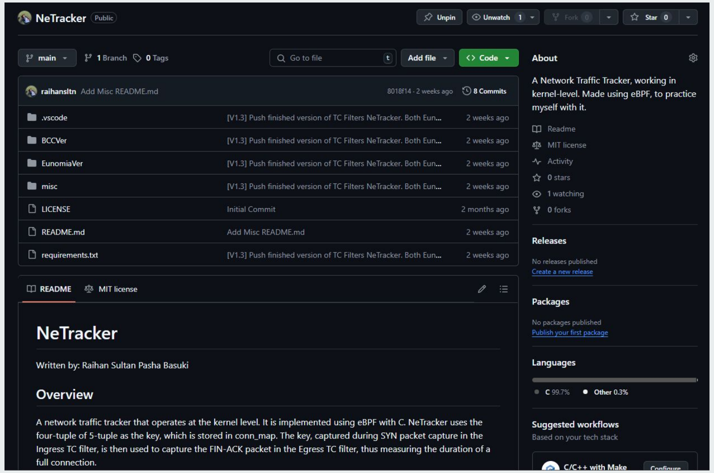
NeTracker

By: Raihan Sultan Pasha Basuki

A network traffic tracker that operates at the kernel level. It is implemented using eBPF with C.

NeTracker eBPF Program is built using C language, and also it has its BCC Loader with Python language.



raihansltn Add Misc README.md 8018f14 · 2 weeks ago 8 Commits

.vscode	[V1.3] Push finished version of TC Filters NeTracker. Both Eun...	2 weeks ago
BCCVer	[V1.3] Push finished version of TC Filters NeTracker. Both Eun...	2 weeks ago
EunomiaVer	[V1.3] Push finished version of TC Filters NeTracker. Both Eun...	2 weeks ago
misc	[V1.3] Push finished version of TC Filters NeTracker. Both Eun...	2 weeks ago
LICENSE	Initial Commit	2 months ago
README.md	Add Misc README.md	2 weeks ago
requirements.txt	[V1.3] Push finished version of TC Filters NeTracker. Both Eun...	2 weeks ago

NeTracker

Written by: Raihan Sultan Pasha Basuki

Overview

A network traffic tracker that operates at the kernel level. It is implemented using eBPF with C. NeTracker uses the four-tuple of 5-tuple as the key, which is stored in conn_map. The key, captured during SYN packet capture in the Ingress TC filter, is then used to capture the FIN-ACK packet in the Egress TC filter, thus measuring the duration of a full connection.

About

A Network Traffic Tracker, working in kernel-level. Made using eBPF, to practice myself with it.

Readme
MIT license
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

C 99.7% Other 0.3%

Suggested workflows

Based on your tech stack

C/C++ with Make Configure

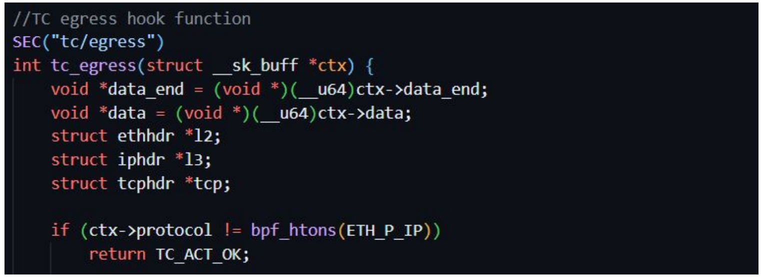
github.com/raihansltn/NeTracker

How does it work?

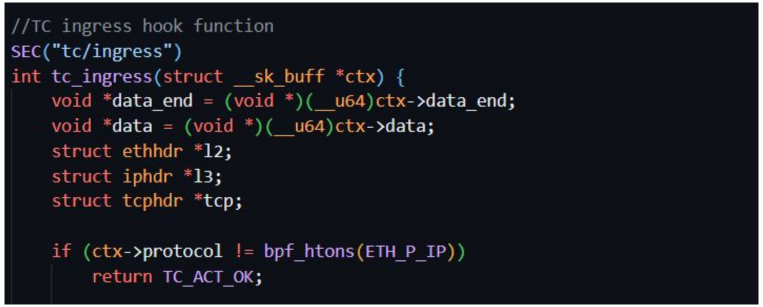
NeTracker utilizes the four-tuple (source IP, destination IP, source port, destination port) as a key, stored in a connection map (conn_map). It captures SYN packets by getting attached at the ingress Traffic Control (TC) filter and FIN-ACK packets at the egress TC filter. By tracking these packets, NeTracker measures the duration of full TCP connections.

The traffic log then readable through /sys/kernel/debug/tracing/trace_pipe or using bpftool

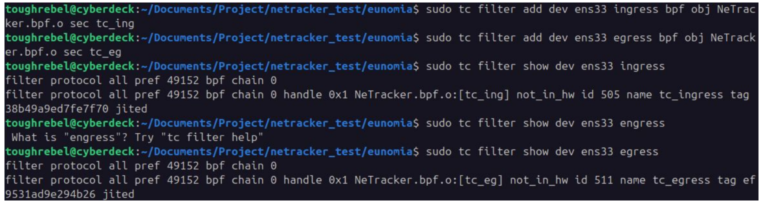
github.com/raihansltn/NeTracker



```c
//TC egress hook function
SEC("tc/egress")
int tc_egress(struct __sk_buff *ctx) {
    void *data_end = (void *)(__u64)ctx->data_end;
    void *data = (void *)(__u64)ctx->data;
    struct ethhdr *l2;
    struct iphdr *l3;
    struct tcphdr *tcp;

    if (ctx->protocol != bpf_htons(ETH_P_IP))
        return TC_ACT_OK;
```



```c
//TC ingress hook function
SEC("tc/ingress")
int tc_ingress(struct __sk_buff *ctx) {
    void *data_end = (void *)(__u64)ctx->data_end;
    void *data = (void *)(__u64)ctx->data;
    struct ethhdr *l2;
    struct iphdr *l3;
    struct tcphdr *tcp;

    if (ctx->protocol != bpf_htons(ETH_P_IP))
        return TC_ACT_OK;
```



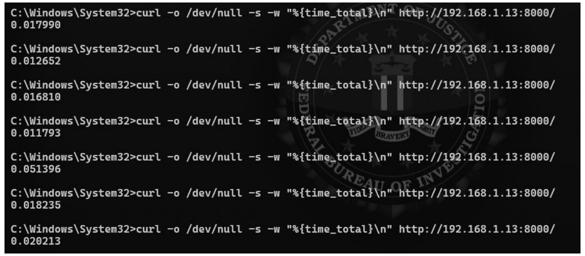
```
toughrebel@cyberdeck:~/Documents/Project/netracker_test/eunomia$ sudo tc filter add dev ens33 ingress bpf obj NeTrac
ker.bpf.o sec tc_ing
toughrebel@cyberdeck:~/Documents/Project/netracker_test/eunomia$ sudo tc filter add dev ens33 egress bpf obj NeTrack
er.bpf.o sec tc_eg
toughrebel@cyberdeck:~/Documents/Project/netracker_test/eunomia$ sudo tc filter show dev ens33 ingress
filter protocol all pref 49152 bpf chain 0
filter protocol all pref 49152 bpf chain 0 handle 0x1 NeTracker.bpf.o:[tc_ing] not_in_hw id 505 name tc_ingress tag
38b49a9ed7fe7f70 jited
toughrebel@cyberdeck:~/Documents/Project/netracker_test/eunomia$ sudo tc filter show dev ens33 engress
 What is "engress"? Try "tc filter help"
toughrebel@cyberdeck:~/Documents/Project/netracker_test/eunomia$ sudo tc filter show dev ens33 egress
filter protocol all pref 49152 bpf chain 0
filter protocol all pref 49152 bpf chain 0 handle 0x1 NeTracker.bpf.o:[tc_eg] not_in_hw id 511 name tc_egress tag ef
9531ad9e294b26 jited
```

Result

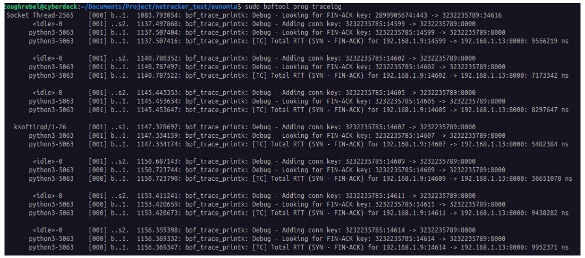
NeTracker successfully demonstrates the capability to track TCP connection durations at the kernel level using eBPF.

As seen, the 2 images on the top are the result of curl time using {time_total} flag representing the total time in seconds, from the start until the transfer is completed, which covered SYN to FIN-ACK. To the bottom, there are 2 images result when I sent a connection through browser to http server. Both tests, resulted in 1.000.000 to 9.000.000 ns differences, because the filters trace time both when SYN hit ingress and FIN ACK hit egress respectively, while the curl measured the total elapsed time of the entire request.

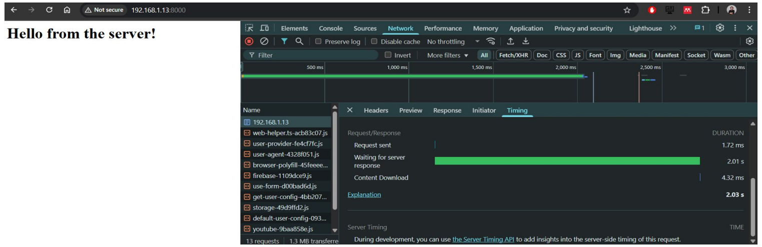
github.com/raihansltn/NeTracker



```
C:\Windows\System32>curl -o /dev/null -s -w "%{time_total}\n" http://192.168.1.13:8000/
0.017990

C:\Windows\System32>curl -o /dev/null -s -w "%{time_total}\n" http://192.168.1.13:8000/
0.012652

C:\Windows\System32>curl -o /dev/null -s -w "%{time_total}\n" http://192.168.1.13:8000/
0.016810

C:\Windows\System32>curl -o /dev/null -s -w "%{time_total}\n" http://192.168.1.13:8000/
0.011793

C:\Windows\System32>curl -o /dev/null -s -w "%{time_total}\n" http://192.168.1.13:8000/
0.051396

C:\Windows\System32>curl -o /dev/null -s -w "%{time_total}\n" http://192.168.1.13:8000/
0.018235

C:\Windows\System32>curl -o /dev/null -s -w "%{time_total}\n" http://192.168.1.13:8000/
0.020213
```



```
toughrebel@cyberdeck:~/Documents/Project/netracker_test/eunomia$ sudo bpftool prog tracelog
 Socket Thread-2565    [000] b..1.  1083.793054: bpf_trace_printk: Debug - Looking for FIN-ACK key: 2899905674:443 -> 3232235789:34616
          <idle>-0     [001] ..s2.  1137.497868: bpf_trace_printk: Debug - Adding conn key: 3232235785:14599 -> 3232235789:8000
    python3-5063       [001] b..1.  1137.507404: bpf_trace_printk: Debug - Looking for FIN-ACK key: 3232235785:14599 -> 3232235789:8000
    python3-5063       [001] b..1.  1137.507416: bpf_trace_printk: [TC] Total RTT (SYN - FIN-ACK) for 192.168.1.9:14599 -> 192.168.1.13:8000: 9556219 ns

          <idle>-0     [001] ..s2.  1140.780352: bpf_trace_printk: Debug - Adding conn key: 3232235785:14602 -> 3232235789:8000
    python3-5063       [001] b..1.  1140.787497: bpf_trace_printk: Debug - Looking for FIN-ACK key: 3232235785:14602 -> 3232235789:8000
    python3-5063       [001] b..1.  1140.787522: bpf_trace_printk: [TC] Total RTT (SYN - FIN-ACK) for 192.168.1.9:14602 -> 192.168.1.13:8000: 7173342 ns

          <idle>-0     [001] ..s2.  1145.445353: bpf_trace_printk: Debug - Adding conn key: 3232235785:14605 -> 3232235789:8000
    python3-5063       [001] b..1.  1145.453634: bpf_trace_printk: Debug - Looking for FIN-ACK key: 3232235785:14605 -> 3232235789:8000
    python3-5063       [001] b..1.  1145.453647: bpf_trace_printk: [TC] Total RTT (SYN - FIN-ACK) for 192.168.1.9:14605 -> 192.168.1.13:8000: 8297647 ns

  ksoftirqd/1-26       [001] ..s1.  1147.328697: bpf_trace_printk: Debug - Adding conn key: 3232235785:14607 -> 3232235789:8000
    python3-5063       [001] b..1.  1147.334159: bpf_trace_printk: Debug - Looking for FIN-ACK key: 3232235785:14607 -> 3232235789:8000
    python3-5063       [001] b..1.  1147.334174: bpf_trace_printk: [TC] Total RTT (SYN - FIN-ACK) for 192.168.1.9:14607 -> 192.168.1.13:8000: 5482384 ns

          <idle>-0     [001] ..s2.  1150.687143: bpf_trace_printk: Debug - Adding conn key: 3232235785:14609 -> 3232235789:8000
    python3-5063       [000] b..1.  1150.723744: bpf_trace_printk: Debug - Looking for FIN-ACK key: 3232235785:14609 -> 3232235789:8000
    python3-5063       [000] b..1.  1150.723790: bpf_trace_printk: [TC] Total RTT (SYN - FIN-ACK) for 192.168.1.9:14609 -> 192.168.1.13:8000: 36651078 ns

          <idle>-0     [001] ..s2.  1153.411241: bpf_trace_printk: Debug - Adding conn key: 3232235785:14611 -> 3232235789:8000
    python3-5063       [000] b..1.  1153.420659: bpf_trace_printk: Debug - Looking for FIN-ACK key: 3232235785:14611 -> 3232235789:8000
    python3-5063       [000] b..1.  1153.420673: bpf_trace_printk: [TC] Total RTT (SYN - FIN-ACK) for 192.168.1.9:14611 -> 192.168.1.13:8000: 9438282 ns

          <idle>-0     [001] ..s2.  1156.359398: bpf_trace_printk: Debug - Adding conn key: 3232235785:14614 -> 3232235789:8000
    python3-5063       [000] b..1.  1156.369332: bpf_trace_printk: Debug - Looking for FIN-ACK key: 3232235785:14614 -> 3232235789:8000
    python3-5063       [000] b..1.  1156.369347: bpf_trace_printk: [TC] Total RTT (SYN - FIN-ACK) for 192.168.1.9:14614 -> 192.168.1.13:8000: 9952371 ns
```


Not secure
192.168.1.13:8000
Elements Console Sources Network Performance Memory Application Privacy and security Lighthouse
Preserve log
Disable cache
No throttling
Filter
Invert
More filters
All Fetch/XHR Doc CSS JS Font Img Media Manifest Socket Wasm Other
500 ms 1,000 ms 1,500 ms 2,000 ms 2,500 ms 3,000 ms
Headers Preview Response Initiator Timing
Explanation

Hello from the server!

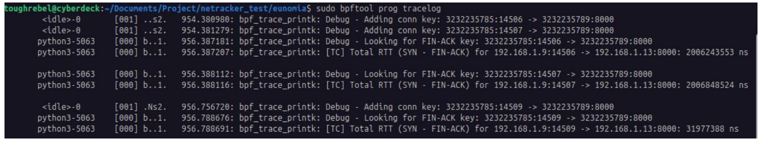
Name
192.168.1.13
web-helper.ts-acb83c07.js
user-provider-fe4cf7fc.js
user-agent-4328f051.js
browser-polyfill-45feeee...
firebase-1109dce9.js
use-form-d00bad6d.js
get-user-config-4bb207...
storage-49d9ffd2.js
default-user-config-093...
youtube-9baa858e.js

13 requests | 1.3 MB transferre

Request/Response	DURATION
Request sent	1.72 ms
Waiting for server response	2.01 s
Content Download	4.32 ms
Explanation	2.03 s

Server Timing — TIME

During development, you can use the Server Timing API to add insights into the server-side timing of this request.



```
toughrebel@cyberdeck:~/Documents/Project/netracker_test/eunomia$ sudo bpftool prog tracelog
          <idle>-0     [001] ..s2.   954.380980: bpf_trace_printk: Debug - Adding conn key: 3232235785:14506 -> 3232235789:8000
          <idle>-0     [001] ..s2.   954.381279: bpf_trace_printk: Debug - Adding conn key: 3232235785:14507 -> 3232235789:8000
    python3-5063       [000] b..1.   956.387181: bpf_trace_printk: Debug - Looking for FIN-ACK key: 3232235785:14506 -> 3232235789:8000
    python3-5063       [000] b..1.   956.387207: bpf_trace_printk: [TC] Total RTT (SYN - FIN-ACK) for 192.168.1.9:14506 -> 192.168.1.13:8000: 2006243553 ns

    python3-5063       [000] b..1.   956.388112: bpf_trace_printk: Debug - Looking for FIN-ACK key: 3232235785:14507 -> 3232235789:8000
    python3-5063       [000] b..1.   956.388116: bpf_trace_printk: [TC] Total RTT (SYN - FIN-ACK) for 192.168.1.9:14507 -> 192.168.1.13:8000: 2006848524 ns

          <idle>-0     [001] .Ns2.   956.756720: bpf_trace_printk: Debug - Adding conn key: 3232235785:14509 -> 3232235789:8000
    python3-5063       [000] b..1.   956.788676: bpf_trace_printk: Debug - Looking for FIN-ACK key: 3232235785:14509 -> 3232235789:8000
    python3-5063       [000] b..1.   956.788691: bpf_trace_printk: [TC] Total RTT (SYN - FIN-ACK) for 192.168.1.9:14509 -> 192.168.1.13:8000: 31977388 ns
```