networks** → *Network security*; • **Social and professional topics** → *Censorship*.

KEYWORDS

active probing, fingerprinting, censorship circumvention, Great Firewall of China

ACM Reference Format:
Gaukas Wang. 2023. SoK: Active Probing and Fingerprinting Techniques. In *Unpublished Thesis, May, 2023, Boulder, CO*. ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

As internet censorship continues to escalate worldwide, the development and use of circumvention tools has become a crucial strategy for individuals and organizations seeking to bypass restrictions on online content. However, censors have also intensified their efforts to detect and block such tools. In the early days, tunneling tools used for censorship circumvention were easily blocked using Deep Packet Inspection (DPI)-based filters. But as more sophisticated tools were developed to emulate popular protocols and applications, DPI-based filters became less effective.

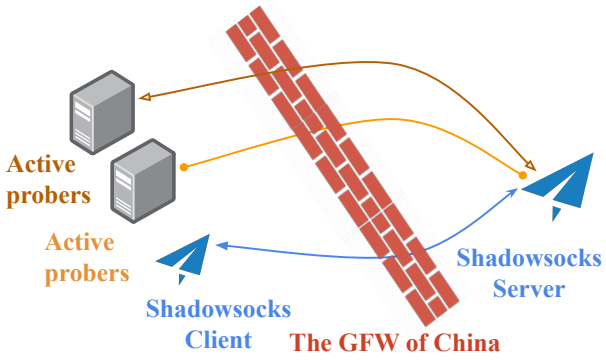

Figure 1: A figure from How China Detects and Blocks Shadowsocks [1] demonstrating how active probing against Shadowsocks Servers works.

One prominent example is Shadowsocks [5], a popular circumvention tool in China that was believed to be unblockable due to its encrypted and pattern-less proxy design. However, the Chinese national firewall (GFW) has been able to detect and block Shadowsocks servers by using active probing and fingerprinting techniques as revealed by Alice et al. [1]: The GFW actively probes suspected Shadowsocks servers by sending multiple types of payloads and comparing the server's response with known fingerprints of Shadowsocks servers, with the topology shown in Figure 1. By doing so, the GFW can minimize the potential collateral damage of blocking innocent servers producing circumvention-like traffic.

Shadowsocks is not the only censorship circumvention protocol prone to active probing and will not be the last circumvention tool blocked by the GFW. Reports show that many circumvention tools/protocols could be identified with some sort of active or passive techniques. And it might explain why are we constantly seeing these circumvention tools getting blocked by censors.

This paper provides a comprehensive overview of the current knowledge and intelligence related to active probing and fingerprinting vulnerabilities of censorship circumvention tools. We revisit and summarize the methods used by strong censors to detect popular circumvention tools using active probing and passive fingerprinting techniques. We also identify design flaws in common circumvention tools that contribute to active probing or other fingerprinting vulnerabilities through case studies on a few popular tools. Furthermore, we evaluate countermeasures that can reduce the risks of active probing and fingerprinting. Based on this knowledge, we establish criteria for a probe-resistant circumvention tool

Unpublished Thesis, May 2023, Boulder, CO

ACM ISBN 978-1-4503-XXXX-X/23/05…$0.00
https://doi.org/XXXXXXX.XXXXXXX

and discuss possible future research paths that could benefit the circumvention community in terms of reducing fingerprintability.

2 BACKGROUND

2.1 Censorship and Circumvention

Censorship and circumvention have been two opposing forces in the field of online communication and information access. Censorship refers to the practice of restricting or controlling access to information or content by governments, organizations, or other entities. Circumvention, on the other hand, involves using various techniques to bypass censorship and gain access to restricted content.

Censorship mechanisms are often deployed against content providers, such as websites or social media platforms, and can be relatively straightforward. With the limited number of well-known providers and their static server IP addresses and domain names, censorship can be achieved using techniques such as DNS pollution, IP blocklists, connection hijacking, and others.

However, circumvention tools are in a more complicated situation for censor. While there are limited kinds of popular circumvention tools, open-source tools like *OpenVPN* [19], *Shadowsocks* [5], and *Trojan* [31] could be self-hosted by users on cloud providers at a relatively low cost. Some centralized providers have also built decentralized tools that connect users to anonymous peer-to-peer networks [20, 29], greatly reducing the effectiveness of static blocking techniques based on known IP addresses.

2.2 Protocol Filtering

Protocol Filtering is one of the simplest Passive Analysis techniques have been used by many censors to detect and block circumvention tools without even needing to establish a connection with either party. For example, the Great Firewall of China (GFW) has installed deep packet inspection (DPI)-based filters in their international outbound/inbound routes to detect and block some popular tunneling protocols such as PPTP/L2TP, IPsec, and the vanilla OpenVPN protocol [36]. These protocols come with plaintext headers telling censors their identity, and are exclusively used for tunneling purposes. The GFW targets such strong indicators of these tools and terminate connections based on which and may even block future connections to the same target address for a given duration [1, 23]. Overall. technique is considered trivial as most popular communication protocols have unique patterns and structures in their packets. Moreover, some protocols even include protocol-specific headers in plaintext, exposing more information to censors.

Protocol Filtering involves matching captured packets or multiple packets extracted from a captured connection to the corresponding protocol. For example, in the case of Secure Shell (SSH) [24–28], as shown in Figure 2 (obtained from [37]), the SSH client and server exchange their protocol and software versions in plaintext during the first roundtrip. More plaintext information is exchanged in the following roundtrips until the client sends its first encrypted service request. Once the adversary intercepts all plaintext communications, they can easily reveal the protocol used in this connection (SSH in our example) since it is a well-known protocol and includes sufficient details in plaintext. Packet analyzers such as Wireshark [35] also employ this technique.

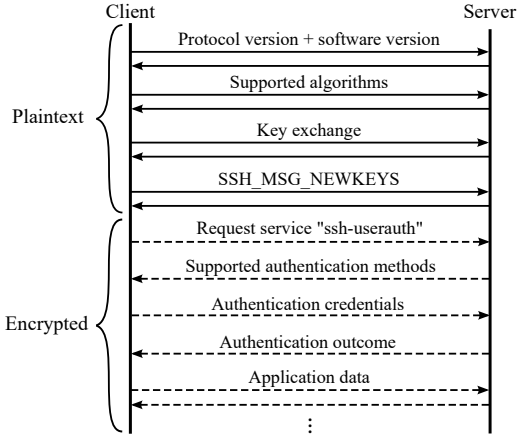

Figure 2: A flow diagram for SSH protocol obtained from [37]

Protocol Filtering could be performed by both OB and OM censors, and even some lower level censors such as local or institutional ones. Censors can widely deploy this technique to build protocol allowlists or blocklists with reasonably high confidence/accuracy.

2.3 Previous Attempts against Fingerprinting

In response to DPI-based protocol filtering techniques, circumvention developers and researchers have proposed various countermeasures. One such approach is *Parroting*, in which a circumvention tool behaves like a benign application to mimic the implementation of the target application, effectively reducing fingerprintability at the protocol level. This approach has been used successfully by many popular circumvention tools, including Tor, Signal, V2Ray, and Trojan, which use **Transport Layer Security (TLS)** as one of their default transports. TLS provides encryption and guarantees integrity and authenticity by default.

However, as Houmansadr et al. pointed out, parroting may increase fingerprintability at the implementation level since it is challenging to create a perfect parrot that mimics all aspects of the target application's implementation [17]. In practice, this may result in a vulnerable parrot that can be detected and blocked by censors. Different implementations for the same protocol sometimes come with different observable features, which are usually unique and exclusive to one implementation or one specific version of the implementation. Failure in copying all observable features may leads to detection vulnerabilities that are prone to more advanced **Passive Fingerprinting** techniques to be introduced in next section. We will also discuss common mistakes that can lead to a vulnerable parrot later in this paper.

Another approach used by some circumvention tools is to completely encrypt their traffic. For example, Shadowsocks encrypts all its traffic, making it difficult to identify based on content alone. In this approach, the communication intercepted by censors becomes fully encrypted, and only the server and client with the pre-shared secret can read the plaintext. However, this approach has some

disadvantages, including increased latency due to the overhead of encryption and the need for a pre-shared secret.

In summary, while parroting and complete encryption are both effective at reducing fingerprintability, they have their own trade-offs and limitations. The development of more effective and efficient countermeasures against passive fingerprinting techniques remains an active area of research.

3 ADVERSARY MODEL

The adversary, in this case, refers to state-level censors who may employ various techniques to fingerprint unknown targets. Based on recent reports about strong censors, we assume two different situations for the adversary: an oblivious censor (**OB**) with limited resources and control over the internet, or an omniscient censor (**OM**) with ample processing power and storage capacity to perform DPI at line speed for an arbitrary duration (both names are from [17]). The OM censor can perform Passive Fingerprinting by recording all traffic for cross-packet/cross-connection analysis offline, and control active probers for Active Fingerprinting. It can also spoof or hijack any IP address for active probing. On the other hand, an OB censor cannot spoof IP addresses and can only perform passive analysis on each connection individually.

In this section, we discuss more about fingerprinting approaches that may be used to detect circumvention tools.

3.1 Implementation Analysis

Implementation Analysis is another popular passive fingerprinting technique and builds on top of **Protocol Filtering** introduced in Section 2. Instead of targeting protocols, it targets sub-protocol level differences existing between different implementations, since each implementation of a single protocol comes with its own flavor, and leads to minor differences in detailed, application-specific data. In general, the more loosely-defined a protocol is, the more likely that strong fingerprintability will come with its various implementations due to different interpretations by developers.

For example, in the Transport Layer Security(TLS) protocol, a client starts the handshake process with a server by sending a `ClientHello` message the server. The ClientHello message advertises TLS specs the client supports, including but not limited to the TLS protocol version, cipher suites, and TLS extensions for extended functionalities. However, different implementation may have their different preferences on multiple settings and implement unique subsets of all possible features defined by the RFC documentations [7, 8, 30], which leads to fingerprintability. Frolov and Wustrow had collected such so-called TLS fingerprints based on ClientHello and provided a website showing all seen fingerprints of TLS protocol [15]. According to the site [14], popular web browsers, such as Google Chrome, Mozilla Firefox, Apple Safari, all come with different fingerprints. And popular TLS libraries in every programming languages have their unique TLS fingerprints as well.

Even though there are no signs of censor massively using this technique to directly identify and block protocols, research has indicated that some specific TLS fingerprints from Tor trigger GFW's active probing [10, 23] and it is believed to be how GFW detects and block hidden Tor bridges. Thus, this technique could benefit OM censors much more than OB censors. The collateral damage for this technique is quite obvious: if another application happen to use the same set of fingerprintable features, then it will be blocked as well. Some ISPs in Iran has allegedly blocked the TLS ClientHello of Chrome 107, which was mimicked and reused by multiple popular circumvention tools.

3.2 Active Probing

While all the adaptations mentioned in Section 2.3 render passive techniques less effective and maybe not enough for censors to detect and block circumvention tools, some strong censors have evolved towards a much more radical ideology which was named *Proactive attacks* in [17] but later referred to more as **Active Probing** or **Active Fingerprinting** by developers and researchers in the field of censorship. Instead of leaving no trace, active probing techniques utilize hosts on the Internet as *probers* to proactively make connections to suspected target hosts and try to identify the unknown entity by sending specifically designed requests and comparing the response for each request to known/expected responses from well-known implementations of benign applications or circumvention tools. Such correlation between expected responses and certain requests also defines the *fingerprint* of each implementation from a different point of view.

Different probes need to be designed for each implementation/protocol, and a possible probe could be either a normal request or even a bad request. Sometimes, a bad request could more efficiently tell the difference between implementations, due to the error handling is usually the gray area where no strictly defined behavior exists and no consensus has been reached among developers. For example, to identify a web server configured to show the same index, a request for `/` will not be really helpful since the same content will be loaded. However, when a random path that does not exist on any of the servers is requested, the error page for different implementation would usually disclose the information of the web server's identity.

And based on its trigger, active probing could be categorized into two categories, *reactive probing* and *proactive probing* [10]. *Reactive probing* is triggered by user behaviors, such as user making a connection to a specific port and/or talks in a specific pattern/protocol with the server. The latter, *proactive probing*, does not require any explicit stimulus, but is instead closer to Internet scanning. Previous research weighs more on *reactive probing*, but it is worth noting that *proactive probing* could also be used by the most radical censors trying to "act early". It is worth noting that, if the active probing technique is not designed to be exclusive to the target implementation, then there's some chance for false positives which causes the blocking of benigh applications.

3.3 Machine Learning driven Fingerprinting

Some reports show that some censorship/security contractors are advertising their traffic analysis system/algorithms with Machine Learning(ML) methods integrated [21]. Although none of the cited sources found convincing evidence for censors employing ML-based traffic classifiers in their system, it is proved by researchers to be practical. A group of researchers claimed that they were able to classify if a TCP connection belongs to Shadowsocks or not with a good accuracy with Random Forest algorithm [6]. Theoretically

speaking, some statistical features in the traffic generated by proxies might be hiding in the traffic, such as packet size, burst length, connection time, and ML-based classifiers could be a really good solution for such classification problems.

Besides how censors may use ML methods for passive analysis, they could also use ML actively for probing. Geneva [3] was introduced by Bock et al. to fight against censorship with ML, and it works like how active probing works: Geneva adapts its strategy to decorate the packets/connection on either a client or a server to reveal possible decoration strategies for a TCP connection, on either client-side or server-side, to bypass the existing censorship. Later it was also adapted to be used in finding possible strategies for TCP amplification attacks which was before considered non-trivial [2]. The same idea could be applied in the opposite direction to help censors directly finding and blocking hidden instances of circumvention servers, as well as finding passive or active fingerprinting techniques to be applied to the censorship infrastructure later.

4 ACTIVE PROBING CASE STUDY

Previous research works have identified active probing techniques a censor used or may use to detect the presence of certain implementation of circumvention tools. In this section, we provide study selected cases and identify the design flaws being exploited by active probing attacks in each case.

4.1 Tor

4.1.1 Background. Tor was among the first batch of protocols getting actively probed and blocked. In December 2011, Wilde reported the observation that shortly after a client in China connects to any Tor bridge outside China, a prober will actively scan the Tor bridge by trying to connect with Tor and if succeeded, the specific `<IP:Port>` tuple on the Tor bridge will then be blocked [33]. This report also indirectly confirmed that China has been employing DPI techniques to detect the presence of Tor and use detected presence to trigger the following active probing. In Wilde's investigation, the stimulus was confirmed to be the specific TLS *ClientHello* message sent by Tor client when connecting to a Tor bridge. In this message, a unique list of cipher suites was found to be exclusive to Tor clients and and therefore becomes a fingerprint. And once this fingerprint is intercepted, active probing will happen after multiples of 15 minutes.

Later, Winter and Lindskog [34] were able to show with their experiments that after the `<IP:port>` tuple has been blocked, the blocked tuple on the bridge will continuously get probed with the same probing strategy by negotiating new Tor connections. The blocking will be removed ONLY after the tuple become unreachable for 12-hours, that is to say, as long as the Tor bridge still works on the blocked tuple, the GFW will not remove the blocking. The research also showed that more than half of the active probing attempts are from one single IP address (`202.108.181.70`) and the rest of 1615 connections were made by probers with 1584 different IPs.

4.1.2 Analysis. The active probing techniques used in detecting hidden Tor bridges were relatively simple and straightforward: since Tor bridges did not incorporate enough identity check for the connecting client and would accept connections from any Tor client, the active prober simply established TLS connections with the Tor bridge and negotiated for a Tor connection. Apart from the critical flaw of the TLS fingerprint included in the ClientHello message, the other major design flaw contributed to the blocking of Tor bridges is the lack of authorization validation. In other words, if the scanned Tor bridges reserves its negotiation service for only certain user with a valid "reservation", such probing attempts would have failed.

4.2 Shadowsocks

4.2.1 Background. Shadowsocks was a very popular open-source proxy tool introduced in 2012 by clowwindy for censorship circumvention purposes [5]. It was known for encrypting the traffic fully and therefore leaves no plaintext context in its connections. The success of Shadowsocks in China gets attention from the national censor of China, the GFW. However, the interference from GFW against Shadowsocks had not been confirmed with solid evidence until the massive blocking happened during political sensitive times since 2017 October. But a more interesting question asked by researchers is How did China detect and block Shadowsocks, which was answered by Alice et al. in 2019 [1].

In the past, Shadowsocks once had several cipher-based vulnerabilities which were fixed with later introduced AEAD ciphers [9, 11] and could be exploited by passive techniques. But GFW used a much different approach this time, by combining active probing and passive analysis techniques. Researchers found that genuine Shadowsocks connection to a remote server will trigger 7 different types of active probes, with 5 of which being replay attacks with no bytes or a few certain bytes changed from what was sent by the shadowsocks client(replay probes), and rest 2 of which not based on that (randon bytes probes). The researcher also identified the complex mechanism behind the probes, including what's triggering them, how are they dependent on each other, and the intention behind the probes.

4.2.2 Analysis. In [1], based on the assumption that Shadowsocks traffic should be indistinguishable from random, researchers revealed that one single data packet after a TCP handshake is enough to trigger the active probing if the packet length meets certain criteria. The active probes for Shadowsocks are much more complex comparing to Tor's. When seeing a random bytes packet, Shadowsocks server behaves differently based on different packet length, salt length and cipher used. For example, for servers configured to use stream ciphers and 8-byte salts, the server will timeout the connection if only received a packet with length less than 8-bytes, or reset the connection if the packet length is greater than 8-bytes. As for replay probes, different implementation respond quite differently and leading to possibilities not only protocol identification (Shadowsocks), but also differentiations between implementations (Outline, Shadowsocks-Go, Shadowsocks-libev, etc). Such specific behavior creates the exclusive fingerprint of Shadowsocks, and could have been avoided by behaving more consistently on error and/or including certain probing detection/prevention mechanism.

4.3 ShadowTLS

4.3.1 Background. ShadowTLS is a novel TLS-like proxy introduced in 2022. Unlike other TLS-based proxies, instead of talking

with the server in real TLS, it uses a different approach and performs real handshake with an allowed Server Name Indication(SNI) value in the TLS ClientHello message through a reflection server to bypass TLS Server Name allowlists, which was reported to be seen in Quanzhou, China [4]. After the forwarded "real handshake", ShadowTLS will continue using the same TCP connection for proxy requests and responses and the communication will look like benign TLS communication as long as the censor is incapable of decrypting TLS. Since the implementation is still new, there is no direct evidence pointing to active probing used by censors.

However, Wang et al. were able to identify a few relatively simple active probing techniques effective against ShadowTLS in [32]. The researchers scanned the Internet to observe behavioral schemes for TLS servers in response to several common probers and compared them with the scheme found in ShadowTLS based on experiments and code analysis. Turns out only a very small fraction (0.05%) of TLS servers on the Internet behaves exactly like ShadowTLS in response to these probes. And therefore concluded that the combination of identified probes is efficient against ShadowTLS.

4.3.2 Analysis. Probes in [32] were fairly common and could be easily observed on the Internet through capturing packets on a TLS server. The most efficient probe identified by Wang et al. was built by using random bytes to fill the payload of a TLS Application Data (instead of ciphertext generated by TLS). Only 0.12% of the public TLS servers on port 443 responded like ShadowTLS: staying silent. A vast majority (88.9%) of the TLS servers sends a TLS Alert and close the connection immediately after it.

Other probes identified in the paper were also similar in idea, basically by triggering errors on purpose and observe the response. The RFC documentation of TLS protocol has well-defined expected behaviors when both ends are benigh, but not quite so in defining error handlings, which effectively makes it possible for different implementations of TLS protocol to have different fingerprints in terms of error handling. As a TLS-server parrot, ShadowTLS is expected to copy the behavior from the remote server it mimicks (where the SNI points to).

5 COUNTERMEASURES

Countering active probing or certain passive analysis techniques are not as simple as "just write better code". Past works have made several attempts such as adding obfuscation and trying to design more complicated system and make it harder for censors to analyze and break. However, any implementation will essentially leave its fingerprints/traces with its presence here and there. And the goal, instead of eliminating the trace, should be focusing on weakening the distinguishability between a circumvention tool's presence and the presence of "popular and must-have" tools, therefore scaling up the collateral damage for block the circumvention tool against censors.

5.1 The Parrot is Dead

Parroting, or mimicking was overrated and has been criticized by Houmansadr et al. in [17]. Since building perfect parrots is almost an impossible job, a better approach might be giving up on parroting. However, it does not necessarily mean developers should stop thinking about blending traffic created by circumvention tools into popular protocols. Poorly designed parrots expose fingerprintable traits and lead to trivial detection and potential legal responsibilities in certain region. And instead of building a look-alike, there are better ways to achieve the goal.

5.1.1 Reusing not Reinventing. In terms of developing the transport for a circumvention tool, rather than spending more time on creating a crude implementation of a popular protocol, a better approach could be reusing the existing popular implementations by either developing based on its code (if it is open-source under a compatible license) or assembling the solution with Application Fronting.

NaiveProxy is a good example for repurposing an open-source project: as a TLS-based proxy, instead of building its own TLS stack, its authors extracted the network stack of Chromium, *cronet*, and repurposed it for tunneling. This effectively reduced the maintenance workload and fingerprintability, as well as improved robustness by avoiding reinventing the wheel. The downside of this approach, however, could be limited portability and programmability, since popular implementations are usually bulky in size and the dependency relation could be complicated. It will not be easy to fully understand it, and not really practical to rewrite it in another programming language.

The other method, Application Fronting, is a popular way to hide the presence of a circumvention server, especially a TLS-based proxy server. Like Domain Fronting [12], Application Fronting uses another application/service as the front-end and have the front-end conditionally forwarding requests. While Domain Fronting usually require a 3rd-party CDN server, Application Fronting could be setup on one server with reverse proxy applications such as Nginx or Caddy. It reuses the implementation of the application without reprogramming it and if configured correctly, the server should behave exactly like the fronting application for most of active probing techniques.

5.1.2 Fingerprint Cloning. Fingerprinting cloning is another way to prevent fingerprinting against a specific implementation. Frolov and Wustrow proposed a TLS fingerprint mimicry module/library called uTLS in [15] and it has become the most popular choice for TLS-based proxies in order to mimick popular web browsers' fingerprints in any TLS client [1]. Currently, uTLS was used in many popular TLS-based proxy projects including V2Ray/V2Fly, XTLS, and Singbox. The existence of libraries like uTLS greatly improved the developing efficiency of new censorship circumvention tools and the same idea could be applied to server-side: a server could choose to mimick one or more popular server implementation of certain protocols and handle all kinds of requests "correctly" by responding exactly like how the target implementation would, unless authenticated to use the circumvention service. Currently, there are only a few circumvention tools designed with similar idea, such as ShadowTLS, Restls, and XTLS Vision.

[1] It is worth noting that, uTLS does not solve all fingerprinting issues but specifically targets the so-called TLS fingerprint in TLS ClientHello. TLS protocol includes other fingerprintable features which are not covered by uTLS.

5.2 Better Authentication/Access Control

Another root cause of the active probing vulnerability is that most of the servers are publically accessible to all probers, and it might make sense to add better authentication design to circumvention tools. Recall Tor bridges' failure to a super simple probe, the reason of the failure is that Tor bridges essentially have no access control and allows anyone, without any prior notification, connect and use the service. This makes it extremely vulnerable to active probing attacks. As for a parrotting server, in order to be fully probing-resistant, it must not behave differently from the implementation it is mimicking until the client successfully authenticates itself with the server. In addition, developers have employed some other techniques which effectively prevents active probing attacks.

5.2.1 Exclusive Access. With an additional registrar/directory server, it is trivial to implement a mechanism which we call *Exclusive Access*. This mechanism should only allow authorized users to connect to a certain service and block all other unauthorized access by not picking up their connections or reflecting its request to a different host which does not run a circumvention tool. The challenges and risks of this method are mainly about how to prevent other fingerprintable traits introduced when deterring connections. The latest version of ShadowTLS uses this idea to prevent active probing attacks.

5.2.2 Summoning-based Server. The idea is similar to the previous one. Properlly authorized clients may be able to summon servers on-demand and have the server to initiate the connection in some ways. We see similar approach in *Conjure* [13], a refraction networking project working at ISP-level for circumvention. In Conjure, clients register with the station and a decoy will be available to it. Such idea could be integrated more with Peer-to-Peer(P2P) communication protocols such as WebRTC or BitTorrent in order to make the traffic looks less suspicious if needed – since P2P protocols are essentially all "summoning-based".

6 DISCUSSION

6.1 Challenges in Active Probing Analysis

It is generally considered non-trivial to detect and measure active probing attacks from censors [10, 18, 22, 23]. The Internet is noisy and there are all kinds of probes and crawlers actively scanning the Internet, which makes it difficult to identify and therefore analyze active probing attacks initiated by censors and specifically against censorship circumvention tools. For *reactive probings* with a very long interval from its stimulus or *proactive probings*, researchers will be required to filter out the "background radiation" before conducting targeted experiment or analysis.

6.2 Myth of Machine Learning

Machine Learning is a topic becoming very popular in recent years. Since it is very powerful in some specific areas of application, its abilities could be overrated by people not doing ML-related research. In the open-source community of censorship circumvention developers, the myth of ML has been circling around for some time. A group of people believes censors have or eventually will use Machine Learning techniques in censorship and *therefore defeat all possible ways of circumvention.*

While the first half could be true, the second half might have been way to pessimistic. The father of the Great Firewall of China, Dr. Fang Binxing has stated publicly that there is likely to be an eternal war between censorship and circumvention.

6.3 Future Works

It has been more than 10 years since the GFW employed Active Probing for the first time against Tor, in 2011. In recent years, we see censors more frequently using active probing techniques against circumvention tools. But as there are many research works relating to how could active probing be used for detections [16, 18], little did we know about how to measure them or what's triggering them. Thus, research works under following categories could be potentially valuable in terms of the topic of active probing.

Detection/Identification of Active Probing. Better methods to detect active probing attacks and/or identify probers could be significant. Being able to detect active probing attacks from a hostile party with better confidence would benefit both researchers and developers by allowing both to respond earlier and come up with a more timely analysis/fix. This would be critical in this cat-and-mouse game between censorship and circumvention. Machine Learning should be seriously considered.

Active Probing Vulnerability Scanning. We could solve the problem in another way, by resolving most of, if not all active probing vulnerabilities before a tool gets released. Researchers may work on collecting more active probing specimens to identify and conclude a more general testing procedural, in the form of a checklist. Optionally, an automated test suite could be developed for common protocol.

Active Parrot. Imperfect parrots are vulnerable to active probing techniques and the essential reason as argued by Houmansadr et al. in [17] is that parrots will eventually behave differently from the mimicking targets if the developer manually copy traits from another application. An alternative solution might be letting the parrots to actively copying behaviors from a *template application*, hosted in either an internal or an external environment. More studies and experiments need to be conducted for such proactive mimicry to validate its effectiveness and performance.

Post-ML Circumvention. The day for the technological singularity is closer thanks to the fast development of Machine Learning. Now we are living in a post-ML world and censors may use or could have used Machine Learning techniques against circumvention tools. To evade detection and blocking, developers and researchers might need to consider building ML-driven circumvention tools against ML-driven censors.

7 CONCLUSION

In this paper, we summarized several popular fingerprinting techniques with a focus on the ones based on active probing. And we revisited a few cases of active probing attacks and/or vulnerability in the real-world implementations to highlight common design flaws contributing to active probing vulnerability.

With the countermeasures we suggested in Section 5, we could conclude that it is a challenging task to build a probe-resistant

circumvention tool. Certain criteria must be met, including the implementation itself should not be trivially traceable (thus, needs a *perfect parrot*) and the need of stricter access control. However, as an old Chinese proverb says, "Untying the bell (on the tiger's neck) requires the person who fastened the bell" [2], censorship is born for majorly political interests and will not stop progressing until all politics ends. The arm race between censorship and circumvention will go on, old techniques will eventually fail and deprecate while the new ones getting in place. There will never be a perfect final solution for either censors or circumventors.

REFERENCES

[1] Alice, Bob, Carol, Jan Beznazwy, and Amir Houmansadr. 2020. How China Detects and Blocks Shadowsocks. In *Proceedings of the ACM Internet Measurement Conference* (Virtual Event, USA) *(IMC '20)*. Association for Computing Machinery, New York, NY, USA, 111–124. https://doi.org/10.1145/3419394.3423644

[2] Kevin Bock, Abdulrahman Alaraj, Yair Fax, K. S. Hurley, Eric Wustrow, and Dave Levin. 2021. Weaponizing Middleboxes for TCP Reflected Amplification. In *USENIX Security Symposium*.

[3] Kevin Bock, George Hughey, Xiao Qiang, and Dave Levin. 2019. Geneva: Evolving Censorship Evasion Strategies. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. Association for Computing Machinery, New York, NY, USA, 2199–2214. https://doi.org/10.1145/3319535.3363189

[4] Chism. 2022. 前几天去泉州体验了一把白名单 - V2EX. https://web.archive.org/web/20220507153405/https://v2ex.com/t/851473.

[5] Shadowsocks contributors. 2012. Shadowsocks | A fast tunnel proxy that helps you bypass firewalls. https://shadowsocks.org/.

[6] Ziye Deng, Zihan Liu, Zhouguo Chen, and Yubin Guo. 2017. The Random Forest Based Detection of Shadowsock's Traffic. In *2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, Vol. 2. 75–78. https://doi.org/10.1109/IHMSC.2017.132

[7] T. Dierks, Independent, E. Rescorla, and RTFM Inc. 2006. RFC 4346 The Transport Layer Security (TLS) Protocol Version 1.1. https://www.rfc-editor.org/rfc/rfc4346.

[8] T. Dierks, Independent, E. Rescorla, and RTFM Inc. 2008. RFC 5246 The Transport Layer Security (TLS) Protocol Version 1.2. https://www.rfc-editor.org/rfc/rfc5246.

[9] edwardz246003. 2020. Redirect attack on Shadowsocks stream ciphers. https://github.com/edwardz246003/shadowsocks.

[10] Roya Ensafi, David Fifield, Philipp Winter, Nick Feamster, Nicholas Weaver, and Vern Paxson. 2015. Examining How the Great Firewall Discovers Hidden Circumvention Servers. In *Proceedings of the 2015 Internet Measurement Conference* (Tokyo, Japan) *(IMC '15)*. Association for Computing Machinery, New York, NY, USA, 445–458. https://doi.org/10.1145/2815675.2815690

[11] David Fifield. 2017. Shadowsocks active-probing attacks and defenses. https://groups.google.com/g/traffic-obf/c/CWO0peBJLGc/m/Py-clLSTBwAJ.

[12] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. 2015. Blocking-resistant communication through domain fronting. *Proc. Priv. Enhancing Technol.* 2015, 2 (2015), 46–64.

[13] Sergey Frolov, Jack Wampler, Sze Chuen Tan, J. Alex Halderman, Nikita Borisov, and Eric Wustrow. 2019. Conjure: Summoning Proxies from Unused Address Space. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. Association for Computing Machinery, New York, NY, USA, 2215–2229. https://doi.org/10.1145/3319535.3363218

[14] Sergey Frolov and Eric Wustrow. 2019. TLSFingerprint.io. https://tlsfingerprint.io/.

[15] Sergey Frolov and Eric Wustrow. 2019. The use of TLS in Censorship Circumvention. *Proceedings 2019 Network and Distributed System Security Symposium* (2019).

[16] Marcin Gregorczyk, Piotr Żórawski, Piotr Nowakowski, Krzysztof Cabaj, and Wojciech Mazurczyk. 2020. Sniffing Detection Based on Network Traffic Probing and Machine Learning. *IEEE Access* 8 (2020), 149255–149269. https://doi.org/10.1109/ACCESS.2020.3016076

[17] Amir Houmansadr, Chad Brubaker, and Vitaly Shmatikov. 2013. The Parrot Is Dead: Observing Unobservable Network Communications. In *2013 IEEE Symposium on Security and Privacy*. IEEE, 65–79. https://doi.org/10.1109/SP.2013.14

[18] Xiaoqiang Huang, Bailing Wang, Yunxiao Sun, Yang Liu, Lianhai Wang, and Shujiang Xu. 2019. Protocol State Based Active Probing Method for Detecting Applications of Proprietary Protocol. *2019 2nd International Conference on Information Systems and Computer Aided Education (ICISCAE)* (2019), 379–384.

[19] OpenVPN Inc. 2001. About OpenVPN Through The Years. https://openvpn.net/about/.

[20] Kee Jefferys, Simon Harman, Johnathan Ross, and Paul McLean. 2018. *Loki: private transactions, decentralised communication.* Technical Report. LOKINET.

[21] katayaburi. 2023. Some hints about Chinese GFW providers, related to the October 2022 large scale blocking. https://github.com/net4people/bbs/issues/239.

[22] Hyeonwoo Kim, Dongwoo Kwon, and Hongtaek Ju. 2014. Analysis of ICMP policy for edge firewalls using active probing. In *The 16th Asia-Pacific Network Operations and Management Symposium*. 1–4. https://doi.org/10.1109/APNOMS.2014.6996591

[23] Pavel Liubinskii. 2020. The Great Firewall's active probing circumvention technique with port knocking and SDN. In *Master Thesis*. Aalto University.

[24] Chris M. Lonvick and Sami Lehtinen. 2006. The Secure Shell (SSH) Protocol Assigned Numbers. RFC 4250. https://doi.org/10.17487/RFC4250

[25] Chris M. Lonvick and Tatu Ylonen. 2006. The Secure Shell (SSH) Authentication Protocol. RFC 4252. https://doi.org/10.17487/RFC4252

[26] Chris M. Lonvick and Tatu Ylonen. 2006. The Secure Shell (SSH) Connection Protocol. RFC 4254. https://doi.org/10.17487/RFC4254

[27] Chris M. Lonvick and Tatu Ylonen. 2006. The Secure Shell (SSH) Protocol Architecture. RFC 4251. https://doi.org/10.17487/RFC4251

[28] Chris M. Lonvick and Tatu Ylonen. 2006. The Secure Shell (SSH) Transport Layer Protocol. RFC 4253. https://doi.org/10.17487/RFC4253

[29] Tor Project. 2006. History. https://www.torproject.org/about/history/.

[30] E. Rescorla and Mozilla. 2018. RFC 8446 The Transport Layer Security (TLS) Protocol Version 1.3. https://www.rfc-editor.org/rfc/rfc8446.

[31] trojan gfw. 2017. Trojan Documentation | trojan. https://trojan-gfw.github.io/trojan/.

[32] Gaukas Wang, Anonymous, Jackson Sippe, Hai Chi, and Eric Wustrow. 2023. Chasing Shadows: A security analysis of the ShadowTLS proxy. (Feb. 2023). https://www.petsymposium.org/foci/2023/foci-2023-0002.php

[33] Tim Wilde. 2012. Great Firewall Tor Probing Circa 09 DEC 2011. https://gist.github.com/twilde/da3c7a9af01d74cd7de7.

[34] Philipp Winter and Stefan Lindskog. 2012. How the Great Firewall of China is Blocking Tor. In *2nd USENIX Workshop on Free and Open Communications on the Internet (FOCI 12)*. USENIX Association, Bellevue, WA. https://www.usenix.org/conference/foci12/workshop-program/presentation/Winter

[35] Wireshark Foundation. [n. d.]. *Wireshark*. https://www.wireshark.org/

[36] Diwen Xue, Reethika Ramesh, Arham Jain, Michalis Kallitsis, J. Alex Halderman, Jedidiah R. Crandall, and Roya Ensafi. 2022. OpenVPN is Open to VPN Fingerprinting. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 483–500. https://www.usenix.org/conference/usenixsecurity22/presentation/xue-diwen

[37] Pavel Čeleda, Petr Velan, Benjamin Král, and Ondřej Kozák. 2019. Enabling SSH Protocol Visibility in Flow Monitoring. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. 569–574.

[2] In Chinese characters, "解鈴還須繫鈴人"