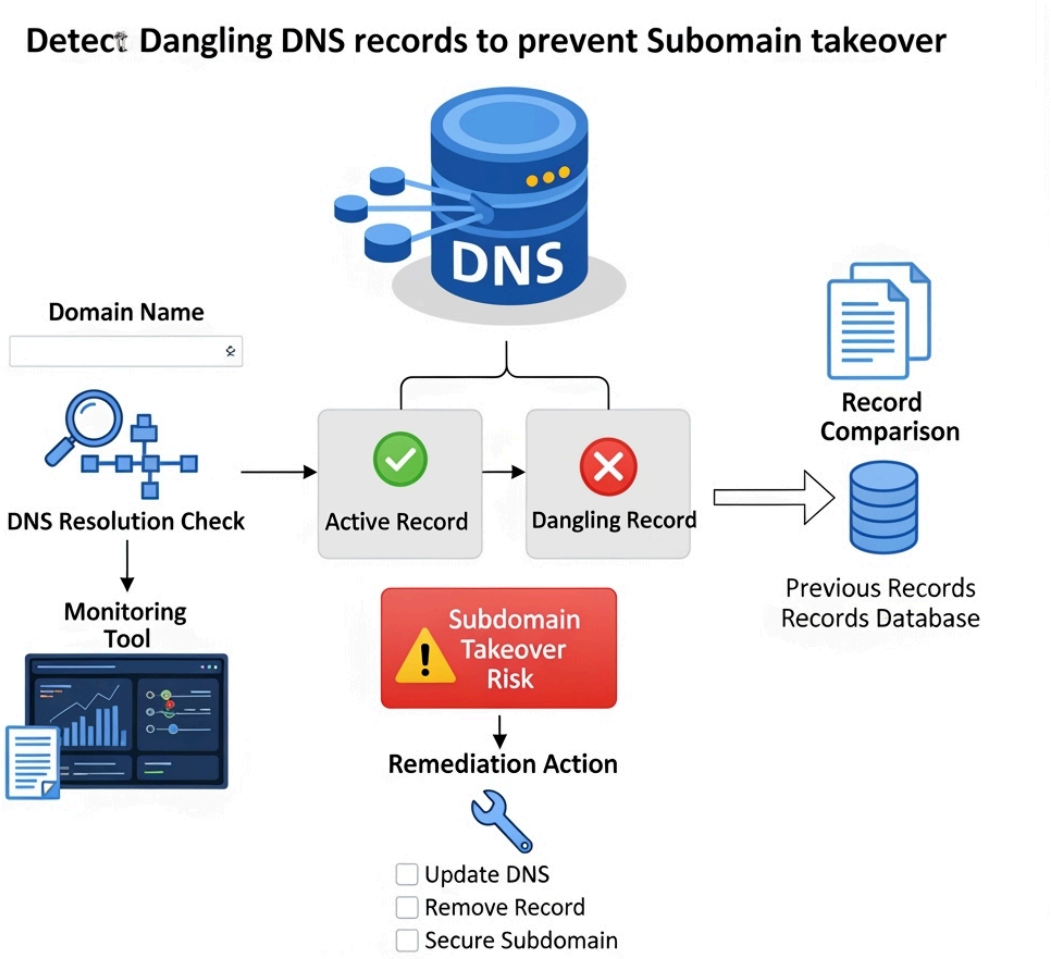

🔎 Definition

- **CNAME** = **Canonical Name Record** in DNS.
- It maps **one domain name (an alias)** to another domain name (the **canonical, or "real" name**).

Think of it as a **nickname pointer** — instead of pointing directly to an IP address, it points to another domain that does.

📌 Example

Suppose you want `www.example.com` to use the same server as `example.com`:

```
www.example.com.    CNAME   example.com.
example.com.        A       192.0.2.123
```

- When a browser looks up `www.example.com`, DNS first resolves the CNAME → `example.com` → then follows the `A` record to get the IP (`192.0.2.123`).

✅ Common Use Cases

1. **Aliases / Subdomains**
    - `blog.company.com → company-blog.squarespace.com`
    - Lets you host services on external platforms but keep your brand domain.
2. **Load balancing / CDNs**
    - `cdn.company.com → cdn.provider.net`
    - CDN providers manage the underlying IP addresses.
3. **Simplified management**
    - Instead of updating IPs in multiple places, point aliases to one canonical record.
4. **Email & SaaS verification**Often used in **domain ownership checks** (e.g., Microsoft 365, Google Workspace, security tools).

⚠️ Security Considerations

- **CNAME chaining** can expose dependencies: if the target domain is abandoned, attackers could hijack it (“dangling CNAME”).
- Misconfigured CNAMEs are a common source of **subdomain takeovers**.
- Always monitor your DNS zone for **orphaned or unused CNAME records**.

🎯 Summary

A **CNAME record** in DNS:

- Creates an **alias** → **canonical name** mapping.
- Helps with flexibility, SaaS integrations, and simplified DNS management.
- Must be carefully managed to prevent **takeover risks**.

✅ — **CNAME records are one of the main ways to detect dangling DNS records**, and they're a common cause of **subdomain takeovers**.

Let's break it down:

🔎 What's a Dangling DNS Record?

- A **dangling DNS record** happens when a DNS entry (like a CNAME or A record) points to a resource that **no longer exists**.
- Example:
  - `app.company.com → CNAME → app.azurewebsites.net`
  - If the Azure resource `app.azurewebsites.net` was deleted, the DNS CNAME still points there, but the target is gone.
  - An attacker could **re-register that Azure resource** and hijack `app.company.com`.

🛠️ How to Use CNAMEs to Detect Dangling Records

1. **Enumerate subdomains**
   - Collect all subdomains for your domain (`sublist3r`, `amass`, ASM/DRP tools like SOCRadar).
2. **Check for CNAME records**
   - Use `dig`, `nslookup`, or scripts to see if they resolve to external providers (Azure, AWS, GitHub Pages, Heroku, etc.):
     ```
     dig CNAME app.company.com
     ```
3. **Verify the target**
   - If the CNAME points to something like `app.azurewebsites.net`, check whether that resource actually exists.
   - If it does **not exist**, that's a dangling record.
4. **Attempt resolution**
   - Query the target hostname. If it doesn't resolve or shows a "not found"/"no such host" error → dangling.
5. **Cross-check against takeover lists**
   - Known takeover-prone providers include:

        - Azure (*.azurewebsites.net, *.blob.core.windows.net)
        - AWS S3 (*.s3.amazonaws.com)
        - GitHub Pages (*.github.io, *.github.com)
        - Heroku (*.herokuapp.com)
        - Squarespace, Shopify, Fastly, etc.

⚠️ Security Implications

- Dangling CNAMEs are the **#1 subdomain takeover vector**.
- Attackers who claim the orphaned resource can:
    - Host phishing pages under your subdomain.
    - Steal cookies/tokens if users trust the domain.
    - Damage brand reputation.

✅ Best Practices

- Regularly audit **all CNAMEs** in your DNS zones.
- Remove CNAMEs pointing to **retired SaaS/cloud services**.
- Use **ASM/DRP tools** (like SOCRadar, Detectify, or Assetnote) that automatically detect dangling DNS records.

🛠️ Step-by-Step Dangling CNAME Detection

1. Find the CNAME record

Use `dig` or `nslookup` to see if a subdomain points to another domain.

```
dig CNAME subdomain.company.com +short
```

Example output:

```
app.company.com.   3600  IN  CNAME  app.azurewebsites.net.
```

👉 This tells you that `app.company.com` is an alias for an Azure App Service.

2. Check resolution of the target

Now query the target directly:

```
dig app.azurewebsites.net +short
```

- If it returns **an IP** → resource exists (likely safe).
- If it returns **nothing / NXDOMAIN** → potential dangling CNAME.

3. Probe the HTTP response

Try to connect with `curl`:

```
curl -I http://app.azurewebsites.net
```

Common signs of dangling targets:

- `404 Not Found` or `No such host`
- Error messages like:
    - *“This site is not configured”*
    - *“No such app”*
    - *“The bucket you’re attempting to access does not exist”*

4. Check if provider allows re-registration

Cross-reference the domain pattern (`*.azurewebsites.net`, `*.github.io`, `*.s3.amazonaws.com`) with known **takeover-prone services**.

- If yes, and the target is gone → an attacker could claim it.

5. (Optional) Use WHOIS / host tools

Double-check whether the target hostname is live anywhere:

```
host app.azurewebsites.net
whois azurewebsites.net
```

✅ Example Workflow

Let's say you suspect `dev.company.com` is dangling:

```
dig CNAME dev.company.com +short
# → dev-herokuapp.herokuapp.com.

dig dev-herokuapp.herokuapp.com +short
# → (no answer)

curl -I http://dev-herokuapp.herokuapp.com
# → "No such app"
```

📌 Result: **Dangling CNAME detected** → `dev.company.com` could be hijacked by registering the missing Heroku app.

🔒 Best Practices

- Run this audit regularly (especially after decommissioning apps).
- Use scripts (`amass`, `subfinder`, `subjack`, `tko-subs`) to automate detection.
- Immediately remove or fix CNAMEs pointing to unused resources.