TCP/IP Fingerprint Spoofing

Using User-Space Networking and Rust

Joe Dye (He/Him)

Ping Proxies

What is TCP/IP fingerprinting?

The Basics

One of the main reasons for using a proxy is a desire for anonymity, regardless of if the proxy is being used for web-scraping or to watch BBC iPlayer from abroad.

> A mismatch between the declared-OS and the detected-OS is a large signal that a proxy is being used.

Majority of proxy servers run Linux and the majority of users of proxies either:

- Run a non-Linux operating system.
- Want to look like they run a non-Linux operating system.

IP Headers

IHL — Internet header length (between 5–15)

Length — Total length of headers + data

ID — Identification field for fragmentation

Flags — DF (don't fragment) and MF (more fragments)

TTL — Number of allowed hops, default 64

Protocol — Encapsulated protocol, always 6 for TCP

Checksum — Error checking for headers

IP Headers

IHL — Internet header length (between 5–15)

Length — Total length of headers + data

ID — Identification field for fragmentation

Flags — DF (don't fragment) and MF (more fragments)

TTL — Number of allowed hops, default 64

Protocol — Encapsulated protocol, always 6 for TCP

Checksum — Error checking for headers

OS-correlated

TCP Headers

SEQ — Initial packet count, random

ACK — Always 0 in first packet

Data Offset — Header size, depends on options set

Window Size — Initial receiving window size

Checksum — Error checking for headers

Urgent PTR — Always 0 in initial packets

Options — 0–40 bytes, contains MSS

MSS — Maximum receivable data, OS specific

TCP Headers

SEQ — Initial packet count, random

ACK — Always 0 in first packet

Data Offset — Header size, depends on options set

Window Size — Initial receiving window size

Checksum — Error checking for headers

Urgent PTR — Always 0 in initial packets

Options — 0–40 bytes, contains MSS

MSS — Maximum receivable data, OS specific

■ OS-correlated

Try it yourself...

M1460-N-W7
M1452-N-W6-N-N-T-S-E-E
M1460-S-T-N-W12
M1412-N-W6-N-N-T-S-E-E
M1460-S-T-N-W7
M1452-N-W6-N-N-T-S-E-E
M1400-S-T-N-W8
M1460-N-W6-N-N-T-S-E-E
M1410-N-W7-S-T
M1460-N-W6-N-N-T-S-E-E
M1452-S-T-N-W9
M1360-N-W6-N-N-T-S-E-E

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Try it yourself...

Linux:

M1460-N-W7
M1460-S-T-N-W7
M1410-N-W7-S-T

Mac OS:

M1452-N-W6-N-N-T-S-E-E
M1452-N-W6-N-N-T-S-E-E
M1460-N-W6-N-N-T-S-E-E

Android:

M1460-S-T-N-W12
M1400-S-T-N-W8
M1452-S-T-N-W9

iOS:

M1412-N-W6-N-N-T-S-E-E
M1460-N-W6-N-N-T-S-E-E
M1360-N-W6-N-N-T-S-E-E

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

M1412-N-W6-N-N-T-S-E-E

M1410-N-W7-S-T

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

M1412-N-W6-N-N-T-S-E-E

M1410-N-W7-S-T

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

M1412-N-W6-N-N-T-S-E-E

M1410-N-W7-S-T

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

M1412-N-W6-N-N-T-S-E-E

M1410-N-W7-S-T

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

M1412-N-W6-N-N-T-S-E-E

M1410-N-W7-S-T

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

M1412-N-W6-N-N-T-S-E-E

M1410-N-W7-S-T

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

M1412-N-W6-N-N-T-S-E-E

M1410-N-W7-S-T

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

M1412-N-W6-N-N-T-S-E-E

M1410-N-W7-S-T

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Mobile Devices and the MSS

Mobile Carrier	MSS
Verizon	1388
T-Mobile	1360
Google Fi	1348

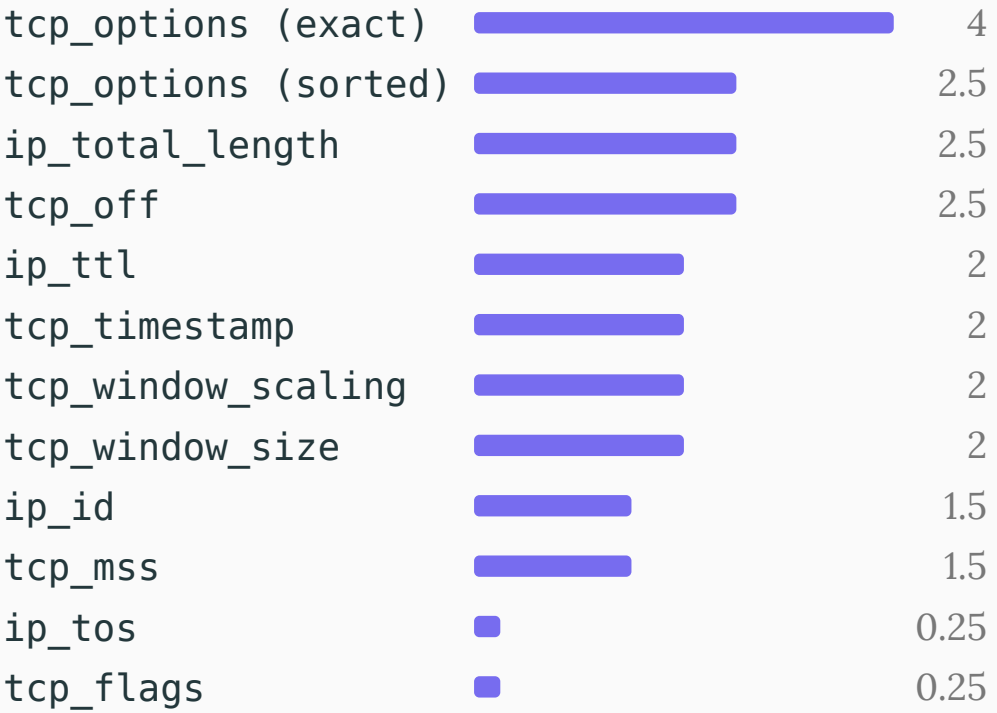
M1360-N-W6-N-N-T-S-E-E

How is TCP/IP fingerprinting actually done?

Zardaxt (Python)

```
score = 0
os_name = entry['os']

if entry['ip_id'] == fp['ip_id']:
    score += 1.5
if entry['ip_tos'] == fp['ip_tos']:
    score += 0.25
if entry['ip_total_length'] == fp['ip_total_length']:
    score += 2.5
if entry['ip_ttl'] == fp['ip_ttl']:
    score += 2
if entry['tcp_off'] == fp['tcp_off']:
    score += 2.5
if entry['tcp_timestamp_echo_reply'] ==
fp['tcp_timestamp_echo_reply']:
    score += 2
if entry['tcp_window_scaling'] == fp['tcp_window_scaling']:
    score += 2
if entry['tcp_window_size'] == fp['tcp_window_size']:
    score += 2
if entry['tcp_flags'] == fp['tcp_flags']:
    score += 0.25
if entry['tcp_mss'] == fp['tcp_mss']:
    score += 1.5
if entry['tcp_options'] == fp['tcp_options']:
    score += 4
elif entry['tcp_options_ordered'] == fp['tcp_options_ordered']:
    score += 2.5
```



Huggin-Net (Rust)

```rust
#[derive(Clone, Debug, PartialEq)]
pub enum Quirk {
    Df,                                 // df     - "don't fragment" set (probably PMTUD)
    NonZeroID,                          // id+    - DF set but IPID non-zero
    ZeroID,                             // id-    - DF not set but IPID is zero
    Ecn,                                // ecn    - explicit congestion notification support
    MustBeZero,                         // 0+     - "must be zero" field not zero
    FlowID,                             // flow   - non-zero IPv6 flow ID
    SeqNumZero,                         // seq-   - sequence number is zero
    AckNumNonZero,                      // ack+   - ACK number non-zero, ACK flag not set
    AckNumZero,                         // ack-   - ACK number zero, ACK flag set
    NonZeroURG,                         // uptr+  - URG pointer non-zero, URG flag not set
    Urg,                                // urgf+  - URG flag used
    Push,                               // pushf+ - PUSH flag used
    OwnTimestampZero,                   // ts1-   - own timestamp specified as zero
    PeerTimestampNonZero,               // ts2+   - non-zero peer timestamp on initial SYN
    TrailingNonZero,                    // opt+   - trailing non-zero data in options
    ExcessiveWindowScaling,             // exws   - excessive window scaling (> 14)
    OptBad,                             // bad    - malformed TCP options
}
```

How will customers use this?

User Experience

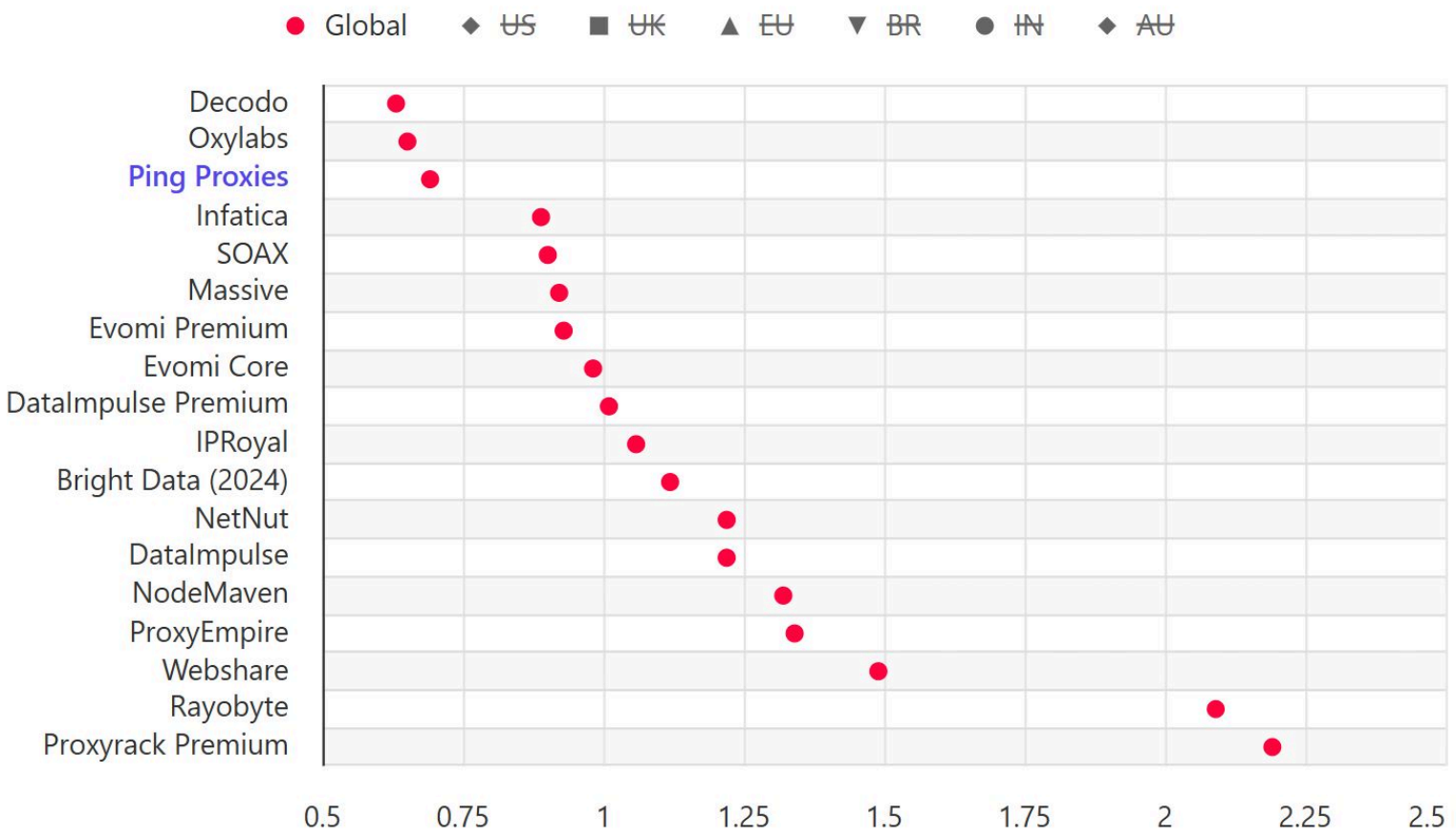
Ideally, mimicry.

- Copy the client fingerprint. Replay it on outgoing connections.

In reality, explicitness.

- Client connects to `USER:PASSS@linux.isp.pingproxies.com:9123`.
- Or as part of their credentials: `USER_windows:PASS@192.168.1.238:9123`

User Experience



How do we beat TCP/IP fingerprinting?

Kernel Bypass

Custom Kernel

Pros:

Quickest MVP
Have experience

Cons:

Not portable
I hate C

eBPF

Pros:

Is cool
Super fast

Cons:

Limited scope
Still C

User-Space TCP

Pros:

Total control
Very portable

Cons:

TCP is hard
TCP is very hard

Kernel Bypass

Custom Kernel

Pros:

Quickest MVP
Have experience

Cons:

Not portable
I hate C

eBPF

Pros:

Is cool
Super fast

Cons:

Limited scope
Still C

User-Space TCP

Pros:

Total control
Very portable

Cons:

TCP is hard
TCP is very hard

User-Space Networking

- Raw sockets let us send and receive bytes by bypassing some of the kernel's TCP/IP stack.
- TUN/TAP devices create a virtual NIC that bypasses most of the kernel's TCP/IP stack.
- DPDK lets us write our own poll-mode drivers and get/put bytes directly off/on the NIC.

Complexity rises as the performance of the kernel-bypass improves.

Smoltcp

> smoltcp is a standalone, event-driven TCP/IP stack that is designed for bare-metal, real-time systems. Its design goals are simplicity and robustness.
>
> smoltcp achieves Gbps of throughput when tested against the Linux TCP stack in loopback mode.

```
    .-[ 127.0.0.1/4321 -> 127.0.0.1/1234 (syn) ]-
|
| client   = 127.0.0.1/4321
| os       = ???
| dist     = 0
| params   = none
| raw_sig  = 4:64+0:0:65495:mtu*1,0:mss,ws,sok,eol+2,eol+1,eol+0:df:0
```

Smoltcp

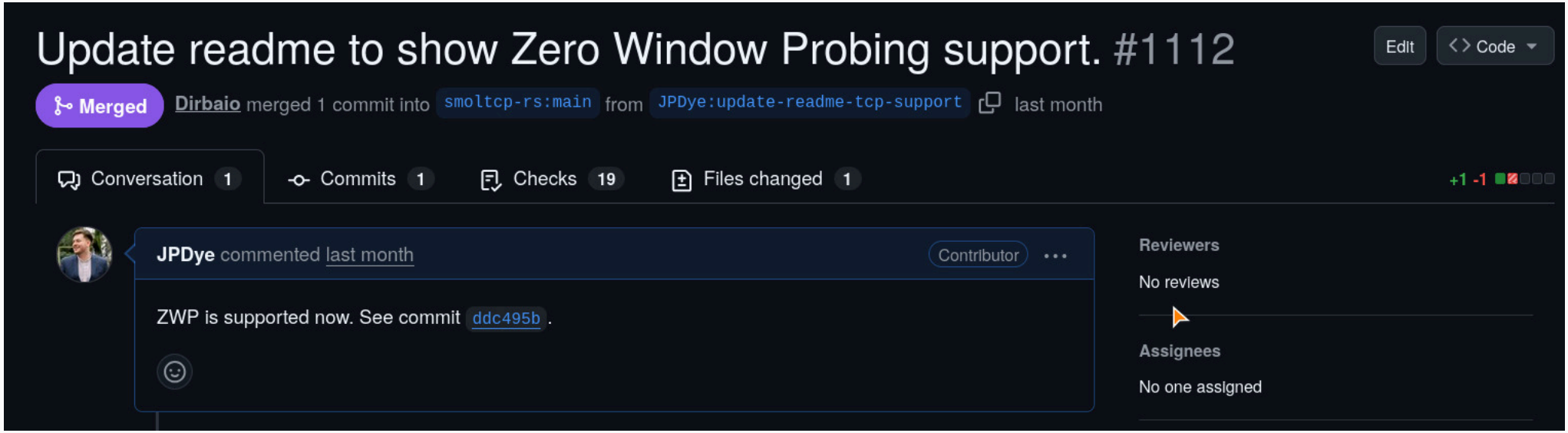
> smoltcp is missing many widely deployed features, usually because no one implemented them yet.

- ✗ Probing Zero Windows is not implemented.
- ✗ Timestamping is not supported.
- ✗ Silly window syndrome avoidance is not implemented.
- ✗ Selective acknowledgements are not implemented.

Smoltcp

```
    .-[ 127.0.0.1/42136 -> 127.0.0.1/1234 (syn) ]-
|
| client   = 127.0.0.1/42136
| os       = Windows/7 or 8
| dist     = 0
| params   = Generic
| raw_sig  = 4:128+0:0:65535:mss*4,0:mss,nop,ws,nop,nop,sok:df,id+:0
```

Zero Window Probes



- ✓ Probing Zero Windows is implemented.

Silly Window Avoidance


```rust
if let Some(last_win) = self.last_scaled_window() {
    let window_len = self.rx_buffer.window();
    let window_capacity = self.rx_buffer.capacity();
    let last_window_len = (last_win as usize) << self.remote_win_shift;
    let window_increase = window_len.saturating_sub(last_window_len);
    (window_increase >= 2 * self.remote_mss
        && (window_increase >= window_capacity / 4
            || window_len <= window_capacity / 8
            || self.remote_mss >= window_capacity / 8))
        || window_increase >= window_capacity / 2
}
```

✓ Silly Window Syndrome avoidance is implemented.

Silly Window Avoidance

Buffer	Packet Loss							
	0%	0.1%	1%	2%	5%	10%	20%	30%
128	0%	0%	0%	0%	0%	0%	0%	0%
256	0%	0%	0%	0%	0%	0%	0%	0%
512	0%	0%	0%	0%	0%	0%	0%	0%
1024	0%	0%	0%	0%	0%	0%	0%	0%
2048	0%	0%	0%	0%	0%	0%	0%	0%
4096	0%	0%	0%	0%	0%	0%	0%	0%
8192	0%	+2.9%	+9%	+10.4%	+11.9%	+12.5%	+5.1%	+3.3%
16384	0%	+0.1%	+1.7%	0%	+6.1%	-1.9%	-8.4%	+14.9%
32768	0%	+0.7%	+3.5%	-8.7%	+15.9%	+23.5%	+91.5%	-19.2%

- ✓ Silly Window Syndrome avoidance is implemented.

FIN.