Hello everyone!
Today, I'd like to explain IP addresses in a way that's easy to relate to. Think of an IP address like your house number. Just like how an online order gets delivered directly to your home using your address, digital information reaches your device using its IP address.

There are two main types of IP addresses: **Public IP** and **Private IP**.
To make it easier to understand, imagine this: your **public IP** is like your official name known to the world, while your **private IP** is more like a nickname used only within your household. At home, people may call you by your nickname, but to the outside world, you are recognized by your real name. Similarly, your device uses a **private IP within your local network**, and a **public IP** to communicate on the internet.

Why do we need both?
Because IPv4 addresses are limited and many of them have already been exhausted, private IPs were introduced to help conserve address space and allow more efficient use of bandwidth. This setup provides a huge advantage, especially in larger networks.

Now, are there any rules for assigning these "nicknames" or private IPs?
Absolutely — and that's where **subnetting** comes in. Subnetting defines how many IP addresses can exist within a private network. While it can be a complex topic, you can think of it as a way of organizing and limiting address distribution within a network.

With that foundation in place, let's dive deeper!

**1. IP (Internet Protocol)**

- **IPv4 vs. IPv6:**
    - **What to learn:** Understand the address formats, how they're assigned (DHCP, static), and the implications of each. For pentesting, knowing how to enumerate both IPv4 and IPv6 addresses is crucial.
    - **Pentesting Relevance:**
        - **IPv4 exhaustion & NAT:** Understand why NAT exists (addressed below).
        - **IPv6 adoption challenges:** Many networks are dual-stacked or have partial IPv6 deployments, which can introduce misconfigurations and unmonitored attack surfaces. Attackers can sometimes bypass IPv4-focused defenses by exploiting IPv6. Learn IPv6 specific tools (e.g., nmap -6, ip -6).
- **Public vs. Private IP Addresses:**
    - **What to learn:** Know the reserved private IP ranges (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16). Understand that these are not routable on the public internet.
    - **Pentesting Relevance:**
        - **Internal vs. External Testing:** This distinction is fundamental. External tests target public IPs, internal tests target private IPs.

        - **Reconnaissance:** Identifying public IP ranges belonging to a target organization helps in scoping. During internal tests, mapping private IP ranges is essential for network discovery.
        - **SSRF (Server-Side Request Forgery):** Understanding private IP ranges is critical for exploiting SSRF to target internal services.
- **Subnetting and CIDR:**
    - **What to learn:** How to calculate network addresses, broadcast addresses, host ranges, and subnet masks given a CIDR notation (e.g., /24, /16).
    - **Pentesting Relevance:**
        - **Network Mapping:** Accurately mapping target networks by identifying subnets. This guides your scanning and enumeration.
        - **Scope Definition:** Understanding CIDR blocks is essential for defining the scope of a penetration test.
        - **Bypassing ACLs/Firewalls:** Sometimes, misconfigured rules might allow access to specific subnets that you can identify

**Classful Addressing: Understanding the Types**

The internet is a massive network of connected computers, and these devices communicate using protocols. While we mostly use just two versions of IP addresses (IPv4 and IPv6), there are actually 15 different types in total.

**IPv4** addresses are 32 bits long and usually written in four parts — like 192.168.0.1.
**IPv6** addresses, on the other hand, are 128 bits long and look like long strings of hexadecimal numbers (e.g., 2001:0db8:85a3:0000:0000:8a2e:0370:7334). Each hexadecimal character can represent values up to 15, which makes it efficient.

Some key differences:

- IPv4 supports **broadcasting** and has **12 header fields**.
- IPv6 does **not support broadcast** and uses only **8 header fields**, making it more lightweight.

---

**IP Address Classes**

IP addresses are grouped into **five classes**:

- **Class A**: 1.0.0.0 to 126.255.255.255
- **Class B**: 128.0.0.0 to 191.255.255.255
- **Class C**: 192.0.0.0 to 223.255.255.255

- **Class D**: 224.0.0.0 to 239.255.255.255
- **Class E**: 240.0.0.0 to 254.255.255.255

Out of these, we use **Class A to D** in networks. **Class D** is used for multicast, while **Class E** is reserved for research and experimental purposes.

Also, any address starting with **127.** is used as a **loopback address**, which means it's used to test the device's own network configuration (like 127.0.0.1 for localhost testing).

---

**How Are IP Addresses Assigned?**

There's a hierarchy in how IP addresses are distributed:

- **IANA** (Internet Assigned Numbers Authority) handles global allocation.
- It delegates regional distribution to **RIRs** (Regional Internet Registries).
- These RIRs then assign IPs to your local **ISP** (Internet Service Provider), who finally allocates it to you.

---

**Public vs. Private IP Addresses**

Let's talk about the difference between public and private IPs.

- **Private IPs** are used **within a local network** and are not routable on the internet.
  These are the common private ranges:
    - 10.0.0.0 to 10.255.255.255
    - 172.16.0.0 to 172.31.255.255
    - 192.168.0.0 to 192.168.255.255
- **Public IPs** are assigned by your **ISP** and are **used to communicate over the internet**.

---

**What if IP Assignment Fails?**

Normally, a service called **DHCP** (Dynamic Host Configuration Protocol) automatically assigns IPs within your network. But if DHCP fails, your device will use something called **APIPA** (Automatic Private IP Addressing), which assigns an IP from a special range (usually starting with 169.254.x.x).

---

**Quick Bytes About IP**

- In IPv4, each of the 4 sections is called a **byte** (8 bits), and the maximum value for each byte is **255**.
- An IP address has two main parts:
    - **Network ID**: Identifies the network segment.

- **Host ID**: Identifies the specific device on that network.

**How to Find Network ID, Subnet Mask & Broadcast Address (Classful IP Addressing)**

Let's break this down step-by-step with real examples so even if you're just getting started with networking, you'll feel confident by the end of this read.

---

**How to Find the Network ID?**

Let's say you're given an IP address: **120.10.90.10**
Now, your first task is to **identify which class** this IP belongs to.

Here's a quick refresher:

- **Class A:** 1.0.0.0 to 126.255.255.255
- **Class B:** 128.0.0.0 to 191.255.255.255
- **Class C:** 192.0.0.0 to 223.255.255.255

So… 120.10.90.10 clearly falls under **Class A** range

Now remember this golden rule:

In **Class A**, the **first octet** (the first number before the first dot) is for the **network** and the rest are for the **hosts**.

So the **network ID** for 120.10.90.10 is simply:
**120.0.0.0**

Let's try one more.

What about: **150.10.90.10**
This falls in **Class B** (128–191 range). In Class B, the first **two octets** are for the network.

So the **network ID** is:
**150.10.0.0**

Nice and easy, right?

---

**Subnet Masks – What Are They?**

Think of a **subnet mask** as a filter — it tells us **which part** of the IP address is reserved for the **network** and which part is for the **host**.

Here's the default subnet mask for each class:

- **Class A:** 255.0.0.0
- **Class B:** 255.255.0.0
- **Class C:** 255.255.255.0

Let’s apply it to an example.

Say we have the IP: **120.15.10.100**

Step 1: Check which class this falls into.
It’s **Class A**, since 120 is in the 1–126 range.

So, the subnet mask for this IP is:
**255.0.0.0**

---

**Broadcast IP – Sending to Everyone!**

Now let’s learn how to find the **broadcast address** — the IP used when you want to send a message to **all devices** on a network.

Let’s go back to our example:
**120.15.10.100**

First, as we already determined, this is Class A.
So the **network ID** is: **120.0.0.0**

Now here’s a tip:

Replace all host octets with **255** to get the **broadcast IP**.

Since in Class A only the **first octet** is the network part, the rest (3 octets) are for the host.

So, the broadcast address becomes:
**120.255.255.255**

And that’s your one-to-all IP!

---

**Quick Recap:**

| IP Address | Class | Network ID | Subnet Mask | Broadcast IP |
|---|---|---|---|---|
| 120.10.90.10 | A | 120.0.0.0 | 255.0.0.0 | 120.255.255.255 |
| 150.10.90.10 | B | 150.10.0.0 | 255.255.0.0 | 150.10.255.255 |

.

**How to Find Usable Hosts**

Let’s say you have an IP address like **180.168.10.100** and want to find out how many systems can be assigned IPs in that range.

Step 1: Identify which class the IP belongs to.
Since the first octet is 180, it falls into **Class B**.

Step 2: Determine the network ID.
For Class B, the network ID uses the first two octets, so the network ID is:
**180.168.0.0**

Step 3: Calculate the number of host bits.
In Class B, the remaining two octets (two zeroes) represent hosts. Each octet is 8 bits, so:

- 8 bits for the third octet
- 8 bits for the fourth octet

Step 4: Calculate the total number of available IP addresses.
The total number of IPs is 28×28=2162^{8} \times 2^{8} = 2^{16}28×28=216. This means **65,536 IP addresses** can be assigned within this network range.

---

**IP Address Construction**

Imagine you need to assign an IP address to a device. If it's for a private network, you typically consider Class A, B, or C ranges. For public IP addresses, allocation is managed differently and often requires automated tools or official registries.

---

**Classless Addressing (CIDR)**

Now, let's look at **classless addressing**, which offers more flexibility than classful addressing.

In classless addressing, the **network ID** is considered as a **block ID**, and there are no fixed classes. A block consists of both the block ID and host ID combined, making a total of 32 bits.

Notation for this is:
**x.y.z.w/n**
where **n** represents the number of bits used for the network portion of the address.

For example, consider:
**200.10.20.40/28**

Here, **/28** means the first 28 bits are for the network, leaving 32−28=432 - 28 = 432−28=4 bits for the host portion. Since 4 bits represent hosts, the number of hosts is 24=162^{4} = 1624=16.

---

**Understanding the Subnet Mask**

The 28 bits for the network can be represented as:
**11111111.11111111.11111111.11110000**
which in decimal is:
**255.255.255.240**

To determine the network this IP belongs to, convert the last octet (40) to binary:
**00101000**

Since the first 28 bits represent the network, we keep those bits fixed and set the remaining 4 bits (for hosts) to zero.

So the host bits **1000** become **0000**, resulting in:
**200.10.20.00100000**

This binary corresponds to:
**200.10.20.32/28**

Therefore, the IP address **200.10.20.40/28** belongs to the network **200.10.20.32/28**.

---

**Rules to Follow in Classless Addressing**

- Addresses within a block must be **contiguous** (no gaps).
- The number of addresses in a block must be a **power of two** (for example, 16 is 242^424 as in the example above).
- The **first address** of every block must be **evenly divisible** by the size of the block.

In the previous example, the first address of the network block is **32**, which is the first valid IP address for the network **200.10.20.32/28**.

**SLAAC (Stateless Address Autoconfiguration)**

When a device uses IPv6, it **does not require a DHCP server** to configure its IP address. Instead, SLAAC allows devices to automatically configure their own IP addresses **statelessly** as soon as they connect to the network.

Here's how it works step-by-step:

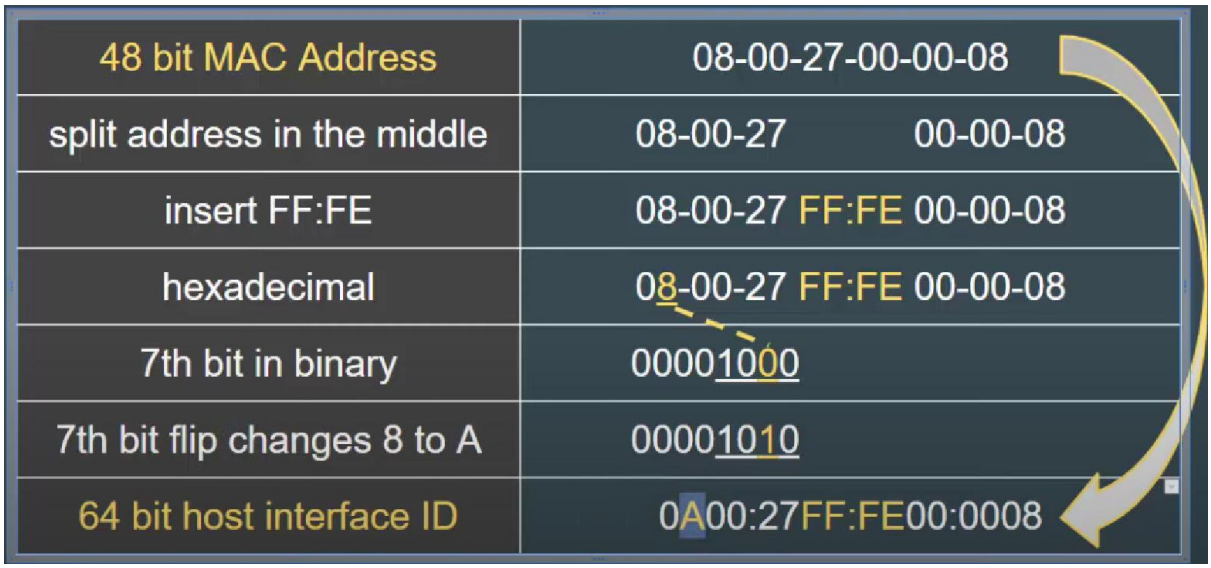
1. When a device (like a PC) joins the network, it first sends out a **Router Solicitation** message. This message is an **ICMPv6** packet essentially asking,
   *"Hey, is there a router on this network?"*
2. The router responds with a **Router Advertisement** message saying,
   *"Yes, I'm here."*
3. Once the device receives this advertisement, it can automatically configure its own IPv6 address. It does this by combining:
   - The **network prefix** and gateway information learned from the router advertisement.
   - An **interface identifier** which can be generated using two methods:
     - **EUI-64 format**, which converts the device's 48-bit MAC address into a 64-bit Interface ID.
     - Or a **randomly generated** identifier for privacy.
4. After the device has configured its IPv6 address, it sends a **Neighbor Solicitation** message to check for address conflicts on the network. This process, called **Duplicate Address Detection (DAD)**, is basically the device "pinging" itself to make sure no other device has the same IP address.

---

**How SLAAC Works Internally: Conversion of 48-bit MAC to 64-bit Interface ID**

One common method for generating the interface ID is to use the device's MAC address. Since MAC addresses are 48 bits and the interface ID needs to be 64 bits, SLAAC uses the **EUI-64 format** to expand the MAC address.

Conversion of 48 bit MAC to 64 bit Interface Id:->




RFC 4861 Neighbor Discovery - SLAAC – ICMPv6

**1. Link-Local Addresses (fe80::...)**

- Every IPv6 device **must** have a link-local address.
- Used for communication within the local network (like NDP and router discovery).
- **Source address** in NS is often a link-local address.

**Two Cases:**

- **Normal NS:** Source = link-local address (fe80::...)
- **DAD NS:** Source = :: (unspecified address)

---

**2. Duplicate Address Detection (DAD)**

- Before using an IPv6 address, a device checks if it's already in use.
- Sends NS with source :: and destination = target's solicited-node multicast.
- If no NA received → address is safe to use.
- If NA received → address is already in use.

---

**3. How Does Device A Know Device B Exists?**

- **It doesn't know upfront!** It learns in one of these ways:
    - From **Router Advertisements** (RAs).
    - From **DNS**, mDNS, or application-level discovery.
    - Manual configuration (rare).

---

**4. How Does Device A Know Where to Send NS?**

- Device A takes **Device B's IPv6 address**, extracts the last 24 bits.
- Builds the **Solicited-Node Multicast address**.
- Sends NS to that multicast.
- **Device B** is already listening to it!

**Example:**

- Device B: IPv6 = fe80::1234 → last 24 bits = 00:12:34
- NS goes to: FF02::1:FF00:1234
- Only Device B hears and responds.

---

**5. Comparison: ARP (IPv4) vs NDP (IPv6)**

| **Feature** | **ARP (IPv4)** | **NDP (IPv6)** |
|---|---|---|
| Protocol Used | ARP (separate) | ICMPv6 |
| Discovery Method | Broadcast | Solicited-node multicast |
| Security | None | Can use Secure NDP (SEND) |
| Functions Supported | Address resolution | Address resolution, DAD, Router Discovery, etc. |

---

**1. What is NDP? (The IPv6 Phonebook)**

Imagine you move to a new neighborhood (IPv6 network). You need to:

- Find your neighbors (other devices).
- Learn their house numbers (MAC addresses).
- Discover who the postmaster is (router).
  **NDP does all this automatically**—it's like a smarter, more efficient version of IPv4's ARP.

---

**2. How NDP Works (Step-by-Step)**

**A. Neighbor Solicitation (NS) – "Who Lives Here?"**

**Scenario:** Your device (Device A) wants to talk to another device (Device B at fe80::1234), but only knows its IPv6 address (not its MAC).

1. **Device A shouts a targeted question:**
    - Instead of yelling to everyone (broadcast like ARP), it sends a **Neighbor Solicitation (NS)** to a **specific multicast group** (FF02::1:FF00:1234).
        - *Think of this like knocking on the door of "House #1234" instead of shouting down the whole street.*
    - **Source address:** Device A's own link-local address (fe80::1).
    - **Destination address:** FF02::1:FF00:1234 (a special group only Device B listens to).
2. **Why multicast?**
    - Every IPv6 device calculates its **unique multicast address** from its own IP.
    - Device B (fe80::1234) automatically listens to FF02::1:FF00:1234.
    - **No prior communication needed!** Device A just *derives* the multicast address from Device B's IP.

**B. Neighbor Advertisement (NA) – "That's Me!"**

When Device B hears the NS:

1. **It replies directly to Device A:**
    - *"Hi! I'm fe80::1234, and my MAC address is 12:34:56:78:9A:BC."*
    - **Source address:** Device B's link-local (fe80::1234).
    - **Destination address:** Device A's address (fe80::1).
2. **If Device B changes its MAC address:**
    - It can announce to everyone (FF02::1 = "all nodes" multicast) like a neighborhood bulletin.

---

**3. Extra Features of NDP**

**A. Duplicate Address Detection (DAD) – "Is This Address Taken?"**

Before Device A uses fe80::1, it checks:

1. Sends an NS **from** :: **(no address yet)** to FF02::1:FF00:1.
2. If no one replies → Address is free!
3. If someone says *"That's mine!"* → Device A picks a new address.

**B. Router & Prefix Discovery**

- **Routers periodically announce:**
  *"I'm your default gateway! Use prefix 2001:db8::/64 for addresses."*
- Devices use this to auto-configure their IPv6 addresses (no DHCP needed!).

**C. Redirects**

If Device A sends traffic to the wrong router, the router says:
*"Hey, send this to Router B instead!"*

**Dual-Stack Misconfigurations and IPv4 to IPv6 Transition**

IPv4 and IPv6 coexist on the internet using several techniques such as **dual stack**, **tunneling**, and **translation**. Let's focus on the **dual stack** approach.

A **dual stack** device supports both IPv4 and IPv6 protocols simultaneously. Such devices can include PCs, servers, or routers.

---

**IPv4 to IPv6 Transition Strategies**

The transition from IPv4 to IPv6 is necessary due to the limitations of IPv4 and the vastly increased address space provided by IPv6. Here are the key points to understand:

- **Limitations of IPv4:**
  IPv4 uses a 32-bit address length, which provides approximately 4 billion unique IP addresses. Despite techniques like classless IP addressing and private IP addresses, the number of available IPv4 addresses is insufficient for the rapidly growing number of internet-connected devices worldwide.
- **Introduction of IPv6:**
  IPv6 uses a 128-bit address length, offering an astronomically larger number of unique IP addresses (2^128), thus addressing the scarcity issue.
- **Challenges of Transition:**
  A sudden, complete migration from IPv4 to IPv6 is not practical because many existing devices and networks still rely exclusively on IPv4. Therefore, the transition must be gradual and smooth to maintain compatibility between IPv4 and IPv6 systems.
- **Transition Strategies:**
  There are three main strategies used to facilitate the transition from IPv4 to IPv6, which are typically discussed in related educational content.

---

**Step-by-Step Example: Accessing a Website with Dual Stack**

1. You type **google.com** into your browser.

2. Your computer queries the **DNS (Domain Name System)**, asking:
   *“What is the IP address of google.com?”*
   It requests both IPv4 (A record) and IPv6 (AAAA record) addresses.
3. The DNS responds with one or both addresses:
   - If the website supports IPv6, the DNS returns an **AAAA record**, for example: 2607:f8b0:4009:80e::200e (IPv6 address).
   - If the website only supports IPv4, the DNS returns an **A record**, for example: 142.250.190.46 (IPv4 address).
4. Your computer then chooses the best option:
   - If IPv6 is available, it uses that address because IPv6 is newer and generally offers better performance.
   - If only IPv4 is available, it falls back to using the IPv4 address.

---

**Why Does DNS Decide?**

DNS acts like a translator that informs your device which IP version to use:

- *“This website supports IPv6, use that.”*
- Or, *“This website only supports IPv4, use the older protocol.”*

This decision ensures a smooth user experience regardless of the underlying IP protocol used by the website.

**1. The Problem: IPv6 Bypassing IPv4 Security**

Many networks focus on securing **IPv4** but **ignore IPv6**, assuming:

- *"We don’t use IPv6, so it’s safe."*
- *"Our firewall only checks IPv4 traffic."*

**Reality:**

- Most modern devices (Windows, Linux, macOS) **enable IPv6 by default**.
- If IPv6 isn’t actively blocked, attackers can **use it to bypass IPv4 security controls**.

**2. How Attackers Exploit Dual-Stack**

**A. IPv6 DNS Hijacking (DNS Spoofing)**

- **Scenario:**
  - Your firewall blocks malicious IPv4 DNS requests.

        - But if the attacker **forces your device to use IPv6 DNS**, they can redirect traffic.
    - **How?**
        - Send a **rogue Router Advertisement (RA)** saying:
          *"Hey, use my malicious IPv6 DNS server!"*

**IPv6 Man-in-the-Middle (MITM)**

- **Scenario:**
    - Your network monitors IPv4 ARP spoofing.
    - But if IPv6 is enabled, attackers use **NDP spoofing** (fake Neighbor Advertisements).
- **How?**
    - Attacker sends fake **Neighbor Advertisement (NA)**:
      *"I'm the router! Send all IPv6 traffic to me!"*
    - Your device updates its neighbor cache → traffic goes to the attacker.

---

**3. Real-World Examples**

✅ **CVE-2020-16898 ("Bad Neighbor")**

- Windows IPv6 stack flaw let attackers execute code via malicious RAs.

✅ **Enterprise Network Breaches**

- Attackers use IPv6 to **bypass NAC (Network Access Control)** that only checks IPv4.

✅ **Malware Using IPv6 for C2 (Command & Control)**

- Some malware (e.g., **APT29**) uses IPv6 to evade detection.

**How IPv6-over-IPv4 Tunneling Works (Simple Explanation)**

Imagine you need to send a **big box (IPv6 packet)** through a **small tunnel (IPv4-only network)**. Since the tunnel only allows **small boxes (IPv4 packets)**, you **put the big box inside a small one** (encapsulation). At the other end, you **unpack it** (decapsulation) to get the original big box back.

---

**Step-by-Step Tunneling Process**

**1. Encapsulation (Entering IPv4 Land)**

- **Scenario:**
    - **Device A (IPv6)** wants to send a packet to **Device B (IPv6)**.
    - But the middle network **only understands IPv4** (e.g., an old ISP or corporate network).
- **What Happens:**

1. Device A prepares its **IPv6 packet** (with source = A, destination = B).
2. Before sending, it **wraps the IPv6 packet inside an IPv4 packet** (like putting a letter inside an envelope).
    - **New IPv4 Header:**
        - **Source IPv4:** Tunnel entry point (e.g., 192.0.2.1).
        - **Destination IPv4:** Tunnel exit point (e.g., 203.0.113.1).
        - **Protocol Field = 41** (This tells the network: *"Hey, there's an IPv6 packet inside me!"*).
    - **Payload:** The original IPv6 packet.

**2. Travel Through IPv4 Network**

- The **IPv4 packet** (with the IPv6 packet inside) travels normally across the IPv4-only network.
- Routers in the middle **only look at the IPv4 header**—they don't care what's inside.

**3. Decapsulation (Exiting IPv4 Land)**

- When the packet reaches the **tunnel endpoint (IPv6-enabled router)**:
    1. It checks the **IPv4 protocol field (41)** and realizes: *"This contains an IPv6 packet!"*
    2. It **strips off the IPv4 header**, revealing the original IPv6 packet.
    3. The IPv6 packet is then **forwarded normally** to Device B.

---

**Key Points**

1. **Protocol 41** is critical—it's the "secret handshake" that tells routers: *"This IPv4 packet carries IPv6!"*
2. **Tunnel Endpoints** must be configured to **encapsulate/decapsulate** (e.g., using 6to4, Teredo, or manual tunnels).
3. **Performance Overhead:** Encapsulation adds extra bytes (IPv4 header), slightly increasing latency.

---

**Real-World Analogy**

- **IPv6 Packet = Train (needs wide tracks).**
- **IPv4 Network = Narrow road.**
- **Solution:** Load the train onto a **truck (IPv4 packet)**. Drive the truck on the road. At the destination, unload the train.

---

**Why This Matters for Security/Pentesting**

- **Tunneling can bypass firewalls** if not properly monitored (e.g., attackers sneak IPv6 traffic through IPv4).
- **Protocol 41 misuse:** Some networks block it to prevent unauthorized tunneling.
- **Tunnel Hijacking:** If attackers control the tunnel endpoint, they can intercept/decapsulate traffic.