VPNalyzer

UNIVERSITY OF MICHIGAN

Consumer Reports

VPNalyzer

Systematic Investigation of the VPN Ecosystem

Reethika Ramesh

LASER, April 2022

Internet traffic is increasingly being **disrupted, tampered with, and monitored** by ISPs, advertisers, and other threat actors

VPNs are on the Rise

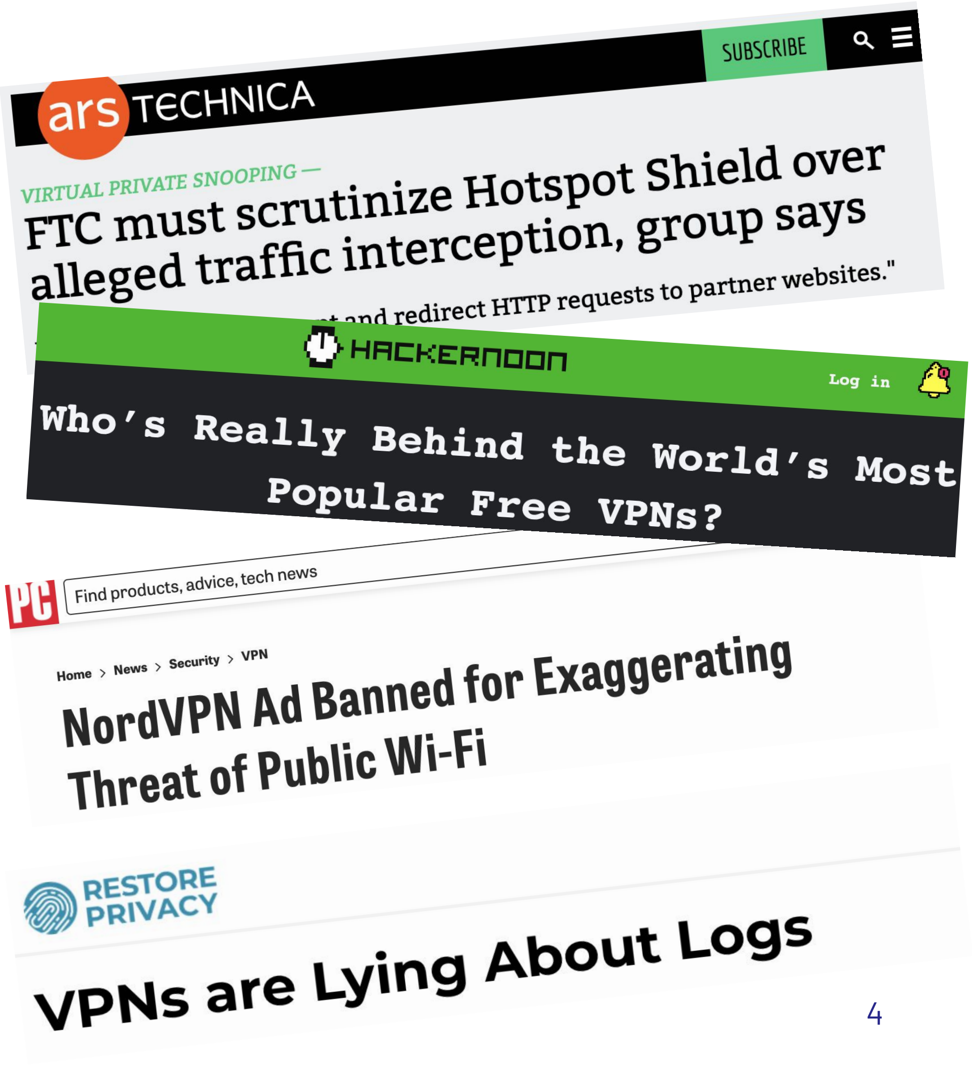
"From 2010 to year-end 2019, the use of VPNs has increased by **approximately four times**" Cybersecurity company PC Matic, 2020

Commercial VPNs are a multi-billion dollar industry; most recently ExpressVPN was acquired for $936 million Reuters, Sep 2021

Reasons for use?

Protection from surveillance, censorship circumvention, accessing work/school/university resources, entertainment etc

This multi-billion dollar industry is **laxly regulated,** rife with **hyperbolic claims,** and **remains severely understudied**



Towards a Systematic Investigation of VPNs

Previous reports are lab-based:

- ↪ Used **inconsistent heuristics**
- ↪ Involved a large amount of **manual effort**
- ↪ **Limited in the scale** and types of VPN products studied

Towards a Systematic Investigation of VPNs

Previous reports are lab-based:

- ↪ Used **inconsistent heuristics**
- ↪ Involved a large amount of **manual effort**
- ↪ **Limited in the scale** and types of VPN products studied

KEY CHALLENGES:

Rigor, Scale, Automation

Bringing transparency and better security to consumer VPNs requires a different approach

We built VPNalyzer

to address these challenges

Building VPNalyzer to Address Key Challenges

Modular, extensible test suite

Repeated VPN evaluations over time should not require starting from scratch

System should evolve alongside the VPN ecosystem: Validating VPN providers' fixes for issues reported as disclosures requires an updatable test suite

Building VPNalyzer to Address Key Challenges

Modular, extensible test suite

Repeated VPN evaluations over time should not require starting from scratch

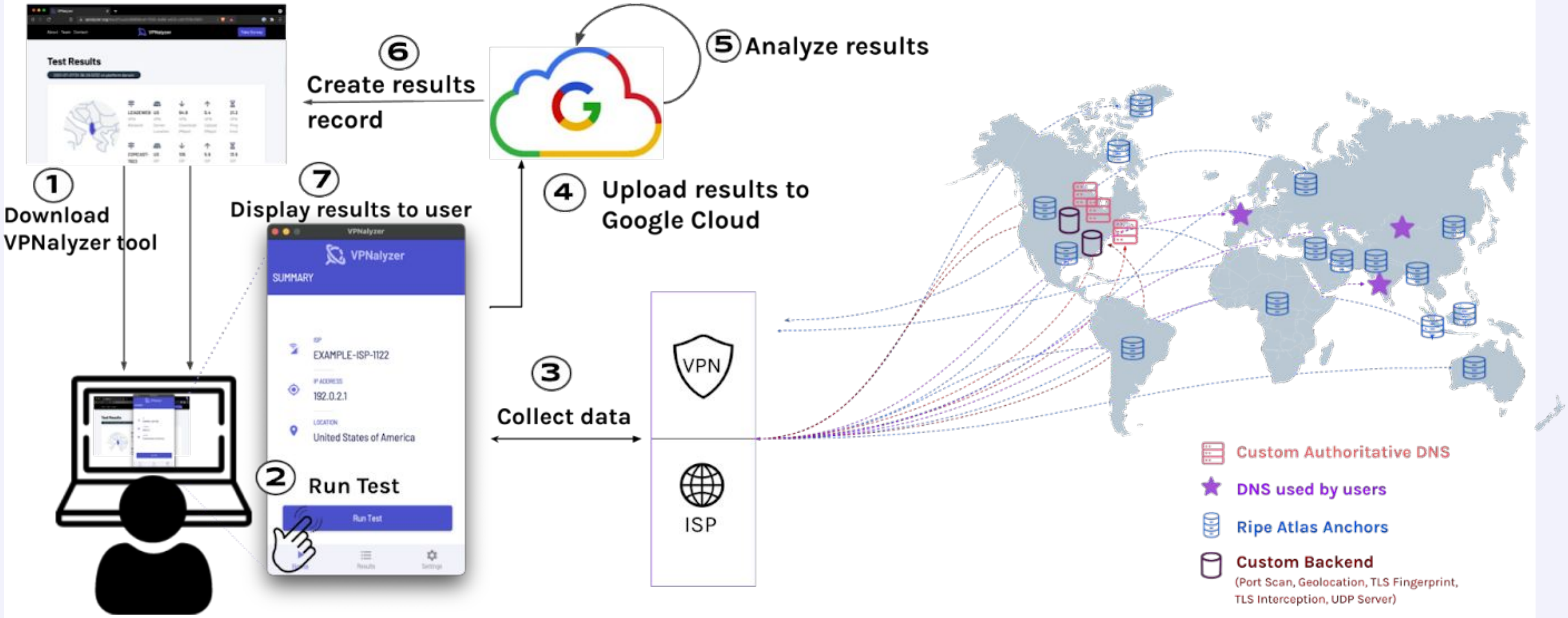
System should evolve alongside the VPN ecosystem: Validating VPN providers' fixes for issues reported as disclosures requires an updatable test suite

Facilitate Crowdsourced Data

Increasing number of VPN providers

Users have varied threat models and use cases, ranging from watching netflix to "anonymity"; they may prefer different VPN products

VPNalyzer System Design



Design and Implementation Considerations

Tradeoffs: Functionality vs Ease

Explored creating web based javascript, browser extension, and native desktop app

Need a sustainable cross-platform solution

Systematic testing demands multiple platform support and specialized development

Conducive to test both VPN and ISP

Making the test suite conducive to test users' VPN and ISP both

Developing test suite and validating tests

Improving upon previous work, testing measurements

Tradeoffs: Functionality vs Ease

What each offers:

Web based tests

Cannot make requests to different websites and services necessary to test features

Browser Extension

Limited functionality to test critical features like leak protection during tunnel failure

Desktop App

Provides the right level of functionality, and fine-grained access for robust measurements

Tradeoffs: Functionality vs Ease

What we can implement:

Web based tests

Bandwidth while on VPN
Static geolocation
DNS leak tests under normal conditions

Browser Extension

DNS Leaks
Can conduct constant measurements

Desktop App

Test for leaks during tunnel failure
Self-contained experiments

Developing the VPNalyzer Tool

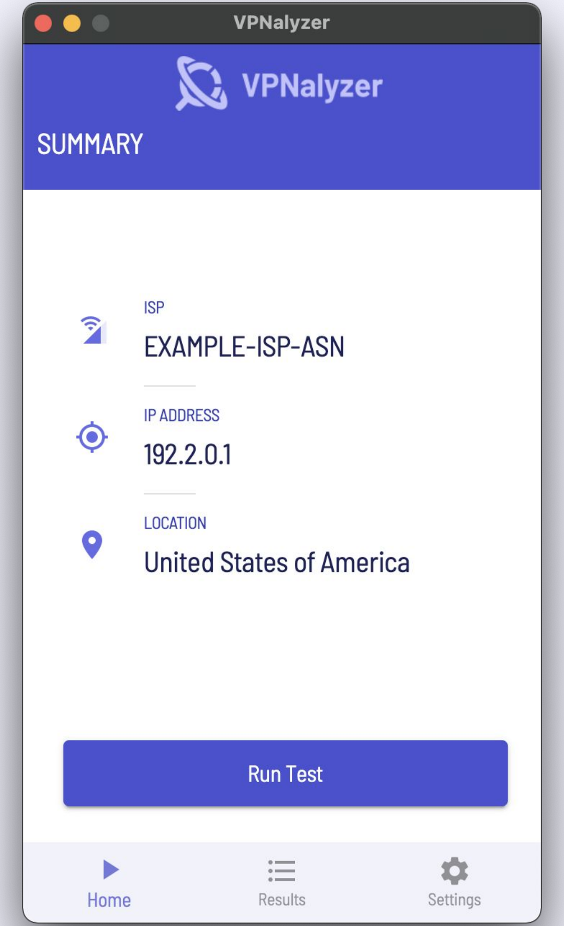
Electron Framework affords Cross-platform compatibility and Native API availability

Measurement code in **Node.js**

Front end in **React**

Available as a **MacOS**, **Windows**, and **Linux** application

One full experiment run (avg time): 20 mins



What do we test with VPNalyzer?

Aspects of Service

- Bandwidth and latency
- Geolocation
- RPKI validation

Misconfiguration and leakages

- DNS leaks
- IPv6 leaks
- Data leaks during tunnel failure

Security and Privacy Essentials

- Port scanning
- Router interface reachability
- Presence of DNS proxy
- QNAME minimization
- DNSSEC validation
- Lack of support for DoH
- TLS Interception

VPNalyzer has a modular, extensible test suite currently containing 15 measurements

Design and Implementation Considerations

Tradeoffs: Functionality vs Ease

Explored creating web based javascript, browser extension, and native desktop app

Need a sustainable cross-platform solution

Systematic testing demands multiple platform support and specialized development

Conducive to test both VPN and ISP

Making the test suite conducive to test users' VPN and ISP both

Developing test suite and validating tests

Improving upon previous work, testing measurements

Systematic Investigation Demands Cross-Platform Support

Protection during tunnel failure is a key privacy feature. However, implementation varies:

- from one VPN to another
- based on operating system

Systematic Investigation Demands Cross-Platform Support

Protection during tunnel failure is a key privacy feature. However, implementation varies:

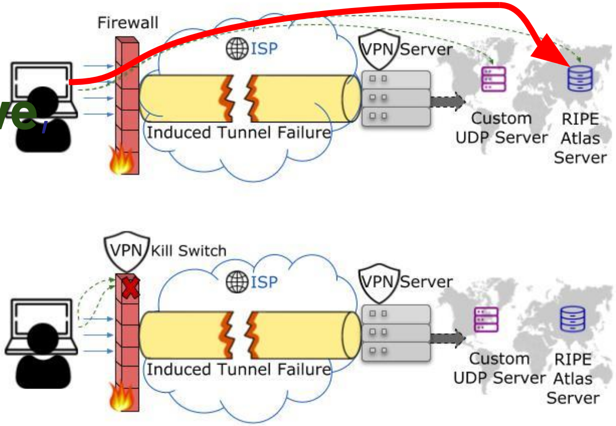
- from one VPN to another
- based on operating system

Testing such features needs cross-platform development and expertise

Detecting Traffic Leaks During Tunnel Failure

Overview: Conceptually, create an “allowlist” of specific hosts, cause a tunnel failure by blocking all traffic except to and from allowlist

If the VPN’s leak protection is **effective**, the traffic to the hosts on the allowlist should also be **blocked**



Detecting Traffic Leaks During Tunnel Failure

↪ **Bootstrap via ISP**: Request administrative privileges, log firewall state before any changes, initiate sessions

Detecting Traffic Leaks During Tunnel Failure

- ↪ **Bootstrap via ISP**
- ↪ **VPN Case**
  - ■ Initialization Phase
    - ↪ Set up necessary platform-specific components

Detecting Traffic Leaks During Tunnel Failure

↪ **Bootstrap via ISP**

↪ **VPN Case**

- Initialization Phase
  - ↪ Set up necessary platform-specific components:
    - Linux: Add chains for `iptables` and `ip6tables`
    - Windows: Log version of `PowerShell` and `NetSecurity` module (Need PowerShell > 2.0)
    - MacOS: Test custom anchors on `pf`, enable `pf`, and obtain token to revert it (`pfctl -X TOKEN`)

Detecting Traffic Leaks During Tunnel Failure

- ↪ **Bootstrap via ISP**
- ↪ **VPN Case**
  - ■ Initialization Phase
    - ↪ Set up necessary platform-specific components
    - ↪ Log the firewall state again

Detecting Traffic Leaks During Tunnel Failure

↪ **Bootstrap via ISP**

↪ **VPN Case**

- Initialization Phase
- Create Allowlist and Induce Tunnel Failure
  - RIPEstat Data API: Whats My IP
  - One of our custom UDP heartbeat servers (ServerA)
  - Authoritative nameservers and public DNS resolvers belonging to Cloudflare, Google, and OpenDNS

Detecting Traffic Leaks During Tunnel Failure

↪ **Bootstrap via ISP**

↪ **VPN Case**

- Initialization Phase
- Create Allowlist and Induce Tunnel Failure
  - RIPEstat Data API: Whats My IP
  - One of our custom UDP heartbeat servers (ServerA)
  - Authoritative nameservers and public DNS resolvers belonging to Cloudflare, Google, and OpenDNS
- Detection Logic

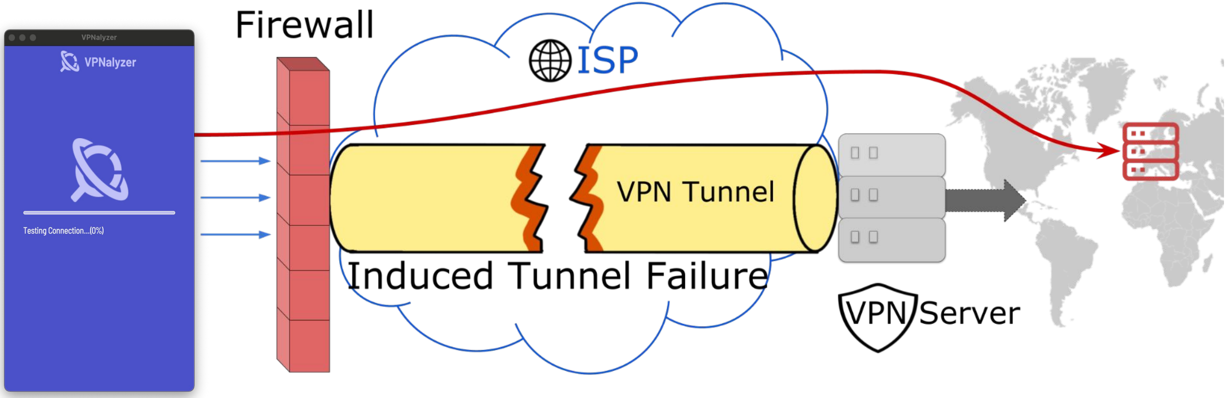
Traffic Leak Detection Logic

Probe for Possible Data Leaks:

↪ For 120s, periodically query the RIPEstat Data API: Whats My IP

If some **data leak protection exists**, queries would time out

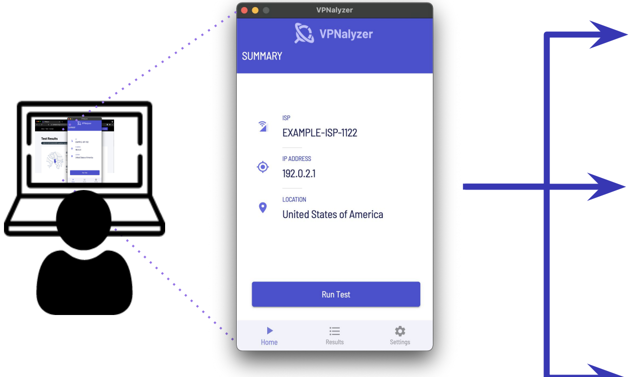
If **there is no data leak protection**, query reaches endpoint and returns user's ISP IP



Detecting Traffic Leaks During Tunnel Failure

- ↪ **Bootstrap via ISP**
- ↪ **VPN Case**
  - Initialization Phase
  - Create Allowlist and Induce Tunnel Failure
  - Detection Logic
- ↪ **ISP Case**
  - No Measurements
  - Log Firewall State

VPNalyzer Experiment Flow



1 Bootstrap via ISP

Request administrative privileges, initialize packet captures, fetch necessary resources, and log firewall state

2 Testing with the VPN on

Test suite is triggered for *VPN case*:
We run Test {1 → X} serially

3 Testing with VPN off

Test suite is triggered again for *ISP case*:
We run Test {1 → X} serially as applies

Design and Implementation Considerations

Tradeoffs: Functionality vs Ease

Explored creating web based javascript, browser extension, and native desktop app

Need a sustainable cross-platform solution

Systematic testing demands multiple platform support and specialized development

Conducive to test both VPN and ISP

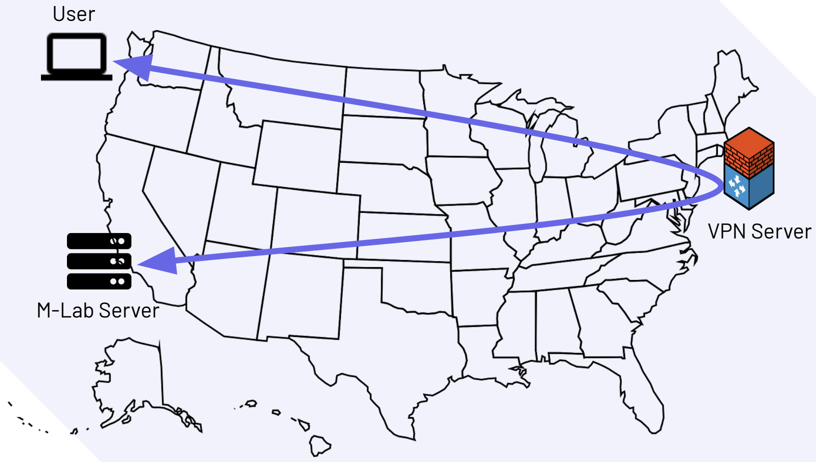
Making the test suite conducive to test users' VPN and ISP both

Developing test suite and validating tests

Improving upon previous work, testing measurements

Validating Measurements Testing from VPN and ISP

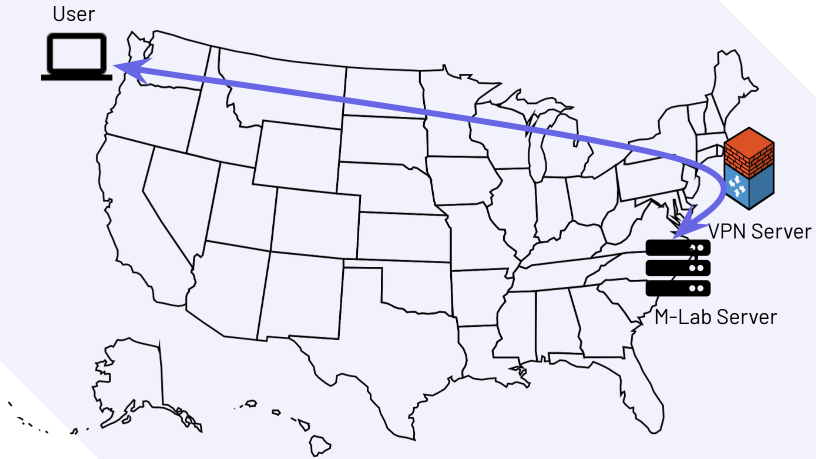
- Bandwidth and Latency Test
  - To calculate performance overhead/enhancement due to their VPN, we need to measure bandwidth in both VPN and ISP case



Validating Measurements Testing from VPN and ISP

- Bandwidth and Latency Test
  - To calculate performance overhead/enhancement due to their VPN, we need to measure bandwidth in both VPN and ISP case

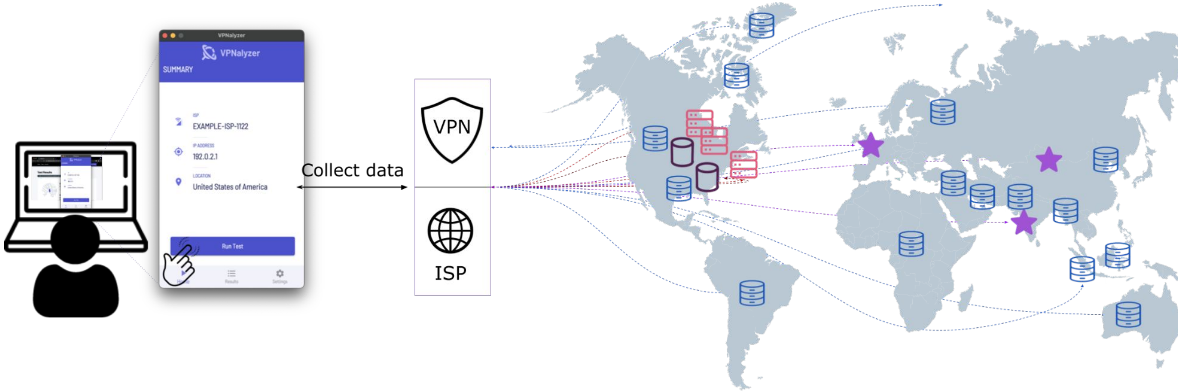
Validating Measurements Testing from VPN and ISP



- Bandwidth and Latency Test
  - Calculate performance overhead or enhancement due to the VPN
  - Need to measure bandwidth in both VPN and ISP case
  - Selecting a non-optimal M-Lab server resulted in bloated performance overhead

Validating Measurements Testing from VPN and ISP

- Compare VPN and ISP case for:
    - DNS servers available in both cases
    - Detecting IP leakages
    - DNS Leaks
    - TLS Fingerprint



Design and Implementation Considerations

Tradeoffs: Functionality vs Ease

Explored creating web based javascript, browser extension, and native desktop app

Need a sustainable cross-platform solution

Systematic testing demands multiple platform support and specialized development

Conducive to test both VPN and ISP

Making the test suite conducive to test users’ VPN and ISP both

Developing test suite and validating tests

Improving upon previous work, testing measurements

Prior Work and Measurements

We looked at prior work and measurements methods

Select inspirations from:

- ↪ Recovery from Tunnel Failure – [Khan, IMC 2018]
- ↪ Using Constraint-based Geolocation – [Weinberg, IMC 2018]
- ↪ QNAME Minimization with custom domains – [de Vries, PAM 2019]
- ↪ TLS Fingerprinting – [Frolov, NDSS 2019]

Prior Work and Measurements

We looked at prior work and measurements methods

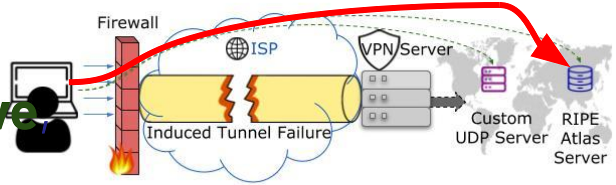
Select inspirations from:

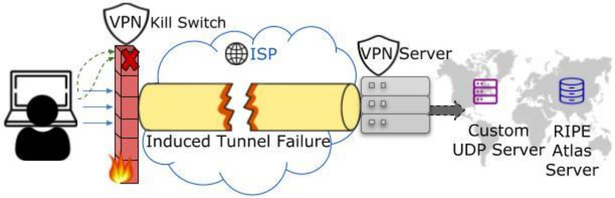
- ↪ **Recovery from Tunnel Failure** – [Khan, IMC 2018]
- ↪ Using Constraint-based Geolocation – [Weinberg, IMC 2018]
- ↪ QNAME Minimization with custom domains – [de Vries, PAM 2019]
- ↪ TLS Fingerprinting – [Frolov, NDSS 2019]

Detecting Traffic Leaks During Tunnel Failure

Overview: Conceptually, create an “allowlist” of specific hosts, cause a tunnel failure by blocking all traffic except to and from allowlist

If the VPN’s leak protection is **effective**, the traffic to the hosts on the allowlist should also be **blocked**





Inducing Tunnel Failure is Tricky

- Should not tamper with user's custom rules
- Must not hinder VPNs' leak protection mechanism

Inducing Tunnel Failure is Tricky

- ↪ Should not tamper with user’s custom rules
- ↪ Must not hinder VPNs’ leak protection mechanism

Our detection mechanism must **co-exist** with other applications (including the VPN) and ensure **test reliability** above all

Background: Data leak during tunnel failure on MacOS

Using `pf` on MacOS, and anchors (collection of rules, and tables)

Ordering of the anchors is **important,**
avoid modifying and using `/etc/pf.conf` directly

Obtain token to revert changes

`pfctl -E` and
`pfctl -X TOKEN`

Experimenting with Multiple VPNs Reveals Clues

Tested first by just adding our anchor at the bottom of the rules

- Anything before our anchor with the `quick` keyword will override our rules

Experimenting with Multiple VPNs Reveals Clues

Tested first by just adding our anchor at the bottom of the rules

- Anything before our anchor with the `quick` keyword will override our rules

Some VPNs upon tunnel failure, reset the firewall and push their anchor to the top

If we add our anchor by resetting all rules, we risk overriding the VPN's protection mechanism

We opted for our measurement to be conservative (avoid false positives)

Examining the VPN rules

- VPNs allowlist DNS queries in their kill switch or firewall implementations
- VPNs create a table with relevant IPs to allowlist

Examining the VPN rules

- VPNs allowlist DNS queries in their kill switch or firewall implementations

- VPNs create a table with relevant IPs to allowlist

```
pass out quick inet proto udp from any port = 68
 to 255.255.255.255 port = 67 no state
pass in quick inet proto udp from any port = 67
to any port = 68 no state
```

```
pass quick from any to <vpn_servers> flags any
keep state
pass quick proto tcp from any to any port = 53
flags any keep state
pass quick proto udp from any to any port = 53
keep state
```

DNS Leak Discovery!

We designed a measurement to capture VPNs that allows DNS queries to leak:

- Allowlist public DNS resolvers and nameservers
- Upon inducing tunnel failure, periodically send `whoami` queries

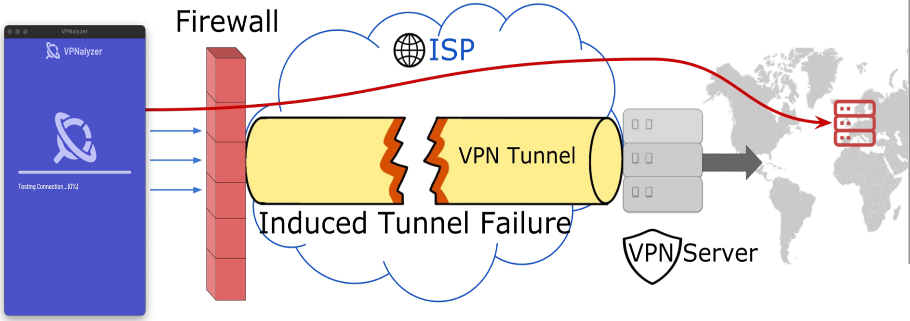
Example Queries and Responses:

```
dig +noedns -t txt whoami.cloudflare.com. @ns3.cloudflare.com.

; <<>> DiG 9.10.6 <<>> +noedns -t txt whoami.cloudflare.com. @ns3.cloudfla
re.com.
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12199
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;whoami.cloudflare.com.         IN      TXT

;; ANSWER SECTION:
whoami.cloudflare.com.  15      IN      TXT     "<USER'S ISP PUBLIC IP>"

;; Query time: 22 msec
;; SERVER: <SERVER IP>#53(<SERVER IP>)
```

```
dig +noedns -t A myip.opendns.com. @resolver{1,2}.opendns.com.

; <<>> DiG 9.10.6 <<>> +noedns -t A myip.opendns.com. @resolver1.opendns.c
om. @resolver2.opendns.com.
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1510
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;myip.opendns.com.              IN      A

;; ANSWER SECTION:
myip.opendns.com.       0       IN      A       <USER'S ISP PUBLIC IP>

;; Query time: 61 msec
;; SERVER: <SERVER IP>#53(<SERVER IP>)
```

DNS Leak:
Testing and Validation

- Allowlist public DNS resolvers and nameservers
- Upon inducing tunnel failure, periodically send `whoami` queries



Example Queries and Responses:

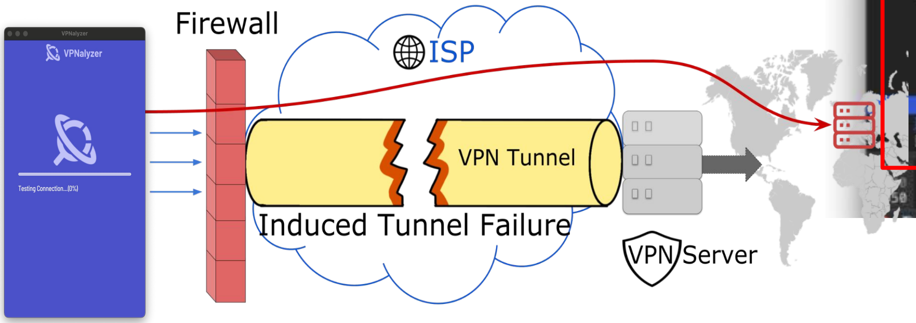
```
dig +noedns -t txt whoami.cloudflare.com. @ns3.cloudflare.com.

; <<>> DiG 9.10.6 <<>> +noedns -t txt whoami.cloudflare.com. @ns3.cloudfla
re.com.
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12199
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;whoami.cloudflare.com.          IN      TXT

;; ANSWER SECTION:
whoami.cloudflare.com.  15       IN      TXT     "<USER'S ISP PUBLIC IP>"

;; Query time: 22 msec
;; SERVER: <SERVER IP>#53(<SERVER IP>)
```

```
dig +noedns -t A myip.opendns.com. @resolver{1,2}.opendns.com.

; <<>> DiG 9.10.6 <<>> +noedns -t A myip.opendns.com. @resolver1.opendns.c
om. @resolver2.opendns.com.
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1510
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;myip.opendns.com.               IN      A

;; ANSWER SECTION:
myip.opendns.com.       0        IN      A       <USER'S ISP PUBLIC IP>

;; Query time: 61 msec
;; SERVER: <SERVER IP>#53(<SERVER IP>)
```

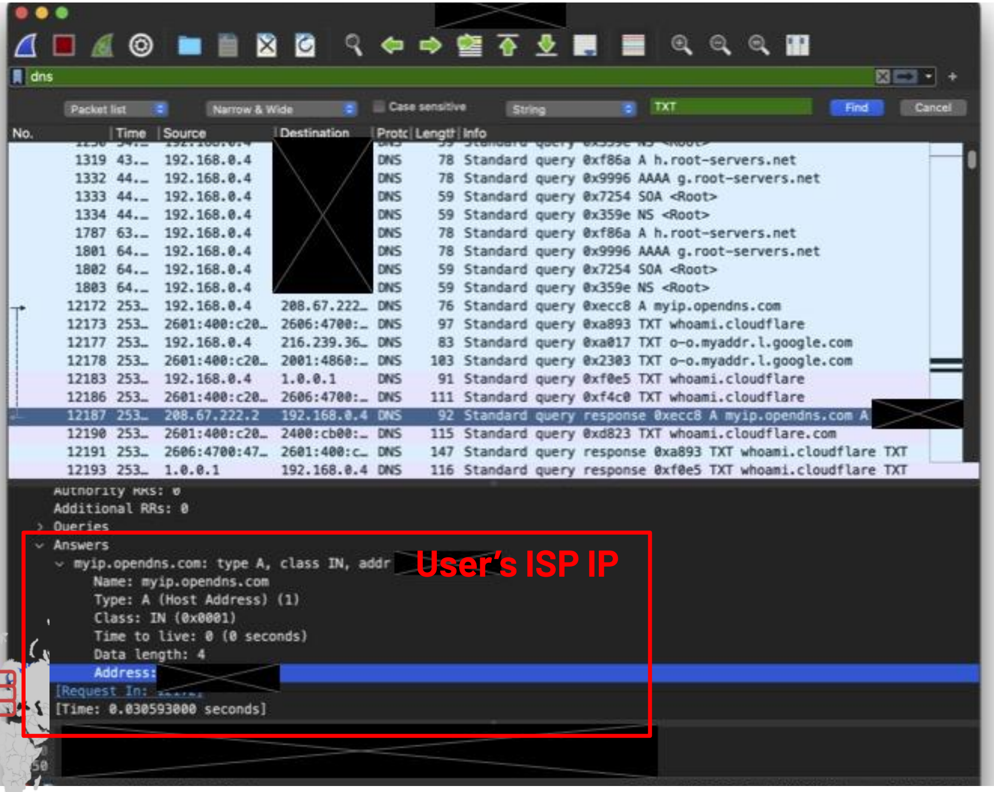
DNS Leak:

Testing and Validation

- Allowlist public DNS resolvers and nameservers
- Upon inducing tunnel failure, periodically send `whoami` queries



Validating Results:



Learning from Experiments

Implementation of features varies b/w VPN providers

After multiple rounds of testing, we found:

- ↪ Attn to ordering of existing or new VPN rules
- ↪ VPNs inserting dynamic rules on the fly
- ↪ VPN “**allowing**” certain types of traffic in their firewall rules

Design and Implementation Considerations

Tradeoffs: Functionality vs Ease

Explored creating web based javascript, browser extension, and native desktop app

Need a sustainable cross-platform solution

Systematic testing demands multiple platform support and specialized development

Conducive to test both VPN and ISP

Making the test suite conducive to test users' VPN and ISP both

Developing test suite and validating tests

Improving upon previous work, testing measurements

We tested **80 popular VPNs** with our VPNalyzer tool and uncovered several previously unreported findings

VPNalyzer in Practice:

Testing 80 popular VPNs

- We tested random servers in each VPN provider, on Windows and MacOS
  - **58 paid** VPN providers
  - **18 free** VPN providers
  - **4 self-hosted** VPN solutions
    (Algo, OpenVPN Access Server on AWS, Outline, Streisand)
- Some results for the same VPN provider may differ based on server selected

Traffic Leakages:

IPv6 Traffic

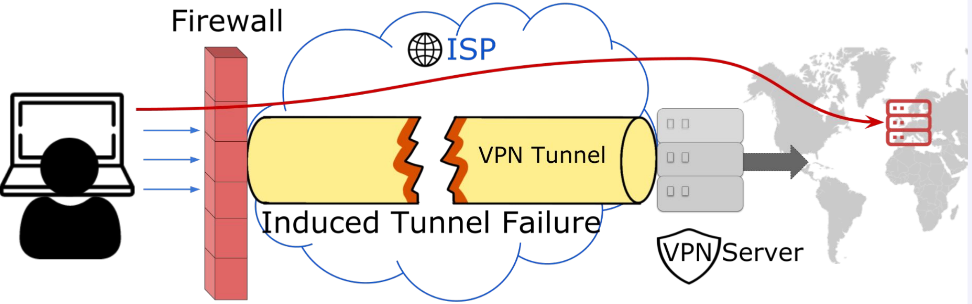
- Only 11 out of 80 VPNs support IPv6

- Five VPNs **leak IPv6 traffic** to the ISP by default
  - **UMich VPN** is among them

Traffic Leakages:

During Tunnel Failure

Upon tunnel failure, 26 providers **leak traffic** to the user's ISP



By default, 26 VPNs lack protection during tunnel failure

Traffic Leakages:

During Tunnel Failure

Upon tunnel failure, 26 providers **leak traffic** to the user's ISP

- ↪ 18 leak all traffic, eight of these leak DNS traffic only
- ↪ Five of these 26 are the ones that leak IPv6

By default, 26 VPNs lack protection during tunnel failure

Traffic Leakages:

Even with a Kill Switch Enabled

Even in their most secure setting, 10 providers **leak traffic** to the user's ISP upon tunnel failure

↪ Six of which even had a **"kill switch" feature** enabled

Even with a "*kill switch*", six VPNs **leak traffic during tunnel failure**

Traffic Leakages:

Insecure Default Configuration

Astrill VPN tunneled **only browser traffic** by default

Psiphon did **not enable "VPN mode"** by default

Default Configuration caused user's (non-browser) traffic to be exposed to the ISP

Filed Responsible Disclosures

Positive Impact

- Our disclosures and communications with VPN providers have already led to positive changes in at least four VPN providers
- Consumer Reports used our VPNalyzer tool for their own investigation to help recommend VPNs to their subscribers
- Served as a real-world evaluation of our tool

VPNalyzer

UNIVERSITY OF MICHIGAN

Consumer Reports

VPNalyzer

Investigating the commercial VPN ecosystem

Reethika Ramesh

LASER, April 2022