

DNS 反射攻击检测调研报告

姜萍

201928018670047

目录

1 DNS 反射攻击原理.....	2
1.1 DNS 反射.....	2
1.2 DNS 反射攻击漏洞存在的原因.....	2
1.3 DNS 反射和 DNS 放大的联系.....	3
1.4 DNS 反射放大攻击.....	3
2 现状.....	5
2.1 安全事件.....	5
2.2 开放 DNS 解析器探测现状.....	6
2.3 防御/减轻 DNS 反射攻击.....	7
3 检测方法.....	9
3.1 基于时间特征的检测.....	9
3.2 基于网络流量统计分析和网络流量控制的检测.....	10
3.3 基于机器学习的检测方案.....	11
3.4 基于 DNS 查询历史匹配.....	12
3.5 基于 SDN 的 DNS 反射放大检测和缓解.....	13
3.6 基于行为的 DNS 反射放大检测.....	15
3.7 基于线性关系的 DNS 反射放大检测.....	16
3.8 新的想法.....	16
4 DNS 反射放大器的特征.....	16
5 探测 DNS 反射放大器.....	17
6 数据集.....	18
参考文献.....	20

1 DNS 反射攻击原理

1.1 DNS 反射

DNS 反射攻击是一种通过向开放的 DNS 服务发送伪造源发地址的查询请求来将应答流量导向攻击目标的一种拒绝服务攻击手段，导致 DNS 服务器充当“反射器”，将所有响应发送到受害者的系统，同时保持攻击者的身份隐藏。

反射攻击有很多种，比如无论是基于 DNS 还是基于 NTP，其最终都是基于 UDP 协议的。在 UDP 协议中正常情况下客户端发送请求包到服务端，服务端返回响应包到客户端，但是 UDP 协议是面向无连接的，所以客户端发送请求包的源 IP 很容易进行伪造，当把源 IP 修改为受害者的 IP，最终服务端返回的响应包就会返回到受害者的 IP，大量的 应答分组将汇聚在受害者端， 堵塞受害者的网络而最终造成 DDoS 攻击，这就形成了一次反射攻击，是一种基于带宽耗尽型的攻击方式。

其中，DNS 反射攻击利用以下三个关键因素，达到了可观的 DDoS 流量放大效果：

- 1) 易于伪造的 UDP 源 IP 地址；
- 2) 互联网上数量庞大的开放解析器；
- 3) DNS 应答大小的高放大比；

事实上许多基于 UDP 的协议都可以用作反射攻击，如 NTP、SSDP 等，近期都被攻击者利用，但互联网上 DNS 的反射源（开放解析器）数量是最多的。

1.2 DNS 反射攻击漏洞存在的原因

互联网上大量 DNS 服务的默认配置缺失，导致 DNS 服务器会响应所有 IP 的 DNS Query 请求。DNS 服务是整个互联网的基础服务，在我们链接互联网的时候，需要通过 DNS 解析将域名转化成对应的 IP 地址，理论上来说 ISP 的 DNS 服务器只响应来自它自己客户 IP 的 DNS Query 响应，但是实际应用中总是用缺省的配置，不进行更细致的检查的判断

DNS 大部分使用 UDP 协议。UDP 协议没有类似 TCP 协议中的握手过程去验证请求的源 IP 的真实性，那么攻击者就可以轻易伪造成受害目标 IP 地址。如图 1 所示，攻击者（实际上是攻击者控制的傀儡机）发送大量伪造了 Victim IP 的请求给 DNS 服务器，DNS 服务器成为放大器将 DNS 响应回复给受害者，造成类似 DoS 攻击。

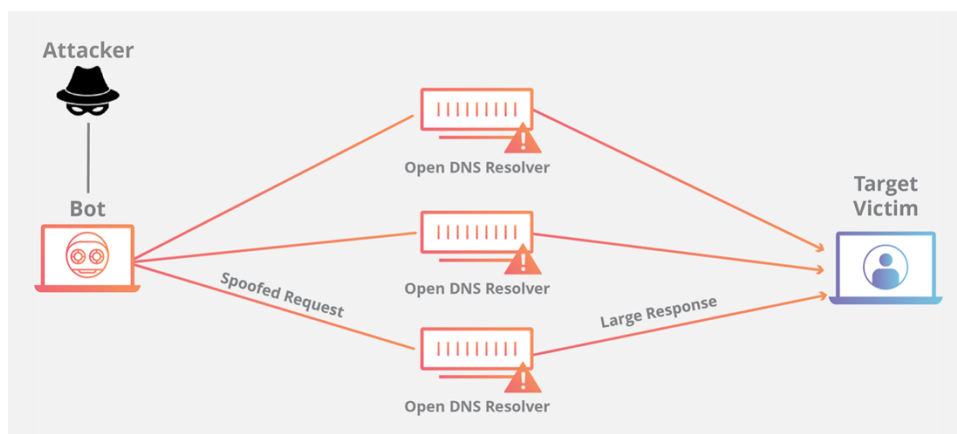


图 1 DNS 反射攻击示意图

1.3 DNS 反射和 DNS 放大的联系

放大攻击是攻击者的一个小请求包最终会产生一比请求包大数倍的响应包给受害者，使攻击者可以利用其自身有限的带宽等资源造成较大的攻击效果，就是一个放大的状态。而因为 DNS 反射攻击中的一个关键点是 DNS 应答包的大小具有高放大比的特点，所以常常将反射和放大连在一起称为 DNS 反射放大攻击。

^[1]根据 DNS 协议（RFC2617 文档）的特点，DNS 应答报文一般是查询报文的几倍甚至几十倍。普通的 DNS 请求包的大小一般 70 字节左右，DNS 应答包最大 512 字节(请求包的查询类型为 ANY)，超过 512 字节会被截断。实施 DNS 反射攻击，攻击流量也能被放大 7 倍多。而启用 DNS 的扩展机制 EDNS0，可以突破 DNS 消息通过 UDP 传送时的 512 字节限制，支持响应 UDP 数据包最大可达 4000 字节，这样攻击流量能被放大 60 倍左右。再通过开放 DNS 递归服务器的流量反射，相当于攻击数据分组增加了数十倍，达到显著的攻击流量放大的效果。

如果一定要区分 DNS 反射和放大攻击，放大攻击更侧重于达到最大的放大效果，也就是实现上文所说的对一个流量的 60 倍放大，这个方法需要攻击者控制第三方的 DNS 服务器，攻击者在这些被控制的权威域名服务器上先发布大字节的 TXT 资源记录，比如 4000 字节的记录文本，用于响应查询请求。然后，攻击者向大量开放递归服务的 DNS 服务器发送带有扩展字段 OPT RR（伪资源记录）的 DNS 查询请求分组时，就能达到最佳的放大效果。

1.4 DNS 反射放大攻击

目标就是辅以最佳的放大效果，通过 DNS 反射形成 DDoS 攻击。

首先看放大的过程，目标是将相对较小的 DNS 请求转换为巨大的响应。典型的 DNS 请求（仅几行文本）很小（通常为几十个字节），并且返回的响应仅略大。如图 2 所示，真正的（非恶意）DNS 响应的放大因子可能为 1.5 或更小。

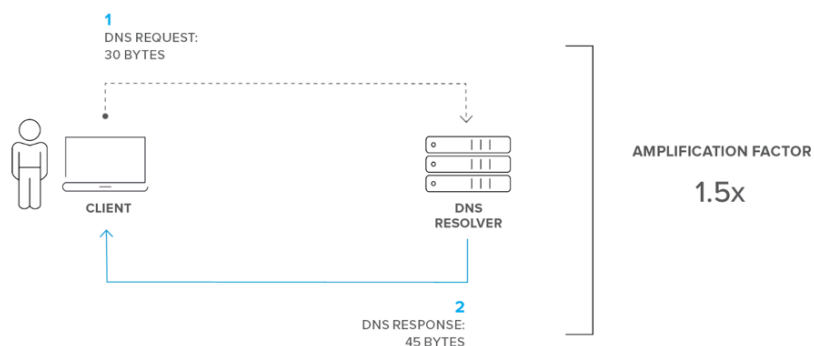


图 2 普通 DNS 放大倍数

为了实现其目标，攻击者以实质上放大响应大小的方式来设计 DNS 请求。一种实现方式是，不仅请求类似 `www.example.com` 这样的网站的 IP 地址，而且请求有关整个域的信息（例如，对记录类型“ANY”使用 DNS 请求），因此响应可能包括有关子域，备份服务器，邮件服务器，别名等的详细信息，一个 10 字节的 DNS 请求就可能会生成 10、20 甚至 50 倍大的响应。如图 3 所示。

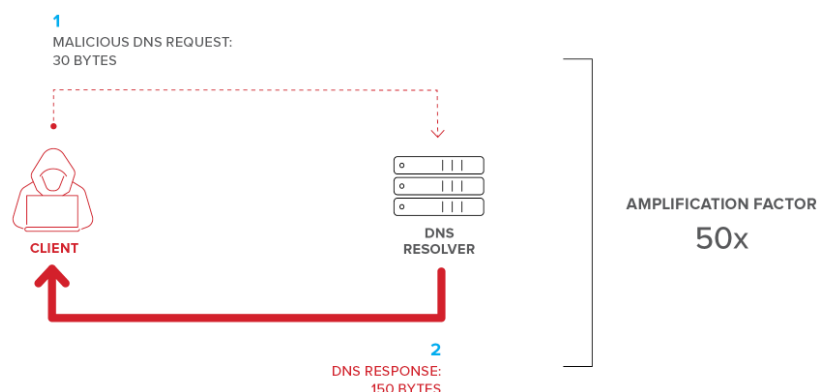


图 3 DNS 高放大比

然后利用 DNS 反射，攻击者在其 DNS 请求中将源 IP 地址伪造（欺骗）为受害者的 IP 地址，并使用多个开放的 DNS 解析器进行此攻击，如图 4 所示。这种攻击的优点在于，它不需要攻击者大量的资源-僵尸网络不是必需的（尽管攻击者当然可以使用一个）。攻击者可以用相对较少的精力和资源来制作 DNS 请求，该请求将以足够的流量轰炸受害者的站点，从而严重损害其性能或将其完全关闭。

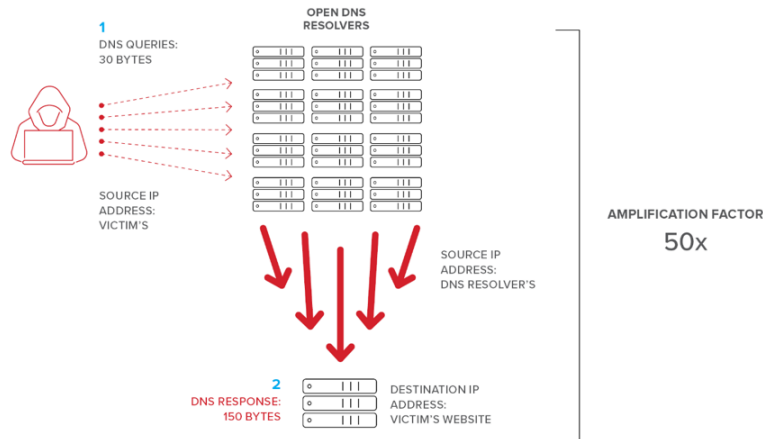


图 4 DNS 反射放大/DDoS 攻击

2 现状

2.1 安全事件

2013 年 Spamhaus 攻击事件中，攻击者向 30 000 多个开放 DNS 服务器发送了对 ripe.net 域名的解析请求，并将源 IP 地址伪造成 Spamhaus 的 IP 地址。DNS 请求数据的长度约为 36 byte，而响应数据的长度约为 3 000 byte，这意味着利用 DNS 反射能够产生约 100 倍的放大效应。Spamhaus 攻击事件中，大量开放 DNS 服务器的响应数据产生了大约 300 Gbit / s 的攻击流量，因此，攻击者只需要掌握和控制一个能够产生 3 Gbit / s 流量的僵尸网络，就能够进行 300 Gbit / s 流量的大规模攻击。^[1]

2016 年 10 月美国 Dyn 公司的 DNS 服务器遭受 DDoS 攻击，导致美国大范围断网。事后的攻击流量分析显示，DNS 反射放大攻击与 SYN 洪水攻击是作为本次造成美国断网的拒绝服务攻击的主力。由于反射放大攻击危害大，成本低，溯源难，被黑色产业从业者所喜爱。^[2]

2020 年由以色列特拉维夫大学和赫兹利亚跨学科研究中心的研究人员发现的存在于 DNS 协议中的 NXNSAttack 漏洞，可影响所有的递归 DNS 解析器，用于造成 DNS 反射 DDOS 攻击。括 CVE-2020-8616 (BIND)，CVE-2020-12662

(Unbound)，CVE-2020-12667 (Knot) 和 CVE-2020-10995 (PowerDNS)。在 NXNSAttack 的场景中，远程攻击者可通过向脆弱的解析器发送 DNS 查询放大网络流量，该解析器会查询由攻击者控制的授权服务器控制器。攻击者的服务器授予虚假的

服务器名称，该名称指向受害者的 DNS 域，造成解析器生成对受害者 DNS 服务器的查询。该攻击会造成放大系数超过 1620。

2.2 开放 DNS 解析器探测现状

Open Resolver Project 持续探测互联网上的开放 DNS 解析器，尽管近一年来开放解析器数量呈缓慢下降趋势，但至今全球依然有超过 2000 万个 IP 地址响应 DNS 请求。2014 年 5 月我们做的教育网范围内的调查也显示，教育网内有 2 万多个 IP 地址响应 DNS 查询，其中 6663 个能够提供正确的递归解析。

^[3]基于 DDoS 反射放大攻击全球探测 2018/03/05 的第四轮与 2019/05/06 CoAP 的第五轮数据，通过 Zoomeye 网络空间探测引擎获取到 2 千万(21,261,177)台 UDP 53 端口相关的主机，对这些主机进行放大倍率探测，实际上只有 384 万(3,847,687)台主机开启了 53 端口，占了扫描总数的 18.1%。在开启了 53 端口的主机中，有 3 万(31,999)台主机放大倍数在 10 倍以上，只占总数的 0.83%，而放大倍数为 1 的主机有 277 万(2,776,027)台^[2]，如图 5。

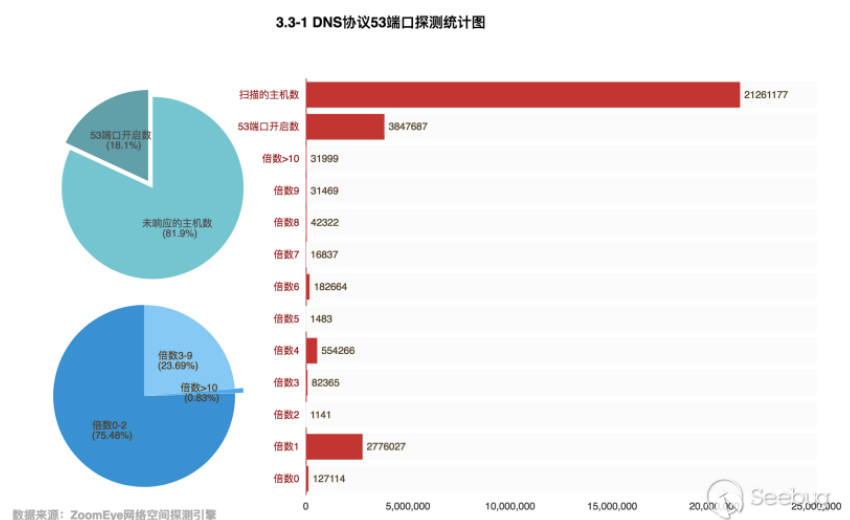


图 5 DNS 反射放大倍数

和以往数据相比，互联网上 DNS 服务器的和可被利用的 DNS 服务器数量均处于下降状态。这 3 万台放大倍数大于 10 的主机全球分布情况，如图 6 所示，全球来看，美国排在第一位。对这些可利用主机在我国的分布情况进行了统计，如图 7 所示，和上一湖北省的 DNS 服务器数量有了明显的提高。

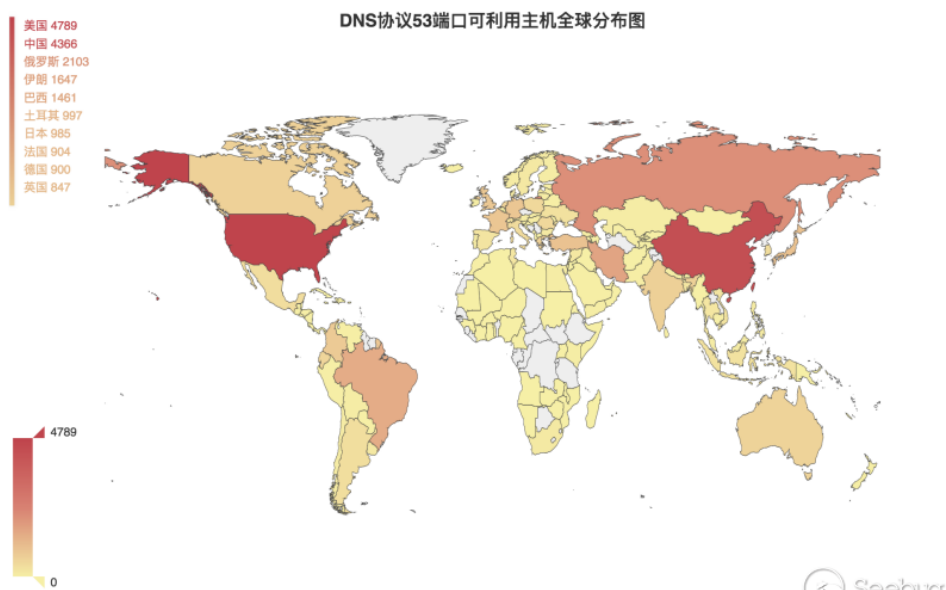


图 6 DNS 协议 53 端口可利用主机全球分布图

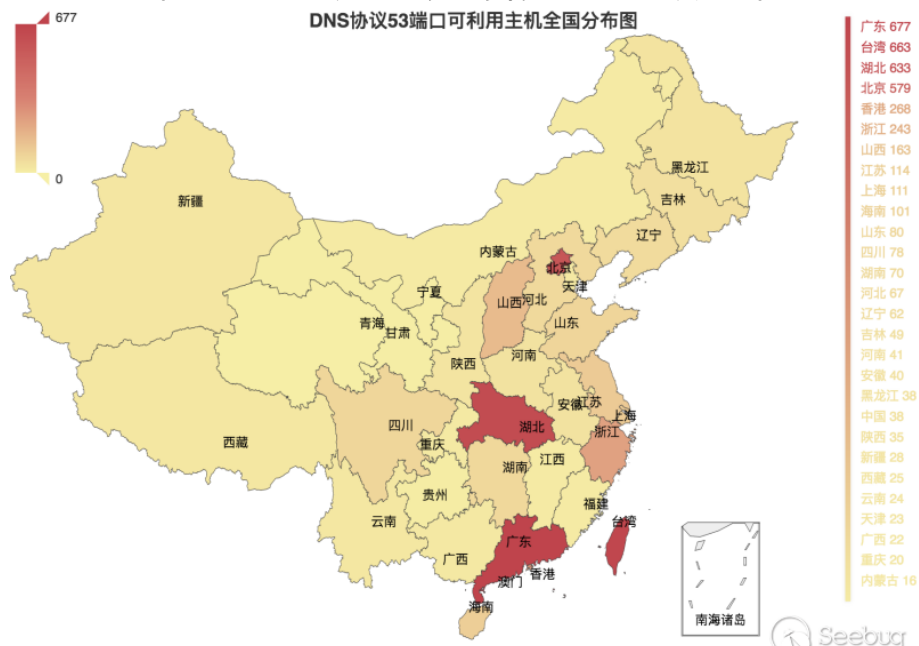


图 7 DNS 协议 53 端口可利用主机全国分布图

2.3 防御/减轻 DNS 反射攻击

1) 简单的，如果是 DNS 服务器的管理员，可以按照下面的配置选择关闭递归功能和限制可查询的 IP 地址。

```
options { recursion no;};
```

```
options { allow-query {192.168.1.0/24;};};
```

如果是客户端，也就是可能的受害者，也许是个人或者企业，首先可以通过网络层的 ACL 规则来防御，或者使用抗 DDoS 系统进行流量清洗，目前大部分的云服务商都有这类功能。

2) 通过 DNS 切换域名对应 IP 来躲避攻击影响。

这种方法很有效，但是也存在一些不足，就是权威 DNS 解析记录的 TTL 会被靠近用户端的 DNS 服务器篡改，这导致不管我们把 ttl 改到多低，等到了用户那还是要等十分钟左右，在这段时间企业的业务是会受到很大影响的。当然还可以从运营商或云服务商那购买这种抗 DDOS 服务，这种方法很有效（至少在流量不是特别大的时候），但是成本较高。

3) Cloudflare 的解决方案 “Gatebot”

借助正确配置的防火墙和足够的网络容量，分散攻击流量以阻止反射攻击。尽管攻击将针对单个 IP 地址，但 Cloudflare 的 Anycast 网络会将所有攻击流量分散到不再造成破坏的地步。Cloudflare 能够利用规模优势在许多数据中心之间分配攻击的重量，平衡负载，从而永远不会中断服务，并且攻击也不会淹没目标服务器的基础架构。在最近六个月的窗口中，Cloudflare 的 DDoS 缓解系统“Gatebot”检测到 6,329 次简单的反射攻击（每 40 分钟一次），网络成功地缓解了所有这些攻击。

<https://blog.cloudflare.com/meet-gatebot-a-bot-that-allows-us-to-sleep/>

4) Imperva 的解决方案

imperva DDoS 保护解决方案利用 Anycast 技术来平衡其高性能交换服务器全球网络中的攻击负载，在该网络中，流量会进行深度包检测，以过滤掉恶意 DDoS 流量。Imperva 服务部署在受保护的 DNS 服务器之前，如图 8，成为所有 DNS 查询的第一个目的地。Imperva 充当安全代理，可防止非法 DNS 查询到达 DNS 服务器，同时将其屏蔽直接 IP 网络层攻击。针对 DNS 反射 DDoS 攻击的保护结合了信誉和基于速率的启发式技术，可以检查传入的查询并过滤出恶意数据包，而不会影响合法访问者。

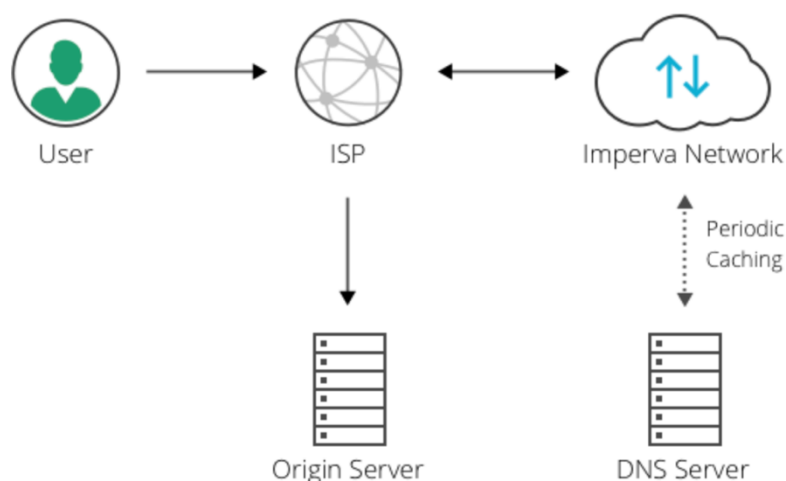


图 8

5) 一些预防性安全措施^[4]

本地 DNS 服务器配置为仅处理组织内部的 DNS 请求；

使用支持 DNS 的防火墙仅允许 DNS 响应进入网络，该响应与从本地 DNS 服务器发送的请求匹配；

使用 DNS Anycast 分发流量并避免任何单个 DNS 服务器超载；

如果条件允许，使用第三方 DDoS 保护（清理）服务。

3 检测方法

3.1 基于时间特征的检测

GuXQ, WangHY, NiTG 等^[5]，在基于网络流量行为时间序列的 DDoS 检测技术中，主要是采用统计分析、关联分析等多种技术手段周期性检测域名服务器端对用户发送和接收的流量行为中的异常模式，判断其是否满足 TCP 和 UDP 行为的时间同步性，识别攻击者和正常用户，从而发现异常行为。

基于生存时间值智能研判的 DNS 攻击检测技术^[1]，主要包括了 DNS 攻击行为检测和伪造源地址域名查询智能抑制两部分。如图 9，正常来说，同一个客户端发送的 DNS 查询请求数据分组在一定时间内经过基本固定的路由到达 DNS 服务器，也就是说在一段时间内，从同一个用户端发给 DNS 服务器的 DNS 请求数据分组的 TTL 值应该是稳定的。由于攻击者发动 DNS 反射攻击时，需要僵尸网络多台傀儡机同时发送伪造源地址的 DNS 请求分组，攻击者虽然可以伪造发向 DNS 服务器的数据分组中的源 IP 地址、TTL 等任何字段的信息，但是由于傀儡机通常分布在不同地区，因

此，攻击者很难正确伪造真实 IP 地址对应的主机发送的数据分组到达 DNS 服务器的跳数，因此，基于 TTL 值智能研判的 DNS 攻击检测技术主要是对来自同一个源 IP 地址的 DNS 请求数据分组 TTL 值进行实时比对，对同一源 IP 地址 TTL 值频繁变动的 DNS 请求分组实施无递归的本地解析或直接丢弃的抑制方法。由于 IP 地址伪造技术是成功实施分布式 DNS 反射 DDoS 攻击的必要条件，利用本方法能够准确发现伪造源 IP 地址分组并抑制域名反射放大攻击，有效解决了原有技术中漏判率和误判率严重的问题。

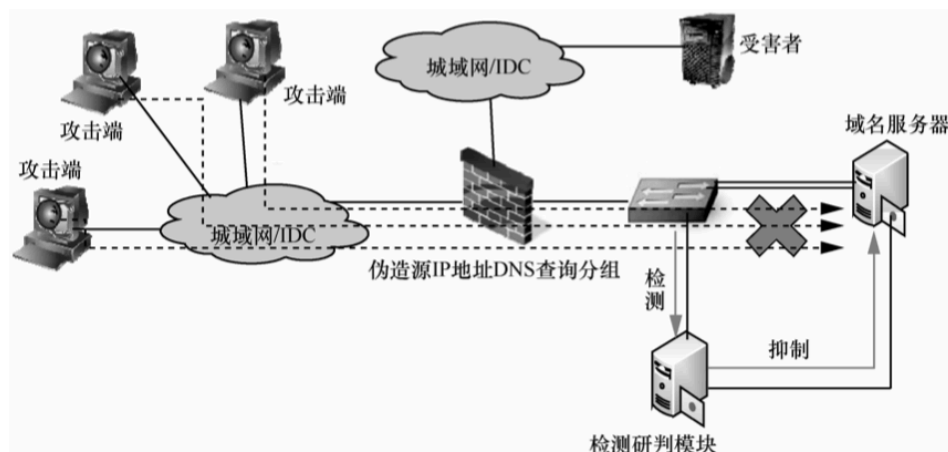


图 9 基于生存时间值智能研判的 DNS 攻击检测技术

Dharma 等^[6]为 SDN 控制器上的 DDoS 检测和缓解提供了基于时间的解决方案。该方法旨在观察高流量，对模式进行聚类并基于阈值进行缓解。

Rudman 和 Irwin^[7]发现源地址和生存时间（TTL）值的变化可通过分析两个大型数据集来检测 NTP 放大攻击。如果来自同一地址的数据包的 TTL 值发生明显变化，则表明源地址是攻击的目标。这是因为攻击主机通常存在于不同的子网中，因此，传输到同一受害者地址的数据包的最终 TTL 值之间必须存在显著差异。

3.2 基于网络流量统计分析和网络流量控制的检测

在基于网络流量统计分析^[8]和网络流量控制的技术^[9]方法中，检测 DNS 拒绝服务攻击之前，需要能够描述 DNS 服务器正常运转情况下的网络流量信息和正常行为分布规律，通过分析 DNS 服务器的正常流量信息特征和正常行为分布规律，建立 DNS 服务器所在网络的正常状态和异常状态模型，通过设定异常状态阈值实现 DNS 恶意流量攻击行为的异常检测。此类检测技术常用的检测特征包括了域名解析失败率发生突然大幅增长、DNS 服务器所在网络流量迅速增大、随机域名出现有规律的均匀分布等

重要检测特征。虽然流量统计分析方法具有一定的检测能力，但是随着攻击流量的隐蔽性增强和攻击形式的层出不穷，流量统计方法的误判率大幅度上升。

K.Kalkan 等^[10]使用统计熵方法检测和预防 SDN 中的 DDoS 攻击。与 ML 算法相比，此方法减少了系统负载。此外，由于采用了统计方法，因此所提出的过程可以检测和缓解未知的攻击类型。K.Kalkan 等，研究了名为 SDNScore^[11]的系统，是一个基于数据包的统计模型，该模型依赖于功能强大的交换机，调查数据包并使用其属性值计算分数。R. Wang 等^[12]，使用统计模型基于合法时期收集的统计特征生成配置文件，并通过将数据包与这些配置文件进行比较来丢弃攻击数据包。Bohatei^[13]等，通过监视虚拟路由器，如果流入受害者的榴莲超过了阈值，则检测 DDoS 攻击。Huistra 等^[14]，观察到通过分析反射器端 DNS 数据包的数量和大小来检测 DNS 放大攻击是可行的。他们通过直接比较 DNS 请求流入的大小和预定义的阈值来确定此流是可疑的还是正常的。此外，他们还比较了 DNS 响应流出的大小和响应数据包的平均大小，并将阈值作为第二指标。

3.3 基于机器学习的检测方案

基于机器学习和支持向量机算法的检测技术^[15]，是大规模、高带宽网络环境下实现对 DNS 网络攻击智能检测的重要手段，系统把服务器收到的正常数据流和 DDoS 攻击流当成一个分类问题进行运算，利用机器学习和支持向量机算法建立模型来进行判断，并对检测结果进行预警。此类技术的优点是利用支持向量机在解决小样本、非线性及高维问题时所具有的良好性能，实现对 DNS 攻击行为的高速检测。但是此类技术存在攻击流检测环境中多个检测器的协作问题，需要结合集成学习、人工免疫等新兴技术来提高检测系统的检测精度和顽健性。

C.-C. Chen 等^[16]提供了一种基于机器学习的 DRDoS 攻击检测。系统使用 SVM 分类器对 SDN 中的 DNS/NTP 数据包进行分类，并阻止他们认为是放大的数据。

Meitei 等^[17]，使用决策树，多层感知器，朴素贝叶斯和支持向量机（SVM）对 DNS 请求流量进行分类，它基于以下几种流量特征，例如来自同一 IP 的数据包到达时间的平均值，数据包的统计信息大小，以及 IP 出现的可能性等。属性选择算法也用于减少冗余功能。

Gao 等^[18]，指出以下特征可以视为 DRDoS 泛洪攻击的攻击指标，例如 IP 数据包的数量，数据包的总大小，所有端口之间的最大数据包数量以及接收到的数据包之

间的差异并由受害者发出。基于上述特征的突然变化，他们使用了基于归一化多项式内核的支持向量机来识别攻击流量。

3.4 基于 DNS 查询历史匹配

Ancy Sherin Jose 和 Binu A^[19]，使用 Hadoop MapReduce 和 Chukwa 自动检测和防止 DNS 反射放大攻击。Chukwa 是 Apache 许可下的 Hadoop 子项目，专门用于分布式日志监视和分析。该检测方法着重于通过使用 chukwa 进行日志收集，然后使用 HDFS 来存储日志并用 Chukwa Demux 完成 Map Reduce 的功能用于检测孤立查询，从而检测 DNS 反射放大攻击，从而通过更新防火墙规则自动防御。由于组织中的任何计算机或节点都容易受到攻击，因此需要监视每台计算机以检测攻击的存在，由于必须监视来自多台计算机的日志或 tcpdump 以进行 DNS 反射放大攻击检测，这超出了单台计算机的计算能力，因此需要 Hadoop。Hadoop 是一个分布式框架，可以存储大型数据并使用 MapReduce 范例处理这些数据。可以将 Chukwa 代理配置为从节点收集日志到 Chukwa Collector，后者可以将这些巨大的日志文件提取到 HDFS，HDFS 是 Hadoop 的文件系统组件。具有 Hadoop 分布式文件系统的集群可用于存储此大型日志数据，而 MapReduce 可用于处理此数据。也就是 Hadoop 将数据的存储和处理划分为多个节点，能够处理少量大文件，完全符合处理 tcpdump / IDS 日志以检测 DNS 放大攻击的要求。

具体来说，该检测方法利用 Chukwa 的 CharFileTailingAdaptorUTF8 来监视受害者的特定文件，并利用此信息来检测攻击。在本文中，Chukwa Agent 与 filetailer。CharFileTailingAdaptorUTF8 模块可用于从必须监视 DNS 放大攻击的工作站收集 Tcpdump 日志。Chukwa 代理将这些日志发送到 Chukwa 收集器，后者将数据存储在 HDFS 中。我们可以对此数据运行 Demux 作业，以检测是否发生了 DNS 放大攻击。本文着重于通过检查要监视的工作站上的单个 DNS 响应来检测 DNS 放大攻击。在进行 DNS 放大攻击时，目标受害者会收到尚未发送 DNS 请求的响应。DNS 请求响应的一对一映射在正常 DNS 操作的情况下发生，而在 DNS 放大攻击的情况下将不存在这种映射。要查找一对一映射，我们可以使用 DNS 响应的查询 ID。如果出现查询 ID 的请求和响应，则这是正常操作。如果仅对 queryId 出现响应，则可能是攻击。检测 DNS 放大攻击的流程图如图 10。

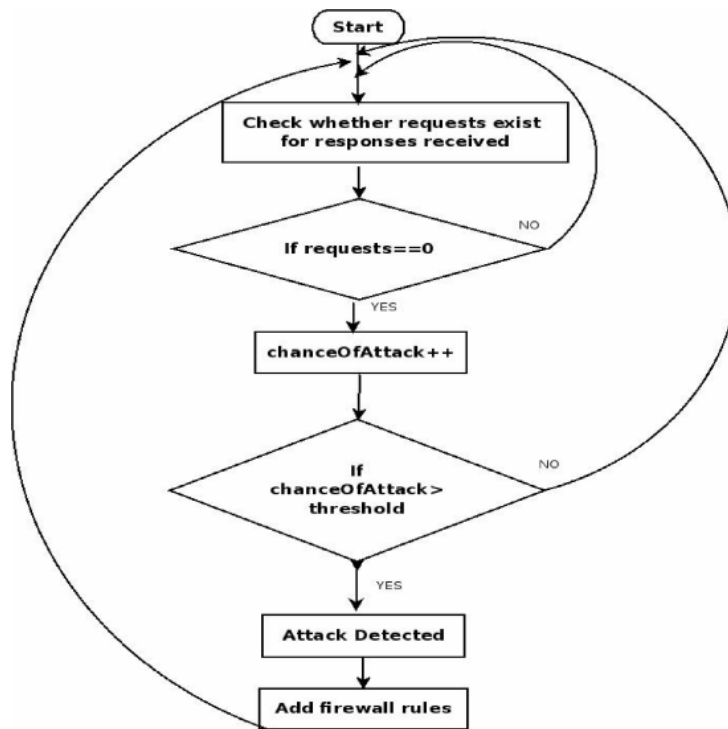


图 10 检测流程

Kambourakis 等^[20]人提出的工作。通过保持 DNS 查询，响应和了解欺骗数据包的一对一匹配，为 DNS 放大提供了一个公平的解决方案。每当可疑数据包到达时，计数器就会增加一，并且当指定的阈值到达时，攻击警报就会起作用。网络中的问题是必须存储很大的数据。

3.5 基于 SDN 的 DNS 反射放大检测和缓解

Kaan "Ozdinc,er 等^[21]，开发了针对 SDN 的 DNS DRDoS 攻击的系统。假定在基于 SDN 的网络中存在一个或多个 DNS 服务器和受害计算机。攻击的检测和缓解将分别进行检查，并且在检测期间将采用两阶段方法，旨在减轻攻击。首先，为了确定 DNS 服务器在网络中的位置，将定期扫描该服务器。为此目的，在此扫描中，只有到端口 53 和来自端口 53 的数据包才被解码到第 7 层，并确定它是否是 DNS 服务器。尽管默认情况下将端口 53 用于 DNS 服务器，但可以选择将其用于其他应用程序。然后分为两阶段，整体流程如图 11 所示。

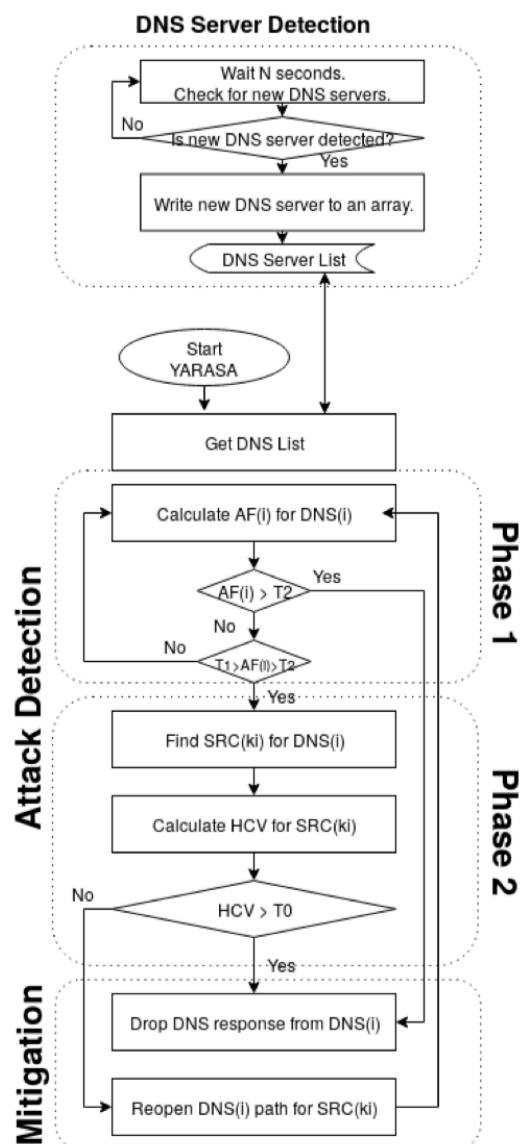


图 11

阶段 1：放大因子监视。通过控制器收集检测到的 DNS 服务器的请求和传出响应的长度，并计算 AF：

$$AF_1 = \text{size}(\text{response}) \div \text{size}(\text{request})$$

$$AF_2 = \text{sumofsize}(\text{response}) \div \text{sumofsize}(\text{request})$$

AF 值增加表示可能正在发生攻击。DNS 放大攻击通常与 ANY 查询一起使用，但是，通过其他查询，例如 DNSKEY，TXT，NS，MX，也会增加 AF 值。因此，增加 AF 并不意味着立即开始攻击，为了消除此歧义问题，针对 AF 检查设置上限阈值和下限阈值，当超过下限阈值时，YARASA 才进入阶段 2，以避免丢弃合法查询，而当超过上限阈值时，将跳过第二阶段，并直接进入缓解措施阶段。

阶段 2，跳数变化监控。因为不同的主机类型，产生的 TTL 初始值，和每一跳减小值都不同，所以可以根据 TTL 值来判断 IP 的真实性。检查 AF 值较高的客户端对服务器的 DNS 请求的 IP 标头中 TTL 值的变化，从而判断此高 AF 值是否是造成攻击的原因。计算 Hop Count 的变化值 HCV，如果 HCV 值高于指定的阈值，则开始缓解阶段。如果不是，则什么也不做，阶段 2 将继续等待阶段 1 的触发。这意味着阶段 1 的 AF 值很高，但这不是 DNS 放大攻击。

Ahmad Ariff Aizuddin 等^[22]，提出的解决方案首先从转发设备收集流数据，然后检查数据包头中的预定义流值，查看流量是否源自 DNS 服务器。sFlow 使聚合流可以通过即时缓存立即导出。如果源端口号不是 53，则将流存储在普通缓存中，否则将其存储在即时缓存中。然后查看 DNS 应用层数据来进一步检查过滤后的 DNS 流量。sFlow 允许用户添加满足检测标准的流特征，比如 DNS 属性。如果流记录中有匹配的请求查询 ID，则将流量分类为正常 DNS 响应，否则，将其标记为可疑。然后将标记的流转发到 SDN 控制器以进行缓解。缓解 DDoS 的另一种适用规则操作是对目标网络上收到的响应进行速率限制。因为它在保持正常 DNS 操作的同时降低了反射放大效果。在 OpenFlow v1.3 中，带来了流量整形的支持，允许 SDN 控制器根据指定的减速参数更改每个流条目中数据包的频率，如图 5，从而使限制操作更有针对性。

3.6 基于行为的 DNS 反射放大检测

Longzhu Cai 等^[23]，提出的解决方案，以 DNS 请求的频率，一个时间段内放大的数据流量的速率（响应流量/请求流量）和一个时间段内增加的数据包数量三个特征为依据，采用 k-means 聚类，提出了一种通过使用模式识别来识别 DNS 反射放大攻击的方法。一个简单的例子说明，如图 12。

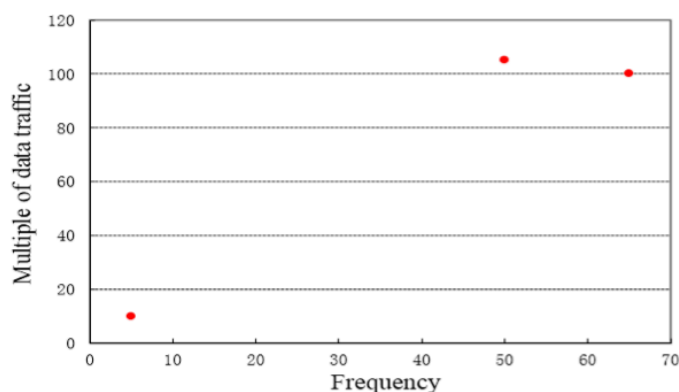


图 12 k-means 示例

X 轴是每单位时间的请求数，Y 轴是响应流量与请求流量的比率，选择这两个特征作为依据进行聚类，左下点是正常组的参考点，右上点是异常组的参考点。对于新数据，我们可以使用聚类方法确定它是正常还是异常，也就是计算数据与每个参考点之间的距离，以确定应将数据分为哪一组。

3.7 基于线性关系的 DNS 反射放大检测

Wei 等^[24]，指出，来自相同反射器的响应流与其他反射器具有线性关系。他们利用由一个路由器发送给受害者的数据包组成的流，通过利用 Spearman 的秩相关系数来区分放大攻击流量和合法流量，来测量流对之间的相关系数。

3.8 新的想法

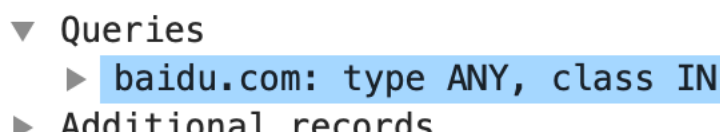
1.在受害者端，基于有状态的防火墙，对于一个 DNS 响应数据包，应该在防火墙中有对应的 DNS 查询记录才认为是合法的，否则可以进行标记为可疑，方便后续工作。

2.利用现有的机器学习和深度学习进行学习识别。

4 DNS 反射放大器的特征

1) any

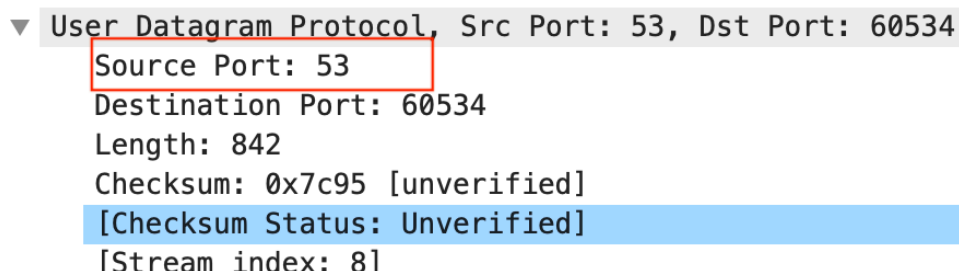
“ANY”类型表示可以响应所有类型，这样响应包才会尽可能的大。



```
▼ Queries
  ► baidu.com: type ANY, class IN
  ► Additional records
```

图 13

2) 53 端口



```
▼ User Datagram Protocol, Src Port: 53, Dst Port: 60534
  Source Port: 53
  Destination Port: 60534
  Length: 842
  Checksum: 0x7c95 [unverified]
  [Checksum Status: Unverified]
  [Stream index: 8]
```

图 14

3) 放大倍数通常较大，比如 50 倍左右：

需要可以控制 DNS 服务，并上传入大的 txt 文本，才能达到这么大的放大倍数。

- 4) 大量的 DNS 响应请求
- 5) 包含一些不存在或者生僻的域名
- 6) 通常经过循环查询从而放大 DNS 流量
- 7) 会将将 OPT RR 字段中的 UDP 报文大小设置为很大的值（如 4096）：

在 dig 命令中加了 “+bufsize=65535” 选项，否则会被截断为 512 字节

- 8) 大量的流量都来自正常的 DNS 服务器

5 探测 DNS 反射放大器

命令：dig +bufsize=65535 +time=1 @DNS_IP any baidu.com

探测脚本：

```
import os
for line in open('/Users/janessa/Desktop/dns_6.txt'):
    line = line.strip('\n')
    c = 'dig +bufsize=65535 +time=1 @'+line+' any baidu.com'
    #print(c)
    os.system(c)
```

图 15

数据包处理程序：

```
# -*- coding: UTF-8 -*-
import dpkt
import socket
import csv

with open("dnsdetection.csv", "a+") as csvfile:
    writer = csv.writer(csvfile)
    #writer.writerow(["client ip", "DNSServer ip", "query len", "response len", "multiple"])

f = open('/Users/janessa/Desktop/DNSDetection_6.pcap', 'rb')
pcap = dpkt.pcap.Reader(f)
num=0
for ts, buf in pcap:
    eth = dpkt.ethernet.Ethernet(buf)
    ip = eth.data
    udp = ip.data
    if(num%2==0):
        dns_ip = socket.inet_ntoa(ip.dst)
        client_ip = socket.inet_ntoa(ip.src)
        query_len = ip.len
        line = [client_ip, dns_ip, query_len]
    else:
        response_len = ip.len
        multiple = response_len/line[2]
        line.append(response_len)
        line.append(multiple)
        print(line)
        writer.writerow(line)
    num=num+1
f.close()
with open("dnsdetection.csv", "a...
```

图 16

探测处理结果如表 1，详见文件 “./探测包/dnsdetection.csv/”，client_ip 表示探测主机 ip，DNSServer_ip 是被探测的 DNS 服务器 ip,query_len 是查询包大小，response_len 是响应包大小，multiple 为放大倍数。

共探测了 668 个 DNS 服务器，其中只有 126 个有响应，具体的放大倍数见表的最后一列”multiple”。

表 1 DNS 反射放大器探测

client_ip	DNSServer_ip	query_len	response_len	multiple
192.168.0.100	8.8.8.8	66	526	7.96969697
192.168.0.100	208.67.222.222	66	343	5.1969697
192.168.0.100	208.67.220.220	66	343	5.1969697
192.168.0.100	208.67.222.123	66	343	5.1969697
192.168.0.100	208.67.220.123	66	343	5.1969697
192.168.0.100	216.146.35.35	66	98	1.48484849
192.168.0.100	216.146.36.36	66	98	1.48484849
192.168.0.100	180.76.76.76	66	416	6.3030303
192.168.0.100	223.5.5.5	66	177	2.68181818
192.168.0.100	223.6.6.6	66	177	2.68181818
192.168.0.100	114.114.114.114	66	526	7.96969697
192.168.0.100	114.114.115.115	66	526	7.96969697
192.168.0.100	114.114.114.119	66	526	7.96969697
192.168.0.100	114.114.115.119	66	526	7.96969697
192.168.0.100	114.114.114.110	66	515	7.8030303
192.168.0.100	114.114.115.110	66	476	7.21212121
192.168.0.100	101.226.4.6	66	98	1.48484849
192.168.0.100	218.30.118.6	66	98	1.48484849
192.168.0.100	168.95.192.1	66	66	1
192.168.0.100	168.95.192.1	92	214	2.32608696
192.168.0.100	168.95.1.1	66	66	1
192.168.0.100	168.95.1.1	92	214	2.32608696
192.168.0.100	208.67.222.222	66	343	5.1969697
192.168.0.100	208.67.220.220	66	343	5.1969697
192.168.0.100	165.87.13.129	66	66	1
.....

6 数据集

1. dns-amp-internet2.txt

来源： <http://blog.huque.com/2013/04/dns-amplification-attacks.html>

时间：2013-04-10

描述：在根域上的 DNS 反射放大 ANY 查询，按响应放大率（最后一列）的降序排序，其中放大率仅是 DNS 响应与请求有效负载之比，不包含封装的 UDP / IP / L2 等标头。

每一行行格式：

zone: 根域名称

ns_name: 域名

ns_address: 服务器地址

query_size: DNS 查询包大小

response_size: DNS 响应包大小

amp1: DNS 响应有效负载与 DNS 查询有效负载之比

amp2: 整个响应数据包与请求数据包的估计比率

示例：

```
ksu.edu nic.kanren.net. 164.113.192.242 36.0 4085.0 113.47 53.99
ksu.edu kic.kanren.net. 164.113.92.250 36.0 4085.0 113.47 53.99
umbc.edu UMBC3.umbc.edu. 130.85.1.3 37.0 4093.0 110.62 53.41
lsu.edu phloem.uoregon.edu. 128.223.32.35 36.0 4016.0 111.56 53.10
lsu.edu bigdog.lsu.edu. 192.16.176.1 36.0 4016.0 111.56 53.10
umbc.edu UMBC5.umbc.edu. 130.85.1.5 37.0 4045.0 109.32 52.80
umbc.edu UMBC4.umbc.edu. 130.85.1.4 37.0 4045.0 109.32 52.80
sdsmt.edu ns5.gratisdns.dk. 85.17.221.46 38.0 4095.0 107.76 52.76
sdsmt.edu ns4.gratisdns.dk. 87.73.3.3 38.0 4095.0 107.76 52.76
sdsmt.edu ns2.gratisdns.dk. 208.43.238.42 38.0 4095.0 107.76 52.76
sdsmt.edu ns1.gratisdns.dk. 109.238.48.13 38.0 4095.0 107.76 52.76
```

图 17

根域: ksu.edu

域名: nic.kanren.net

DNS 服务器 ip: 164.113.192.242

36.0: 查询包大小为 36 字节

4085.0: 响应包大小为 4085 字节

113.47: DNS 响应有效负载是 DNS 查询有效负载的 113.47 倍

53.99: 整个响应数据包的大小是请求数据包的 53.99 倍

2. impact 数据集

来源: <https://ant.isi.edu/datasets/dns/>（没有下载下来，注册信息不足）

DITL_B_Root-20170411

DITL_B_Root-20130528

YYYY 是年份（2017）、MM 是月（04）、DD 是每月的某天、HH 是小时（00-23）、MM 是分钟（00-59）、SS 是秒（00-59）、NNNNNNNN 是一个序列号、NNNNNNNNNN 是 Broot 的任播站点。

描述: 是使用 CICFlowMeter-V3 收集并提取了 80 多种流量特征, 包含了 FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort 和协议, 以及如例如持续时间, 数据包数量, 字节数, 分组的长度等流量特征, 示例:

```
Unnamed: 0,Flow ID, Source IP, Source Port, Destination IP, Destination Port, Protocol, Timestamp, Flow Duration, Total Fwd Packets, Total Backward Packets>Total Length of Fwd Packets,  
Total Length of Bwd Packets, Pkts Count Length Max,Fwd Packet Length Min, Fwd Packet Length Mean, Fwd Packet Length Std,Bwd Packet Length Max, Bwd Packet Length  
Mean, Bwd Packet Length Std,Flow Bytes/s, Flow Packets/s, Flow IAT Mean, Flow IAT Std, Flow IAT Max, Flow IAT Min, Fwd IAT Total, Fwd IAT Mean, Fwd IAT Std, Fwd IAT Max, Fwd IAT Min,Bwd  
IAT Total, Bwd IAT Mean, Bwd IAT Std, Bwd IAT Max, Bwd IAT Min,Fwd PSH Flags, Bwd PSH Flags, Fwd URG Flags, Bwd URG Flags, Fwd Header Length, Bwd Header Length,Fwd Packets/s, Bwd  
Packets/s, Min Packet Length, Max Packet Length, Packet Length Mean, Packet Length Std, Packet Length Variance,FIN Flag Count, SYN Flag Count, RST Flag Count, PSN Flag count, ACK Flag  
Count,Urg Flag Count, CQE Flag Count, ECE Flag Count, DownUp Ratio, Average Packet Size, Avg Fw Segment Size, Avg Bwd Segment Size, Fwd Header Length 1,Fwd Avg Bytes/Bulk, Fwd Avg  
Packets/Bulk, Fwd Avg Bulk Rate, Bwd Avg Bytes/Bulk, Bwd Avg Packets/Bulk,Bwd Avg Bulk Rate,Subflow Fwd Packets, Subflow Fwd Bytes, Subflow Bwd Packets, Subflow Bwd Bytes,  
Init_Win_bytes_forward, Init_win_bytes_backward, act_data_pkt_fwd, min_seq_size_forward,Active Mean, Active Std, Active Max, Active Min,Idle Mean, Idle Std, Idle Max, Idle Min,  
SimilarHTTP_Inbound_Label
```

```
15798,172.16.0.5-192.168.50.4-9401-15931-17,172.16.0.5,9401,192.168.50.4,15931,17,2018-11-03 10:42:57,176671,1,2,0,2560,0,0,1280,0,1280,0,0,0,0,0,0,0,0,0,0,2.56E9,2000000,0.1,0,  
\<0,0,1.0,1.0,1.0,1.0,0,0,1.0,0,0,0,0,0,0,0,0,0,0,-2,0,2000000,0,0,1280,0,1280,0,1280,0,0,0,0,0,0,0,0,0,0,0,0,1920,0,1280,0,0,-2,0,0,0,0,2,2560,0,0,-1,-1,-1,-1,0,>  
<,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,MSSQL  
  
110891,172.16.0.5-192.168.50.4-9402-29997-17,172.16.0.5,9402,192.168.50.4,29997,17,2018-11-03 10:42:57,176673,0,2,0,816,0,0,408,0,408,0,408,0,0,0,0,0,0,0,Infinity,Infinity,0,0,0,  
.0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-2,0,0,0,0,408,0,408,0,408,0,0,0,0,0,612,0,408,0,0,-2,0,0,0,0,2,816,0,0,-1,-1,-1,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
MSSQL  
  
66956,172.16.0.5-192.168.50.4-9403-29887-17,172.16.0.5,9403,192.168.50.4,29887,17,2018-11-03 10:42:57,176727,1,2,0,810,0,0,405,0,405,0,405,0,0,0,0,0,0,0,0,InfB,2000000,0.1,0,0,1,
```

详细说明见：<https://www.unb.ca/cic/datasets/ddos-2019.html>、<https://www.unb.ca/cic/research/applications.html#CICFlowMeter>

参考文献

- [1] Luo Z Q, Shen J, Jin H M. Detection and control technology of distributed DNS reflective DDoS attack. Telecommunications Science, 2015270
- [2] 【ZoomEye 专题报告】DDoS 反射放大攻击全球探测分析[E].
<https://paper.seebug.org/898/#31chargen>.2019.04.24
- [3] 章思宇, 姜开达. DNS 拒绝服务攻击与对策[E].
https://www.edu.cn/xxh/fei/zxz/201503/t20150305_1235269.shtml. 2015-03-05
- [4] imperva. DDoS Protection for DNS. <https://www.imperva.com/products/dns-ddos-protection-services/>.
- [5] GuXQ, WangHY, NiTG, et al. Detection of application-layer DDoS attack based on time series analysis. Journal of Computer Applications, 2013, 33 (8) : 2228~2231
- [6] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM, 13(7): 422–426, 1970.
- [7] L. Rudman, B. Irwin. Characterization and analysis of NTP amplification based DDoS attacks Proceedings of Information Security for South Africa (2015), pp. 1-5
- [8] Luo ZQ, Shi GS, Shen J, et al. Research on mobile Internet security technology. Telecommunications Science, 2013 (Z1) : 254~256, 261
- [9] Tang H, Luo ZQ, Shen J. Research and application of the active defense technology of DDoS attack in botnet. Telecommunications Technology, 2014 (11) : 76~80
- [10] K. Kalkan, L. Altay, G. G"ur, and F. Alag"oz, "Jess: Joint entropybased ddos defense scheme in sdn," IEEE Journal on Selected Areas in Communications, vol. 36, no. 10, pp. 2358–2372, 2018.
- [11] K. Kalkan, G. G"ur, and F. Alag"oz, "Sdnscore: A statistical defense mechanism against ddos attacks in sdn environment," in 2017 IEEE Symposium on Computers and Communications (ISCC). IEEE, 2017, pp. 669–675.
- [12] R. Wang, Z. Jia, and L. Ju, "An entropy-based distributed ddos detection mechanism in software-defined networking," in 2015 IEEE Trustcom/BigDataSE/ISPA, vol. 1. IEEE, 2015, pp. 310–317.
- [13] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and elastic ddos defense," in 24th {USENIX} Security Symposium ({USENIX} Security 15), 2015, pp. 817–832.

- [14] D. Huistra Detecting reflection attacks in DNS flows Proceedings of Twente Student Conference on IT (2013)
- [15] Xu T, Luo Y, He D K. Detecting DDoS attack based on multi-class SVM. Journal of University of Electronic Science and Technology of China, 2008, 37 (2) : 274~277
- [16] C.-C. Chen, Y.-R. Chen, W.-C. Lu, S.-C. Tsai, and M.-C. Yang, “Detecting amplification attacks with software defined networking,” in 2017 IEEE conference on dependable and secure computing. IEEE, 2017, pp. 195–201.
- [17] IL Meitei , KJ 辛格, T. 德分类算法检测 DDoS DNS 放大攻击国际信息学和分析学会议论文集, 2016
- [18] Y.X. Gao, Y.K. Feng, J. Kawamoto, K. Sakurai A machine learning based approach for detecting DRDoS attacks and its performance evaluation Proceedings of Asia Joint Conference on Information Security (2016), pp. 80-86
- [19] Ancy Sherin Jose, Binu A. Automatic Detection and Rectification of DNS Reflection Amplification Attacks with Hadoop MapReduce and Chukwa.IEEE, 2014
- [20] G. Kambourakis, T. Moschos, D. Geneiatakis, and S. Gritzalis, “A fair solution to dns amplification attacks,” in Second International Workshop on Digital Forensics and Incident Analysis (WDFIA 2007). IEEE, 2007, pp. 38–47. G.
- [21] Kaan “Ozdinc,er, etc. SDN-based Detection and Mitigation System for DNS Amplification Attacks.IEEE.2019
- [22] Ahmad Ariff Aizuddin, etc. DNS Amplification Attack Detection and Mitigation via sFlow with Security-Centric SDN. IMCOM. 2017
- [23] Longzhu Cai, etc.A Behavior-based Method for Detecting DNS Amplification Attacks.IEEE.2016
- [24] W. Wei, F. Chen, Y.J. Xia, G. Jin A rank correlation based detection against distributed reflection DoS attacks IEEE Commun. Lett., 17 (1) (2013), pp. 173-175