Bypassing Tunnels:
Leaking VPN Client Traffic by Abusing Routing Tables

Nian Xue, Yashaswi Malla, Zihang Xia, Christina Pöpper, and Mathy Vanhoef

USENIX Security, 9-11 August 2023, USA
Usage of VPNs: watch videos from other country
Usage of VPNs: protect your traffic

› Identify website visits: IP address, plaintext DNS, SNI,…
› Attack TLS: no cert check, sslstrip, academic attacks,…
Usage of VPNs: protect your traffic

› Defend against untrusted Wi-Fi & compromised core routers

› Research goal: can we trick the client into leaking packets?
  » Yes, by manipulating the client’s routing table \(\rightarrow\) \(\sim66\%\) vulnerable!
  » Attacks are independent of the crypto protocol
1. By default, send packets over tun0 = over the VPN tunnel
1. By default, send packets over tun0 = over the VPN tunnel

2. **LocalNet exception**: local network is directly accessible
Background: VPN client routing table

$ ip route
1  default via tun0
2  192.168.1.0/24 via eth0
3  2.2.2.2 via eth0

1. By default, send packets over tun0 = over the VPN tunnel
2. **LocalNet exception**: local network is directly accessible
3. **ServerIP exception**: avoid re-encryption of VPN packets
We assume secure DNS behavior

Can’t trust the network’s DNS server

```
$ cat /etc/resolv.conf
nameserver 6.6.6.6
```
We assume secure DNS behavior

Can’t trust the network’s DNS server. Once connected:

1. The VPN client sets a trusted DNS server
2. DNS is sent through the VPN tunnel
   + we assume other routing-based attacks are prevented

```bash
$ cat /etc/resolv.conf
nameserver 2.2.2.3
```
LocalNet attack

Local network is 1.2.3.0/24

Create VPN tunnel with 2.2.2.2

Set trusted DNS server

default via tun0
1.2.3.0/24 via eth0

Target.com

1.2.3.4

2.2.2.2
LocalNet attack

default via tun0
1.2.3.0/24 via eth0

Visit random.com

Visit target.com

Send to 1.2.3.4

Intercept traffic!
LocalNet attack: 195 experiments

<table>
<thead>
<tr>
<th>VPN Provider</th>
<th>Class</th>
<th>OS</th>
<th>Version Number</th>
<th>LAN Setting</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Default LAN Access</td>
<td></td>
</tr>
<tr>
<td>OS Built-in VPN</td>
<td>Free</td>
<td>Windows</td>
<td>Windows 10 Pro</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Free</td>
<td>Windows</td>
<td>Windows 11 Pro</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Free</td>
<td>macOS</td>
<td>Ventura 13.0.1</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Free</td>
<td>iOS</td>
<td>iOS 16.1.1</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Free</td>
<td>Android</td>
<td>Android 8.1.0</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Free</td>
<td>Android</td>
<td>Android 12</td>
<td>No</td>
<td>N/A</td>
</tr>
</tbody>
</table>

1.1.1.1

|               | Free | Windows | 2022.10.106.0 | No | N/A | ✔ | ![Image](image.png) |
|               | Free | Linux   | 2022.9.591   | No | N/A | ✔ | ![Image](image.png) |
|               | Free | macOS   | 2022.10.107.0 | No | N/A | △ | ![Image](image.png) |
|               | Free | iOS     | 6.16          | No | N/A | ❌ | ![Image](image.png) |
|               | Free | Android | 6.17          | No | N/A | ✔ | ![Image](image.png) |
LocalNet attack: 195 experiments

<table>
<thead>
<tr>
<th>Network Manager</th>
<th>Operating System</th>
<th>Version</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>EvolveVPN</td>
<td>Windows</td>
<td>12.37.0</td>
<td>Yes</td>
</tr>
<tr>
<td>EvolveVPN</td>
<td>Linux</td>
<td>3.36</td>
<td>No</td>
</tr>
<tr>
<td>ExpressVPN</td>
<td>macOS</td>
<td>11.12.0</td>
<td>Yes</td>
</tr>
<tr>
<td>ExpressVPN</td>
<td>iOS</td>
<td>11.70.0</td>
<td>Yes</td>
</tr>
<tr>
<td>ExpressVPN</td>
<td>Android</td>
<td>10.63.2</td>
<td>Yes</td>
</tr>
<tr>
<td>VPN Proxy Master for iPhone</td>
<td>Free</td>
<td>iOS</td>
<td>2.1.5</td>
</tr>
</tbody>
</table>
LocalNet attack: summary

- **Android**: 21.4% Vulnerable, 12.6% Blocks non-RFC1918 local traffic, 56% Secure
- **Linux**: 35.7% Vulnerable, 2.3% Blocks non-RFC1918 local traffic, 62% Secure
- **Windows**: 66.7% Vulnerable, 13.3% Blocks non-RFC1918 local traffic, 20% Secure
- **macOS**: 87.5% Vulnerable, 5% Blocks non-RFC1918 local traffic, 7.5% Secure
- **iOS**: 100% Vulnerable, 0% Blocks non-RFC1918 local traffic, 0% Secure
The iOS case

Attacks can be prevented by setting `includeAllNetworks`
  › But causes reliability issues
  › Vendors very hesitant to enable it

Result is that **iOS remains less secure**
  › Context: VPNs on iOS were already known to leak traffic in certain scenarios.
ServerIP attack

DNS request for vpn.com
Spoof DNS reply: 1.2.3.4
Create VPN tunnel with 1.2.3.4
Set trusted DNS server
default via tun0
1.2.3.4 via eth0
Redirect to 2.2.2.2

Target.com
1.2.3.4
2.2.2.2
ServerIP attack

default via tun0
1.2.3.4 via eth0

Target.com
1.2.3.4

2.2.2.2

VPN
ServerIP attack

default via tun0
1.2.3.4 via eth0

Visit random.com

Visit target.com

Send to 1.2.3.4

Intercept traffic!
ServerIP attack: 53 experiments

› Many **built-in clients** are affected (Windows, macOS, Linux)
› Legacy built-in VPN on **Android 11 and below** was affected
› Most iOS/Android apps not vulnerable

Impact: can leak traffic to single IP address

› Can target the DNS server set by the VPN client 😊
› Or repeat the attack…
Defenses & Disclosure

1. **LocalNet attack**: disable local network access when it’s using public IP addresses

2. **ServerIP Attack**: send all traffic over VPN, except packets generated by VPN process

- Reported to CERT/CC on May 10, 2023
- Contacted vendors that had a security contact
- Practically all acknowledged the issue
Conclusion

- Two wide-spread flaws in VPN clients
- In hindsight easy attack, but ~66% vulnerable
- Bad integration of protocols into real systems

- Defense: more carefully configure routing tables
- OS should have API to create VPN tunnels
Questions?

› Two wide-spread flaws in VPN clients
› In hindsight easy attack, but ~66% vulnerable
› Bad integration of protocols into real systems

› Defense: more carefully configure routing tables
› OS should have API to create VPN tunnels