Report: DNS Cache Poisoning Vulnerability

By Xiang Li from Network and Information Security Laboratory, Tsinghua University

On Oct 10, 2021

Overview

We found a vulnerability in the bailiwick checking procedure of your resolver, which allows specific attackers to inject `NS` records of any domain (even TLDs) into the cache and conduct a DNS cache poisoning attack. The effects of an exploit would be widespread and highly impactful, as this injection could hijack any subdomain under the poisoned domain, including TLDs, e.g., `com` and `net`.

Vulnerability Details

When the resolver is configured with a conditional forwarding zone and a forwarder, it will query all domains under the zone targeting the forwarder. For example, we give a brief config file in the following, where queries of any domain under `attacker.com` will be sent directly to `x.x.x.x` for resolution. While for the other domains, they are processed by the resovler itself through recursive queries, such as `com` and `victim.com`.

```
options {
  recursion yes;
  dnssec-validation yes;
}
zone "attacker.com" {
       type forward;
       forwarders { x.x.x.x port 53; };
};
```

If an attacker query `A` records of `x.attacker.com`, and the resolver receives a response with an `AA` flag in the following. The bogus `NS` records of `com` will be cached. As a result, queries of any domain under `com` in the future will be sent directly to the fake name server `ns.attacker.com`, thus hijacked by attackers.

```
+------------+--------------------------------+
| Field      | Value                          |
+------------+--------------------------------+
| Question   | x.attacker.com A               |
| Answer     | x.attacker.com A 1.2.3.4       |
| Authority  | com NS ns.attacker.com         |
+------------+--------------------------------+
```

Root Cause Analysis

Through source code review, we locate the root cause of the vulnerability.

Specifically, `Query.zone` is the key to bailiwick checking logic. Any resource records whose name is not under or same as `Query.zone` during responses processing will be removed.

However, when forwarding, `Query.zone` is initialized with the "closest" name of `NS` records following the recursive resolution process.

As for the forwarding zone, there is no directly corresponding `NS` records. Therefore, for example, when querying `attacker.com`, Query.zone is `com` or the root `.`.

Consequently, the fake domain is under `com` and `.`, which is considered to be in-bailiwick.

Proof-of-Concept

We give two exploiting methods and show our real attack videos in the link (https://cloud.tsinghua.edu.cn/d/faddf16c281e41a7b216/) with password: `bind_attack_pre_21`.

For ethical considerations, we build the whole DNS testing environment, including the attacker machine, the conditional DNS server, the upstream resolver, and our authoritative server. And we use `attacker.attack` as an example domain to attack NS records of `com`.

1. If the forwarder `x.x.x.x` is held by an attacker, he can return the bogus response directly from the authoritative server. We show our attack steps below, and the video is named `bind_attack_fd.mp4` in the folder.
    1. From 00:00, we show the latest BIND version.
    2. From 00:05, we start BIND.
    3. From 00:10, we query `github.com` and obtain correct `com` NS records through recursive queries.
    4. From 00:20, we query the forwarded domain `attacker.attack` and receive fake `com` NS records.
    5. From 00:28, bingo, fake `com` NS records are cached successfully.
2. If the forwarder `x.x.x.x` is not held by an attacker, he should inject the bogus response directly from the off-path. And we leverage the SAD DNS attack to poison the resolver successfully. We show our attack steps below, and the video is named `bind_attack_fu.mp4` in the folder.
    1. From 00:00, we show the latest BIND version and Linux version.
    2. From 00:06, we start BIND.
    3. From 00:11, we query `github.com` and obtain correct `com` NS records through recursive queries.
    4. From 00:21, we query the forwarded domain `[random_prefix].attacker.attack` and start SAD DNS attacks.
    5. From 33:10, bingo, after 1,642 attempts costing 1,970s, fake `com` NS records are cached successfully.

Threat Surface

We test the latest version `9.16.21`, until Oct. 10, 2021, which is vulnerable.

Mitigation

Consider setting `Query.zone` with the forwarding zone, such as `attacker.com`.