Review + Presentation of

OpenVPN is Open to VPN Fingerprinting

By Diwen Xue, Reethika Ramesh, Arham Jain, Michalis Kallitsis, J.Alex halderman, Jedidiah R. Crandall, Roya Ensafi

What is the problem?

VPN adoption has seen steady growth over the past decade due to increased public awareness of privacy and surveillance threats. In response various ISPs , advertisers and governments are now seeking to track or block VPN traffic in order to maintain visibility and control over the traffic within their jurisdiction. Examples:

1. Binxing Fang, the designer of the great firewall of China(GFW) said there is an “eternal war” between the Firewall and VPNs and the country has ordered ISPs to report and block personal VPN usage
2. ISPs, even less-powerful ones, now have access to technologies such as carrier-grade deep packet inspection(DPI) with which they can implement more sophisticated modes of detection based on protocol semantics.

In this paper, they seek to answer 2 research problems:

a. Can ISPs and governments identify traffic flows as OpenVPN connections in real time?
b. Can they do so at scale without incurring significant collateral damage from false positives?

Summary

- The paper develops mechanisms to accurately fingerprint OpenVPN connection based on 3 protocol features:

1. Byte Pattern
2. Packet Size
3. Server Response

- Playing the role of an attacker, they develop a 2 phased framework which performs passive fingerprinting followed by active proving, in sequence. This is inspired by the architecture of the Great Firewall consisting of *Filter* and *Prober* components.

A *filter* performs passive filtering over passive network trafficking real time, exploiting protocol quirks identified in the OpenVPN’s handshake stage

After a flow is flagged by a *Filter,* the IP and port information is passed to a *Prober* that performs active probing as confirmation. The probers send a set of pre-defined probes specifically designed to fingerprint an OpenVPN server.

Finally, probed servers that are confirmed as Openvpn servers are logged for manual analysis

This 2 phase framework is capable of processing ISP-scale traffic at line-speed with an extremely low positive rate. They evaluate their framework in partnership with a million user

ISP-MeritY and find that they identify 85% of OpenVPN connections with negligible false positives, suggesting that OpenVPN services can be effectively blocked with little collateral damage. Additionally the framework identifies 34 out of 41 obfuscated connections as well

Obfuscated VPN:
Obfuscated VPN services, whose operators often tout them as “invisible” and “unblockable”, typically use OpenVPN with an additional obfuscation layer to avoid detection. Techniques:

1. OpenVPN XOR patch: Originally developed by ClayFace, the XOR patch scrambles a packet by either XOR-ing the bytes with a pre-shared key, reversing the order of bytes, xoring each byte with its position, or a combination of these steps.
2. OpenVPN encrypted tunnels:
   Some VPN services wrap OpenVPN traffic inside encrypted tunnels to prevent DPI fingerprinting. Some of the obfuscation tunnels are Obfsproxy, Stunnel, Websocket tunnel etc
3. Proprietary protocols

- They identify two-thirds of obfuscated OpenVPN flows. Most of the obfuscated traffic resemble the vanilla open VPN with an XOR patch
- Lack of random padding and co-location with vanilla Openvpn servers make the obfuscated servers more vulnerable to detection.

(Presentation)**Technologies used:**

1. **Real World deployment setup, for filters:**

- For packet processing: We use PFRing (https://www.ntop.org/products/packet-capture/pf_ring/pf_ring-zc-zero-copy/) in xero-copy mode for fast packet processing by parallelized *Filters*. The framework is setup on a 16-core server inside *Merit* with 2 mirroring interfaces that have an aggregated 20 GBps bandwidth. Due to large traffic volume, they optimise the deployment with PF_RING in order to improve packet processing speed.They employ PF_RING in zero copy mode and spread the traffic load across a Zeek cluster of 15 workers.

To put things in perspective, event hough due to limited CPU resources they only sample 12.5% of all TCP and UDP flows arriving at the network interfaces in order to minimize the effect of packet loss, the filters still handle over **15 Terabytes of traffic from over 2 billion flows on an average day on a single server.**

2. Real world deployment setup, for probers:

- Nim for probers

(Presentation)**Filter:**
Filters are implemented in Zeek, an open source network monitoring tool. (https://zeek.org/)

1. Exploiting byte patterns: opcode based fingerprinting
   Key insight: **A packet field taking a fixed number of values can be easy to fingerprint**

   Each OpenVPN packet has a header of 24 bits in TCP mode and 8 bits in UDP mode, which is *not* part of the encrypted payload. Each of this header starts with an opcode that specifies the message type of the current packet and a key ID that refers to a new TLS session. This opcode field can take over 10 different values.
   A typical OpenVPN session starts with a client sending a **client reset** packet. The server then responds with a **server reset** packet, and a TLS handshake follows. OpenVPN packers that carry TLS ciphertext have **P_control** as their message type. Each P_control packet is explicitly acknowledged by **P_ack** packets. Finally actual payloads are transmitted as **P_data** packets. (atleast 5 opcodes)

   They fingerprint OpneVPN's handshake sequence by analysing each opcode byte for the first N packets of a flow. The filter flags a flow if the number of opcodes observed accords with the protocol(5+ opcodes) and the client and server resets are not seen once the handshake is complete.

2. Exploiting packet length: ACK-based fingerprinting
   Key insight: **For OpenVPN, the presence of explicit ACK packets, uniform in size and only seen in some parts of a session provides another fingerprintable feature.**

   OpenVPN engages in TLS style handshake with it's peer over the control channel. Since TLS is designed to operate over a reliable layer, OpenVPN implements an explicit acknowledgement and retransmission mechanism for its control channel messages. Specifically, incoming P_Contol packets are acknowledged by P_Ack packets, which do not carry any TLS payloads and are uniform in size. Moreover these ACK packets are seen mostly in the early stages of a flow, during the handshake phase, and are not used in the actual data transfer channel, which can run over an unreliable layer.

   Specifically, they identify a likely ACK packet of a session by locating an initial packet exchange sequence of C->S(client reset), S->C(server reset), C->S(ack), C->S(control). For vanilla open VPN packets and XOR-based obfuscation, the first ACK packet usually appears as the third(data) packet transmission of the session. For tunnels or obfuscators that have their own handshake or key exchange process(eg stunnel, SSH tunnel, or ObfsProxy) this counting is offset by the number of tunnel handshake packets.

   Don't understand: next we group packets into 10 packet bins….

3. Exploiting server behaviour

(Presentation)**Probers:**
Probers are implemented in Nim(https://nim-lang.org/). Probers are based on exploiting server behaviour.
Key Insight: Although an application may not respond to probing, an attacker may still be able to fingerprint application-specific thresholds at TCP level, such as timeouts or RST thresholds.

1. Typically, OpenVPN servers respond to a client rest with an explicit server reset, thereby giving away their identity.
2. However, most commercial providers now have adopted *tls-auth* or *tls-crypt* options. These add an additional HMAC signature-signed by a pre-shared key- to every control channel packet for integrity verification including initial reset packets. With either of these enabled an OpenVPN server would not respond to an unauthenticated client with a server reset, but drop such packets. Similar HMAC mechanisms are used by more popular “probe-resistance” proxies, such as obfs4. However, unlike obfs4 which waits for a server-specific random delay before dropping an unauthenticated connection, OpenVPN always *immediately* closes the connection if a valid HMAC cannot be located. **We use this to design 2 probes to trigger an OpenVPN server into different code paths - which results in different connection timeouts- and measure the time elapsed before the server responds or terminates the connection.**

    Base Probe 1 carries a typical 16-byte OpenVPN *client reset,* while *base probe 2,* has the same payload with the last byte stripped. The assumption is that since our 2 probes only differ on the last byte, non OpenVPN servers will respond to the 2 probes the same way, while OpenVPN servers with HMAC enabled with go down 2 different code paths for the 2 of them. Connection for the first probe will be dropped immediately because the OpenVPN packet is reassembled and a valid HMAC could not be located. The second probe will not receive an immediate response, as the server will wait for an additional byte to arrive for reassembly. The connection will stay idle until a server specific handshake timeout has passed, after which the connection will be dropped. As such, the first probe will be dropped at the decryption routine, while the second probe will be dropped at the packet reassembly routine.

Key Insights

- Unlike circumvention tools like Tor and Refraction networking, which employ sophisticated techniques to avoid detection, robust obfuscation techniques have been absent from the VPN ecosystem. In the long term, a cat-and-mouse game similar to that

between Great Firewall and Tor is imminent in the VPN ecosystem. Hence, there is a marked demand for an emerging class of services called “stealth” or “obfuscated” VPN, especially from users in countries with heavy censorship or laws against personal VPN usage.(project Idea??)

- **A packet field taking a fixed number of values can be easy to fingerprint**
- **For OpenVPN, the presence of explicit ACK packets, uniform in size and only seen in some parts of a session provides another fingerprintable feature.**
- Although an application may not respond to probing, an attacker may still be able to fingerprint application-specific thresholds at TCP level, such as timeouts or RST thresholds.
-

Strengths

- They evaluated the practicality of their framework in partnership with a mid-tier ISP.
- They put a lot of stress on Ethics, privacy and responsible disclosure since raw networ traffic contains real user’s data and is highly sensitive:

a. They cleared their research plan with university counsel ans IRB. Although IRB determined that the work is not regulated, they take measure to minimise potential risk for end user
b. *Fiter* analysed only the first payload byte, completely ignoring the remainder of the payload. The raw snapshot was never inspected by humans.
c. The logs were stored and analysed on server that is securely maintained by Merit

Limitations/Weaknesses

- Source code was not published or could not be published due to the fear of malicious agents

Summary of Key Results

- Tracking and blocking the use of Open-VPN, even with the most current obfuscation methods, is straightforward and within reach of any ISP or network operator, as well as any nation state adversary. Overall they identified 1718 out o 2000 vanilla flows corresponding to 39 out of 40 unique configurations
- 4 out of 5 VPN providers use XOR-based obfuscation which is easily fingerprintable. They identify over two thirds of all obfuscated flows, corresponding to 34 out of 41 obfuscated configurations.
- UDP and obfuscated servers often share infrastructure with vanilla TCP servers leaving them “guilty by association” and hence identifiable.

Open Questions?

- Can VPN providers develop more standardised obfuscation solutions, such as Pluggable transports
- Will the VPN ecosystem see the same cat-and-mouse game?

Tools:

1. Massscan : https://github.com/robertdavidgraham/masscan
   TCP port scanner, spews SYN packets asynchronously, scanning entire Internet in under 5 minutes
2. https://github.com/ntop/nDPI