

Forum:[Hack Forums](#)

Thread: [DNS Cache Poisoning.](#)

youoffline - February 22, 2025, 7:22 am

First time posting, sorry if this is not the appropriate thread.

I've been researching about DNS cache poisoning but couldn't find anything regarding my situation.

Since UDP doesn't check NS validity, can't i just spam packets claiming to be the authoritative nameserver with a fake IP address?

If I'm not missing something, this should cache the fake IP address instead of the real one. Users who visit this site will be redirected to the fake website.

Have any of you tried this?

DramaKing - February 22, 2025, 12:06 pm

So in theory, you can make a DNS request to a resolver that has not cached the domain requested or uses DNSSEC, then forge a reply to that request with the upstream resolver's IP address as the source IP and your IP address as the answer.

Just make sure when picking a target site to run 'nmap -p 443 --script http-security-headers <domain.tld>' and verify that it returns "HSTS not configured for HTTPS Server." Otherwise, the most that will happen is a DoS.

youoffline - February 23, 2025, 3:40 pm

(Feb 22, 2025 01:06 PM)DramaKing Wrote: [\[CITE\]](#) So in theory, you can make a DNS request to a resolver that has not cached the domain requested or uses DNSSEC, then forge a reply to that request with the upstream resolver's IP address as the source IP and your IP address as the answer.

Just make sure when picking a target site to run 'nmap -p 443 --script http-security-headers <domain.tld>' and verify that it returns "HSTS not configured for HTTPS Server." Otherwise, the most that will happen is a DoS.

Yeah, the HSTS is not configured for this website, it doesn't use DNSSEC, I made a quick script that sends these forged requests, sent a couple of packets but it sadly doesn't work. I think the problem I'm having here is that this website has weird configuration?

Basically, I found the authoritative nameserver (ns1.target.com), then I tried to find the recursive resolver (the thing I'm actually trying to attack), so I ran:

```
dig A ns1.target.com +short
```

and got the IP address, (for example, 123.45.67.1), then, logically:

```
dig @123.45.67.1 target.com
```

and this is where it gets interesting, this returned that:

```
...;; WARNING: recursion requested but not available
```

even though, running this IP in Shodan shows that Recursion is enabled on port 53.

am i tripping? I got really confused, the authoritative name server is the recursive resolver too?
Either dig or Shodan is messing it up.

Here's a code snippet:

target_ns = 123.45.67.1 (It's supposed to be the recursive resolver)
fake_ip is the IP I'm trying to replace it with.

```
Code def craft_packet(self) -> IP:
    return IP(dst=self.target_ns, src="123.45.67.1") /\
        UDP(dport=53, sport=RandShort()) /\
        DNS(
            id=RandShort(),
            qr=1,
            aa=1,
            rd=1,
            ra=1,
            qdcount=1,
            anccount=1,
            qd=DNSQR(
                qname=self.target_domain,
                qtype="A",
                qclass="IN"
            ),
            an=DNSRR(
                rname=self.target_domain,
                type="A",
                ttl=3600,
                rdata=self.fake_ip
            )
        )
```

DramaKing - February 23, 2025, 4:05 pm

(Feb 23, 2025 04:40 PM)youoffline Wrote: [\[CITE\]](#) Yeah, the HSTS is not configured for this website, it doesn't use DNSSEC, I made a quick script that sends these forged requests, sent a couple of packets but it sadly doesn't work. I think the problem I'm having here is that this website has weird configuration?

Websites don't use DNSSEC. Literally no DNS security is configured for individual websites.

(Feb 23, 2025 04:40 PM)youoffline Wrote: [\[CITE\]](#) Basically, I found the authoritative nameserver (ns1.target.com), then I tried to find the recursive resolver (the thing I'm actually trying to attack), so I ran:

dig A ns1.target.com +short

and got the IP address, (for example, 123.45.67.1), then, logically:

dig @123.45.67.1 target.com

and this is where it gets interesting, this returned that:

...;; WARNING: recursion requested but not available

even though, running this IP in Shodan shows that Recursion is enabled on port 53.

am i tripping? I got really confused, the authoritative name server is the recursive resolver too? Either dig or Shodan is messing it up.

It's not dig or Shodan. It's you. All you're doing is querying the authoritative nameserver's IP address and using it to query the target site's IP address. Recursive resolvers are not associated with specific websites, other nameservers, hosting services, or what-have-you. To get the recursive resolver, run 'cat /etc/resolv.conf.'

As for the error, it seems that the authoritative nameserver has recursion disabled, which makes sense as it exists to resolve static DNS entries. Adding '+norec' to the command will disable recursion and should give you the exact same response as 'dig target.com.' ?

(Feb 23, 2025 04:40 PM)youoffline Wrote: [\[CITE\]](#) Here's a code snippet:

target_ns = 123.45.67.1 (It's supposed to be the recursive resolver)
fake_ip is the IP I'm trying to replace it with.

```
Code def craft_packet(self) -> IP:
    return IP(dst=self.target_ns, src="123.45.67.1") /\
        UDP(dport=53, sport=RandShort()) /\
        DNS(
            id=RandShort(),
            qr=1,
            aa=1,
            rd=1,
            ra=1,
            qdcount=1,
            anccount=1,
            qd=DNSQR(
                qname=self.target_domain,
                qtype="A",
                qclass="IN"
            ),
            an=DNSRR(
                rname=self.target_domain,
                type="A",
                ttl=3600,
                rdata=self.fake_ip
            )
        )
```

I can't code anything in C, but this doesn't appear to me to do anything. Regardless, I think that you're trying to take the authoritative name server and have it recursively lookup one of the domains that its authoritative for and poison the entry for 'self.target_domain'.

This is a perfect example of why wannabe hackers should understand protocols like DNS before going off half-cocked.

youoffline - February 24, 2025, 4:51 am

(Feb 23, 2025 05:05 PM)DramaKing Wrote: [\[CITE\]](#) Websites don't use DNSSEC. Literally no DNS

security is configured for individual websites.

I think it is pretty clear that a WEBSITE wont use DNSSEC. It was implied that DNSSEC is not configured for this website.

(Feb 23, 2025 05:05 PM)DramaKing Wrote: [\[CITE\]](#) To get the recursive resolver, run 'cat /etc/resolv.conf.'

Afaik this only works if you're on the same network as the target. Which is not the case here.

(Feb 23, 2025 05:05 PM)DramaKing Wrote: [\[CITE\]](#) I can't code anything in C, but this doesn't appear to me to do anything. Regardless, I think that you're trying to take the authoritative name server and have it recursively lookup one of the domains that its authoritative for and poison the entry for 'self.target_domain'.

The code snippet is written in Python, this is the main part of the whole code and it handles creating the packet which should be sent to the DNS server, I deliberately did not include the rest of the code because it does not serve any purpose for the question I'm asking.

The script aims to race against the legit DNS responses, impersonates the authoritative nameserver and tries to attack the recursive resolver (this is what I'm trying to find) which will cache the fake response.

(Feb 23, 2025 05:05 PM)DramaKing Wrote: [\[CITE\]](#) This is a perfect example of why wannabe hackers should understand protocols like DNS before going off half-cocked.

I mean, we're here to learn, right? ?

DramaKing - February 24, 2025, 5:11 am

(Feb 24, 2025 05:51 AM)youoffline Wrote: [\[CITE\]](#) I think it is pretty clear that a WEBSITE wont use DNSSEC. It was implied that DNSSEC is not configured for this website.

I'll concede the point. Domains have to be signed for DNSSEC. It isn't enough for the DNS servers to have it configured now that I look into it.

(Feb 24, 2025 05:51 AM)youoffline Wrote: [\[CITE\]](#) Afaik this only works if you're on the same network as the target. Which is not the case here.

That only works on your machine. Let's say that you're not using systemd-resolved and that there's an address or two set by DHCP. This is likely your router, which is running a DNS resolver. You'll then need to check your router to see what its DNS servers are, probably what your ISP is using.

If you're targeting a specific victim, then you'll mostly have to guess what recursive resolver is being used.

(Feb 24, 2025 05:51 AM)youoffline Wrote: [\[CITE\]](#) The script aims to race against the legit DNS responses, impersonates the authoritative nameserver and tries to attack the recursive resolver (this is what I'm trying to find) which will cache the fake response.

I mean, we're here to learn, right? ?

Of course everyone has to start somewhere. I just feel the need to call out people who seem like they're getting ahead of themselves. What it still sounds like to me is that you're querying for the authoritative nameserver's IP address, confusing that for the recursive resolver (which, again, any recursive resolver can query the TLD servers which receive records from authoritative name servers. There is no relationship between these kinds of servers.)

If that isn't enough, I found this script: <https://github.com/gr3yc4t/dns-poisoning-tool>. It does have a switch to specify an authoritative name server, so I could be missing something.

youoffline - February 24, 2025, 7:27 am

(Feb 24, 2025 06:11 AM)DramaKing Wrote: [\[CITE\]](#) What it still sounds like to me is that you're querying for the authoritative nameserver's IP address, confusing that for the recursive resolver. I agree on this, I kinda got confused because I looked up the authoritative name server's IP address on Shodan and it stated that recursion was enabled, so I thought this would be enough to conclude that I was dealing with a recursive resolver. I did some dig commands, also tried this:

```
Code dig @<suspected_resolver_ip> google.com
```

Since this didn't provide an answer, I figured it wasn't a recursive resolver.

I don't know how effective this check was, any who, looks like the only option here is to guess.

(Feb 24, 2025 06:11 AM)DramaKing Wrote: [\[CITE\]](#) any recursive resolver can query the TLD servers which receive records from authoritative name servers. There is no relationship between these kinds of servers.

This statement seems a bit misleading. Specifically, TLD servers do not receive records from authoritative name servers, they provide records that point to the authoritative name servers for domains under that TLD.

Thanks for the script btw, maybe I'll use that instead of gluing my own. ?

DramaKing - February 24, 2025, 8:15 am

(Feb 24, 2025 08:27 AM)youoffline Wrote: [\[CITE\]](#) I agree on this, I kinda got confused because I looked up the authoritative name server's IP address on Shodan and it stated that recursion was enabled, so I thought this would be enough to conclude that I was dealing with a recursive resolver. I did some dig commands, also tried this:

```
Code dig @<suspected_resolver_ip> google.com
```

Since this didn't provide an answer, I figured it wasn't a recursive resolver.

I don't know how effective this check was, any who, looks like the only option here is to guess.

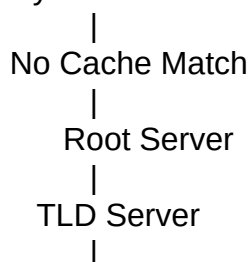
This statement seems a bit misleading. Specifically, TLD servers do not receive records from authoritative name servers, they provide records that point to the authoritative name servers for domains under that TLD.

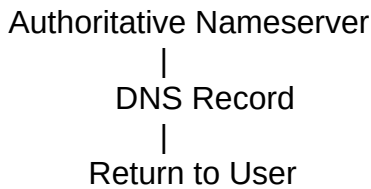
Thanks for the script btw, maybe I'll use that instead of gluing my own. ?

I can't explain why Shodan gave the result that it did, but since the nameserver is authoritative for the domain, it would never do recursion on that domain. Recursion only happens when the resolver does not have the requested records.

I checked with Copilot for the step-by-step recursion process. You are correct that the TLD servers only store the authoritative nameservers (it's been awhile since Network+).

User Query -> DNS Resolver -> Cache Check





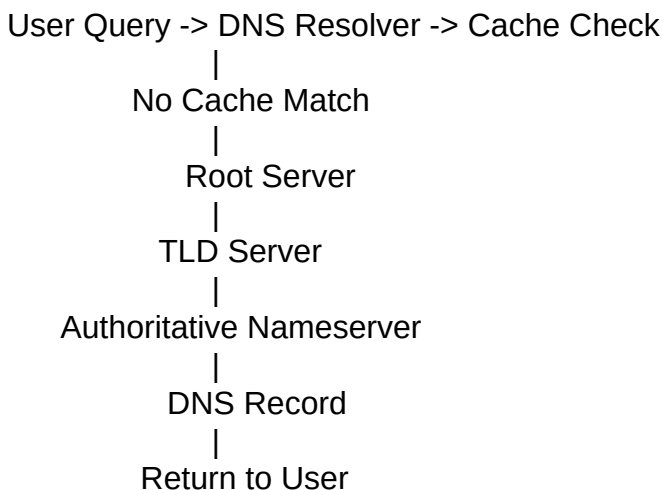
So you have the IP address for ns1.hosting.com. You use that as the source IP, and the destination IP should be, say, 71.10.216.1. Any DNS servers using that recursive resolver will receive the poisoned record from cache for the duration of TTL, or the poisoned server deletes it.

To beat a dead horse, let's say that that you're trying to target a specific user. You send an IP logger that reveals the target is using a Comcast IP address. You decide that it's a safe bet that the user is either using the ISP-provided DNS or 8.8.8.8, maybe Private DNS. So you attack 75.75.75.75. As long as the former is true, and you beat the race condition, the target should be directed to your server.

youoffline - February 25, 2025, 2:05 am

(Feb 24, 2025 09:15 AM)DramaKing Wrote: [CITE](#) I can't explain why Shodan gave the result that it did, but since the nameserver is authoritative for the domain, it would never do recursion on that domain. Recursion only happens when the resolver does not have the requested records.

I checked with Copilot for the step-by-step recursion process. You are correct that the TLD servers only store the authoritative nameservers (it's been awhile since Network+).



So you have the IP address for ns1.hosting.com. You use that as the source IP, and the destination IP should be, say, 71.10.216.1. Any DNS servers using that recursive resolver will receive the poisoned record from cache for the duration of TTL, or the poisoned server deletes it.

To beat a dead horse, let's say that that you're trying to target a specific user. You send an IP logger that reveals the target is using a Comcast IP address. You decide that it's a safe bet that the user is either using the ISP-provided DNS or 8.8.8.8, maybe Private DNS. So you attack 75.75.75.75. As long as the former is true, and you beat the race condition, the target should be directed to your server.

All of this is very clear, but the part I'm stuck at is the destination IP, which should be the recursive resolver.

Do you have any methods on how to guess the recursive resolver's IP? I've tried setting the destination to ISP's primary DNS, but that didn't work.

DramaKing - February 25, 2025, 4:06 am

(Feb 25, 2025 03:05 AM)youoffline Wrote: [\[CITE\]](#) All of this is very clear, but the part I'm stuck at is the destination IP, which should be the recursive resolver.

Do you have any methods on how to guess the recursive resolver's IP? I've tried setting the destination to ISP's primary DNS, but that didn't work.

Like I said, there won't be any specific DNS resolver. There's just no guarantee that it will work.

Forgot to mention that my router doesn't seem to forward packets not coming from the local subnet. I suggest testing the source IP spoofing, then sorting out vulnerable DNS servers.