Regular expressions in XDP

IVAN KOVESHNIKOV

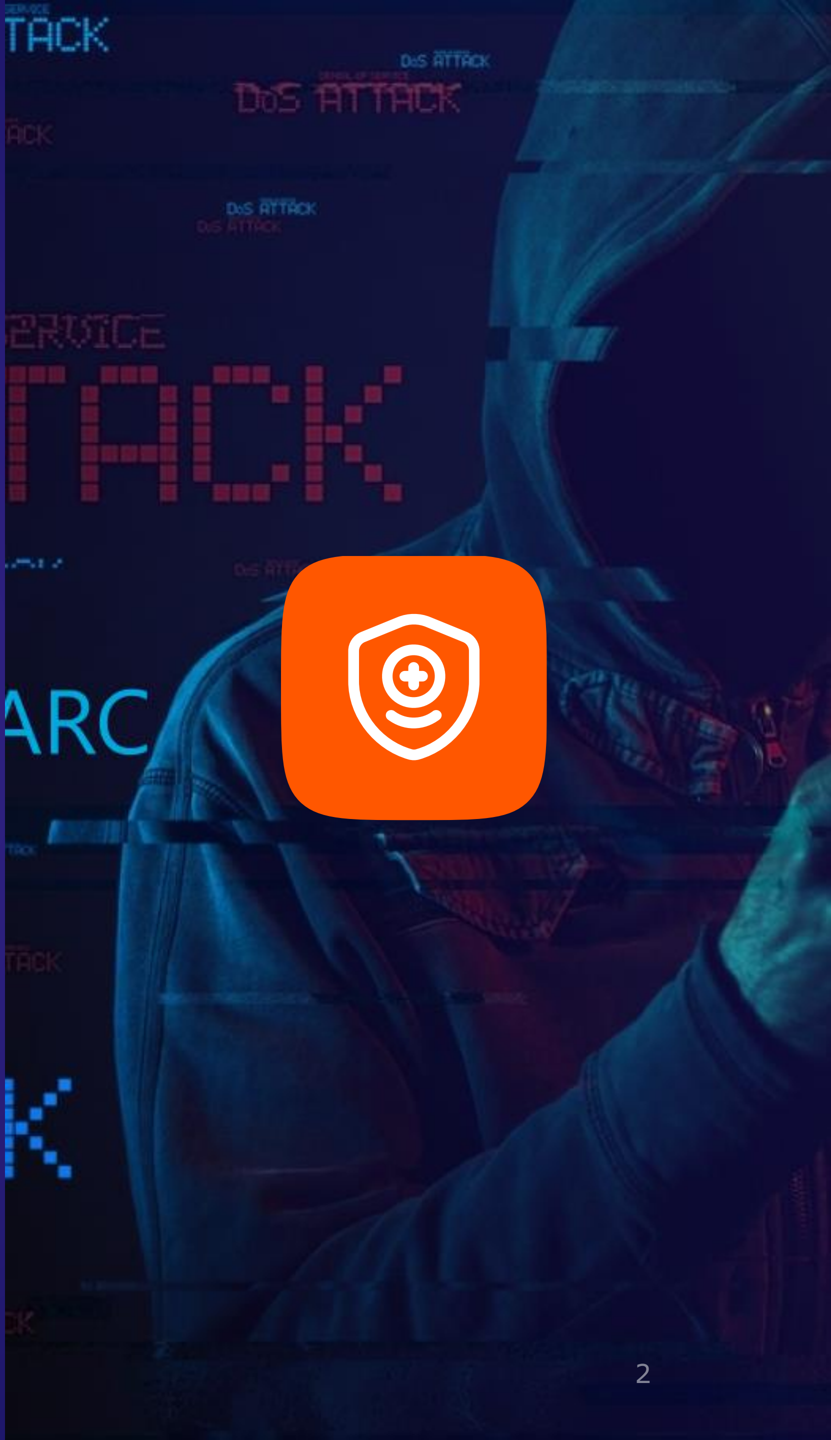
Gcore

SERGEY NIZOVTSEV

Tempesta Technologies

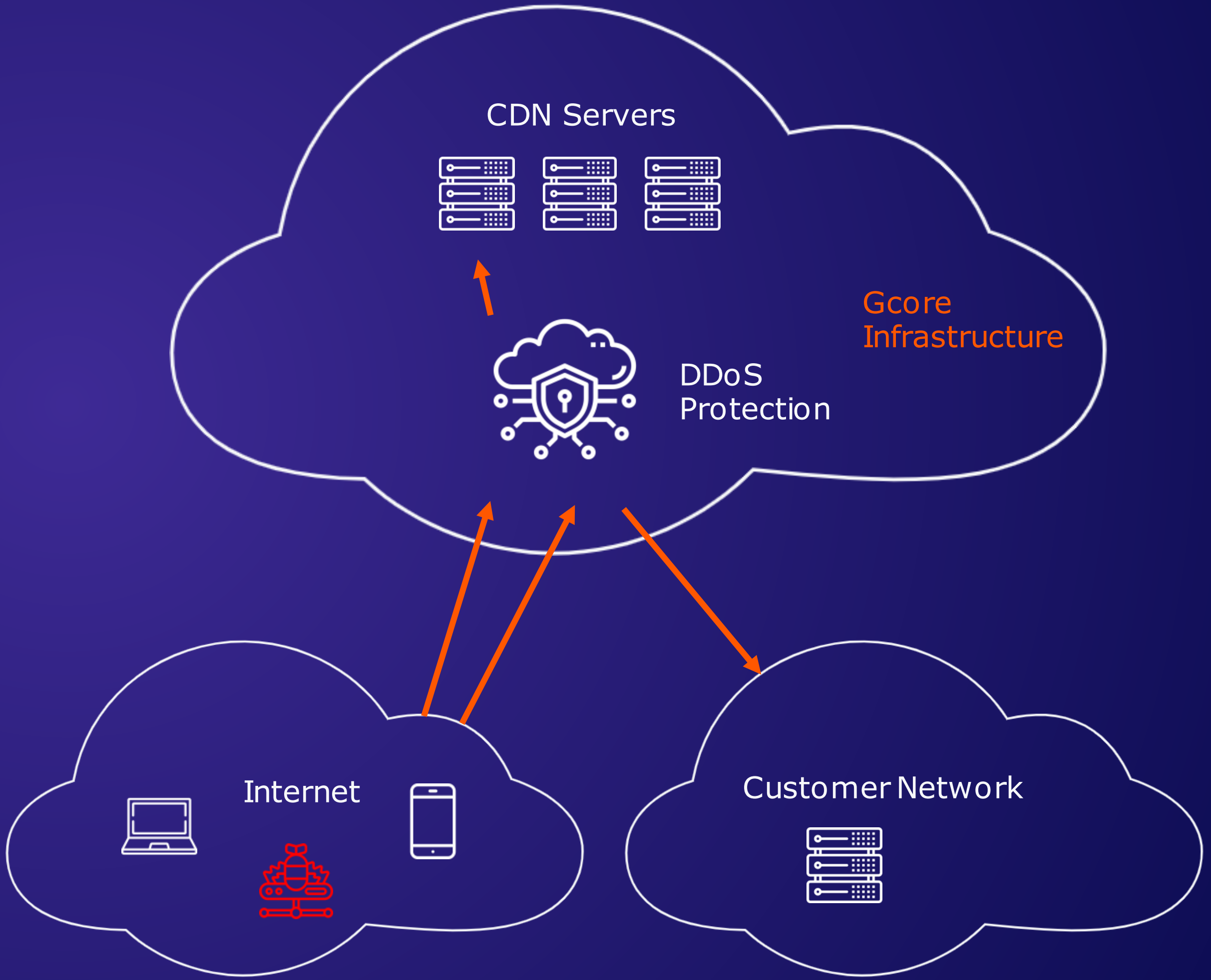
Regular expressions in XDP

- ✓ Security and DDoS Protection as a service
- ✓ XDP pipeline with REGEX
- ✓ Benchmarks
- ✓ Collaboration and further work



Our traditional network design

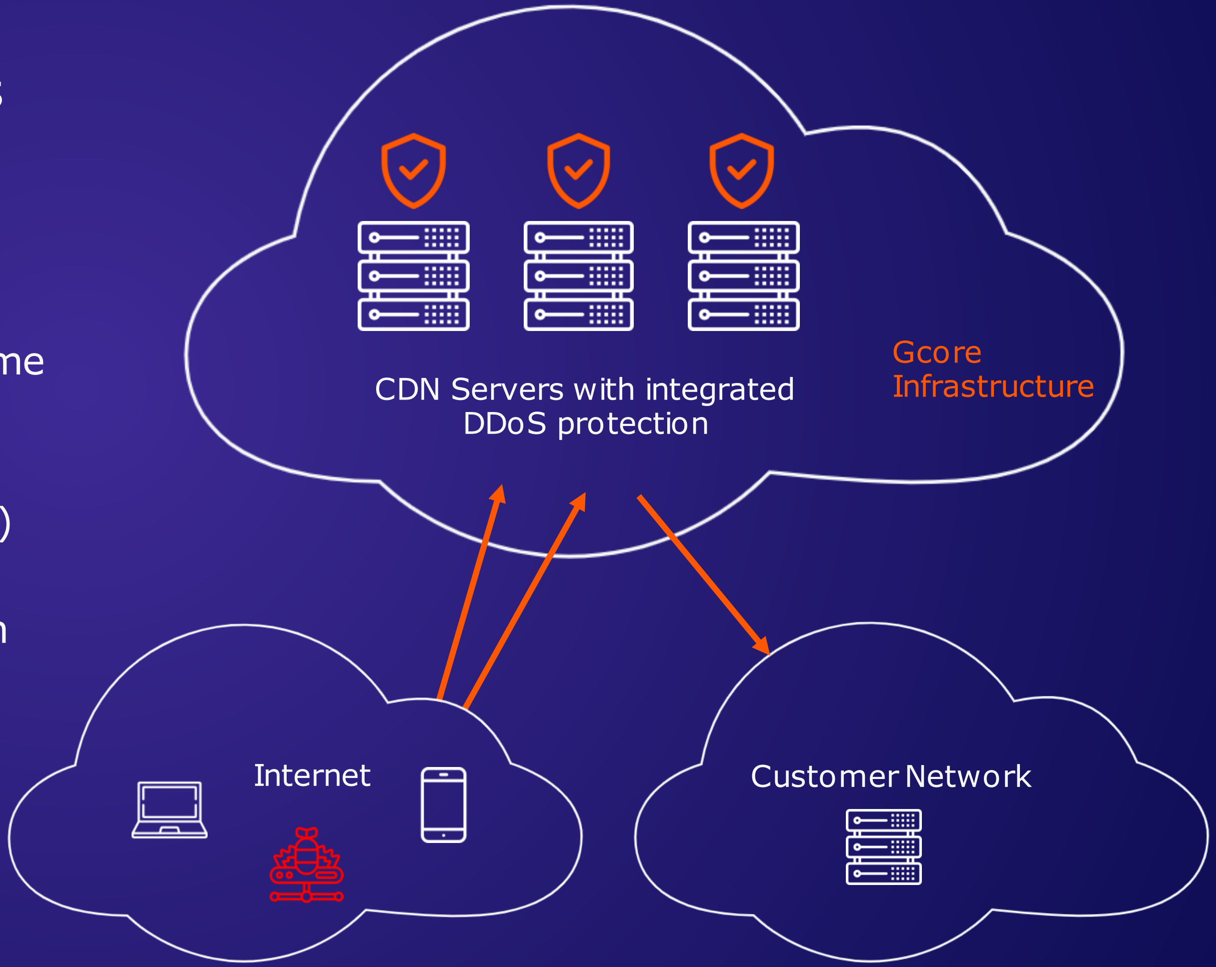
- ✓ Hundreds of CDN servers, dozens of standalone DDoS protection servers
- ✓ Protection servers only at selected locations
- ✓ Third-party solution with DPDK
- ✓ Asynchronous ingress/egress on both CDN and DDoS protection servers



CORE

Our new distributed network design

- ✓ Hundreds of CDN servers, each comes with DDoS protection
- ✓ XDP
- ✓ Multiple network-intensive applications on the same nodes
- ✓ Closer to client end-points (and DDoS generators)
- ✓ Standalone servers are still used during transition

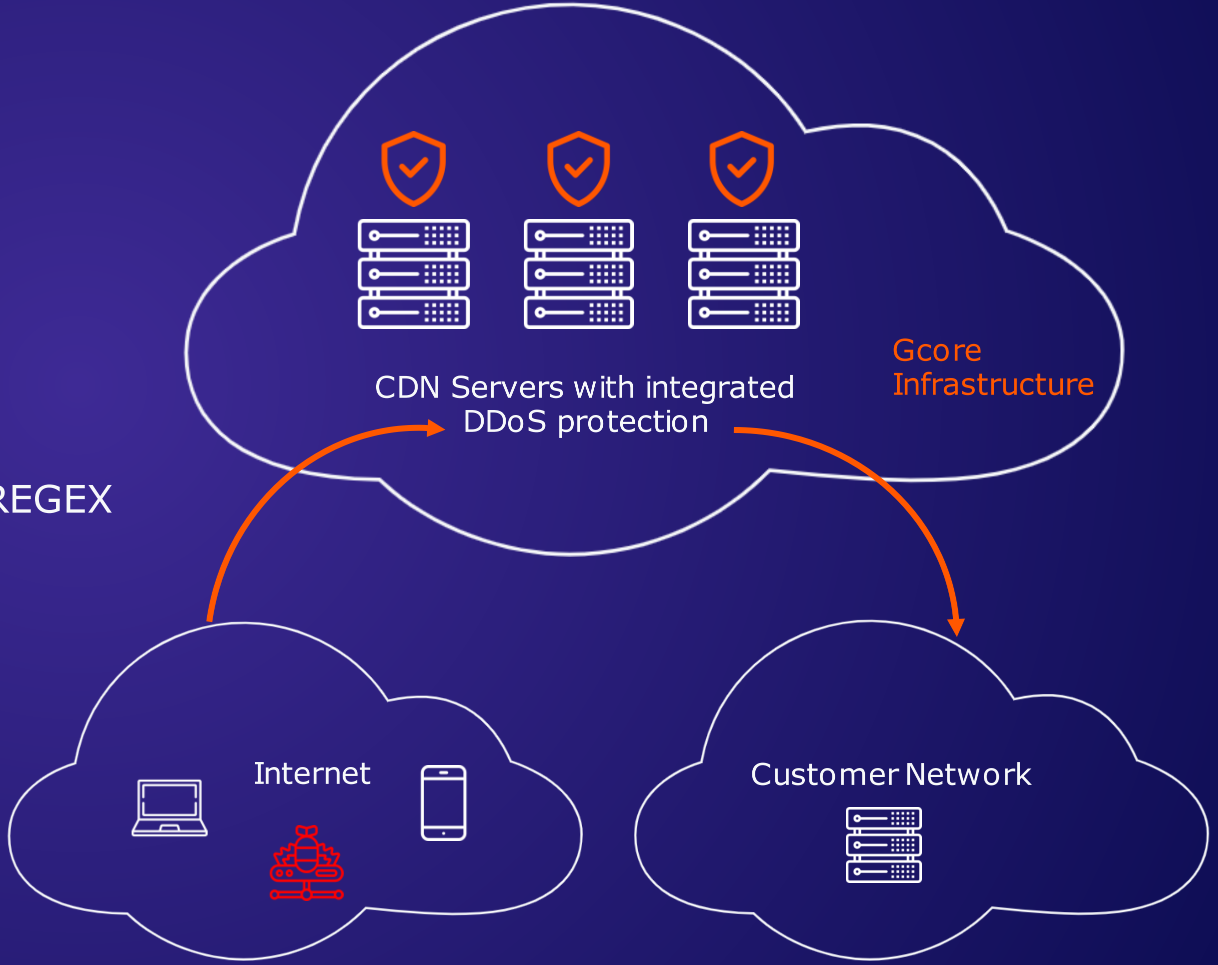


CORE

When we use regular expressions

Mostly game traffic is a subject
for protection by regular expressions

- ✓ UDP
- ✓ MTU < 1500
- ✓ Game protocols: strict format can be verified by REGEX
- ✓ Reaction to attacks: drop on pattern match

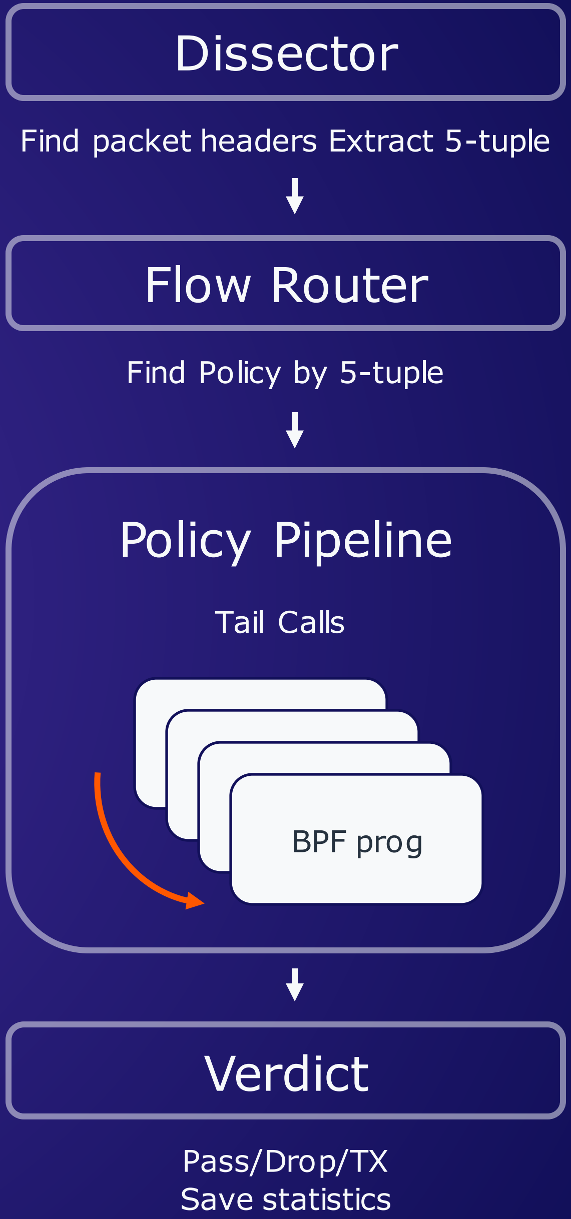


Regular expressions in XDP

- ✓ Security and DDoS Protection as a service
- ✓ XDP pipeline with REGEX
- ✓ Benchmarks
- ✓ Collaboration and further work

Our XDP pipeline

- ✓ Cover configuration for thousands of customers
- ✓ Order of countermeasures may differ
- ✓ REGEX is one of the countermeasures
- ✓ Not the first
- ✓ Not the last
- ✓ At the end traffic is either processed locally or sent to a customer network



REGEX in XDP: runtime

- ✓ Too big and complex: better to use an existing implementation
- ✓ Cannot fit eBPF limitations and pass through the verifier
- ✓ Efficient implementations require vector operations
- ✓ May have different performance, depending on patterns and traffic
- ✓ Understanding budgets while processing REGEX is crucial—performance degradation for one customer may lead to service degradation for all customers

REGEX runtime: Hyperscan

- ✓ BSD License
- ✓ Simultaneous matching of large numbers of regular expressions
- ✓ DPI as a common usage scenario
- ✓ Self-contained C runtime for scanning
- ✓ No memory allocations in hot path
- ✓ Can process multiple packets in one batch (not supported by XDP)

In-module eBPF helpers

A kernel module can define an eBPF helper function and dynamically extend capabilities of kernel and eBPF.

- ✓ Work started in version 5.16
- ✓ It was finalized in 5.18
- ✓ An eBPF helper cannot be registered for XDP until 5.18
- ✓ We had version 5.17 :-(

REGEX in XDP: vector operations in runtime

XDP runs in SoftIRQ, FPU is not used there

Need to save and restore FPU state:

- ✓ Per-packet inside XDP helper OR
- ✓ NAPI-wide

Other kernel subsystems work with FPU too, now FPU load/store operations must also disable interrupts and preemption

eBPF API

```
struct rex_scan_attr attr = {
    .database_id = regex_id,
    .handler_flags = REX_SINGLE_SHOT,
    .nr_events = 0,
    .last_event = {},
};

err = bpf_xdp_scan_bytes(xdp, payload_off, payload_len, &attr);
if (err < 0)
    return XDP_ABORTED;

return (attr.nr_events > 0) ? XDP_DROP : XDP_PASS;
```

REGEX in XDP: configuration

eBPF maps:

- ✓ All synchronization is already implemented
- ✓ Fixed entry size
- ✓ Application-specific

Configfs:

- ✓ Need to implement synchronization between management plane and data plane
- ✓ Flexible entry size
- ✓ More generic

REGEX in XDP: configuration

- ✓ Create a node using `mkdir` under `/sys/kernel/config/rex`
- ✓ Compile pattern database

```
echo '101:/foobar/' > patterns.txt
echo '201:/a{3,10}/' >> patterns.txt
build/bin/hscollider -e patterns.txt -ao out/ -n1
```

- ✓ Upload compiled regex to the `/sys/kernel/config/rex/<node>/database`

```
dd if=$(echo out/``.db) of=/sys/kernel/config/rex/hello/database
```

- ✓ Read or set new regex identifier at: `/sys/kernel/config/rex/<node>/id`
- ✓ Transfer regex identifier to eBPF program and use as a helper argument

Regular expressions in XDP

✓ Security and DDoS Protection as a service

✓ XDP pipeline with REGEX

✓ Benchmarks

✓ Collaboration and further work

Test Lab

intel

System under test, 1x server with 400G connectivity:

- ✓ 2x Intel Xeon Gold 6348 @ 2.60GHz
- ✓ 2x Intel E810-2cqda2 (2x 100G ports)

Traffic generators, 2x servers with 200G connectivity:

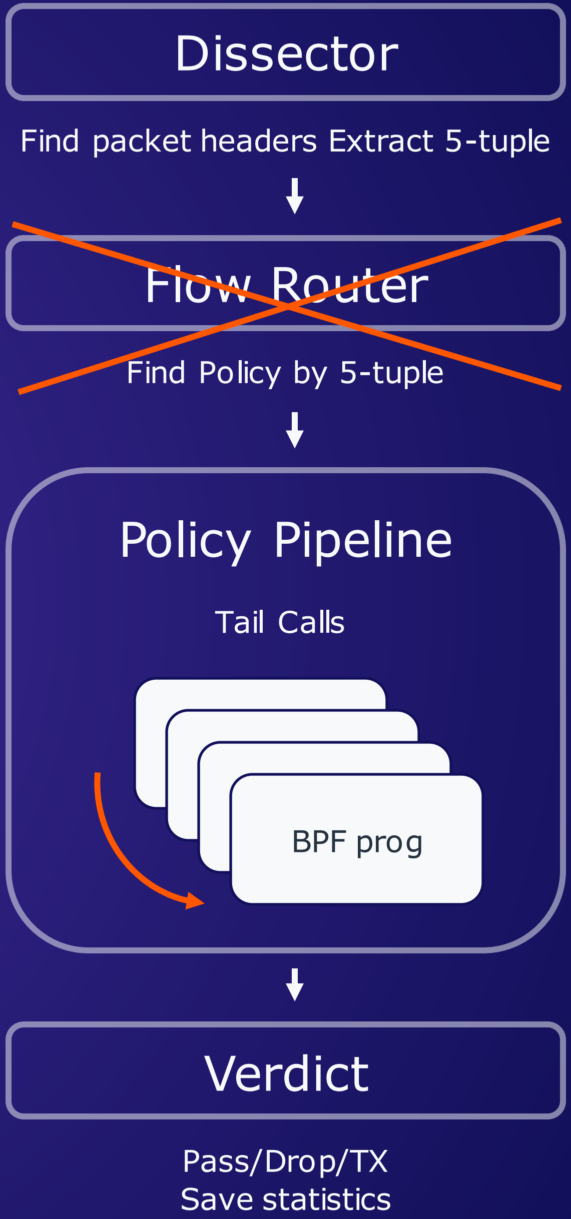
- ✓ 2x Intel Xeon Gold 6242R @ 3.10GHz
- ✓ 2x Intel E810 (Only one of 2x 100G ports is connected)

Test cases

- ✓ Base XDP throughput.
- ✓ Search for a literal inside of a packet payload. Regex: /private/s; Corpus: printable characters
- ✓ Search with access to the whole payload. Regex /pri.*ate/sH; Corpus: printable characters
- ✓ Our real-life regular expressions. 10 regexes in parallel, backtracking, search from the payload beginning
- ✓ Both XDP_DROP and XDP_TX actions tested

Program under test

- ✓ Dissect packet headers
- ✓ The flow router is CPU-hungry, disable it
- ✓ Only REGEX countermeasure is enabled
- ✓ Verdict is only XDP_TX or XDP_DROP
- ✓ Collect statistics in XDP



Dissector

Find packet headers Extract 5-tuple

~~Flow Router~~

~~Find Policy by 5-tuple~~

Policy Pipeline

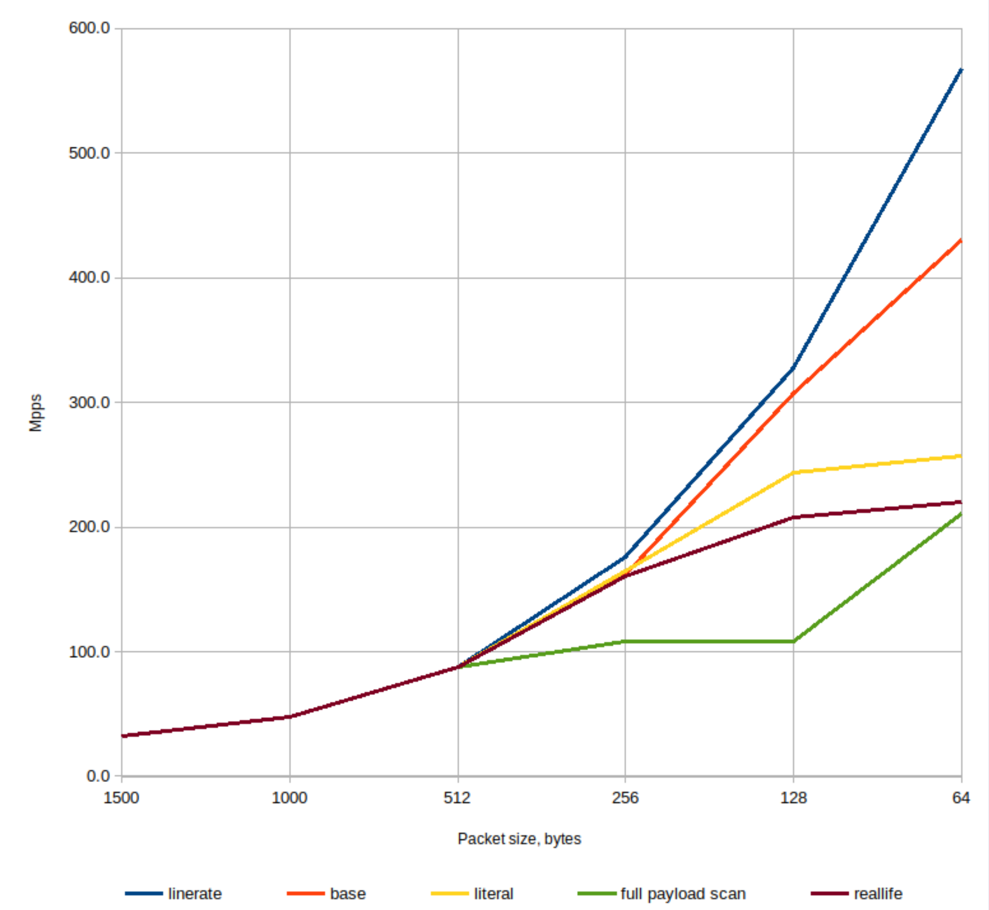
Tail Calls

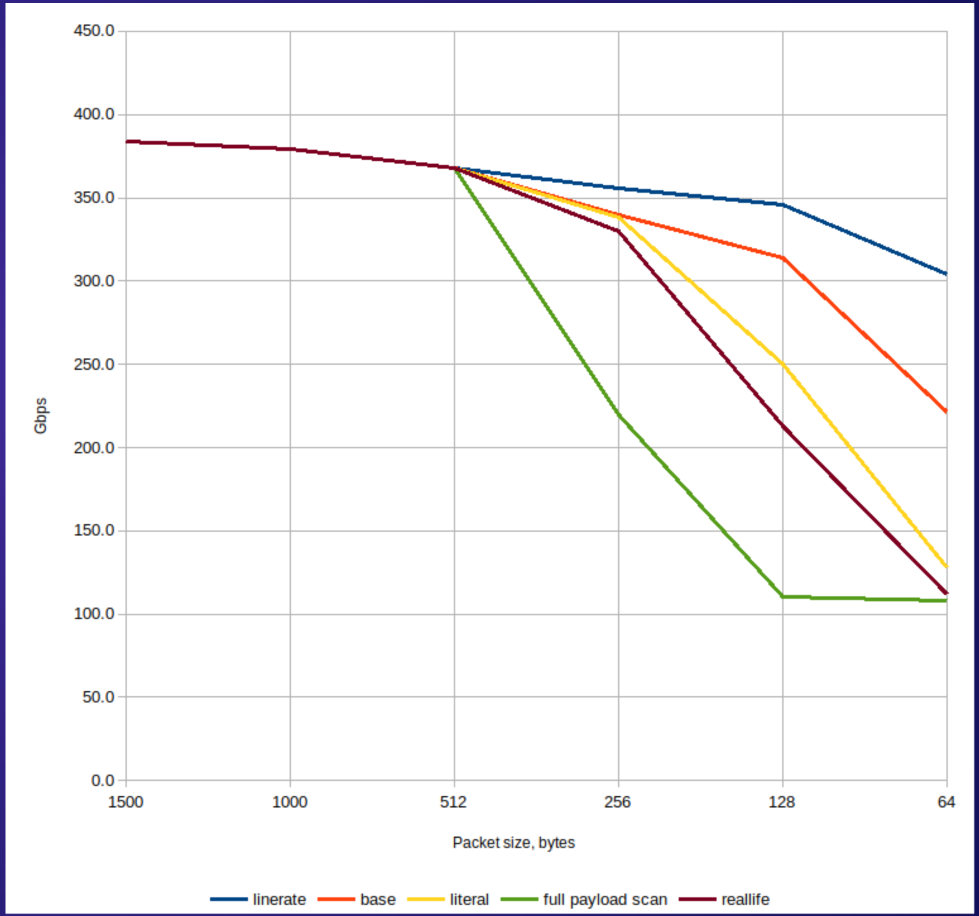
BPF prog

Verdict

Pass/Drop/TX
Save statistics

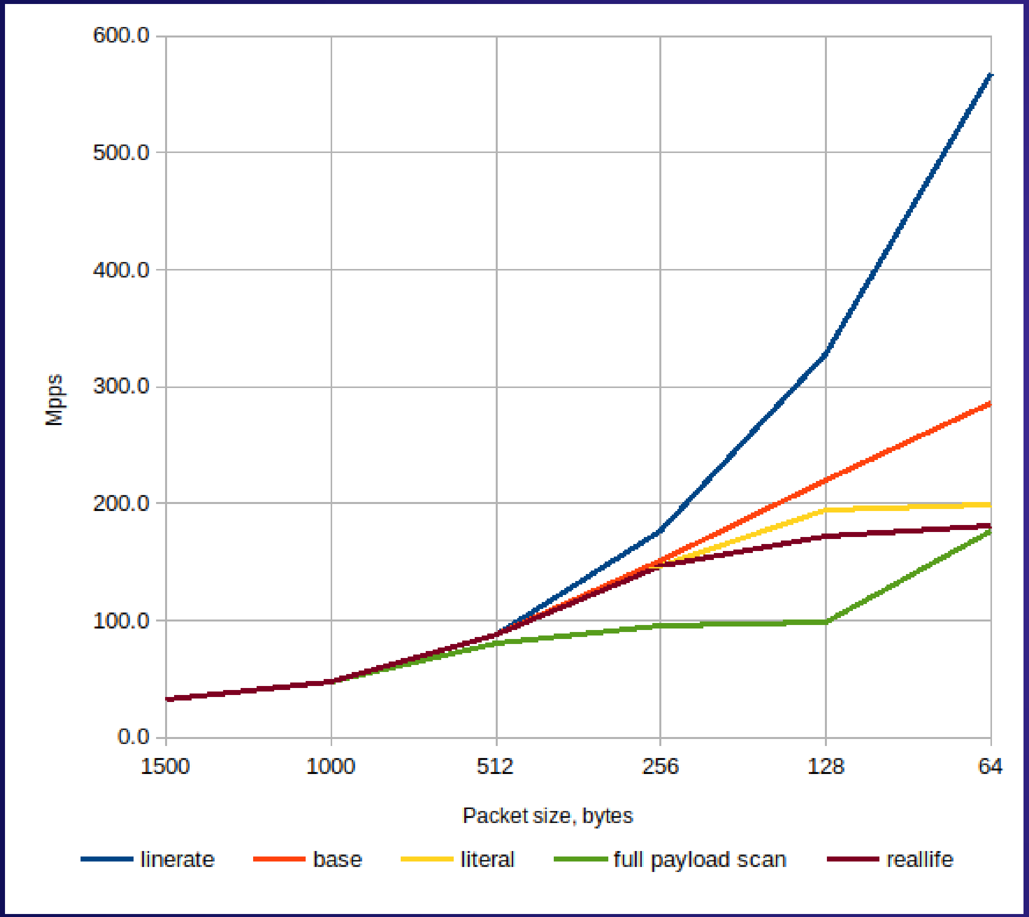
Benchmarks: XDP_DROP

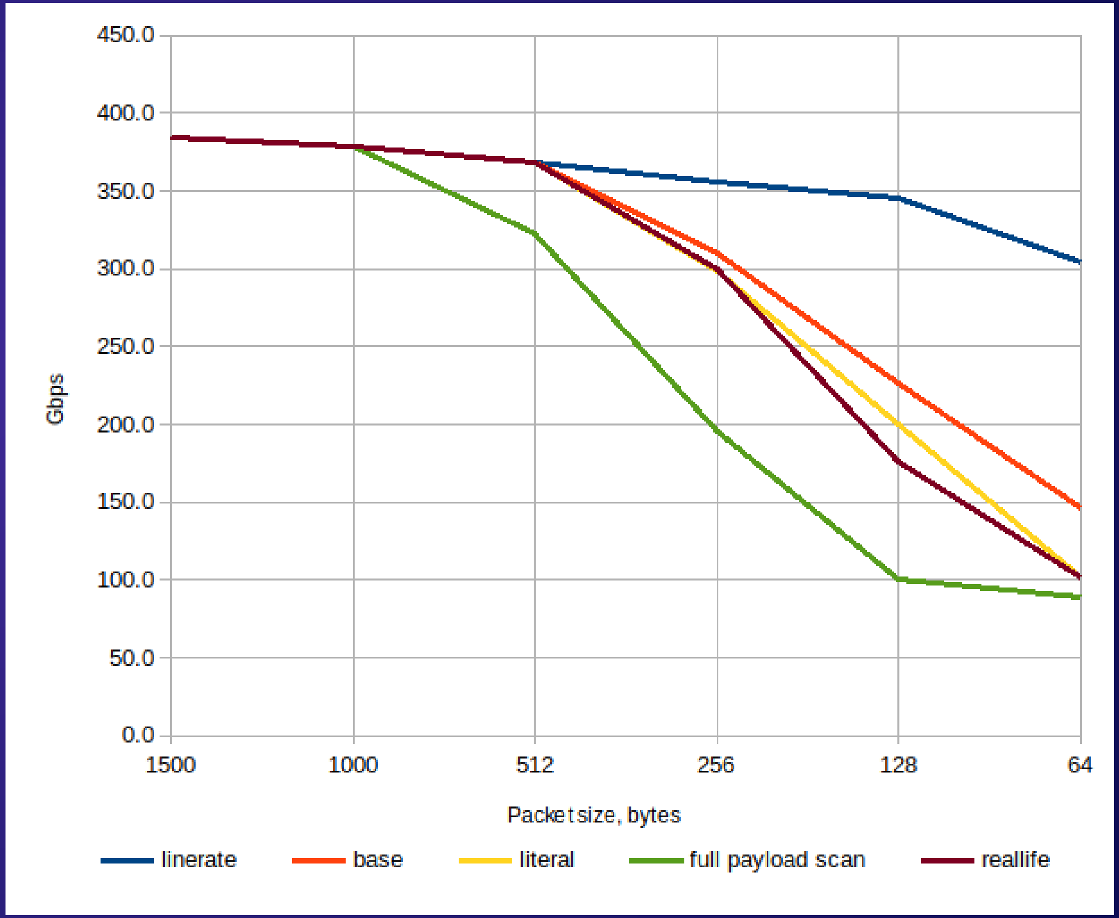




CORE

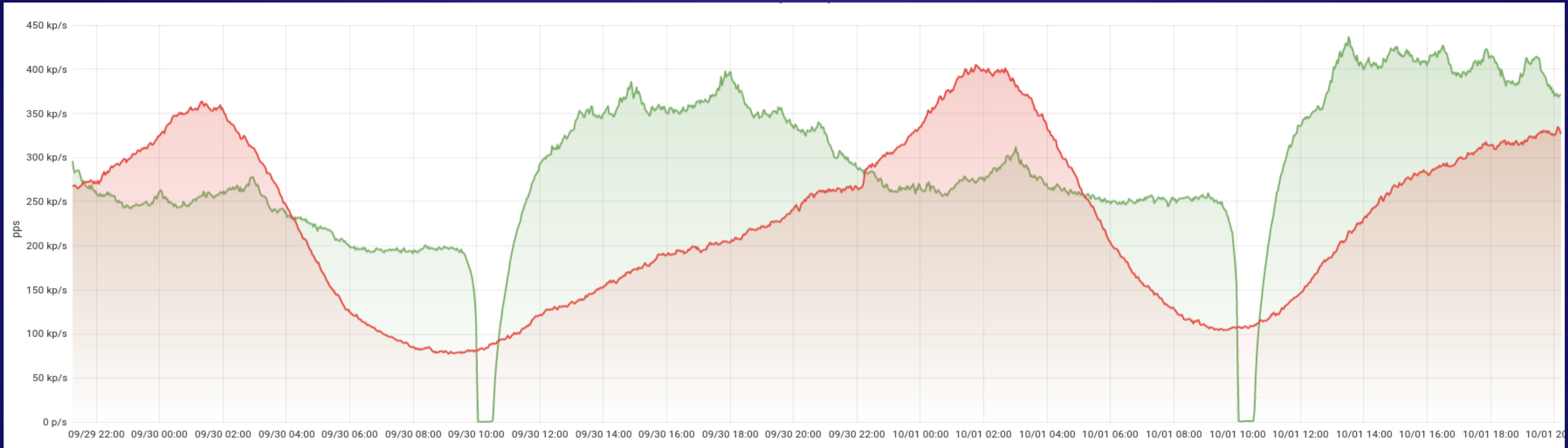
Benchmarks: XDP_TX





CORE

Already on production servers

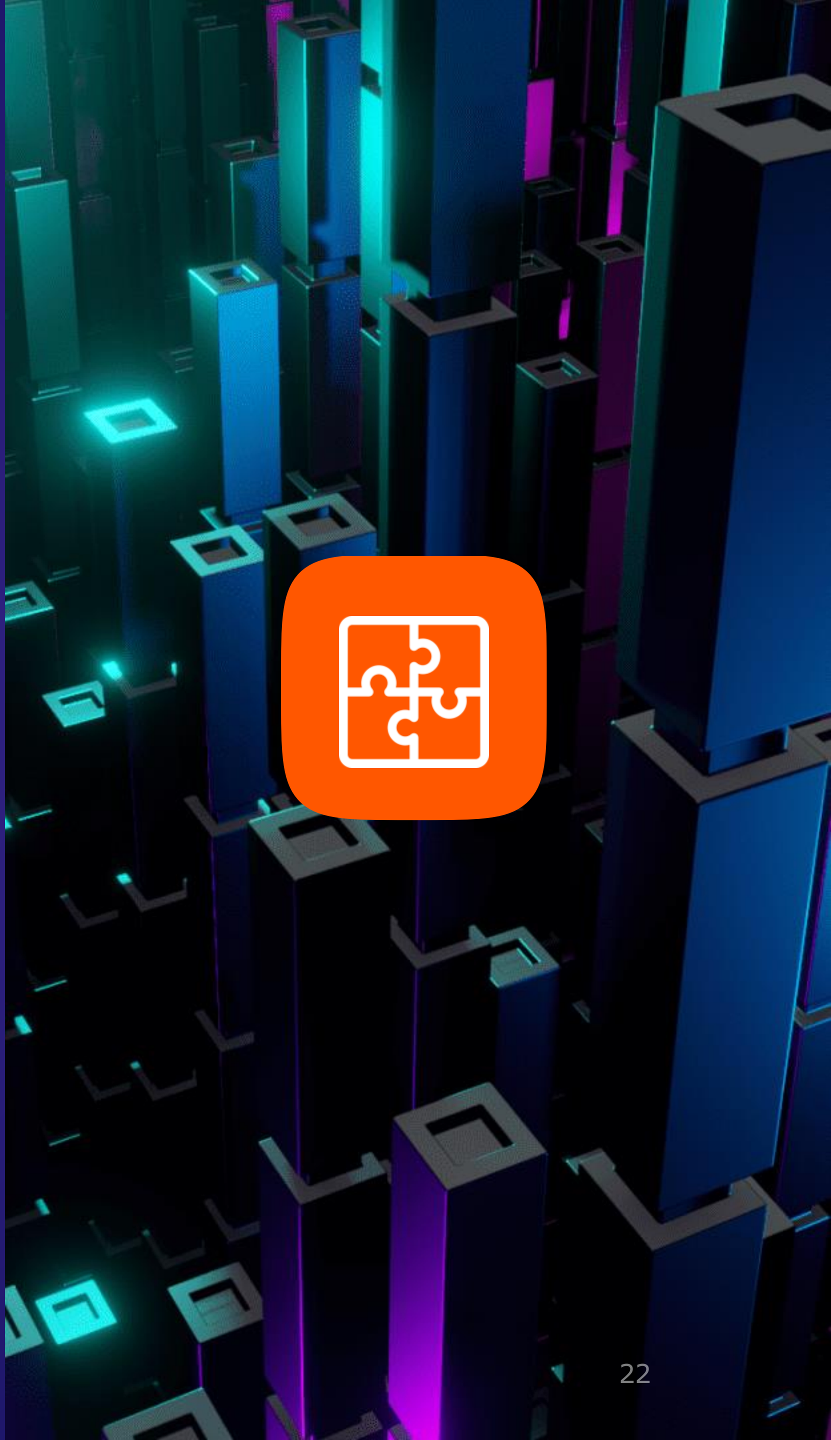


Very first tests (1 Mpps, 1 Gbps) on the real production traffic.
No latency/throughput degradation was found by customers.

CORE

Regular expressions in XDP

- ✓ Security and DDoS Protection as a service
- ✓ XDP pipeline with REGEX
- ✓ Benchmarks
- ✓ Collaboration and further work



CORE

XDP is golden but still under the DPDK shade

- ✓ No offloading. XDP Hints to the rescue
- ✓ NIC vendors have reference benchmarks for DPDK but not for XDP
- ✓ No configuration tuning guides from NIC vendors
- ✓ Some NICs require proprietary drivers, some still have no XDP support

But! NIC vendors are always open to help. Special thanks to colleagues from Intel: Piotr Raczynski, Michal Swiatkowski, Maciej Fijalkowski.

Further work

- ✓ Port to newer kernel version (5.18+ have better API for in-module eBPF helpers)
- ✓ Compare per-packet and NAPI-wide FPU save/restore approach
- ✓ REGEX budgeting. Automatically react if REGEX evaluation consumes too many CPU time
- ✓ Limit REGEX on configuration side to deny expressions, that ruin performance (REGEX bombs)

Source code

G-Core/linux-regex-module

CORE

Thank you!

Go Global Faster with Gcore CDN

IVAN KOVESHNIKOV

ivan.koveshnikov@gcore.com

SERGEY NIZOVTSEV

sn@tempesta-tech.com

We're hiring!

Extras

- ✓ Security and DDoS Protection as a service
- ✓ XDP pipeline with REGEX
- ✓ Benchmarks
- ✓ Collaboration and further work
- ✓ Extras

Raw results: XDP_DROP

pkt size	Linerate, mpps	base, mpps	literal, mpps	full payload	reallife, mpps
1500	32	32	32	32	32
1000	48	48	48	48	48
512	88	88	88	88	88
256	176	162	165	108	161
128	328	307	244	108	208
64	568	430	257	211	220

pkt size	Linerate, gbps	base, gbps	literal, gbps	full payload	reallife, gbps
1500	384	384	384	384	384
1000	379	379	379	379	379
512	368	349.5	349.5	349.5	349.5
256	356	340	338	220	330
128	346	314	250	110	213
64	304	221	128	108	112

Raw results: XDP_TX

pkt size	Linerate, mpps	Base, mpps	Literal, mpps	full payload	Reallife, mpps
1500	32	32	32	32	32
1000	48	48	48	48	48
512	88	88	88	80	88
256	176	151	146	95	146
128	328	220	195	98	172
64	568	286	199	176	181

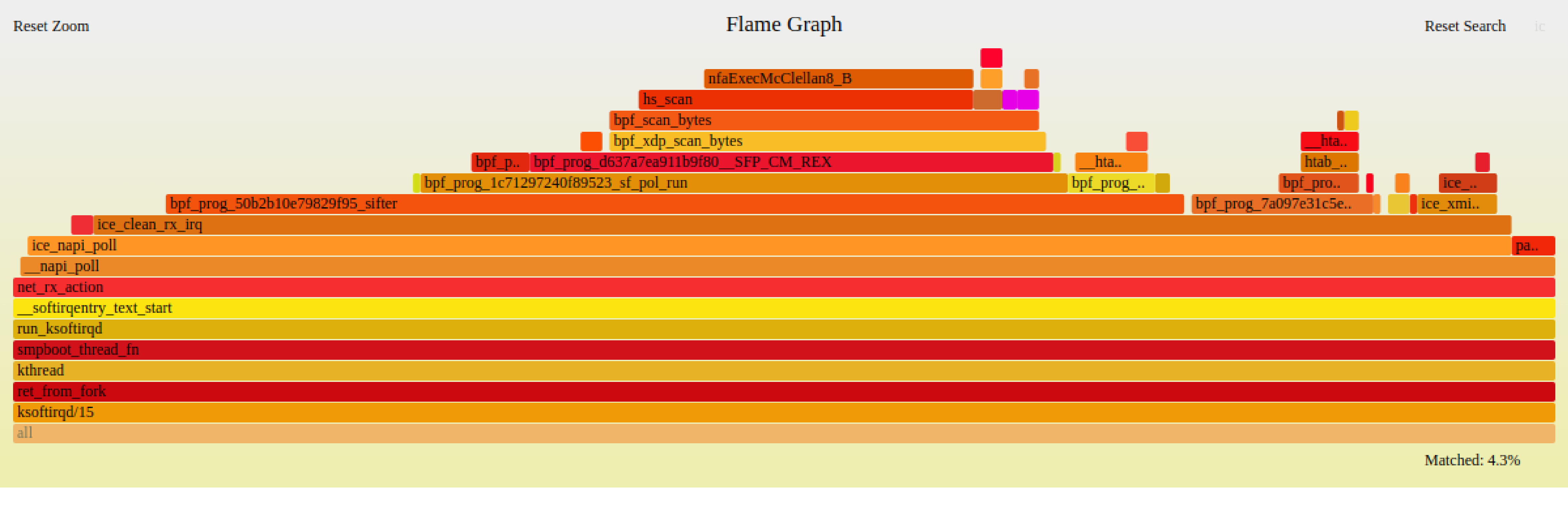
pkt size	Linerate, gbps	base, gbps	literal, gbps	full payload	reallife, gbps
1500	384	384	384	384	384
1000	379	379	379	379	379
512	368	349.5	349.5	323	349.5
256	356	310	299	196	300
128	346	226	200	100	176
64	304	146	102	89	101

Raw results: CPU usage. DROP vs TX

pkt size	base	literal	full payload scan	reallife
1500	6%	14%	33%	15%
1000	8%	20%	48%	22%
512	16%	34%	80%	38%
256	28%	68%	98%	74%
128	51%	98%	98%	98%
64	75%	98%	98%	99%

pkt size	base	literal	full payload scan	reallife
1500	10%	19%	40%	18%
1000	12%	28%	56%	26%
512	22%	48%	98%	45%
256	36%	75%	98%	86%
128	53%	98%	98%	99%
64	83%	98%	98%	99%

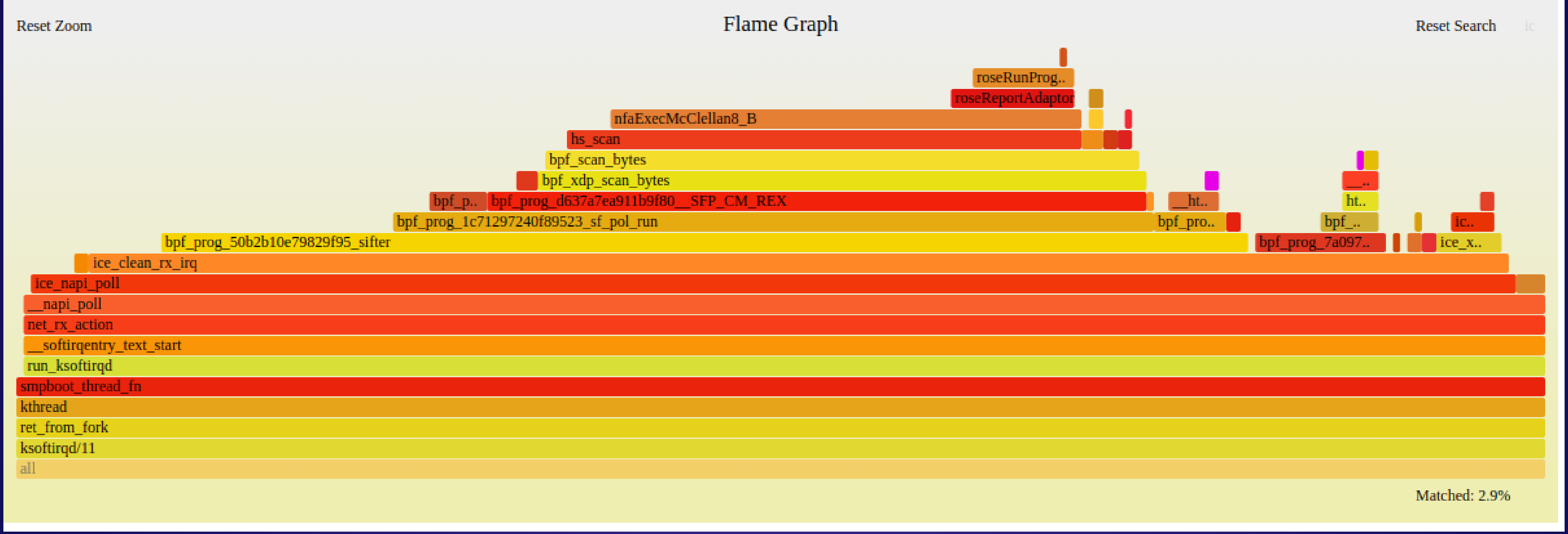
Flame graphs: XDP_DROP



BPF prog: 75%   Hyperscan: 28%   FPU load/store: 2.1%

CORE

Flame graphs: XDP_TX



BPF prog: 78% Hyperscan: 40% FPU load/store: 1.7%

CORE