浅谈 DNS 重绑定漏洞

作者：R1dd1er 合天智汇

原创投稿活动： 重金悬赏 | 合天原创投稿等你来

前言

DNS Rebinding 是一个比较早就出现的漏洞, 本人也是在最近的一些 CTF 比赛的 web 题见识到, 便记录学习一下相关的内容

DNS

DNS 是 Domain Name Service 的缩写, 计算机域名服务器, 在 Internet 上域名与 IP 地址之间是一一对应的，域名虽然便于人们记忆，但机器之间只能互相认识 IP 地址，它们之间的转换工作称为域名解析，而域名解析需要由专门的域名解析服务器来完成，这就是 DNS 域名服务器。

更具体详细的介绍可以参考 http://www.ruanyifeng.com/blog/2016/06/dns.html

DNS TTL

TTL 值全称是 “生存时间（Time To Live)”，简单的说它表示 DNS 记录在 DNS 服务器上缓存时间，数值越小，修改记录各地生效时间越快。

当各地的 DNS(LDNS) 服务器接受到解析请求时，就会向域名指定的授权 DNS 服务器发出解析请求从而获得解析记录；该解析记录会在 DNS(LDNS) 服务器中保存一段时间，这段时间内如果再接到这个域名的解析请求，DNS 服务器将不再向授权 DNS 服务器发出请求，而是直接返回刚才获得的记录；而这个记录在 DNS 服务器上保留的时间，就是 TTL 值。

常见的设置 TTL 值的场景

1. 增大 TTL 值，以节约域名解析时间

通常情况下域名解析记录是很少更改的。我们可以通过增大域名记录的 TTL 值让记录在各地 DNS 服务器中缓存的时间加长，这样在更长的时间段内，我们访问这个网站时，本地 ISP 的 DNS 服务器就不需要向域名的 NS 服务器发出解析请求，而直接从本地缓存中返回域名解析记录, 从而提高解析效率。 TTL 值是以秒为单位的，通常的默认值都是 3600，也就是默认缓存 1 小时。我们可以根据实际需要把 TTL 值扩大，例如要缓存一天就设置成 86400。

2. 减小 TTL 值，减少更新域名记录时的不可访问时间

更换空间因为 TTL 缓存的问题，新的域名记录，在有的地方可能生效了，有的地方可能等上一两天甚至更久才生效。结果就是有的人访问到了新服务器，有的人访问到了旧服务器。如果原来的域名 TTL 值设置的小，各地的 ISP 域名缓存服务器服务器就会很快的访问你域名的权威 DNS 解析服务器，尽快把你域名的 DNS 解析 IP 返回给查询者。

DNS Rebinding

在网页浏览过程中，用户在地址栏中输入包含域名的网址。浏览器通过 DNS 服务器将域名解析为 IP 地址，然后向对应的 IP 地址请求资源，最后展现给用户。而对于域名所有者，他可以设置域名所对应的 IP 地址。当用户第一次访问，解析域名获取一个 IP 地址；然后，域名持有者修改

对应的 IP 地址；用户再次请求该域名，就会获取一个新的 IP 地址。对于浏览器来说，整个过程访问的都是同一域名，所以认为是安全的。这就造成了 DNS Rebinding 攻击。

具体步骤

- 攻击者控制恶意的 DNS 服务器来回复域的查询, 如 rebind.network
- 攻击者通过一些方式诱导受害者加载 http://rebind.network
- 用户打开链接, 浏览器就会发出 DNS 请求查找 rebind.network 的 IP 地址
- 恶意 DNS 服务器收到受害者的请求, 并使用真实 IP 地址进行响应, 并将 TTL 值设置为 1 秒, 让受害者的机器缓存很快失效
- 从 http://rebind.network 加载的网页包含恶意的 js 代码, 构造恶意的请求到 http://rebind.network/index , 而受害者的浏览器便在执行恶意请求
- 一开始的恶意请求当然是发到了攻击者的服务器上, 但是随着 TTL 时间结束, 攻击者就可以让 http://rebind.network 绑定到别的 IP, 如果能捕获受害者的一些放在内网的应用 IP 地址, 就可以针对这个内网应用构造出对应的恶意请求, 然后浏览器执行的恶意请求就发送到了内网应用, 达到了攻击的效果

同源策略的失效

对于 WEB 的同源策略相信大家都很熟悉, 如果两个页面的协议，端口（如果有指定）和域名都相同，则两个页面具有相同的源, 而不同源的客户端脚本在没有明确授权的情况下，不能读写对方资源。

当然, 页面中的链接，重定向以及表单提交是不会受到同源策略限制的, 并且, 跨域资源的引入是可以的。但是 js 不能读写加载的内容。

同源策略确实提高了 web 的安全性, 但是对于 DNS Rebinding 来说是没有作用的, 因为同源策略看的是域名, 并不是背后的 IP 地址, 虽然两次的请求 IP 地址不同, 但是由于 DNS 服务器的绑定, 域名都是一样的, 那么自然不违反同源策略.

CTF 实战应用

balsnCTF2019 韩国鱼

本题的一个步骤也涉及到了 DNS 重绑定的利用, 这里就截取部分代码说明, 完整的 wp 可以查看 https://mp.weixin.qq.com/s/ToORsrR_1fh1gnnO2cM_VQ

部分涉及到的代码

```
$dst = @$_GET['KR'];
$res = @parse_url($dst);
$ip = @dns_get_record($res['host'], DNS_A)[0]['ip'];
...
$dev_ip = "54.87.54.87";
if($ip === $dev_ip) {
    $content = file_get_contents($dst);
    echo $content;
}
```

很明显, 题目是想定死我们域名的 ip 为 54.87.54.87, 但是这样就没法读到内网的信息了, 如果我们域名解析的 IP 直接为 127.0.0.1, 那就无法过 `$ip === $dev_ip` , 接下来就得利用 DNS Rebinding

这是一个测试 dns 重绑定漏洞的网站, 可以让一个域名随机的绑定两个 IP
https://lock.cmpxchg8b.com/rebinder.html?tdsourcetag=s_pctim_aiomsg

让我们在本地测试一下吧

```
➜  ~ host 7f000001.36573657.rbndr.us
7f000001.36573657.rbndr.us has address 127.0.0.1
➜  ~ host 7f000001.36573657.rbndr.us
7f000001.36573657.rbndr.us has address 127.0.0.1
➜  ~ host 7f000001.36573657.rbndr.us
7f000001.36573657.rbndr.us has address 54.87.54.87
➜  ~ host 7f000001.36573657.rbndr.us
7f000001.36573657.rbndr.us has address 54.87.54.87
➜  ~ host 7f000001.36573657.rbndr.us
7f000001.36573657.rbndr.us has address 54.87.54.87
➜  ~ host 7f000001.36573657.rbndr.us
7f000001.36573657.rbndr.us has address 54.87.54.87
➜  ~ host 7f000001.36573657.rbndr.us
7f000001.36573657.rbndr.us has address 127.0.0.1
➜  ~ host 7f000001.36573657.rbndr.us
7f000001.36573657.rbndr.us has address 127.0.0.1
```

所以思路很明确, 先让 $ip 为 54.87.54.87, 然后在 `file_get_contents` 的时候我们 GET 的域名又绑定到 127.0.0.1, 这样就绕过了 if 判断语句, 可以读内网信息, 当然, 由于是随机的, 所以要多打几次.

SECCON 2019 Web: Option-Cmd-U

题目地址: http://ocu.chal.seccon.jp:10000/index.php

```
<!-- src of this PHP script: /index.php?action=source -->
                    <!-- the flag is in /flag.php, which permits access only from internal network
```

```
:-) -->
                    <!-- this service is running on php-fpm and nginx. see /docker-compose.yml -->
```

根据源码提示, 访问 `http://ocu.chal.seccon.jp:10000/index.php?action=source` 读取源码, 我们重点看 PHP 代码部分

```php
<?php
if (isset($_GET['url'])){
    $url = filter_input(INPUT_GET, 'url');
    $parsed_url = parse_url($url);
    if($parsed_url["scheme"] !== "http"){
        // only http: should be allowed.
        echo 'URL should start with http!';
    } else if (gethostbyname(idn_to_ascii($parsed_url["host"], 0, INTL_IDNA_VARIANT_UTS46)) ===
gethostbyname("nginx")) {
    // local access to nginx from php-fpm should be blocked.
        echo 'Oops, are you a robot or an attacker?';
    } else {
    // file_get_contents needs idn_to_ascii(): https://stackoverflow.com/questions/40663425/
        highlight_string(file_get_contents(idn_to_ascii($url, 0, INTL_IDNA_VARIANT_UTS46),
               false,
               stream_context_create(array(
                   'http' => array(
                       'follow_location' => false,
                       'timeout' => 2
                   )
    )));
    }
}
?>
```

我们提交 `http://ocu.chal.seccon.jp:10000/flag.php` , 返回 `Forbidden. Your IP: 172.25.0.1` , 看来我们的 ip 被定为 `172.25.0.1` , 根据代码, 我们得先知道 nginx 的 ip, 简单 fuzz 后, 提交

`http://172.25.0.3/flag.php` 会触发 `Oops, are you a robot or an attacker?` , 确认了 nginx 的 ip 后, 跟上题一样, 我们在上一题给出的网站构造域名, 一个填 `172.25.0.3` , 另一个 IP 就随意了, 原理也跟上题类似, 当然也是随机的, 多点几次就访问到 flag 了.

当然本题还有其他一些有趣的解法, 比如以下的一些 payload 都是可以的

```
http://nginx：80/flag.php
http://@nginx/flag.php
http://nginx／flag.php
```

具体的原理大家可以去 google 其他人写的 wp

结束语

目前来看在 CTF 中出现的 DNS Rebinding 利用相对来说还是比较简单的, 也可能是我做题做的不够多没有遇到更复杂的情况, 网上也有一些讲述关于实战的文章, 由于本人水平有限, 暂时还没进行过实战的利用, 如果文章有纰漏, 还请师傅们纠正.

参考链接

https://bbs.pediy.com/thread-230047.htm

写下你的评论...

不太理解。

意思就是社工目标点你的网页，让他帮你发送 payload 吗

还是说直接控制对方常访问域名的解析权