

**Information extraction of devices behind NAT using  
WebRTC**

Shivasankaran V P  
19110104

**Abstract:**

IP addresses are the core part of the modern Internet, and they are necessary to communicate with other devices on the network. The Internet is expanding at a tremendous rate, and along with it, security flaws are also growing. Malicious users on the Internet constantly exploit these security flaws and take advantage of them, and in most cases, they cannot be identified. Identifying users on the Internet is a complex task mainly due to the number of layers in the network and technologies like VPNs, which specifically provide services to hide a user's identity. The first layer is the NAT, and there is no direct method to identify the user behind the NAT. This article will discuss a solution to find information about the user, including the private IP behind a NAT device from a browser session. The solution exploits WebRTC, which is a popular technology used in peer-peer communication without any intermediary server. The method is demonstrated on a virtual network in Mininet with multiple hosts trying to communicate with a server through a NAT. The feasibility of the solution is shown by comparing results across popular browsers, and a network map is formed displaying all the private IP addresses hidden behind the Public IP of NAT.

**Section 1 - Introduction**

Every device on the Internet is given unique identifiers called Internet protocol (IP) addresses. With the help of these IP addresses, each node can communicate over the network to other nodes. IP version 4 (IPv4) addresses were assigned to nodes at the start of the Internet, and there are a total of  $2^{32}$  IPv4 addresses. The depletion of IPv4 addresses has been anticipated since the late 19th century, and various new technologies like network address translator (NAT) was developed as a workaround to the issue of depletion<sup>1</sup>.

A NAT device performs network address translation from one address space to another. With the help of NAT, it was possible to create several private networks with their own IP address space, which are unique within the network and communicate with the Internet through the NAT<sup>2</sup>. When a node sends a packet to anywhere outside its private network, it goes via the NAT, and the NAT sends the packet to the corresponding destination with its assigned public IPv4 address on the Internet. When the response from the destination comes back, the NAT again maps and forwards the packet to the correct node within its private network. This solves the problem of running out of IPv4 addresses as now multiple devices could communicate with the Internet over just a single IP, and this IP is called Public IP, and the IP of the node within its network is called its private IP. A few things to note are the destination does not know the real source of the packet. It only knows the Public IP address of the NAT which forwarded the packet. So for

---

<sup>1</sup> "Available Pool of Unallocated IPv4 Internet Addresses Now ... - icann." 3 Feb. 2011, <https://www.icann.org/en/news/releases/release-03feb11-en.pdf>. Accessed 27 Nov. 2021.

<sup>2</sup> "rfc2663 - IETF Tools." <https://tools.ietf.org/html/rfc2663>. Accessed 27 Nov. 2021.

the destination, it will assume that the NAT is the source, and effectively the real source behind the NAT gets hidden from the Internet. This feature of NAT has its pros and cons. NAT does provide privacy to the user, and the server has very restricted knowledge about the user in many ways. On the contrary, this feature could be misused. There is no direct way to track the source from which a connection came from, and users could use this anonymity that NAT provides to do illegal activities on the Internet.

Several other factors affect traceability on the Internet, and it's out of the scope of this article. In this article, a simple network topology shown in Fig 4 was used and also assuming that the communication between the nodes happens over a Hypertext Transfer Protocol (HTTP).

The method discussed exploits the famous Web Real-Time Communication (WebRTC), which is a technology developed for peer-peer communication (P2P) without involving any intermediary server. WebRTC uses an Interactive connectivity establishment protocol (ICE)<sup>3</sup>. In order to establish P2P communication, ICE gathers information about its network and the local IP of the machine. The script tricks the browser into giving this information<sup>4</sup>.

The remainder of this article is organized as follows:

Section 2 discusses WebRTC and related concepts.

Section 3 talks about the solution and working methodology.

Section 4 describes the working setup which will be required to replicate the results.

Section 5 showcases the key findings and results.

Section 6 concludes the report.

## **Section 2.1 - STUN and TURN protocols**

Session Traversal Utilities for NAT (STUN) enables a user to find his public IP and the allocated public port by NAT. The STUN protocol can also be used to keep the NAT bindings. A STUN server is a node that runs the STUN protocol and serves clients with their public IP and port information<sup>5</sup>.

Traversal Using Relays around NAT (TURN) has the same basic functionality as STUN protocol. However, besides providing the public IP and port allotted by the NAT, it also acts as a relay server<sup>6</sup>. We will see why this relay server functionality is important in section 2.2. These STUN and TURN servers are deployed all over the Internet. For example, `stun.l.google.com:19302` is a STUN server deployed by google.

---

<sup>3</sup> "rfc7478 - IETF Tools." <https://tools.ietf.org/html/rfc7478>. Accessed 27 Nov. 2021.

<sup>4</sup> "rfc8445 - IETF Tools." <https://tools.ietf.org/html/rfc8445>. Accessed 27 Nov. 2021.

<sup>5</sup> "rfc5389 - IETF Tools." <https://tools.ietf.org/html/rfc5389>. Accessed 27 Nov. 2021.

<sup>6</sup> "rfc5766 - IETF Tools." <https://tools.ietf.org/html/rfc5766>. Accessed 27 Nov. 2021.

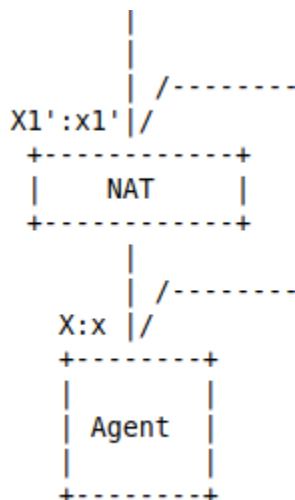


Fig 1:- A STUN or TURN request to an external STUN or TURN server will give the response as  $X1':x1'$   
Image is taken from [src](#)

## Section 2.2 - Interactive connectivity establishment protocol

More commonly known as ICE, interactive connectivity establishment protocol is widely used in setting up P2P connections between two nodes behind NAT. RFC 8445 describes ICE protocol. The core working principle of ICE protocol is finding a pair of IP addresses, and port numbers jointly referred to as a candidate. Without the interference of a NAT device, the straightforward candidate would be the IP address of its network interface and suitable port, but since the node is present behind a NAT device, ICE scans for multiple possible candidates in the network through which a connection could be established with the node.<sup>7</sup>

Server-reflexive candidates are those candidates which correspond to the translated addresses in the public side of NAT. In order to obtain the server-reflexive candidates, a STUN request or a TURN request is sent to the corresponding STUN or TURN server. As the TURN server acts as a relay server, the addresses of relayed candidates act as valid candidates for the node.

<sup>7</sup> "rfc8445 - IETF Tools." <https://tools.ietf.org/html/rfc8445>. Accessed 27 Nov. 2021.

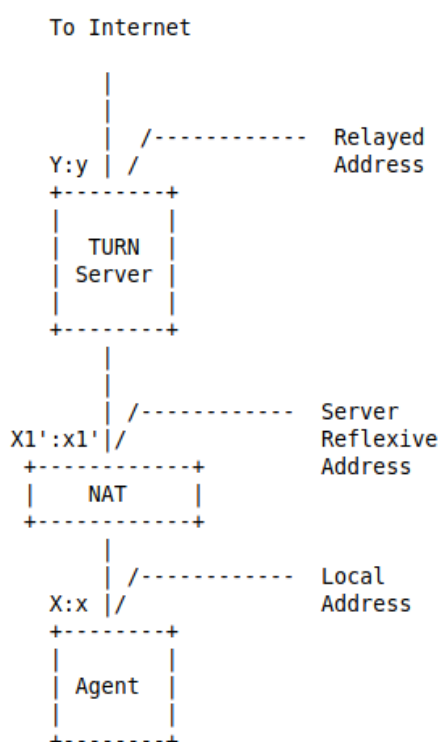


Fig 2:- Depicts different types of candidates; Local candidate (X,x); Server reflexive candidate (X1'x1'); Relayed candidate (Y,y)  
Image is taken from [src](#)

The same process is followed for the other node to which a P2P connection has to be established. Once the nodes gather all candidates for the connection, the lists are shared using Session Initiation Protocol (SIP)<sup>8</sup>. Each node iteratively tries to connect to one of the candidates of the other node. Once a valid candidate pair is found P2P connection is established if no valid candidate pair is found, then a P2P connection cannot be established.

One important observation is that the local IP is a compulsory candidate in the candidate list independent of the type of router, network topology etc. The candidate list is only retrieved and shared when another node wants to set up a P2P connection, more commonly known as an “offer”. The usual flow of the process is as follows the user would visit a URL of a server providing a certain service. The server would then act as a temporary intermediary node to start the P2P connection process between the two nodes.

<sup>8</sup> "rfc3261 - IETF Tools." <https://tools.ietf.org/html/rfc3261>. Accessed 27 Nov. 2021.

### Section 2.3 - Web Real-Time Communication

WebRTC is an open-source project that allows real-time communication between two browsers without an external server or plugins<sup>9</sup>. WebRTC workflow is as follows when two users connect to a web service through their browsers and want to transfer data between them, then the server which provides the web services initiates a peer-peer connection between the two users. This peer-peer connection is established using the ICE protocol mentioned in section 2.2, where one of the users creates an ICE offer to the other user. ICE sets up the channel, and data is transferred between the two users using this channel. Services like VoIP calls, live video conferencing use WebRTC to set up communication.

### Section 3 - Working methodology

The method exploits a technical flaw in WebRTC, to be more specific, an important part of the ICE protocol which WebRTC uses to set up its communication channel. When the ICE protocol gathers its candidate list, the first candidate is always the local IP address. As we saw in section 2.2 that the server initiates the “offer” in one client to the other, we could use a script to replicate this behaviour and create a fake offer from the script to the browser and tricking the browser into revealing its candidates.

A custom script was written in javascript using libraries that support ICE protocols. A peerConnection object is created, which is then bonded with the local machine. The ICE protocol starts gathering its candidates, and once it finds its first candidate, we immediately close the connection and process the candidate which will belong to the local IP address of the system.

2018 onwards, browsers no longer show the IP address in a raw text format. Instead, they are encrypted into .local objects. The workaround to this problem is that allowing media permissions to the website forces ICE protocol to reveal its candidate details in raw text format. This exploit was used in gaining information about the users local IP address and information about the network to which the user is connected<sup>10</sup>.

---

<sup>9</sup> "WebRTC." <https://webrtc.org/>. Accessed 27 Nov. 2021.

<sup>10</sup> "WebRTC IP Address Handling Recommendations - IETF Tools." <https://tools.ietf.org/id/draft-shieh-rtcweb-ip-handling-00.xml>. Accessed 27 Nov. 2021.

```

(16) ['candidate:343882694', '1', 'udp', '2122260223', '10.7.11.38', '39
▼334', 'typ', 'host', 'generation', '0', 'ufrag', '358f', 'network-id',
'I', 'network-cost', '10'] ⓘ
  0: "candidate:343882694"
  1: "1"
  2: "udp"
  3: "2122260223"
  4: "10.7.11.38"
  5: "39334"
  6: "typ"
  7: "host"
  8: "generation"
  9: "0"
 10: "ufrag"
 11: "358f"
 12: "network-id"
 13: "1"
 14: "network-cost"
 15: "10"
  length: 16
▶ [[Prototype]]: Array(0)

```

Fig 3:- An example Candidate of ICE protocol. Element 3 and 4 indicate the IP address and port number of the candidate

## Section 4 - Experimental setup and tools used

The tools required to replicate the results presented in this report

- Mininet VM
- Python
- Flask development environment
- HTML
- Javascript
- Sqlite3
- Browsers
- [IPinfo.io](#). API.

The virtual network topology simulated in Mininet is shown in Fig 4

The project could be found [here](#)

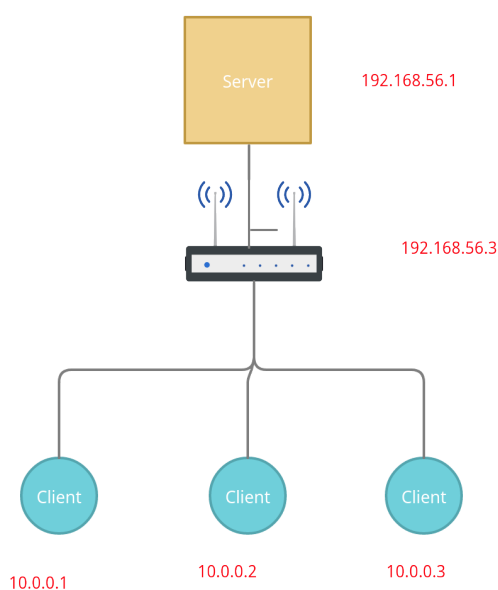


Fig 4:- Network topology

All the experiments were done in a 64-bit system.

Steps to set up the environment:-

- Install Google-chrome, Opera, Brave, Firefox, Microsoft-edge via their respective installation procedures for Ubuntu 64 bit system.
- Start the Flask app at 192.168.56.1:5000
- Create a single network topology with 3 hosts and 1 switch in the Mininet using the following command
  - Sudo mn --nat --topo single,3
- SSH into one of the hosts and run a browser of your choice and visit 192.168.56.1:5000/MyIP.
- Click the button which should display the public IP address of the host and send the private IP address and client system information to the server. If the private IP address is encrypted then it would ask for microphone permission.
- The information gets stored in a SQL database after querying [IPinfo.io](https://ipinfo.io) with the obtained public IP address. The database can be accessed by visiting 192.168.56.1:5000/admin.



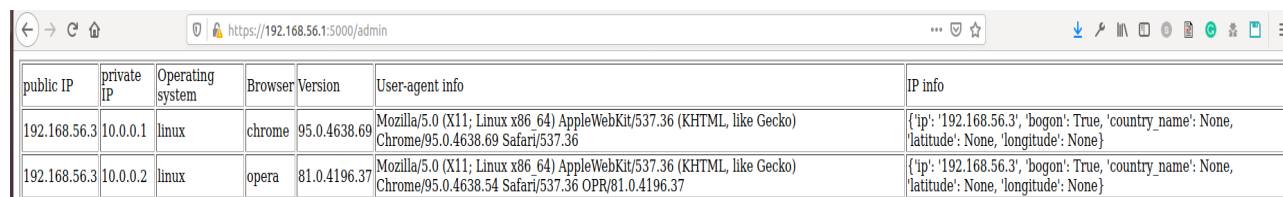
## Section 5 - Key findings

Comparison between different browsers:

	WebRTC supported	Anonymize local IP By default	Anonymize local IP For media channel
Google chrome (v95.0.4638.69)	Yes	Yes	No
Microsoft edge (v97.0.1069.0)	Yes	Yes	No
Opera (v81.0.4196.37)	Yes	Yes	No
Brave (v1.32.106)	Yes	Yes	No
Firefox (v94.0)	Yes	Yes	No

Table 1: Shows the results across different browsers

Extracted client data:



The screenshot shows a web browser window with the address bar displaying 'https://192.168.56.1:5000/admin'. The main content area contains a table with the following data:

public IP	private IP	Operating system	Browser	Version	User-agent info	IP info
192.168.56.3	10.0.0.1	linux	chrome	95.0.4638.69	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Safari/537.36	{'ip': '192.168.56.3', 'bogon': True, 'country_name': None, 'latitude': None, 'longitude': None}
192.168.56.3	10.0.0.2	linux	opera	81.0.4196.37	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.54 Safari/537.36 OPR/81.0.4196.37	{'ip': '192.168.56.3', 'bogon': True, 'country_name': None, 'latitude': None, 'longitude': None}

Fig 5: Extracted client data when viewed from the admin page

## Section 6 - Conclusion

This report showcases the amount of information a server could get about the client without the client noticing anything suspicious beyond microphone permissions. We were able to extract information that could be used to find the physical location of the machine.

From table 1, it is clear that browsers do know about this exploit and have Anonymized local IP addresses but giving media permissions reveals the local IP address. WebRTC is considered the next Adobe flash due to the number of possible exploits that could be done. Only recently, vulnerabilities in WebRTC have been getting the spotlight. Only after 2018 do browsers anonymize local IPs by default stopping any IP leaks. WebRTC is one of the most important web technologies in use as it makes the process of setting P2P connections smooth without worrying about the end-user systems. On April 7 2021, WebRTC became the official web standard for audio and video communication<sup>11</sup>, and if anything from here on, the use of WebRTC will rise and widen. During this boom, It is equally important to track the vulnerabilities of WebRTC even though they may not look that deadly and educate people on the same. WebRTC is not a very stable technology in the long run, mainly because ICE protocol gathers information about the system and network. This information could be leaked in different ways if we are not careful.

### Related work

Al-Fannah, Nasser Mohammed. "One leak will sink a ship: WebRTC IP address leaks." In *2017 International Carnahan Conference on Security Technology (ICCST)*, pp. 1-5. IEEE, 2017.

Reiter, Andreas, and Alexander Marsalek. "WebRTC: your privacy is at risk." In *Proceedings of the Symposium on Applied Computing*, pp. 664-669. 2017.

Johnston, Alan, John Yoakum, and Kundan Singh. "Taking on webRTC in an enterprise." *IEEE Communications Magazine* 51, no. 4 (2013): 48-54.

Rescorla, Eric. "WebRTC security architecture." *Work in Progress* (2013).

Feher, Ben, Lior Sidi, Asaf Shabtai, and Rami Puzis. "The security of WebRTC." *arXiv preprint arXiv:1601.00184* (2016).

---

<sup>11</sup> "WebRTC IP Address Handling Recommendations - IETF Tools." <https://tools.ietf.org/id/draft-shieh-rtcweb-ip-handling-00.xml>. Accessed 27 Nov. 2021.

## References

1. <https://datatracker.ietf.org/doc/html/rfc7478>
2. <https://datatracker.ietf.org/doc/html/rfc8445>
3. <https://datatracker.ietf.org/doc/html/rfc5389>
4. <https://datatracker.ietf.org/doc/html/rfc5766>
5. [https://www.tutorialspoint.com/webrtc/webrtc\\_rtcpeerconnection\\_apis.htm](https://www.tutorialspoint.com/webrtc/webrtc_rtcpeerconnection_apis.htm)
6. <https://www.article19.org/resources/privacy-and-consent-in-the-age-of-browsers-the-question-of-webrtc/>
7. <https://www.icann.org/en/news/releases/release-03feb11-en.pdf>