

# Survey and Experimentation of Web Circumvention Techniques

Ana CEBAN, Lowel BUZZI, Gautier GEORGEON  
TELECOM Nancy, Master in Computer Science School,  
Part of Université de Lorraine,  
193 Avenue Paul Muller, 54600 Villers-Lès-Nancy, France

Email: - ana.ceban@telecomnancy.eu  
- lowel.buzzi@telecomnancy.eu  
- gautier.georgeon@telecomnancy.eu

**Abstract**—With the rapid expansion of the internet and the ubiquity of web-based services, controlling access to specific web content has become a critical concern for various entities, from governments and corporations to individual users. This paper surveys the most prominent web filtering techniques, including IP filtering, DNS filtering, proxy-based filtering, and Server Name Indication (SNI) filtering. For each method, we detail its functional mechanisms, typical implementers, and possible circumvention strategies. We further explore the implications of encryption technologies like TLS and encrypted DNS on filtering capabilities. Lastly, we present experimental insights into the effectiveness and limitations of each method, providing practical evaluations that highlight their strengths, weaknesses, and resilience against circumvention.

**Index Terms**—web filtering, web circumvention, tcp, ip, dns, proxy, sni, tls, https, virtual private network

## I. INTRODUCTION

Web contents delivered through Internet have become a critical part of our daily lives. Accessing but also creating a website has never been simpler. However, people can be interested in blocking a website access, no matter their motivation: parental filtering, governmental decision, company security policy enforcement, etc.

Filtering techniques comply with the government rules to block access to inappropriate or illegal content but also for censorship purposes. This especially applies to some authoritative countries where political content is also inclined to be prohibited. [1]

Network filtering being a vast subject, we will only focus on web filtering in this article.

This article is intended for an intermediate audience, since we will remind some base knowledge about network packets and internet, and also work to obtain an interesting result for people who might want to filter or escape filtering using the techniques we surveyed.

The first step is to define the websites to be filtered, based on various criteria. The second step consists in choosing a filtering method based on the part of the network that is under our control and our filtering capability depending the rights and technical means we have. Ideally, the process should restrict

the targeted websites access for the intended scope. However, depending on the filtering method, bypass mechanisms can be set more or less easily. Also, some sites not designated can be wrongly restricted (false positives). [2]

Our goal is to survey the main website filtering techniques and then to present and evaluate for each possible circumvention solutions. In particular we will consider DNS-based, proxy-based and SNI-based filtering techniques.

The files are available at: [https://github.com/gautier-g/web\\_filtering\\_study](https://github.com/gautier-g/web_filtering_study).

The rest of this article is structured as follow. Section II presents an exhaustive summary of the existing techniques for filtering content on the internet, and section III displays the implementations of web circumvention that we decided to test. Each part ends with a table as a conclusion.

## II. EXISTING FILTERING TECHNIQUES

### A. Web Content Filtering

1) *Who's likely to implement it:* Nowadays, the main actor who's likely to use this feature is the user himself. That is due to the fact that HTTPS (HTTP within TLS) has restricted the possibility to intercept and read packets during transmission and thus filter web content so that the web content can only be filtered at the end-host side as long as the encryption is not broken by a third party. The HTTP model has been clearly abandoned due to its lack of confidentiality (fig. 1): as we can see the message body appears clearly in the transmitted packets.

2) *Functional description:* At first glance, we could think that this process is deprecated due to the systematic use of HTTPS nowadays, but in fact, many client-side features are available in order to achieve web content filtering [2]. Indeed, the user could either install a browser add-on or an external software. Here, we are accessing the website, but before displaying its content, we check on our computer if it is acceptable.

However, if clear-text HTTP is used, deep packet inspection (DPI) techniques analyzing the web content are still possible. For instance a internet service provider (ISP) could filter web pages based on their content. Another possibility to inspect

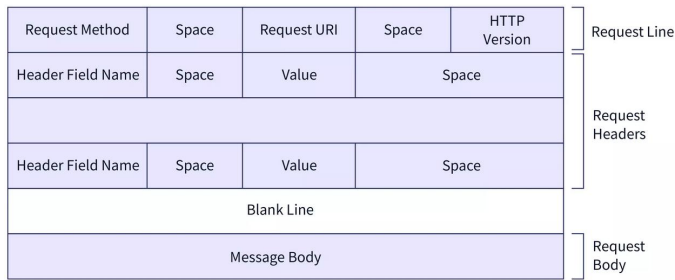


Fig. 1. The HTTP model.

HTTP content is to place a filtering proxy between the client and the web server.

Such proxy has a list of forgiven sites, regularly updated, and when a user polls the proxy, his request is either accepted or refused based on the requested URL (indirect header filtering).

3) *Bypassing techniques*: Today, web content filtering can be implemented through two ways: select the authorized connections before using HTTPS (use an intermediate connection with a proxy), or establish a connection and decide to display the pages or not. That's why there are no bypassing methods unless in the case the user decides to stop or add censorship on its computer. For HTTPS proxies, the only way is to find another way in the network to pass through. Sometimes it is impossible and sometimes you can use a custom DNS resolver [3] or a VPN so that the content is hidden.

## B. HTTPS Proxy Filtering

1) *Who's likely to implement it?*: Almost everyone can nowadays run a cheap proxy on their computer, so that they're able to filter their own content. But that is different from a real server proxy used by companies or the government.

2) *Functional description*: Proxies are ideal for filtering connections considering they're intermediate servers, allowing both to separate a machine from others, or process a HTTP request. (http proxy)

HTTP proxies generally need the user to do an action in order to be used.

They are often used at company scale, in order to permit some content filtering before the proxy sends the request to the desired website.

HTTPS proxies are working the same as HTTP proxies but they act as an intermediate by setting up a TLS connection between the two machines that are communicating. Indeed, when a user asks for the server, the proxy intercepts the request and sends a valid certificate. The TLS handshake is done with the proxy server. Considering the proxy has already established a TLS handshake with the server, the proxy is able to decrypt requests and answers and filter them. The classic HTTPS proxy (forward proxy) used in companies (fig. 2) receives HTTPS requests from the inside (LAN) and eventually filters the requests after decrypting them. For a reverse proxy, it stills the same but the requests go from WAN to LAN.

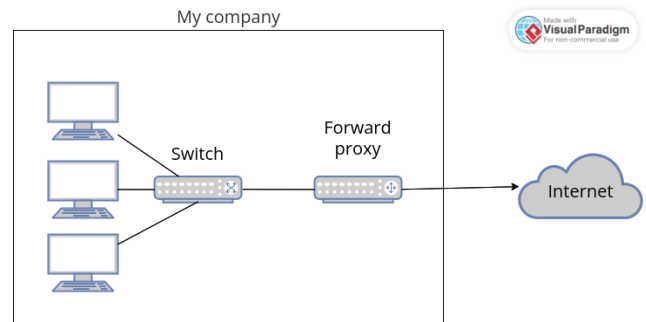


Fig. 2. A classic company proxy.

3) *Bypassing techniques*: These are the same techniques as show in the previous subsection.

In terms of personal use, since proxies are now pretty affordable for personal use on the online market, everyone can try to connect to different websites, switching parameters of the proxy in order to see different results.

## C. TCP/IP Header Filtering

1) *Who's likely to implement it*: Anyone who has the right to modify the rule of a router, or add a proxy server, can implement this feature.

2) *Functional description*: The header filtering technique consists in blocking a blacklist of IP addresses in a router. It is usually made with a firewall. Because of the web aspect, we are interested in blocking the HTTPS port (443). This mechanism is pretty simple but not precise enough: several distinct websites can be hosted on a same server, thus sharing a same IP address, causing legal websites to be forbidden [2]. IP addresses may be brought to change regularly, that's why the blacklist needs to be updated frequently to remain accurate. The TLS protocol (fig. 3) used with HTTP allows a confidential exchange between two parties, especially thanks to the TLS handshake where keys are shared privately to encrypt data.

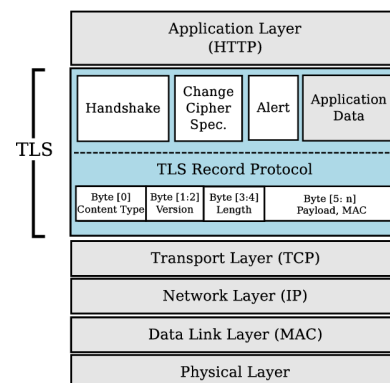


Fig. 3. The TLS model.

3) *Bypassing techniques*: It is sometimes easy to add a proxy in order to modify the IP address. This is the same idea with a virtual private network when the filtering is based on

the IP destination. TOR is also a great technique to bypass IP filtering since the requests are blent between different servers before accessing its destination.

#### D. DNS Filtering

1) *Who's likely to implement it:* DNS (Domain Name System) filtering is mostly implemented in order to regulate the content available online. It is often done by the ISP as it has the default DNS resolver unless specifically changed by the user.

Organizations, such as companies and educational institutions, also implement DNS filtering to restrict internet usage. They may block websites that are not work related, sites that are categorised as social media or illegal content. Additionally, they may want to block domains associated with malware, phishing or other cybersecurity attacks.

2) *Functional description:* The process starts when the user types a URL query into the address bar of their web browser then there are a couple of things that happen:

- first, the domain name query is sent to a server called the DNS resolver, usually provided by the ISP (Internet Service Provider) or a third-party service;
- second, the DNS resolver will then first look into its cache to see if the domain name has been already resolved or it will forward the domain query to another DNS server, each responsible for certain hierarchy level in the domain name system until it obtains the IP address of the server hosting the requested website and send the result back to the user
- third, the user can establish a connection with the website's server

Consequently no website can be accessed without a proper DNS resolution, unless of course the user inserts directly the IP address. The way that DNS filtering work is by intercepting the DNS query and preventing the resolution of the filtered domain name.

The domain name is being analysed, meaning the DNS checks if it contained in a *blocklist* that may be fixed by the organisation filtering policies or another criteria, or if it corresponds to a category that is prohibited (ex: social media for a company).

If the site is blocked, the URL cannot be accessed and the DNS filter either redirects to a safe page or displays an error or another predefined page.

DNS filtering can work also with an *allowlist* instead of a blocklist, meaning only domains that are on the list can be accessed and others are blocked by default. With the DNS filtering (fig. 4), the DNS request sent by the client can't succeed to find the corresponding IP (blocked website) and thus it results in an error (unknown domain) or a safe website displayed.

##### 3) Bypassing techniques:

a) *Third-party DNS:* , one common method to bypass DNS filtering is by using third-party DNS resolvers. As previously mentioned, the default DNS resolver is typically

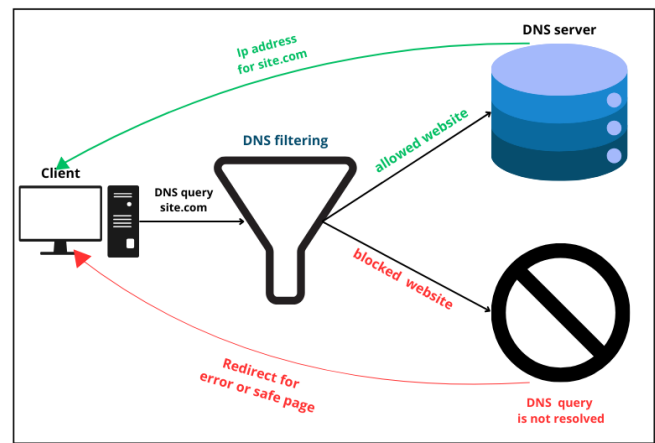


Fig. 4. DNS filtering

provided by Internet Service Provider (ISP). This default resolver often lacks advanced security features such as protection against malicious domain names while being more susceptible to government censorship. So users frequently switch to third-party DNS services like Google DNS (8.8.8.8) (approximately 50% of global DNS traffic). But they are often used as a way to bypass censorship inflicted by local government policies. But this may not be an adequate solution to access illegal website as most DNS services include filters for this kind of content. Third party DNS is also used to have better protection against malware, cyber security attacks but a study based on IST data in Denmark estimated that only 1.1 % to 1.5% of users employ third-party DNS resolvers for content filtering, this and the fact that DNS responses for censored domains are found to be at least two orders of magnitude more prevalent on third-party resolvers than on ISP-provided DNS, indicating that many users switch resolvers primarily to circumvent censorship and regulations.

b) *Encrypted DNS Protocols::* traditional DNS transmits queries in plain, unencrypted text, exposing the user activity and the domains being accessed. A solution to this would be to use encrypted DNS protocols such as DoH (DNS over HTTPS) or DoT (DNS over TLS).

- **DNS over HTTPS:** this protocol encrypts DNS queries using HTTPS tunneling [4], it does therefore offer better privacy by preventing monitoring users traffic, meanwhile preventing DNS filtering as the domain name can not longer be obtained;
- **DNS over TLS:** work similarly to DoH but it uses a dedicated TLS connection instead.

c) *Other Bypassing Methods:* In addition to the methods presented, user may use alternative techniques to circumvent DNS filtering, in this paper we will not further investigate those methods, we will simply mention them.

- **Virtual Private Networks (VPN)** VPNs encrypt all internet traffic and route it through remote servers, masking the user's IP address. You can bypass DNS filtering since you use the DNS server of your VPN.

- **Proxy Servers** Using a proxy with a specific DNS Resolver configuration could allow a user to bypass DNS filtering.
- **Accessing directly through the IP address** however many websites rely on virtual hosting and require domain based access.

#### E. TLS Server Name Indication Filtering

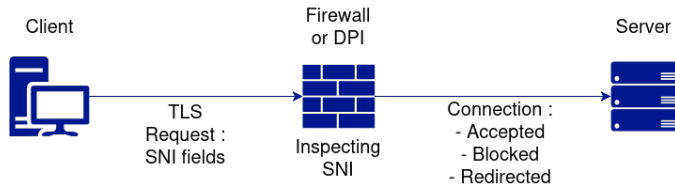


Fig. 5. SNI in TLS protocol

1) *Who's likely to implement it:* Governments, internet service providers, and corporate networks are the primary entities that implement HTTPS SNI filtering. Authoritarian regimes, such as China, often use it for censorship by blocking access to specific websites, while internet service providers and organizations may deploy it to enforce network policies, restrict access to harmful sites, or optimize bandwidth usage. Companies may also employ SNI filtering for security purposes, especially to prevent employees from accessing phishing sites or dangerous domains.

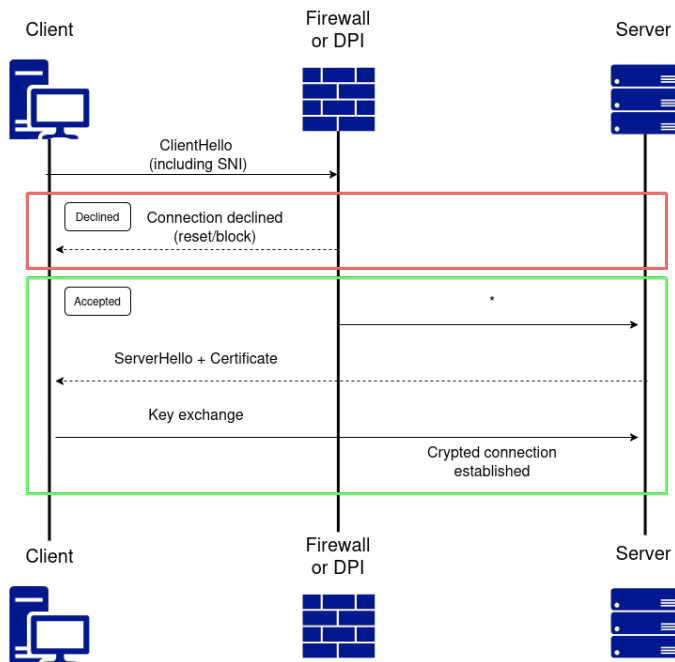


Fig. 6. Functioning of SNI Filtering

2) *Functional description:* SNI is an extension of the TLS protocol that allows a client to indicate the hostname of the server it intends to connect to during the TLS handshake. HTTPS SNI filtering operates by inspecting this plaintext

field before encryption takes place. Firewalls or deep packet inspection (DPI) tools analyze the SNI field in ClientHello messages and apply string matching filtering rules based on blacklists, whitelists, or keyword-based blocking mechanisms. Since this process occurs before full encryption, it enables network administrators to regulate traffic without decrypting the payload. However, this method has limitations, such as inability to filter connections using encrypted SNI (ESNI).

3) *Bypassing techniques:* Encrypted SNI (ESNI) and Encrypted ClientHello (ECH): One of the most effective countermeasures against SNI filtering is ESNI (Encrypted SNI). [5] Unlike traditional SNI, ESNI encrypts the server name within the TLS handshake, preventing middleboxes and firewalls from inspecting or blocking connections based on the requested domain. TLS 1.3 handshakes that include ESNI. — TLS 1.3 introduced ECH, which encrypts the entire ClientHello message, including the SNI field, making traditional filtering ineffective.—

Bypassing with backward compatibility enforcement : SNI filtering can be bypassed thanks to compatibility concerns. Some systems allow fallback to older TLS versions. [6] Attackers can exploit this by forcing a downgrade attack, causing the client to negotiate an older protocol version where SNI filtering is either less effective or absent. This technique is particularly useful in environments where TLS 1.3 is partially adopted but not universally enforced.

Bypassing on shared certificate hosting : A significant limitation of SNI filtering arises when multiple domains share the same TLS certificate. Many Content Delivery Networks (CDNs) and cloud service providers host numerous domains under a single certificate, making it difficult for filtering entities to block one specific domain without affecting others. Users can access restricted domains by requesting a different but valid hostname that resolves to the same IP address and TLS certificate when a common certificate exists between multiples domains. This method is particularly effective when CDNs serve both blocked and permitted domains, especially when forbidding access to all ip addresses could prevent users from accessing services that are not concerned.

Criteria	IP Filtering	DNS Filtering	TLS/SSL Interception	Proxy Filtering
<b>False négative /False positive</b>	High – blocks entire IP ranges.	Moderate – shared IPs may cause issues.	Moderate – encrypted traffic analysis may misclassify.	Low – precise filtering based on full requests.
<b>Performance</b>	High – very fast , minimal overhead.	High – fast lookups, low resource use.	Low – TLS decryption adds overhead.	Moderate – proxy load-dependent.
<b>Evasion Resistance</b>	Low – bypassable via VPNs/CDNs.	Moderate avoidable using encrypted DNS (DoH/DoT).	High – can analyze encrypted traffic, but ECH can evade.	High – controls application-layer traffic.
<b>Implementation</b>	Easy – simple firewall rules.	Easy – DNS server configuration.	Hard – requires TLS decryption setup.	Moderate – requires proxy server setup.
<b>Efficiency</b>	High – works at the network level.	High – low overhead.	Low – computationally expensive.	Moderate – depends on proxy capacity.
<b>Deployment Feasibility</b>	Easy – implemented in firewalls, routers.	Easy – network-level DNS filtering.	Hard – requires endpoint configuration.	Moderate – requires central proxy infrastructure.
<b>Detection Method</b>	Blocks based on IP addresses.	Blocks based on domain names.	Inspects decrypted TLS traffic.	Analyzes full HTTP/S requests.
<b>Cost</b>	Low – simple IP blocklists.	Low – free/open DNS filtering services exist.	High – requires decryption infrastructure.	Variable – self-hosted vs. commercial solutions.
<b>Privacy &amp; Compliance</b>	High – does not inspect content.	High – only blocks domains, not content.	Low – raises privacy concerns due to decryption.	Moderate – proxy logs may be intrusive.

### III. IMPLEMENTATION

This section describes the practical implementation of the various techniques used both for filtering mechanisms and their circumvention discussed in the third part of our article. Each component was developed as a proof-of-concept to demonstrate specific bypass or filtering strategies. All scripts were implemented in Python 3.11 under a Linux-based environment. The following Python libraries were used:

- dnspython: for dns queries
- requests: for DoH and HTTP communication
- ssl and socket: for TLS handshakes
- subprocess/os: for system commands

We run the test on a couple of domain names that were likely to be banned or filtered by a local ISP such as Torrents sharing or streaming websites but we also made sure to include well known sites from who the access would not be restricted for reference.

#### A. Bypassing DNS Filtering with Third-Party DNS

DNS filtering is one of the most common methods used by the ISPs and network administrators to block access to restricted websites.

To demonstrate this, we developed a script that attempts to resolve domain names using both the **default DNS resolver assigned by the ISP** and manually specified DNS providers such as **Google's 8.8.8.8** or **Cloudflare's 1.1.1.1**. If the domain is being filtered at the DNS level, the resolution with the ISP's DNS fails (the program will output an error), while the resolution with a public DNS server succeeds.

Methodology:

- 1) The script attempts to read the local DNS resolver used by the system (usually provided by the ISP) from the `/etc/resolv.conf`
- 2) Using the `dns.resolver` library, the script attempts to resolve each domain that we provide in a predefined list, it then returns the ip address if the resolution was successful and an error otherwise.

1) *Results:* During testing, several domains failed to resolve using the ISP's default DNS server. However, when using a third-party DNS resolver the same domains resolved correctly, returning a valid IP address. This behaviour can be explained by the practice of ISP's restricting access to certain categories of websites, most commonly **torrenting platforms, piracy-related content, or adult sites**.

#### B. Bypassing DNS Filtering using DoH

To further investigate the bypassing of DNS filtering, we tested **DNS-over-HTTPS** as an alternative method to resolve domains that failed using the default DNS server.

For this test, we used **Cloudflare's DoH service** and we concluded that the same domains that previously failed to resolve using the ISP's default DNS successfully resolved and returned valid IP addresses, indicating that DoH was able to bypass the DNS filtering imposed by the default ISP server. However, not all websites support **DoH** and can only be

resolved through traditional DNS. This can lead to issues where DoH may not be effective for certain domains.

Bypassing

#### C. Bypassing SNI Filtering

To investigate if a domain might be filtered by its **SNI certificate**, we attempted to create a TLS handshake to see whether the server returns a valid certificate for the domain.

We implemented the test using the following approach:

- 1) First we use the `supports_https` function to verify if the domain supports HTTPS by attempting a standard TCP connection on port 443
- 2) Then, if the domain supports https we use a function to create a secure connection and to attempt to get the **SNI certificate**
- 3) If the **TLS handshake** is successful, we get the certificate and we print its **Common Name**
- 4) Otherwise, if the handshake fails, this could indicate that the domain is being filtered

1) *Results:* From the domain that we tested we have not found a domain that was filtered at the SNI level. This could be explained by the fact that **SNI filtering** usually is combined with other filtering techniques and is used mostly in corporate environment, making it difficult to test unless we implement the filtering ourselves.

This method of detecting SNI filtering is also not bulletproof, we may get false positive if there could be issues with the server of the network that would lead to connection problems, so additional tests and techniques may be needed to confirm the presence of filtering at this level.

#### D. Proxy Filtering on Domain Names

To test **proxy-based filtering** we used **mitmproxy**, a free and open source interactive HTTPS proxy that allows us to inspect and modify HTTP/HTTPS traffic in real time. We used mitmproxy along with a custom script that is supposed to emulate the way a proxy server might filter access to specific websites based on their domain names.

Methodology:

- 1) The `request` function intercepts each HTTP request and check if the requested domain is in a blocklist that we manually composed
- 2) If the domain is blocked, the proxy sends a **403 Forbidden** response and states that the domain is blocked
- 3) If the domain is allowed, the proxy allows access and responds with code 200

1) *Results:* We noticed the expected behaviour, a series of requests were made through the proxy to different target URL and those that were part of the blocked URLs list were successfully blocked and the rest of the websites were allowed. It was also possible to filter based on content type, URL parameters with this type of approach. Therefore, using a proxy can be an effective way to control access to specific websites, especially in corporate environments.



### E. IP and DNS filtering bypass using VPN

1) *IP filtering bypass*: In this part we looked after the usage of virtual private networks for bypassing filtering techniques. Here we didn't use any script nor local proxies. The hyview client provided by DIATEAM allowed us to connect to a network topology provided by Thibault Cholez, described as follows. (fig. 7) The pfsense firewall (fig. 8) is linked to the network at the top, and with a switch at the bottom, linked with both a linux server and client. We are able to make packet inspection on a machine as desired. In fact, only the client will be used.

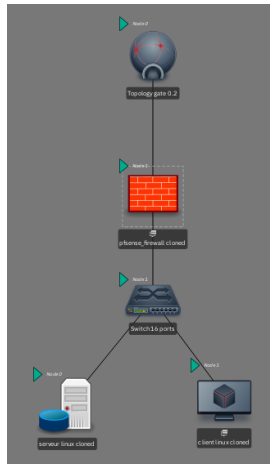


Fig. 7. The set of virtual machines we used.

```
Starting syslog...done.
Starting CROM...done.
pfSense 2.4.2-RELEASE (Patch 1) amd64 Tue Dec 12 13:45:26 CST 2017
Bootup complete

FreeBSD/amd64 (pfSense.localdomain) (ttyv0)

pfSense - Netgate Device ID: 2bf82b895d6a18d33b92

*** Welcome to pfSense 2.4.2-RELEASE-p1 (amd64) on pfSense ***

WAN (wan)      -> em0      -> v4/DHCP4: 172.16.1.220/20
LAN (lan)      -> em1      -> v4: 192.168.1.1/24

0) Logout (SSH only)          9) pfTop
1) Assign Interfaces          10) Filter Logs
2) Set interface(s) IP address 11) Restart webConfigurator
3) Reset webConfigurator password 12) PHP shell + pfSense tools
4) Reset to factory defaults    13) Update from console
5) Reboot system              14) Enable Secure Shell (ssh)
6) Halt system                15) Restore recent configuration
7) Ping host                  16) Restart PHP-FPM
8) Shell

Enter an option: █
```

Fig. 8. The pfsense firewall.

The client is set to only use the DNS Resolver of the pfsense router. (nameserver 192.168.1.1 in the /etc/resolv.conf file) Using nslookup with youtube.com, we get the resolved IP of youtube.com: 216.58.215.46. It is then easy to block this specific IP destination from the pfsense web interface. (fig. 9)

After pfsense and the linux machines are reset, writing "ping 216.58.215.46" on the client obviously results in a full packet loss. (fig. 10)

Using openvpn, we then set up a vpn after creating an account on protonvpn.com. The installation of the openvpn

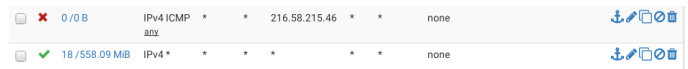


Fig. 9. The set of rules of the LAN interface.

```
hns@debian:~$ ping 216.58.215.46
PING 216.58.215.46 (216.58.215.46) 56(84) bytes of data.
^C
--- 216.58.215.46 ping statistics ---
180 packets transmitted, 0 received, 100% packet loss, time 180432ms
```

Fig. 10. The pfsense interface filters the youtube IP.

binaries was a little hard because of the version of the client running on debian jessie (needed a manual install of openssl and lz4). We finally set up the VPN connection (fig. 11), the ping is now working (fig. 12) and it is consistent that the client uses the VPN server 87.249.134.133 on port 7770.







```
[sudo] password for hns:
2025-05-14 16:58:49 Multiple --up scripts defined. The previously configured script i
s overridden.
2025-05-14 16:58:49 Multiple --down scripts defined. The previously configured scrip
t is overridden.
2025-05-14 16:58:49 OpenVPN 2.6.11 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EP
OLL] [MH/TKINTFO] [AEAD]
2025-05-14 16:58:49 library versions: OpenSSL 1.0.2u 20 Dec 2019, LZO 2.08
Enter Auth Username:198i107fsl96tXRh
Enter Auth Password:
2025-05-14 16:59:14 NOTE: the current --script-security setting may allow this configu
ration to call user-defined scripts
2025-05-14 16:59:14 TCP/UDP: Preserving recently used remote address: [AF_INET]87.249.
134.133:7770
2025-05-14 16:59:14 Attempting to establish TCP connection with [AF_INET]87.249.134.13
3:7770
2025-05-14 16:59:14 TCP connection established with [AF_INET]87.249.134.133:7770
2025-05-14 16:59:14 TCPv4 CLIENT link local: (not bound)
2025-05-14 16:59:14 TCPv4 CLIENT link remote: [AF_INET]87.249.134.133:7770
2025-05-14 16:59:15 WARNING: this configuration may cache passwords in memory -- use t
he auth-nocache option to prevent this
2025-05-14 16:59:15 [node-us-234.protonvpn.net] Peer Connection Initiated with [AF_INE
T]87.249.134.133:7770
2025-05-14 16:59:16 TUN/TAP device tun0 opened
2025-05-14 16:59:16 net iface mtu set: mtu 1500 for tun0
2025-05-14 16:59:16 net iface up: set tun0 up
2025-05-14 16:59:16 net addr v4 add: 10.98.0.4/16 dev tun0
2025-05-14 16:59:16 /etc/openvpn/update-resolv-conf tun0 1500 0 10.98.0.4 255.255.0.0
init
2025-05-14 16:59:16 Initialization Sequence Completed
```

Fig. 11. The VPN connection is established.

```
hns@debian:~$ ping 216.58.215.46
PING 216.58.215.46 (216.58.215.46) 56(84) bytes of data.
64 bytes from 216.58.215.46: icmp_seq=1 ttl=114 time=186 ms
64 bytes from 216.58.215.46: icmp_seq=2 ttl=114 time=186 ms
64 bytes from 216.58.215.46: icmp_seq=3 ttl=114 time=186 ms
64 bytes from 216.58.215.46: icmp_seq=4 ttl=114 time=188 ms
^C
--- 216.58.215.46 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 186.210/186.783/188.160/0.959 ms
hns@debian:~$ █
```

Fig. 12. Ping working.

2) *DNS filtering bypass*: Here we will now modify the DNS resolve of youtube.com using the pfsense interface. (fig. 13) Youtube.com is now resolved by the client as 127.0.0.1. Thus a telnet request using port 443 will fail.

Host Overrides				
Host	Parent domain of host	IP to return for host	Description	Actions
client	localdomain	192.168.1.101	client linux	<a href="#"></a> <a href="#"></a>
server	localdomain	192.168.1.102		<a href="#"></a> <a href="#"></a>
youtube	com	127.0.0.1		<a href="#"></a> <a href="#"></a>



Technique	Objective	Tools/Libraries	Methodology	Results
DNS Filtering Bypass (Public DNS)	Bypass ISP default DNS filtering	dnspython	Resolve domains using both ISP's DNS and public resolvers (8.8.8.8, 1.1.1.1)	Some domains fail with ISP DNS but succeed with public resolvers
DNS-over-HTTPS (DoH)	Encrypt DNS queries to bypass filtering	requests, DoH API	Query Cloudflare's DoH server for previously blocked domains	Blocked domains resolve correctly through DoH
SNI Filtering Test	Detect filtering based on SNI/TLS handshake	ssl, socket	Attempt TLS handshake and fetch the server certificate	No domains found to be blocked; method may produce false positives
Proxy Filtering	Simulate HTTP/HTTPS domain filtering via proxy	mitmproxy, Python script	Intercept requests and block based on a blacklist	Blacklisted domains blocked (403), others allowed (200)
IP Filtering via Firewall	Block access to a specific IP address (e.g., YouTube)	pfSense (GUI)	Add firewall rule to block outgoing traffic to a specific IP	Ping fails with 100% packet loss
Bypass IP Filtering via VPN	Tunnel traffic to bypass IP filtering	OpenVPN, ProtonVPN	Connect to VPN to route traffic through external IP	Ping succeeds via VPN, bypassing firewall restriction
DNS Filtering via Hosts (pfSense)	Simulate malicious DNS resolution (e.g., youtube.com → 127.0.0.1)	pfSense DNS Resolver	Add custom DNS entry redirecting domain to localhost	DNS resolution points to 127.0.0.1, blocking access
Bypass DNS Filtering via VPN DNS	Restore correct DNS resolution via VPN's resolver	OpenVPN, update-resolv-conf	Update /etc/resolv.conf automatically upon VPN connection	Correct DNS resolution restored; filtering bypassed

#### IV. CONCLUSION

Web filtering remains a vital mechanism for enforcing access control over internet content, whether for regulatory, security, or organizational reasons. Each filtering technique brings specific advantages and limitations: IP filtering is fast but overly broad, DNS filtering is simple but easily bypassed, proxies offer granular control but at performance costs, and SNI filtering provides balance but is increasingly undermined by encryption advancements like ESNI and ECH. Our analysis shows that while these techniques are effective to varying degrees, most can be circumvented using widely available tools such as VPNs, custom DNS resolvers, or encrypted protocols. As privacy-enhancing technologies continue to evolve, particularly with the widespread adoption of TLS 1.3 and encrypted DNS, traditional filtering mechanisms face growing challenges. Future filtering systems must adapt to these developments by combining techniques or integrating machine learning to maintain effectiveness without overblocking or infringing on privacy.

#### ACKNOWLEDGMENT

We wish to thank Thibault Cholez, our tutor, for the help he provided to us during researches and practical experiments, and also TELECOM Nancy for allowing us to make this paper.

#### REFERENCES

- [1] W. Stol, H. Kaspersen, J. Kerstens, E. Leukfeldt, and A. Lodder, "Governmental filtering of websites: The dutch case," *Computer Law Security Review*, vol. 25, no. 3, pp. 251–262, 2009. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0267364909000582>
- [2] R. Deibert, J. Palfrey, R. Rohozinski, and J. L. Zittrain, *Access Denied: The Practice and Policy of Global Internet Filtering*. The MIT Press, 01 2008. [Online]. Available: <https://doi.org/10.7551/mitpress/7617.001.0001>
- [3] M. Fejrskov, E. Vasilomanolakis, and J. M. Pedersen, "A study on the use of 3rd party dns resolvers for malware filtering or censorship circumvention," in *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer, 2022, pp. 109–125.
- [4] S. A. Samarakoon, "Bypassing content-based internet packages with an ssl/tls tunnel, sni spoofing, and dns spoofing," 2022. [Online]. Available: <https://arxiv.org/abs/2212.05447>
- [5] Z. Chai, A. Ghafari, and A. Houmansadr, "On the importance of Encrypted-SNI (ESNI) to censorship circumvention," in *9th USENIX Workshop on Free and Open Communications on the Internet (FOCI 19)*. Santa Clara, CA: USENIX Association, Aug. 2019. [Online]. Available: <https://www.usenix.org/conference/foci19/presentation/chai>
- [6] W. M. Shbair, T. Cholez, A. Goichot, and I. Chrisment, "Efficiently bypassing sni-based https filtering," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 990–995.