DNS 缓存毒化调研报告

王一航
哈尔滨工业大学

日期：2021 年 7 月 6 日

摘　　要

DNS 协议自从 1983 年开始计划设计至今已经有接近四十年的历史。目前 DNS 协议已经成为整个互联网的基础设施，各种互联网的关键服务如 CDN 等都依赖于 DNS 协议，这也意味着一旦 DNS 协议被攻击，将会影响到不可估量的互联网设备与服务，甚至导致整个互联网的瘫痪。然而安全性并没有作为设计 DNS 协议时的考虑因素，因此 DNS 协议的广泛应用也吸引了无数攻击者与研究者对其安全性进行研究。目前 DNS 协议最主要也是最严重的威胁来自于 DNS 缓存毒化攻击。DNS 缓存毒化攻击即将错误的域名到 IP 的映射引入 DNS 的缓存中，从而将用户误导至错误的地址。本文将会对 DNS 协议的发展历程中产生的各种缓存毒化的攻击方式，如：Kaminsky Attack [1], 利用 ICMP Side Channel 进行端口探测 [2] 以及利用 IP 分片绕过 DNS 随机化措施 [3-4] 等，与对应的缓解措施，如：0x20 encoding[5], WSEC [6] 以及 XQID [7] 等进行梳理，并提出 DNS 缓存毒化与网络协议安全的一些不成熟的想法。

关键词： DNS 缓存毒化，DNS 协议，Kaminsky 攻击，生日悖论

目录

1 研究背景

Domain Name System (DNS) 是一个将对人类友好的域名映射到对机器友好的 IP 地址的分布式系统。从 1987 年首个 RFC 1034 [8] 公布至今已经有 34 年的历史。现在，DNS 协议承担了现代互联网的关键基础设施的角色。但其在设计之初并未将安全性作为其设计目标之一 [8]，因此 DNS 成为了黑客的重点攻击目标，在 DNS 协议 34 年的历史中，各种各样对 DNS 的攻击层出不穷。其中最重要且最危险的针对 DNS 的攻击即 DNS 缓存毒化（投毒）攻击 [9]。

攻击者利用 DNS 缓存毒化攻击能够造成多种对用户与网站的危害。例如：攻击者可以将一些关键网站（如：政府或银行网站）的 DNS 记录毒化为 (1) 不存在的地址。当正常用户访问目标网站时，将会被引到到不存在的地址，导致访问失败，从而实现了对目标网站的拒绝服务攻击；(2) 攻击者控制的恶意地址。互联网上的许多信任链都基于域名系统，这使得攻击者只要控制域名系统，基于这些信任的系统的安全性就荡然无存。攻击者在将目标网站的 IP 地址毒化为攻击者控制的 IP 地址之后，可以发起网站钓鱼攻击 [10]、发送垃圾邮件或钓鱼邮件，另外攻击者也可以利用用户对于关键网站域名的信任进行中间人攻击或者恶意程序分发等。具有攻击动机的主体可能并不只是黑客团体，政府或者互联网服务提供商（ISP）也可能会主导 DNS 缓存毒化攻击 [11] 对民众进行网络审查。

2 攻击方式

DNS 缓存毒化的本质是攻击者通过各种方法使 DNS 解析器的缓存中存在无效或恶意的 DNS 记录。

一个典型的 DNS 缓存毒化攻击的流程如图 1 所示。

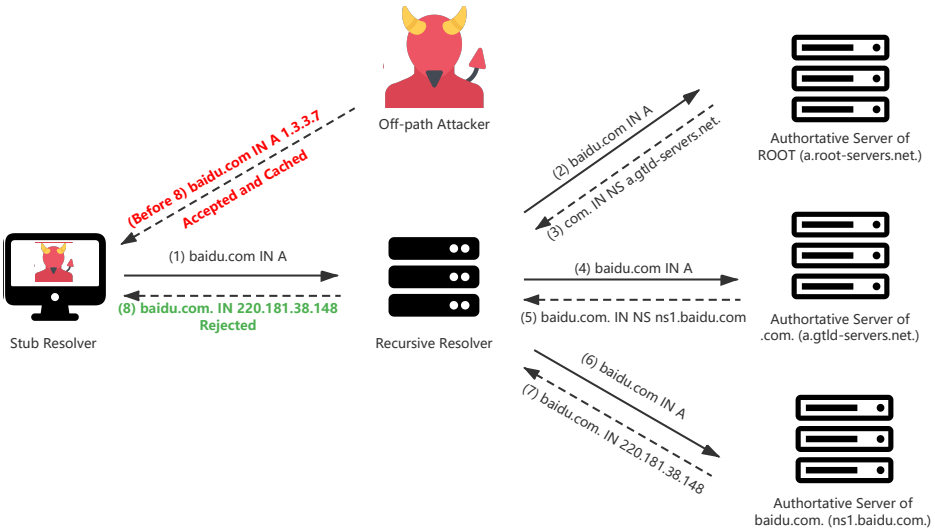

图 1: DNS 缓存毒化攻击

攻击者首先通过流量嗅探或通过控制 Stub Resolver 的行为使其发出一个 DNS 请求，在 Stub Resolver 发出 DNS 请求之后，攻击者立刻需要构造一个合法的 DNS 响应，并且攻击者需要赶在递归解析器将正确的 DNS 响应发送给 Stub Resolver 之前将伪造的响应发给 Stub Resolver。由于

Stub Resolver 会接收并缓存第一个合法的响应，后续到达的响应即使其是合法的，也会被 Stub Resolver 丢弃掉。

从上面的例子，我们可以观察发现，攻击者能够发起一次成功的 DNS 缓存毒化攻击需要满足三个必要条件，我们分别将其称为 时机约束、字段约束 和 时间约束。

- 时机约束：确定 Resolver 何时发起 DNS 请求包
- 字段约束：构造一个合法的 DNS 响应包
- 时间约束：在合法的 DNS 响应返回并被接受之前抢先发给 Resolver

目前大部分的 DNS 缓存毒化攻击都围绕着如何有效地满足这些必要条件展开。下面我们将会针对这三个必要条件进行具体的分析，并梳理在历史上都出现过哪些可以使攻击者更容易满足这三个条件的攻击方式。

2.1 时机约束

为了满足时机约束，攻击者需要确定 Resolver 何时发起 DNS 请求包。攻击者可以通过下述三种方法来确定。

2.1.1 等待缓存过期

一个有耐心的攻击者可以不间断地向 Resolver 发送伪造的 DNS 响应包，直到恰好某一个时刻，Resolver 的缓存过期并发起一个 DNS 查询请求。

2.1.2 流量嗅探

攻击者可以利用中间人攻击等方法进行流量嗅探，从而精确获知解析器何时发起 DNS 请求。

2.1.3 主动触发

攻击者可以通过影响那些使用 Resolver 提供的 DNS 解析服务的其他主体的行为来促使 Resolver 发起一次 DNS 查询，例如：在一个企业的网络环境内部，攻击者可以通过像企业的员工发送一封包含有外部媒体的邮件，当处于企业内部网络的员工打开这封邮件时，邮件客户端将针对外部媒体的域名会进行一次 DNS 请求 [1]，如果攻击者控制了外部媒体所在域名的权威域名服务器，那么攻击者也可以精确获知解析器何时发起 DNS 请求，在这种情况下，事实上攻击者是掌握着发令枪的 [1]。

2.2 字段约束

DNS 查询请求基于 UDP 协议，不同于可以维护连接状态的 TCP 协议，RFC 8085 中要求应用层自己维护会话的状态 [12]。由 UDP 协议本身无法保证接收到的报文具有合法的来源，因此任何人都可以伪造任意源地址、任意源端口号的 UDP 报文，并添加任意的载荷。

[1]攻击者可以简单通过将域名设置为 `$random.attacker.com` 来确保该域名不存在于任何一个层级的解析器缓存中，即递归解析器在迭代查询过程中必然会请求到 `$random.attacker.com` 的权威域名服务器。

攻击者可以通过精心挑选报文中的各个字段，如表 2 所示，构造出一个合法的 DNS 响应报文并发给 Stub Resolver。

IP 头部	源 IP (32-bit)	192.168.1.1
	目的 IP (32-bit)	192.168.1.2
UDP 头部	源端口 (16-bit)	53
	目的端口 (16-bit)	37233
DNS 头部	Transaction ID (16-bit)	12894
DNS 包体	Query Section	baidu.com. IN A
	Answer Section	**baidu.com IN A 1.3.3.7**
	Authority Section	-
	Additional Section	-

图 2: 攻击者构造的恶意 DNS 响应报文

攻击者需要构造如下字段：

- 网络层
  - 响应报文 IP 头部的**源 IP** = 请求报文 IP 头部的**目的 IP**
  - 响应报文 IP 头部的**目的 IP** = 请求报文 IP 头部的**源 IP**
- 传输层
  - 响应报文 UDP 头部的**目的端口** = 请求报文 UDP 头部的**源端口**
  - 响应报文 UDP 头部的**源端口** = 请求报文 UDP 头部的**目的端口**
- 应用层
  - 响应报文 DNS 头部的 **Transaction ID** = 请求报文 DNS 头部的 **Transaction ID**
  - 响应报文 DNS 报文的 **Query Section** = 请求报文 DNS 报文的 **Query Section**
  - 响应报文 DNS 报文的其他字段，须要满足 **Bailiwick 规则**

下面我们将对针对 DNS 响应报文中的每个字段的伪造方式进行梳理。

2.2.1 构造响应包中的源 IP (32-bit)

响应包中的源 IP (32-bit) 即为递归解析器的 IP 地址。

攻击者可以通过注册域名 attacker.com，将该域名的权威域名服务器设置为攻击者控制的机器。然后攻击者促使 Stub Resolver 发起 `$random.attacker.com` 的 DNS 请求，此时 Recursive Resolver 在迭代查询过程中会将 DNS 查询报文发送到攻击者控制权威域名服务器，此时攻击者通过观察权威域名服务器接收到 DNS 查询报文的源 IP 即可获得递归解析器的源 IP 地址。

2.2.2 构造响应包中的目的 IP (32-bit)

响应包中的源 IP (32-bit) 为 Stub Resolver 的 IP 地址。

由于 DNS 缓存毒化的实施过程是鱼叉攻击，因此攻击者必然知道 Stub Resolver 的 IP 地址。如果 Stub Resolver 是局域网内部的一台专门的 DNS 服务器，并且攻击者与 Stub Resolver 在同一个网络内，并且网络中有 DHCP 服务，攻击者可以 (1) 通过 DHCP 服务获取网络管理员预先配

置的 Stub Resolver 地址。(2) 利用 2.1.3 中介绍的方法来触发 Stub Resolver 进行递归查询（若提供）自己控制的域名。

2.2.3 构造响应包中的源端口 (16-bit)

响应包中的源端口 (16-bit) 即 Recursive Resolver 提供 DNS 查询服务所监听的端口，默认为 53 号 UDP 端口。DNS 协议不像 HTTP 协议可以通过浏览器指定某个端口，在 DNS Resolver 的实现中并没有提供可以指定解析路径中 DNS 协议端口的方法。

2.2.4 构造响应包中的目的端口 (16-bit)

响应包中的目的端口 (16-bit) 即 Stub Resolver 在发出 DNS 查询请求的时候使用的 UDP 端口。

2008 年之前的大部分 DNS 软件实现中，请求包的 UDP 源端口号为固定的 53 号端口。事实上，早在 Paul Vixie 于 1995 年发表在 USENIX Security 的论文 [13] 中已经指出 UDP 端口号和 Transaction ID 在 DNS 查询报文中仅占用 16 bits，对于攻击者来说，预测它们并不是一件很困难的事，并且攻击者也可以通过推测这些字段的规律来进行快速预测，即使这些字段已经被添加了随机噪声，攻击者依然非常容易尝试所有值。在对 UDP 端口进行随机化操作之后，攻击者发起一次成功攻击的时间复杂度为 2^{32}，这使得攻击者发起一次成功攻击的复杂度似乎来到了一个不可接受的高度。但是在实践中，Linux 系统上 UDP 的源端口一般都在 $(32768, 60999]$ 区间内，复杂度仅 2^{31} [14]，Windows 和 MacOS 系统上的 UDP 端口号根据 RFC 6056 [15] 范围仅为 $(49152, 65535]$。

即使现在的 DNS 解析器在实现的时候已经对源端口进行了充分的随机化，但是攻击者依然可以通过一些另类的方式对字段的值进行猜测。

在 2021 年加州大学河滨分校钱志云与清华大学段海新团队的论文 [2] 中利用了 ICMP 的目的地址不可达报文与 Linux 内核中对 ICMP 全局速率的限制进行侧信道攻击，探测目标解析器开放的 UDP 端口。同样讽刺的是，Linux 内核中 ICMP 全局速率限制原本也是为了防止 ICMP 报文被用于 DDoS 攻击，但它却变成了攻击者用来探测端口的利器。

在 2020 年以色列巴伊兰大学的 Amit Klein 的论文 [16] 中，对 Linux 内核的伪随机数生成器 `prandom` 进行分析，Amit Klein 发现只需要对 `prandom` 的输出值进行至多 131 次连续观测，并解一个 $GF(2)$ 上的 131 元方程，即可推测出该伪随机数生成器中四个线性反馈移位寄存器的状态。由于在 Linux 的网络协议栈中，许多随机化字段的值都来自于 `prandom` 函数，如：IPID、UDP 端口号等。故利用此种方法也可以对 UDP 源端口号进行预测。攻击者可以利用文中的攻击技术将 DNS 缓存毒化的效率提高 3000 到 6000 倍。实验结果表明，使用该方法发起一次成功的 DNS 缓存毒化攻击的平均时间不超过 1 分钟。

当 Resolver 处于 NAT 设备之后时，NAT 设备可能会破坏 Resolver 的源端口号随机化机制，导致源端口变得重新可预测 [17-18]，这种情况与我们往常以为的 “NAT 是一个天然的防火墙，在 NAT 设备后的服务都很安全。” 刻板印象不符。

另外，攻击者也可以在恶意软件的辅助下，对 Resolver 的 UDP 端口进行耗尽操作 [19]，这样攻击者需要猜测的 UDP 端口值的空间将会大大减小。

2.2.5 构造响应包中的 Transaction ID (16-bit)

DNS 查询请求基于 UDP 协议，不同于可以维护连接状态的 TCP 协议，RFC 8085 中要求应用层自己维护会话的状态 [12]。在 DNS 协议中，Transaction ID 被用来对请求和响应进行关联，解析器发出的每一个 DNS 查询请求都会对应一个 Transaction ID。解析器会维护一个等待列表，其中存放等待中的请求的 Transaction ID。解析器只会接收那些 Transaction ID 的在等待列表中的响应报文，对于不满足要求的报文会直接丢弃。

Transaction ID 在 DNS 报文中共 16 比特，与 UDP 源端口号一起构成攻击者需要完成的重要挑战。在这两个字段都充分随机化的情况下，攻击者需要至多猜测 2^{32} 次才可能攻击成功。然而，在互联网的早期阶段，这两个字段并没有实现充分地随机化。

1997 年，CERT 发布了 CA-1997-22 其中描述了 Bind 的一个漏洞，人们意识到 Bind 并没有对 Transaction ID 进行随机化，事实上，Bind 的 Transaction ID 完全是自增的 [20]。2002 年，研究者 Vagner Sacramento 发现 Bind 会同时从同一个 IP 地址发送多个递归查询，这导致攻击者仅仅需要发送数百次即可将攻击成功的概率提高到接近 100%（基于“生日悖论”）[21]。同年，研究者 Michal Zalewski 在 [22-23] 的启发下，对 Vagner Sacramento 进行进一步研究，并提出或许可以利用上述类似技术对 Transaction ID 的随机性进行分析的假设。2003 年，[24] 通过对这些字段的分布进行可视化的方法验证了 Michal Zalewski 的假设。2007 年，[25] 中也对 Bind 8 和 Bind 9 的多个 Transaction ID 生成算法进行了分析，发现其 DNS 请求包的 UDP 源端口号与 Transaction 仍然可以被有效地预测（理想情况下预测的成功率在 25% 与 43% 之间）。2020 年，[16] 的工作中提到，在 Linux 操作系统下，Transaction ID 的生成也依赖于内核提供的 $prandom$ 函数。因此攻击者可以利用相同的方法对 Transaction ID 进行预测。

2.2.6 构造响应包中的 Query Section

响应包中的 Query Section 即 Stub Resolver 查询的域名对应的 Query Section，并且绝大多数情况 DNS 请求报文的 Type 为 A/AAAA 记录。攻击者可以利用 2.1.3 中的方法通过影响 Stub Resolver 的用户的行为，使其发起指定 DNS 请求，此时攻击者即可轻易构造出 Query Section 与请求包一致的响应包。

2.2.7 构造响应包中的其他 Section

正如 2.2 中提到的，攻击者所构造的响应包还需要满足 Bailiwick 规则。长期以来，大家都认为目前的 DNS 协议在对字段进行充分随机化之后，在 Bailiwck 规则的加持下，已经对 DNS 缓存毒化攻击免疫。

通常情况下，攻击者都希望能够对一些关键网站的 DNS 进行污染，但这些关键网站的 DNS 资源记录往往都广泛存在于 DNS 的缓存中，尽管攻击者可以通过查询一个随机的子域名来绕过缓存机制，但是污染这些子域名的造成的影响往往不大 [2]。

[2]攻击者在污染随机子域名之后可能会对某些 Web 网站的 Cookie 的同源策略存在威胁。如：当某网站主站的 Cookie 的 Path 被设置为 * 并且网站存在 XSS 漏洞的时候，攻击者可以在不违背 Cookie 同源策略的情况下，将用户的机密 Cookie 发送至攻击者控制的服务器。

直到 2008 年，Dan Kaminsky 提出了一种巧妙的绕过 Bailwick 规则的攻击方式才打破了人们的固有观念 [1]。这种攻击方式在利用随机域名前缀来绕过缓存之后，并未直接在响应包的 Answer Section 中写入需要被毒化的资源记录。而是将伪造的资源记录通过 Glue Record 的形式添加在了 Additional Section 中。如图 3 所示，响应包中的其他 Section 并没有未被 Bailiwick 规则。并且，一旦该响应报文被解析器接收，在缓存的有效期内，攻击者将直接接管整个 `baidu.com` 域。

IP 头部	源 IP (32-bit)	192.168.1.1
	目的 IP (32-bit)	192.168.1.2
UDP 头部	源端口 (16-bit)	53
	目的端口 (16-bit)	37233
DNS 头部	Transaction ID (16-bit)	12894
DNS 包体	Query Section	$random.baidu.com. IN A
	Answer Section	-
	Authority Section	**baidu.com IN NS www.baidu.com**
	Additional Section	**www.baidu.com IN A 1.3.3.7**

图 3: Dan Kaminsky 构造的攻击报文

2.2.8 另类构造方法

由于 DNS 协议并不提供**数据加密**与**完整性校验**，因此攻击者如果具有流量嗅探的能力，上述所有基于 DNS 基于随机化的校验手段将会全部失效。另外在具体的实现中，真正的随机性很难达到，攻击者可以通过各种方式对其进行预测，并且某些网络设备（如：NAT 设备，网络防火墙）也可能成为破坏随机性的重要原因。最后，攻击者可以同时向 DNS 解析器发出多个伪造的响应。

有一些研究者提出，如果我们可以触发 DNS 响应报文对 IP 分片进行伪造。1987 年，[26] 率先提出 IP 分片有可能导致网络性能降低等问题。2011 年到 2014 年，来自以色列巴伊兰大学的研究者 Herzberg 等发表了 3 篇“Fragmentation Considered”系列文章 [27, 3, 28]，首次提出了利用 IP 分片技术来进行 DNS 缓存毒化攻击，直接绕过传输层上层所有基于随机化的安全措施。2014 年，Kyle Schomp 等发现某些 DNS Forwarder 对 DNS 响应报文的校验并不完善，将会导致 DNS 缓存毒化漏洞 [29]。2020 年，清华大学段海新老师团队的论文 [4] 中就利用了 IP 分片技术与 DNS Forwarder 对响应报文检测不完善的特性成功实现了 DNS 缓存毒化攻击。

2.3 时间约束

攻击者需要在合法的 DNS 响应返回并被 Resolver 接受之前抢先发给 Resolver。攻击者可以 (1) 提高自己的发包速率。攻击者可以购置更大的带宽，使用多个设备同时进行攻击等。(2) 延长时间窗口。

2.3.1 延长时间窗口

攻击者可以通过触发 Resolver 的 Response Rate Limiting (RRL) [3] 来使 Resolver 无法对正常的 DNS 查询请求进行响应，从而延长时间窗口，提高攻击的成功率 [30, 2]。

攻击者也可以通过自己控制的权威域名服务器以极低的速度向递归解析器返回一条 CNAME 链，让递归解析器不断对攻击者控制的权威域名服务器进行查询，但始终得不到最终的结果，最终在多次尝试后放弃 [2]，而这个多次尝试的过程就为攻击者延长了时间窗口。

3 防御措施

在上文中，我们对攻击者发起一次成功攻击的三个必要条件进行了分析，为了对抗对 DNS 缓存毒化攻击，我们只需要破坏三个必要条件中的其中任意一个即可。下面我们将分别针对如何破坏 *时机约束*、*字段约束* 与 *时间约束* 这三个必要条件来进行介绍。

3.1 破坏时机约束

破坏时机约束即使得攻击者无法获知 Resolver 发起 DNS 请求的时机。

当攻击者具有向直接或者间接 Stub Resolver 发起 DNS 请求的能力的时候，该条件无法被破坏。

3.2 破坏字段约束

破坏字段约束即使得攻击者构造出一个有效 DNS 响应包的成本变得不可接受或不可能做到。

可以分为两个思路，(1) 对攻击者需要猜测的空间进行扩大，并在值选取的时候尽可能引入随机化的特性。(2) 利用密码学的手段对报文进行签名校验，使其无法伪造。

3.2.1 防止攻击者猜中响应包中的源 IP (32-bit)

源 IP 表示 Resolver 所使用的上游 DNS 服务器的 IP。基于克服单点故障和性能的考虑，可以对解析器配置多个上游 DNS 服务器，在缓存未命中的情况下，根据某种特定算法（如：随机选择）选择上游 DNS 服务器进行查询 [31-32]。这提高了攻击者发起一次成功攻击的难度。因为其为了构造有效的 DNS 响应，还需要猜测解析器所使用的上游 DNS 服务器的 IP 地址。

然而攻击者可以首先利用 [32] 中提出的 RDNS 发现的方法获得所有 RDNS，然后利用 [3] 中提出了 NS Blocking 方法对 RDNS 服务器进行拒绝服务攻击，这将导致基于 RDNS 的随机化手段无效。

3.2.2 防止攻击者猜中响应包中的目的 IP (32-bit)

目的 IP 即 Resolver 的 IP 地址，攻击者在攻击之前就必然已知，无法防止攻击者猜中。

[3]该机制原本是被设计用以防范攻击者利用 DNS Resolver 发起分布式拒绝服务攻击的措施，通过限制一个时间片内 Resolver 可以发出的 DNS 响应报文数量。但讽刺的是，这个缓解措施却引入了新的安全威胁。

3.2.3 防止攻击者猜中响应包中的源端口 (16-bit)

源端口号即 DNS 服务器监听的端口号，根据 2.2.3，该字段固定，无法防止攻击者猜中。

3.2.4 防止攻击者猜中响应包中的目的端口 (16-bit)？

目的端口即 Resolver 在发出 DNS 请求的时候使用的 UDP 源端口号。

UDP 源端口号经历了固定 53 端口、序列递增直到随机化的发展历程，安全性在逐步增加。攻击者可以通过随机数预测或端口扫描的方法对 UDP 端口进行猜测。

因此，为了防止攻击者对 UDP 端口号有效预测，应该使用密码学安全的随机数来作为操作系统的 UDP 源端口号。另外，应该采取手段防止攻击者进行 UDP 端口扫描。例如：为了防止攻击者通过 ICMP 侧信道的方式对 UDP 端口号进行扫描，管理员应该尽可能禁用 DNS 服务器的 ICMP 响应（或者至少禁用 ICMP 目的地址不可达响应），或者对 ICMP Global Rate Limiting 进行随机化减少操作 [2]。

3.2.5 防止攻击者猜中响应包中的 Transaction ID (16-bit)

在 DNS 协议设计之初，Transaction ID 被设计为 16 比特，其空间仅有 $2^{16} = 65536$，为了防止攻击者对该字段进行暴力破解，[7] 提出一种叫 `Extended QID (XQID)` 的方法，该方法利用 QNAME 字段对 QID 进行扩展，向域名中添加一个精心设计的前缀，使原本 16 bits 的 QID 的空间大小从原本的 2^{16} 被扩展为 $2^{16} + (|\{a, b, \ldots, z\}| + |\{0, 1, \ldots, 9\}|)^{24}$ ~$2^{16} + (|\{a, b, \ldots, z\}| + |\{0, 1, \ldots, 9\}|)^{63}$ [4]，从而极大降低攻击者猜中的概率。

但 [9, 33] 提出攻击者可以通过构造一个接近域名长度极限的 DNS 查询请求来绕过这种缓解措施。

3.2.6 防止攻击者猜中响应包中的 Query Section

根据 2.2.6，攻击者可以通过影响 Stub Resolver 的行为来猜测 Query Section。

RFC 1034 [8] 中规定了一个域名由多个 label 组成，每一个 label 都必须以字母开头，字母数字或短横线作为主体，并以字母或者数字结尾。并域名可以被以任意的大小写组合进行存储，即在存储时，两个拥有相同拼写但是不同大小写的域名是等价的 [5]。但是在进行比较的时候，需要以大小写敏感的方式进行。如此设计主要是为了兼容未来可能出现二进制的域名。

因此 2008 年 [5] 基于这个假设，提出 Resolver 可以在发出 DNS 查询请求之前对 Query Section 中的 QNAME 字段中的英文字母进行随机大小写操作。如此一来，攻击者不仅需要猜对受害用户查询的域名，还需要猜对受害用户的 DNS 解析器在发送 DNS 请求的时候 Query Section 中的 QNAME 的大小写规则。从一定程度上提高了攻击者猜中的难度，但对于那些英文字母较少的域名该方法提供的随机性明显不足 [9, 33]。

2009 年 [6] 提出可以通过对查询时的域名添加随机前缀的方法来添加更多随机性。

[4] XQID 为了保持与当前 DNS 协议的兼容性，因此其扩展的 QID 不区分大小写

[5] 如：`BaiDU.coM` 与 `baidu.cOm` 是等价的，但是若 DNS 请求报文中的 QNAME 是 `BaiDU.coM`，响应包中 Query Section 中的 QNAME 也必须是 `BaiDU.coM`。

然而，与应对 XQID 的方式类似，对于这种利用 QNAME 字段来添加扩展爆破空间的方式，攻击者可以通过构造一个极限长度的域名，并对其进行 DNS 查询请求来绕过这种缓解措施[9, 33]。

3.2.7 提高攻击者构造响应包中的其他 Section 的难度

为了防止恶意的 Name Server 返回不属于自己管辖的域名信息。1993 年 Bind 实现了 Bailiwick 规则，用以决定是否接受某权威域名服务器返回的结果。[34, 24]。

但目前 Bailiwick 依然不是 DNS 标准的一部分，其具体规则取决于 Resolver 的实现，因此在实现的时候不同的 DNS 软件会存在差异。如：[1] 中 Dan Kaminsky 通过巧妙构造 DNS 响应绕过了 Bailiwick 规则。

2006 年来自加州大学戴维斯分校的 Lihua Yuan 等提出一种基于 P2P 的 DNS 缓存毒化缓解方案 [35]，通过多个 Peer 的信息来对 DNS 响应包的正确性进行检验和矫正。

在 2016 年发布的 RFC 7873 [36] 中，提出了一种用来验证 DNS 响应包是否来自预期的 DNS 服务器的机制，称为 DNS Cookie，DNS Cookie 分为客户端 Cookie 与服务端 Cookie，两者都利用了 EDNS(0) 中的 OPT Pseudo-RR [37] 来进行传输。可以用来防御来自链路外攻击者的 DDoS 攻击与缓存毒化攻击。

前面几节中描述的方案都是基于非密码学的缓解手段，都是在 DNSSEC 全面部署前的 **权宜之计**。DNS 并不具备**机密性**，当攻击者可以进行**侧信道攻击**或者**中间人攻击**时，攻击者可以轻易绕过**所有基于随机化的缓解措施**。

因此上述**通过增加随机性来降低攻击者猜对的概率**只能作为 DNSSEC 全面部署之前的临时代替方案。

为了弥补 DNS 协议并不提供**完整性**的缺陷，RFC 4033 [38] 提出了 DNSSEC，利用非对称密码学体系对 DNS 报文进行完整性校验。但是目前 DNSSEC 的部署进展非常缓慢，如图 **??** 所示，可见目前距离 DNSSEC 全面部署依然有很长的路要走 [6] 。

并且 DNSSEC 若未进行正确部署，可能导致 DNS 缓存毒化攻击 [40, 3, 41]。另外 DNSSEC 的数据包通常较长，导致攻击者可以利用这个特性来发起 DDoS 攻击 [42] 。

为了弥补 DNS 协议并不提供**机密性**的缺陷，2000 年，[**?**] 设计了 Secret Key Transaction Authentication (TSIG) 为 DNS 响应提供加密签名，但是目前 Systemd-Resolved 和 glibc 均不支持 TSIG。2015 年，Frank Denis 与 Yecheng Fu 设计了一个名为 DNSCrypt 的协议用来提高 DNS 的安全性，它使用加密签名来验证响应是否来自选定的 DNS 解析器并且未被篡改，DNSCrypt 是一个开放的规范，具有免费和开源的参考实现，并且也不隶属于任何公司或组织 [43]。2016 年，RFC 7858 [44] 中描述了如何使用 TLS 来为 DNS 协议添加机密性。2018 年，RFC 8484 [45] 中定义了一种基于 HTTPS 的 DNS 协议。这使得 DNS 流量可以直接利用 HTTPS 的加密的特性来对流量进行加密。目前 Chrome[7] / Firefox 等浏览器已经提供了对 DNS over HTTPS 的支持 [46-47]。Windows 操作系统的 Insider 版本也在 Build 19628 之后的版本中提供了对 DNS over HTTPS 支持 [48]。

[6]截图来自 [39]，日期：2021 年 06 月 27 日

[7]Chrome 从 Chrome 83 之后开始提供对 DNS over HTTPS 的支持。

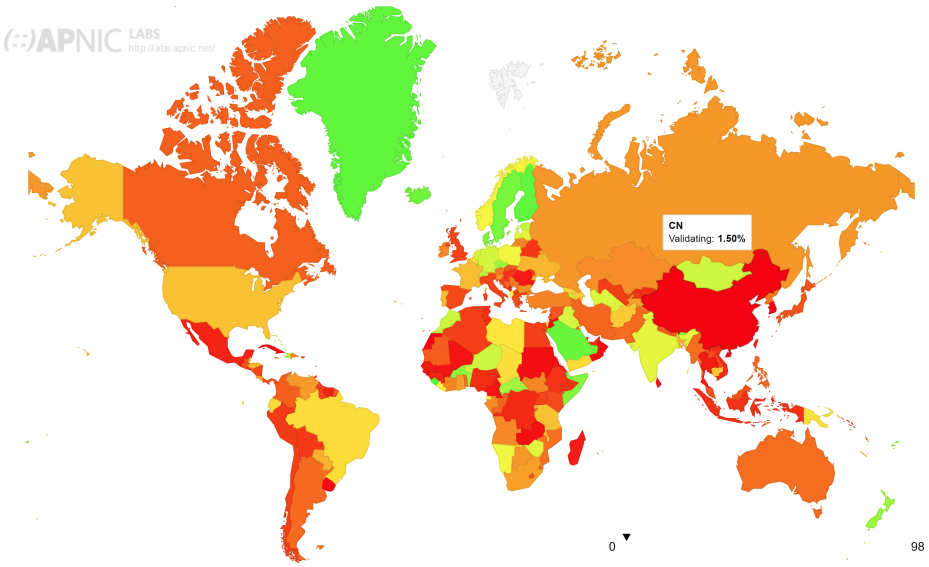

图 4: 全球 DNSSEC 部署情况图

2020 年，[49] 通过对 DoH 的链接进行重置等操作，使得浏览器认为 DoH 服务器不可用，转而使用传统的 DNS 进行查询，而传统 DNS 查询是不提供加密功能的，因此使得用户的 DNS 流量被明文暴露在网络中。并且当浏览器放弃尝试 DoH 转而使用明文 DNS 的时候，用户并不会得到通知。

3.2.8 禁用 IP 分片

在 2.2.8 中，攻击者可以通过 IP 分片来进行 DNS 缓存毒化攻击。IETF 在 2019 年开始推动对 DNS 报文不进行分片的提案，目前仍处于草稿阶段 [50-51]。

对于 Resolver 可以采取 [3] 中提出的减小缓存分片的最大值的方案。对于 Name Server 可以针对每一个响应，都对其添加一个随机的资源记录，这将会导致攻击者无法正确计算第二个 IP 分片的校验和，从而防止伪造 IP 分片的攻击。

3.3 破坏时间约束

RRL 的引入是为了对基于 DNS 协议的 DDoS 攻击进行缓解，但 RRL 的部署引入了新的安全问题。攻击者可以利用 RRL 来对 Resolver 进行拒绝服务攻击，此时则需要进行权衡。[2] 中讨论了对 RRL 进行设置的最佳实践，另外也提出了两种缓解措施，(1) 采用更激进的态度来设置 DNS 的超时时间（例如：少于 1 秒）。(2) 利用任播技术来提高攻击者对权威域名服务器进行 DDoS 的难度。

4 结论

本文梳理了历史上曾经出现过的 DNS 缓存毒化的各种攻击方式与缓解措施。目前 DNS 协议用来防止 DNS 缓存毒化攻击的方法绝大部分依然是基于添加各种随机化机制来提高攻击者猜中的难度，这些措施可以作为 DNSSEC 全面部署前的临时方案，最终为了完全防止 DNS 缓存毒

化攻击，需要利用对消息进行签名验证的 DNSSEC 或对 DNS 流量进行加密的 DNS over HTTPS / DNS over TLS。但目前 DNSSEC 的部署情况依然并不乐观，我们呼吁应该加强对 DNSSEC 的普及。

参考文献

[1] DAN K. It’s the end of the cache as we know it[EB/OL]. Las, Vegas, 2008. https://www.blackhat.com/html/bh-usa-08/bh-usa-08-speakers.html#Kaminsky.

[2] MAN K, QIAN Z, WANG Z, et al. DNS Cache Poisoning Attack Reloaded: Revolutions with Side Channels[C/OL]// LIGATTI J, OU X, KATZ J, et al. CCS ’20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020. ACM, 2020: 1337-1350. https://doi.org/10.1145/3372297.3417280.

[3] HERZBERG A, SHULMAN H. Fragmentation considered poisonous, or: One-domain-to-rule-them-all.org[C/OL]// 2013 IEEE Conference on Communications and Network Security (CNS). IEEE, 2013: 224-232. DOI: 10.1109/CNS.2013.6682711.

[4] ZHENG X, LU C, PENG J, et al. Poison Over Troubled Forwarders: A Cache Poisoning Attack Targeting {DNS} Forwarding Devices[C/OL]//2020: 577-593[2021-03-31]. https://www.usenix.org/conference/usenixsecurity20/presentation/zheng.

[5] DAGON D, ANTONAKAKIS M, VIXIE P, et al. Increased DNS forgery resistance through 0x20-bit encoding: security via leet queries[C/OL]//Proceedings of the 15th ACM conference on Computer and communications security. ACM, 2008: 211-222. DOI: 10.1145/1455770.1455798.

[6] PERDISCI R, ANTONAKAKIS M, LUO X, et al. WSEC DNS: Protecting recursive DNS resolvers from poisoning attacks[C/OL]//2009 IEEE/IFIP International Conference on Dependable Systems & Networks. IEEE, 2009: 3-12. DOI: 10.1109/DSN.2009.5270363.

[7] JESPER G H. Anti DNS spoofing - Extended Query ID (XQID)[EB/OL]. 2008[2021-06-09]. http://www.jhsoft.com/dns-xqid.htm.

[8] MOCKAPETRIS P. RFC 1034: Domain names-concepts and facilities[J]. 1987.

[9] HERZBERG A, SHULMAN H. Security of Patched DNS[C/OL]//FORESTI S, YUNG M, MARTINELLI F. Lecture Notes in Computer Science: Computer Security –ESORICS 2012. Berlin, Heidelberg: Springer, 2012: 271-288. DOI: 10.1007/978-3-642-33167-1_16.

[10] BANU M N, BANU S M. A comprehensive study of phishing attacks[J]. International Journal of Computer Science and Information Technologies, 2013, 4(6): 783-786.

[11] DNS Cache Poisoning in the People’s Republic of China - Research - ViewDNS.info[EB/OL]. 2011[2021-06-25]. https://viewdns.info/research/dns-cache-poisoning-in-the-peoples-republic-of-china/.

[12] POSTEL J. RFC 8085: User datagram protocol[J]. 1980.

[13] VIXIE P. DNS and BIND Security Issues.[C]//Usenix Security Symposium. [S.l.: s.n.], 1995.

[14] ALHARBI F, CHANG J, ZHOU Y, et al. Collaborative client-side DNS cache poisoning attack[C]//IEEE INFOCOM 2019-IEEE Conference on Computer Communications. [S.l.]: IEEE, 2019: 1153-1161.

[15] LARSEN M, GONT F. RFC 6056: Recommendations for transport-protocol port randomization[R]. [S.l.]: BCP 156, RFC 6056, January, 2011.

[16] KLEIN A. Cross Layer Attacks and How to Use Them (for DNS Cache Poisoning, Device Tracking and More)[J/OL]. CoRR, 2020, abs/2012.07432. https://arxiv.org/abs/2012.07432.

[17] Full Disclosure: DNS and Checkpoint[EB/OL]. 2008[2021-06-26]. https://web.archive.org/web/20080908123854/http://seclists.org/fulldisclosure/2008/Jul/0104.html.

[18] CROSS T. DNS cache poisoning and network address translation[J/OL]. Post at IBM’s Frequency X blog, 2008. http://blogs.iss.net/archive/dnsnat.html.

[19] ROEE H, YAIR A. DNS Poisoning Via Port Exhaustion Packet Storm[EB/OL]. 2011[2021-06-14]. https://packetstormsecurity.com/files/105964/dnsp_port_exhaustion.pdf.

[20] CA C A. CA-1997-22: BIND - the Berkeley Internet Name Daemon[R/OL]. CERT Division, 1997: 142. https://resources.sei.cmu.edu/asset_files/WhitePaper/1997_019_001_496176.pdf.

[21] SACRAMENTO V. Vulnerability in the sending requests control of Bind versions 4 and 8 allows DNS spoofing[M/OL]. November, 2002. https://securiteam.com/unixfocus/6n00o2060k/.

[22] MICHAL Z. Strange Attractors and TCP/IP Sequence Number Analysis[EB/OL]. 2001[2021-06-13]. https://lcamtuf.coredump.cx/oldtcp/tcpseq.html.

[23] MICHAL Z. Strange Attractors and TCP/IP Sequence Number Analysis - One Year Later[EB/OL]. 2002[2021-06-13]. https://lcamtuf.coredump.cx/newtcp/.

[24] STEWART J. DNS cache poisoning-the next generation[M]. [S.l.: s.n.], 2003.

[25] KLEIN A. BIND 9 DNS cache poisoning[J]. Report, Trusteer, Ltd, 2007, 3.

[26] KENT C A, MOGUL J C. Fragmentation considered harmful: volume 17[M]. [S.l.: s.n.], 1987.

[27] GILAD Y, HERZBERG A. Fragmentation considered vulnerable: blindly intercepting and discarding fragments[C]// Proceedings of the 5th USENIX conference on Offensive technologies. [S.l.: s.n.], 2011: 2-2.

[28] SHULMAN H, WAIDNER M. Fragmentation considered leaking: port inference for dns poisoning[C/OL]//International Conference on Applied Cryptography and Network Security. Springer, 2014: 531-548. DOI: 10.1007/978-3-319-07536-5_31.

[29] SCHOMP K, CALLAHAN T, RABINOVICH M, et al. Assessing DNS vulnerability to record injection[C]//International Conference on Passive and Active Network Measurement. [S.l.]: Springer, 2014: 214-223.

[30] Blocking DNS Messages is Dangerous[J/OL]. DNS-OARC (Indico)[2021-06-21]. https://indico.dns-oarc.net/event/1/contributions/38/.

[31] Performance Benefits | Public DNS[J/OL]. Google Developers[2021-06-27]. https://developers.google.com/speed/public-dns/docs/performance.

[32] SCHOMP K, CALLAHAN T, RABINOVICH M, et al. On measuring the client-side DNS infrastructure[C]//Proceedings of the 2013 conference on Internet measurement conference. [S.l.: s.n.], 2013: 77-90.

[33] HERZBERG A, SHULMAN H. Security of Patched DNS, technical report 12-04[M]. [S.l.: s.n.], 2012.

[34] SCHUBA C L, SPAFFORD E H. Addressing Weaknesses in the Domain Name System Protocol[M]. [S.l.: s.n.], 1993.

[35] YUAN L, KANT K, MOHAPATRA P, et al. DoX: A peer-to-peer antidote for DNS cache poisoning attacks[C]//2006 IEEE International Conference on Communications: volume 5. [S.l.]: IEEE, 2006: 2345-2350.

[36] EASTLAKE 3RD D, ANDREWS M. Domain Name System (DNS) Cookies[J/OL]. 2016[2021-06-27]. https://www.rfc-editor.org/info/rfc7873.

[37] DAMAS J, GRAFF M, VIXIE P. Extension Mechanisms for DNS (EDNS(0))[J/OL]. 2013[2021-06-27]. https://www.rfc-editor.org/info/rfc6891.

[38] ARENDS R, AUSTEIN R, LARSON M, et al. RFC 4033: DNS security introduction and requirements[R]. [S.l.: s.n.], 2005.

[39] DNSSEC World Map[EB/OL]. [2021-06-27]. https://stats.labs.apnic.net/dnssec.

[40] ARIYAPPERUMA S, MITCHELL C J. Security vulnerabilities in DNS and DNSSEC[C]//The Second International Conference on Availability, Reliability and Security (ARES'07). [S.l.]: IEEE, 2007: 335-342.

[41] SHULMAN H, WAIDNER M. One Key to Sign Them All Considered Vulnerable: Evaluation of ${$dnssec$}$ in the Internet[C]//14th ${$USENIX$}$ Symposium on Networked Systems Design and Implementation (${$NSDI$}$ 17). [S.l.: s.n.], 2017: 131-144.

[42] AFEK Y, BREMLER-BARR A, SHAFIR L. NXNSAttack: Recursive ${$dns$}$ Inefficiencies and Vulnerabilities[C]// 29th ${$USENIX$}$ Security Symposium (${$USENIX$}$ Security 20). [S.l.: s.n.], 2020: 631-648.

[43] DNSCrypt - Official Project Home Page[EB/OL]. [2021-06-27]. https://www.dnscrypt.org/.

[44] HU Z, ZHU L, HEIDEMANN J, et al. Specification for DNS over Transport Layer Security (TLS)[J/OL]. 2016 [2021-06-27]. https://www.rfc-editor.org/info/rfc7858.

[45] HOFFMAN P, MCMANUS P. DNS Queries over HTTPS (DoH)[J/OL]. 2018[2021-06-27]. https://www.rfc-editor.org/info/rfc8484.

[46] A safer and more private browsing experience with Secure DNS[J/OL]. Chromium Blog[2021-06-27]. https://blog.chromium.org/2020/05/a-safer-and-more-private-browsing-DoH.html.

[47] Firefox DNS-over-HTTPS | Firefox Help[EB/OL]. [2021-06-27]. https://support.mozilla.org/en-US/kb/firefox-dns-over-https.

[48] Windows Insiders can now test DNS over HTTPS[J/OL]. TECHCOMMUNITY.MICROSOFT.COM, 2020[2021-06-27]. https://techcommunity.microsoft.com/t5/networking-blog/windows-insiders-can-now-test-dns-over-https/ba-p/1381282.

[49] HUANG Q, CHANG D, LI Z. A Comprehensive Study of DNS-over-HTTPS Downgrade Attack[C]//10th ${$USENIX$}$ Workshop on Free and Open Communications on the Internet (${$FOCI$}$ 20). [S.l.: s.n.], 2020.

[50] FUJIWARA K. Measures against cache poisoning attacks using IP fragmentation in DNS[J/OL]. Working Draft, IETF Secretariat, Internet-Draft draft-fujiwara-dnsop-fragment-attack-01, 2019. https://datatracker.ietf.org/doc/html/draft-fujiwara-dnsop-fragment-attack.

[51] FUJIWARA K. Fragmentation Avoidance in DNS: oposes to avoid IP fragmentation in DNS.[R/OL]. 2020. https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-avoid-fragmentation.