

TAK T00R
ANDAZ 5
شکار زنبر



پرروزه دوره

تکتورانداز

پرروزه

تیم شبکه و زیرساخت ابری
کروه عزم افزایی ساره

پروژه دوره

تک توراندازه

شرح پروژه پیاده سازی stateful Natting in ebpf

فاطمه یوسفی

مدرس دوره: علی جاویدی

پروردگار دوره

تک توراندازه

فهرست

۴.....	مقدمه
۴.....	راهنمای شروع
۵	مفاهیم اساسی
۹.....	راهنمای استفاده
۱۰.....	مستند فنی
۱۷.....	مراجع و منابع

تک توراندازه

: مقدمه

در این دوره، ابتدا با مفاهیم شبکه آشنا شدیم و سپس به مسیریابی با استفاده از `nftables` پرداختیم. در ادامه، مفاهیم زبان C را مورد بررسی قرار دادیم و در نهایت با `eBPF` آشنا شدیم.

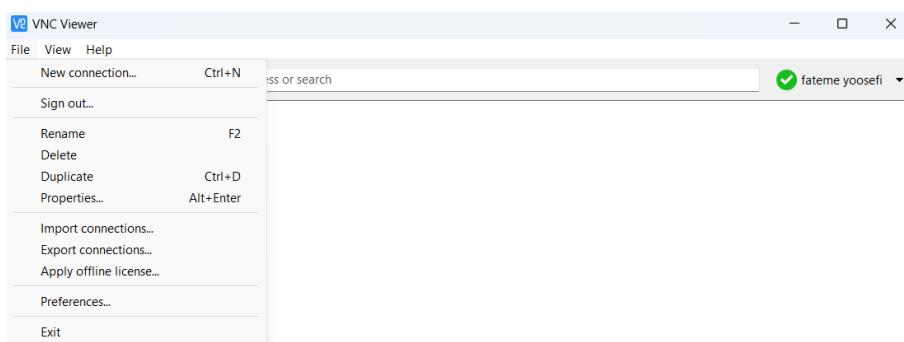
در نهایت پروژه ای تعریف گردید که خلاصه آن به شرح زیر است:

ما یک سیستم کلاینت با **10.2.6.1** : `ip:10.1.6.1` داریم که قرار است از اینترنت که `ip:10.2.6.2` با `ens4` وجود دارد و بین اینترنت و ماشین مجازی ما میگذرد . از طرفی طبق عکس زیر بین کلاینت و ماشین مجازی **10.2.6.2** با `ip:10.1.6.2` با `ens3` وجود دارد و کلاینت بتواند اینترنت را پینگ بگیرد.

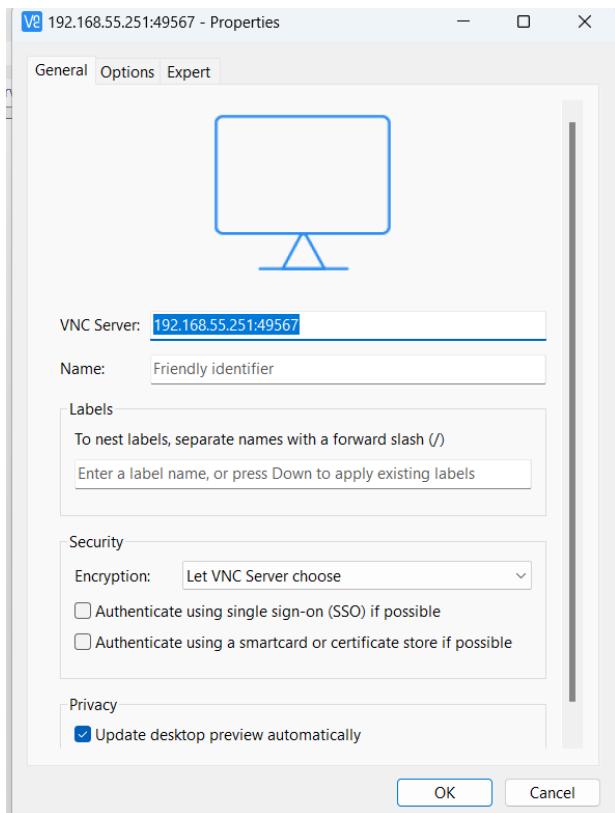
```
root@taktoor-5-dev:/home/college# ip -br ad
lo          UNKNOWN      127.0.0.1/8 ::1/128
ens3         UP          10.1.6.2/30 fe80::5203:ff:fe06:0/64
ens4         UP          10.2.6.2/30 fe80::5203:ff:fe06:1/64
ens5         UP          192.168.55.206/26 fe80::5203:ff:fe06:2/64
```

: شروع راهنمای

در ابتداء برنامه `vnc viewer` را نصب میکنیم. سپس مانند شکل زیر یک ارتباط جدید ایجاد میکنیم.



تک توراندازه



و مانند شکل مقادیر را وارد میکنیم.

و میتوانیم با این کار به ماشین کلاینت متصل شویم.

مفاهیم اساسی:

:NAT ✓

NAT یا Network Address Translation یک تکنولوژی در شبکه‌های کامپیوتری است که برای ترجمه آدرس‌های IP استفاده می‌شود. این تکنولوژی معمولاً در روترهای فایروال برای مدیریت ترافیک شبکه استفاده می‌شود.

تک توراندازه

هدف اصلی از NAT، تبدیل آدرس‌های IP داخلی داخل شبکه به آدرس‌های IP قابل دسترسی از خارج شبکه و بالعکس است. این کار به عنوان یک واسط بین شبکه داخلی و شبکه‌های عمومی انجام می‌شود و به امکان ارتباط با اینترنت از طریق آدرس‌های IP عمومی کمک می‌کند.

نکات کلیدی در مورد NAT شامل موارد زیر می‌شوند:

۱. ترجمه آدرس‌های IP: NAT آدرس‌های IP در بستر IP پکت‌ها را ترجمه می‌کند. این ترجمه می‌تواند از آدرس‌های IP عمومی به آدرس‌های IP داخلی و بالعکس باشد.

۲. پورت‌های TCP/UDP: NAT همچنین می‌تواند پورت‌های TCP و UDP را نیز ترجمه کند، اجازه می‌دهد که چندین دستگاه از یک آدرس IP عمومی استفاده کنند.

۳. نوع NAT: وجود چندین نوع NAT از جمله Static NAT و Dynamic NAT. PAT (Port Address Translation) که هر کدام ویژگی‌ها و مزایای مختلفی دارند.

۴. مزایای امنیتی: NAT می‌تواند به عنوان یک نقطه کنترل دسترسی به شبکه عمل کند و حملات مخرب از شبکه خارجی را کاهش دهد.

به طور کلی، NAT یک ابزار مهم در مدیریت ترافیک شبکه است که امکان ایجاد اتصالات امن و افزایش کارایی را فراهم می‌کند.

تک توراندازه

: Stateful ✓

یک تکنولوژی است که معمولاً در روتراها و فایروال‌ها استفاده می‌شود تا آدرس‌های IP داخلی را به آدرس‌های IP عمومی ترجمه کند و بالعکس. وقتی که NAT به صورت **stateful** است، به این معناست که دارای ذاکرهای (state table) است که وضعیت (state) هر اتصال شبکه را نگهداری می‌کند.

در NAT stateful، وضعیت هر اتصال شبکه شامل اطلاعاتی مانند آدرس IP مبدأ و مقصد، پورت‌های مبدأ و مقصد، وضعیت اتصال (مثلاً اتصال فعال یا غیرفعال) و سایر جزئیات مربوط به اتصال است. این اطلاعات در یک جدول state table نگهداری می‌شوند.

وقتی که بسته‌های شبکه از داخل شبکه به بیرون ارسال می‌شوند، NAT این اطلاعات را برای هر بسته بررسی می‌کند و مطابق با آنها آدرس‌های IP و پورت‌ها را ترجمه می‌کند. به عبارت دیگر، NAT با استفاده از اطلاعات موجود در state table، بسته‌های خروجی را به طور دینامیک ترجمه می‌کند.

استفاده از NAT stateful امکان ایجاد اتصالات برای ارتباطات دوطرفه و موثرتر میان داخلی و خارجی را فراهم می‌کند. این ویژگی مخصوصاً در محیط‌های شبکه‌های بزرگ و پیچیده که نیاز به مدیریت دقیق ترافیک دارند، بسیار مهم است.

تک توراندازه

:Ping ✓

Ping یک ابزار دستوری در سیستم‌های عامل است که برای بررسی اتصال و عملکرد شبکه به کار می‌رود. وقتی که شما یک دستور Ping را اجرا می‌کنید، سیستم شما یک بسته ICMP Echo Request (پیام درخواستی) به یک دستگاه دیگر در شبکه ارسال می‌کند.

دستگاهی که بسته را دریافت کرد، به عنوان پاسخ یک بسته ICMP Echo Reply (پیام پاسخی) به دستگاه فرستنده ارسال می‌کند. زمانی که دستگاه فرستنده بسته‌های پاسخ را دریافت می‌کند، آنها را تجزیه و تحلیل کرده و اطلاعاتی مانند زمان پاسخ‌دهی (ping time) و وضعیت اتصال به دستگاه مقصد را به کاربر نشان می‌دهد.

Ping یکی از ابزارهای معمول برای بررسی ارتباط شبکه است و معمولاً برای اطمینان از اتصال به یک دستگاه یا بررسی عملکرد شبکه به کار می‌رود. از آن برای تشخیص مشکلات احتمالی در شبکه، مثلًا بررسی عملکرد اتصال اینترنت یا بررسی اتصال به یک دستگاه خاص در شبکه، استفاده می‌شود.

تک توراندازه

راهنمای استفاده:

```
clang -O2 -g -target bpf -c final.bpf.c -o final.bpf.o ; rm /sys/fs/bpf/final  
; bpftool prog load final.bpf.o /sys/fs/bpf/final
```

```
ip --force link set dev ens3 xdp obj final.bpf.o sec nat
```

```
cat /sys/kernel/tracing/trace_pipe
```

```
root@taktoor-5-dev:/home/college# bpftool map list  
2: array name NAT_ARR flags 0x0  
|   key 4B value 4B max_entries 2 memlock 4096B  
11: array name NAT_ARR flags 0x0  
|   key 4B value 4B max_entries 2 memlock 4096B  
257: array name final.rodata flags 0x80  
|   key 4B value 34B max_entries 1 memlock 4096B  
|   btf_id 686 frozen
```

```
# bpftool map update id 263 key hex 00 00 00 00 value hex 00 11 22 33
```

```
root@taktoor-5-dev:/home/college# bpftool map dump id 263  
key: 00 00 00 00 value: 00 11 22 33  
key: 01 00 00 00 value: 00 00 00 00  
key: 02 00 00 00 value: 00 00 00 00  
key: 03 00 00 00 value: 00 00 00 00  
Found 4 elements  
root@taktoor-5-dev:/home/college#
```

تک توراندازه

مستند فنی:

در ابتدا میدانیم که وقتی بسته از ماشین کلاینت خارج شود و به اینترفیس ۴ برسد، آی پی سورس بسته همان آی پی اینترفیس ۴ میشود یعنی ۱۰.۲۶.۲ و از طرفی وقتی جواب وارد ماشین ما میشود قبل از آن از اینترفیس ۳ خارج شده است پس آی پی سورس آن برابر با ۱۰.۱۶.۲ خواهد شد. پس برای فیلتر بسته ها باید در نظر داشته باشیم که `bpfhelper` میکند و از طرفی چون آی پی عوض شده است چکسام هم تغییر میکند و باید چک سام جدید را محاسبه کرد برای این کار از یکی از توابع استفاده میکنیم :

```
long bpf_l3_csum_replace(struct sk_buff *skb, u32 offset, u64 from, u64 to, u64 size)
```

Description

Recompute the layer 3 (e.g. IP) checksum for the packet associated to `skb`. Computation is incremental, so the helper must know the former value of the header field that was modified (`from`), the new value of this field (`to`), and the number of bytes (2 or 4) for this field, stored in `size`. Alternatively, it is possible to store the difference between the previous and the new values of the header field in `to`, by setting `from` and `size` to 0. For both methods, `offset` indicates the location of the IP checksum within the packet.

This helper works in combination with `bpf_csum_diff()`, which does not update the checksum in-place, but offers more flexibility and can handle sizes larger than 2 or 4 for the checksum to update.

A call to this helper is susceptible to change the underlying packet buffer. Therefore, at load time, all checks on pointers previously done by the verifier are invalidated and must be performed again, if the helper is used in combination with direct packet access.

Return 0 on success, or a negative error in case of failure.

از طرفی نت باید با نگهداری وضعیت انجام شود که برای این کار از `map` استفاده میکنیم :

```
// Connection state table
struct {
    __uint(type, BPF_MAP_TYPE_HASH);
    __uint(key_size, sizeof(struct connection_state));
    __uint(value_size, sizeof(struct connection_state));
```

تک توراندازه

```
    __uint(max_entries, 1024);
    __uint(pinning, LIBBPF_PIN_BY_NAME);
} connection_state_map SEC(".maps");
```

```
s struct connection_state
{
    __u32 src_ip;
    __u32 dest_ip;
    __u32 translated_src_ip;
    __u32 translated_dst_ip;
    time_t current_time;
};
```

علت نگهداری تایم کنونی این است که اگر بیش از ۳۰ ثانیه از رسیدن جواب گذشته باشد باید این استیت حذف شود و مپ آپدیت شود.

تابع زیر نیز کار حذف کانکشن های خارج از موعد را انجام میدهد.

تکنوراندازه

```
int delete_old_connections(struct __sk_buff *skb) {
    struct connection_state *conn_state;
    __u32 key = 0; // Key to access the map

    // Iterate over the connection state map
    for (int i = 0; i < 1024; i++) {
        // Get the connection state at index i
        conn_state = bpf_map_lookup_elem(&connection_state_map, &key);
        if (!conn_state)
            break; // End of map

        // Get the current time

        time_t now = bpf_ktime_get_ns() / 1000000000;

        // Check if the current time is more than 30 seconds after the connection
        time
        if (now - conn_state->current_time > 30) {
            // Delete the old connection state
            bpf_map_delete_elem(&connection_state_map, &key);
        }

        // Move to the next key
        key++;
    }

    return 0;
}
```

تک توراندازه

کد زیر نیز ابتدا با کمک تابع (`bpf_hdr_pointer()`) هر بار چک میکند که از `data_end` خارج نشده باشد.

```
static inline void *bpf_hdr_pointer(const struct __sk_buff *skb, __u32 offset)
{
    void *data = (void *)(long)skb->data;
    void *data_end = (void *)(long)skb->data_end;

    if (data + offset <= data_end)
        return data + offset;

    return NULL;
}
```

سپس شرایط مورد نظرش را چک میکند یعنی :

Ip packet	✓
Icmp packet	✓
Ping	✓

سپس به دنبال این میگردد که در مپ این کانکشن وجود داشته یا نه !!!

```
// Get connection state
struct connection_state key = {
    .src_ip = bpf_ntohl(ip->saddr),
};

struct connection_state key1 = {
    .dest_ip = bpf_ntohl(ip->daddr),
};

struct connection_state *conn_state =
bpf_map_lookup_elem(&connection_state_map, &key);
struct connection_state *conn_state1 =
bpf_map_lookup_elem(&connection_state_map, &key1);
```

تکنوراندازه

در صورتی که کانکشن موجود نبود:

```

if (!conn_state && !conn_state1)
{
    // New connection, perform NAT translation
    __u32 translated_src_ip;
    __u32 translated_dst_ip;

    if (ip->saddr == htonl(0x0a020602) && ip->daddr == bpf_htonl(0x0A010601))
    {
        translated_src_ip = bpf_htonl(10 << 24 | 1 << 16 | 6 << 8 | 2);
        // Update connection state with translated IP
        struct connection_state new_conn_state = {
            .src_ip = key.src_ip,
            .dest_ip = bpf_ntohl(ip->daddr),
            .translated_src_ip = translated_src_ip,
            .translated_dst_ip = bpf_ntohl(ip->daddr),
            .current_time = bpf_ktime_get_ns() / 1000000000};

        bpf_skb_store_bytes(skb, offsetof(struct __sk_buff, data) +
offsetof(struct icmphdr, un.echo.id),
                           &translated_src_ip, sizeof(translated_src_ip),
BPF_F_RECOMPUTE_CSUM);
        bpf_map_update_elem(&connection_state_map, &key, &new_conn_state,
BPF_ANY);
    }
    else if (ip->saddr == bpf_htonl(0x0a010602) && ip->daddr ==
bpf_htonl(0x0a020602))
    {
        translated_dst_ip = bpf_htonl(10 << 24 | 2 << 16 | 6 << 8 | 1);
        // Update connection state with translated IP
        struct connection_state new_conn_state = {
            .src_ip = key1.src_ip,
            .dest_ip = bpf_ntohl(ip->daddr),
            .translated_src_ip = bpf_ntohl(ip->saddr),
            .translated_dst_ip = translated_dst_ip,
        };
    }
}

```

```
.current_time = bpf_ktime_get_ns() / 1000000000};
```

```
bpf_skb_store_bytes(skb, offsetof(struct __sk_buff, data) +
offsetof(struct icmphdr, un.echo.id),
&translated_dst_ip, sizeof(translated_dst_ip),
BPF_F_RECOMPUTE_CSUM);
bpf_map_update_elem(&connection_state_map, &key1, &new_conn_state,
BPF_ANY);
}
}
```

یک ارتباط میسازد و به مپ می افزاید.

و در غیر اینصورت:

```
else
{
    if (ip->saddr == htonl(0x0a020602) && ip->daddr == bpf_htonl(0x0A010601))
    {
        ip->saddr = bpf_htonl(conn_state->translated_src_ip);
        conn_state->current_time = bpf_ktime_get_ns() / 1000000000;
    }
    else if (ip->saddr == bpf_htonl(0x0a010602) && ip->daddr ==
bpf_htonl(0x0a020602))
    {
        ip->daddr = bpf_htonl(conn_state1->translated_dst_ip);
        conn_state1->current_time = bpf_ktime_get_ns() / 1000000000;
    }
}
```

تکنورانداز

و بعد نیز چکسام بسته را اصلاح و بسته را فوروارد میکند:

```

if (ip->saddr == htonl(0x0a020602) && ip->daddr == bpf_htonl(0x0A010601))
{
    if (bpf_l3_csum_replace(skb, offsetof(struct __sk_buff, data) +
offsetof(struct iphdr, saddr), &ip->saddr, &ip->daddr, sizeof(ip->saddr)) == 0)
    {
        // Forward packet
        return bpf_redirect(skb->ifindex, 0);
    }
    else
        return BPF_DROP;
}

else if (ip->saddr == bpf_htonl(0x0a010602) && ip->daddr ==
bpf_htonl(0x0a020602))
{
    if (bpf_l3_csum_replace(skb, offsetof(struct __sk_buff, data) +
offsetof(struct iphdr, daddr), &ip->daddr, &ip->daddr + 1, sizeof(ip->daddr)) == 0)
    {
        // Forward packet
        return bpf_redirect(skb->ifindex, 0);
    }
    else
        return BPF_DROP;
}

```

پیروزه دوره

تکنوراندازه

مراجع و منابع:

<https://manpages.ubuntu.com/manpages/jammy/en/man.htmlv/bpf-helpers.v>

<https://eunomia.dev/tutorials/-tc/>

<https://github.com/cilium/ebpf/discussions/>

Learning eBPF Programming the Linux Kernel for Enhanced Observability, Networking, and Security

chat gpt 😊