

Wildcard DNS Records: Use Cases and Associated Risks

Posted on April 7, 2025

As an organization's internet infrastructure grows, it requires more and more configuration, setup, and decision-making—from selecting a reliable name server to determining the right types of DNS records and setting the appropriate time-to-live (TTL) values. One of those decisions is whether or not to create a wildcard DNS record, a feature that most hosting providers, registrars, and DNS providers offer.

In this post, we discuss what a wildcard DNS record is, how it helps, and any risks associated with it.

What Is a Wildcard DNS Record?

A wildcard DNS record is a special kind of DNS record that allows domain administrators to point non-existent or random subdomains to a specific IP address or other DNS resource data. So, for example, when someone tries to access a subdomain that isn't defined in the domain's DNS settings, the user would still receive a response instead of an error—the user would be routed to an IP address defined in the wildcard A record.

In a nutshell, a wildcard DNS record serves as a catch-all and is a way to automatically handle invalid subdomain requests. Specifying a wildcard domain is simple, with a few considerations. Here are a few rules to keep in mind:

- **Only a single asterisk (*) is considered as the wildcard operator.** DNS wildcard records do not support other characters or expressions. For example, ***.example.com** is a correct

wildcard record, but ***1.example.com** is not.

- **The asterisk must be at the leftmost label in the domain name.** For example, ***.example.com** is correct, but **example.*com** is not.
- **Only one asterisk is allowed per record.** Wildcards cannot be created on multiple levels in the domain, so if you set a wildcard DNS record for ***.example.com**, most DNS providers will consider it to be invalid. Also, a wildcard like ***.example.com** will not work for a non-defined subdomain such as **a.b.example.com** — wildcards only work for one level.
- **Any existing DNS record will take precedence over the wildcard record.** This means that if there is a specified DNS record for **blog.example.com**, that record will be used even if there is a wildcard record for ***.example.com**. Wildcards only work as a fallback mechanism.

How To Create a Wildcard DNS Record

The exact steps for creating a wildcard DNS record vary depending on the DNS provider, but the general flow remains the same.

1. Log in to your account and access the DNS management console.
2. Add a new DNS record and specify the record type.
3. Enter the wildcard subdomain (e.g., ***.example.com**) and its destination or value, which can be an IP address, URL, or another subdomain.

It may take some time—from a few minutes to up to 48 hours—for the new DNS record to take effect. Also, keep in mind that not all types of DNS records support wildcards. They are supported and widely used with A, AAAA, and TXT records, but although technically permissible, wildcards are less common with CNAME, ALIAS, URL Redirect, HTTP Redirection, and MX records. Other record types, such as NS, SOA, and PTR records, don't support wildcards at all.

Legitimate Use Cases for Wildcard DNS Records

Now you know how to create a wildcard DNS record. The next question is, do you need to create one? To better understand its functionality, here are some legitimate use cases for Wildcard DNS records.

Multi-Tenant Applications

With multi-tenant software, a single instance of the application serves multiple customers, so each of them would have their own subdomains — `customer1.sampleapp.com`, `customer2.sampleapp.com`, `customer3.sampleapp.com`, and so on.

Using a wildcard DNS record (`*.sampleapp.com`) allows the same application to handle all subdomains without having to individually configure the DNS for each and every subdomain.

Blogging Platforms

Blogging platforms like WordPress.com allow users to create custom subdomains, such as `mydailyblog.wordpress.com`. These platforms often use wildcard DNS records (e.g., `*.wordpress.com`) so that any subdomain automatically resolves to a shared IP address.

When a user enters a subdomain in their browser, a DNS lookup is performed, and the wildcard DNS record returns the same IP for all such subdomains. The platform's server infrastructure receives the request and uses the HTTP Host header (e.g., `mydailyblog.wordpress.com`) to determine which blog content to serve.

This architecture enables WordPress to efficiently host millions of user blogs on shared infrastructure, without needing to create separate server instances or deployments for each blog. The system dynamically renders each blog's unique content and design based on the subdomain in the request.

Temporary Usage for New Top-Level Domains (TLDs)

The Internet Corporation for Assigned Names and Numbers (ICANN), which oversees the

Internet's domain name system, mandates that new gTLDs publish special DNS responses for a minimum of 90 days (as outlined in the [Name Collision Occurrence Management Framework](#)) for non-existent domain names, functioning in a limited way like catch-alls, but without using wildcard records.

It's worth noting that ICANN generally [prohibits registries](#) from using wildcard DNS for non-existent or unregistered domains. When someone tries to access such domains, registries are required to have DNS servers return an NXDOMAIN response (i.e., the site does not exist). However, the framework served as an exception and was designed to avoid [name collisions](#)—an unwanted event where a fully qualified domain name residing on the new TLD conflicts with an identical name used in an internal network.

When name collisions happen, users on a private network can encounter an error or be directed to the wrong web page on the public Internet. A temporary DNS record that is similar to a wildcard can help avoid this. Specifically, the A record for 127.0.53.53 (a reserved IPv4 address for name collision warning) is used and will appear in the system logs to warn administrators about a potential issue in their network.

Are There Any Problems With Using Wildcard DNS?

When you think about it, wildcard DNS records help ensure a smooth user experience since visitors would still be routed to a working web page even after requesting a non-existent or undefined subdomain.

However, they should be used with caution. The general rule of thumb is: "If you don't have a legitimate use case for wildcard DNS, don't use it." Here are some of the reasons.

- **Risk of DNS errors:** Wildcard DNS records can lead to unexpected and confusing traffic routing, particularly in environments where internal systems use domain search paths. For example, if a computer on your network is named `computer.example.com`, it may automatically configure `example.com` as a search domain (e.g., in `/etc/resolv.conf`). This means that when applications or users try to access external services, such as `www.google.com`, a failed lookup may fall back to `www.google.com.example.com`. If a

wildcard DNS record like *.example.com is in place, this fallback subdomain will resolve—potentially pointing to your infrastructure. As a result, instead of failing cleanly, requests are routed internally, causing broken functionality, misrouted traffic, and hard-to-diagnose errors. In some cases, this can even affect server-to-server communication, where outbound API calls silently fail or behave unpredictably.

- **Subdomain takeover:** Subdomain takeover: For example, a wildcard DNS entry like *.company1.com may route all subdomains to infrastructure hosted on a public cloud provider—such as an AWS CloudFront distribution or Azure Blob Storage. If that infrastructure is ever deleted but the wildcard DNS remains in place, an attacker can claim the unassigned resource on the same cloud platform. As a result, subdomains like login.company1.com or reset-password.company1.com—which resolve via the wildcard—can end up serving attacker-controlled content.
- **Phishing:** Since a wildcard DNS is essentially a catch-all record, requests for misspelled subdomains can still get routed to a default IP address specified in the wildcard. This means that even mistyped subdomains like logIn.company1.com (instead of login.company1.com) will resolve and load your real homepage or a default page. Threat actors can exploit this behavior in phishing campaigns. For example, they might send users to <https://logIn.company1.com>, which resolves and loads without errors thanks to the wildcard. The page itself might look generic but still uses your domain, HTTPS padlock, and branding — giving a false sense of legitimacy. The attacker can then follow up with a prompt like: “If this link didn’t load properly, click here to reset your password,” directing the user to an external phishing site that collects credentials or personal data. Because the first link looked real and secure, the user is more likely to trust the second step and fall for the phishing attempt.

Wildcard DNS versus Wildcard SSL: What’s the Difference?

As we’ve established, wildcard DNS records control how subdomains resolve to an IP address, mail server, and other DNS resource records.

There are also wildcard SSL certificates—these are used to secure HTTPS connections for multiple subdomains residing on a root domain (e.g., *.example.com). While convenient, they increase the attack surface: if any one subdomain is compromised, the attacker can serve content under that

subdomain using the valid wildcard certificate.

This means visitors will see a valid HTTPS padlock, and many security systems will trust the connection — even though the content might be malicious. Because the certificate is valid for the whole domain, there's no way to differentiate between legitimate and compromised subdomains at the certificate level.

Finding Wildcard DNS Records


Security professionals often need to know where a wildcard DNS is used, such as when mapping the organization's attack surface for vulnerability assessments. WhoisXML API offers plenty of DNS products based on both passive DNS and real-time DNS data that identifies if a DNS record is a wildcard entry.

Getting Wildcard DNS Information with DNS Chronicle API

[DNS Chronicle API](#) lets you retrieve the historical A and AAAA records of any domain. The output also includes a wildcard field, telling you if the returned DNS record is part of a wildcard entry. Here's how a query looks like:

```
curl --location 'https://dns-history.whoisxmlapi.com/api/v1' \
  --header 'Content-Type: application/json' \
  --data '{
    "apiKey": "<your_API_key>",
    "searchType": "forward",
    "recordType": "a",
    "domainName": "0-2nask-us.turbotaxweb.profile.basecamp.app"
  }' >> dns_chronicle_sample.json
```



The output is displayed like this in the API demo page:



Search by **Domain name**, IPv4, IPv6

The demo is limited to 10 records

```
"count": 4,  
"records": [  
  {  
    "date": "2024-10-12",  
    "ips": [  
      {  
        "ip": "13.248.252.114",  
        "wildcard": true  
      },  
      {  
        "ip": "99.83.138.213",  
        "wildcard": true  
      }  
    ]  
  },  
  ],  
},
```



Build complete API lookup

Since the wildcard field indicates “True,” it means that the domain has a wildcard DNS configuration, specifically its A record. If the field displays “False,” then the record is not a wildcard. There may also be records whose wildcard field is “null,” which means that the tool has not yet

checked the DNS record for this domain.

These values are the same as those of DNS Database Download files—our passive DNS repositories containing eight DNS record types. Below is a snapshot of a sample A record file:

d	du	ips	wildcard	active
0-11235.com	1728663261	43.250.142.134	FALSE	TRUE
0-40.nl	1728768594	109.235.32.116	FALSE	TRUE
0-9.au	1728741527	43.250.142.43	FALSE	TRUE
0-a.com	1728774816	64.190.63.222	FALSE	TRUE
0-apps.net	1728755461	80.190.158.32	FALSE	TRUE
0-1f.1kapp.com	1728757237	220.181.168.129 220.181.168.249	TRUE	TRUE
0-2nask-us-file.ebaystrategies-0-2nasc-us.basecamp.app	1728728401	13.248.252.114 99.83.138.213	TRUE	TRUE
0-2nask-us.turbotaxweb.profile.basecamp.app	1728744401	13.248.252.114 99.83.138.213	TRUE	TRUE
0-2nask-usguideler.basecamp.app	1728700735	13.248.252.114 99.83.138.213	TRUE	TRUE
0-fod.infobase.com.ksclib.keene.edu	1728626782	216.17.113.200	TRUE	TRUE
0-fv.df.gov.br	1728859806	131.72.222.176	TRUE	TRUE
0-incites.clarivate.com.wam.seals.ac.za	1728642873	196.21.233.4	TRUE	TRUE
0-j6.df.gov.br	1728811294	131.72.222.176	TRUE	TRUE
0-0-----logopod.ru	1725049256	217.107.219.29		TRUE
0--1.net	1725227709	157.7.189.251		TRUE
0--6.hb.cldmail.ru	1724661484	95.163.53.117		TRUE

Getting Wildcard DNS Information with Reverse DNS API

Another tool that provides additional information about a DNS record being part of a wildcard entry is the [Reverse DNS API](#). The API allows users to look for domains connected to an SOA, TXT, or CNAME record.

Here is an example query:

```
curl --location 'https://reverse-dns.whoisxmlapi.com/api/v1' \
--header 'Content-Type: application/json' \
--data '{
  "apiKey": "<your_API_key>",
  "limit": 1000,
  "includeAdditionalChecks": 1,
  "recordType": "soa",
  "terms": [
```



```
{
  "field": "domain",
  "term": "example.com"
}
]' >> reverse_dns_sample.json
```

The output looks like this:

reverse_dns						
result_value	result_wildcard	result_active	result_name	result_first_seen	result_last_visit	size
ns.icann.org. noc.dns.icann.org. 2025011303 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	19
ns.icann.org. noc.dns.icann.org. 2025011502 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	
ns.icann.org. noc.dns.icann.org. 2025011507 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	
ns.icann.org. noc.dns.icann.org. 2025011510 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	
ns.icann.org. noc.dns.icann.org. 2025011512 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	
ns.icann.org. noc.dns.icann.org. 2025011513 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	
ns.icann.org. noc.dns.icann.org. 2025011514 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	
ns.icann.org. noc.dns.icann.org. 2025011516 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	
ns.icann.org. noc.dns.icann.org. 2025011521 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	
ns.icann.org. noc.dns.icann.org. 2025011525 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	
ns.icann.org. noc.dns.icann.org. 2025011528 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	
ns.icann.org. noc.dns.icann.org. 2025011530 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	
ns.icann.org. noc.dns.icann.org. 2025011533 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	
ns.icann.org. noc.dns.icann.org. 2025011534 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	
ns.icann.org. noc.dns.icann.org. 2025011535 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	
ns.icann.org. noc.dns.icann.org. 2025011537 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	
ns.icann.org. noc.dns.icann.org. 2025011543 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	
ns.icann.org. noc.dns.icann.org. 2025011545 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	
ns.icann.org. noc.dns.icann.org. 2025011546 7200 3600 1209600 3600	FALSE	FALSE	example.com	1696159926	1740083630	

Conclusion

A wildcard DNS record is a catch-all entry that helps make sure DNS requests to non-existent subdomains still point to a valid DNS resource. While it is helpful during the initial setup of new gTLDs and is used by multi-tenant applications and blogging platforms, most system

administrators generally avoid wildcard DNS. They can cause misrouting of DNS traffic and can potentially direct users to malicious content.

If you want to know if a domain uses or has used a wildcard DNS record, you can use our passive DNS products, such as [DNS Chronicle API](#) and [Reverse DNS API](#).