DNS 缓存中毒攻击与防范

绿盟科技开发中心　郭大兴　周向荣

摘要：DNS 简及 DNS 缓存中毒攻击分析与防范。

关键词：DNS　缓存中毒；攻击；防护

0 前言

为了在网络上标识一个实体，TCP/IP 协议使用了 IP 地址，由于 IP 地址难于记忆，需要一种容易记忆的方法，于是就产生了一种名字到地址或地址到名字的映射机制。在 Internet初期,这种映射可以使用一个主机文件来保存,文件只需要包括名字和地址这两列，当程序或用户想把名字映射为地址时，去查找这个主机文件可以了。但是网络迅速的发展，已经不可能再继续使用主机文件来保存这种映射关系，因为主机文件会太大而无法存储所有信息，而且，每当出现变化时，也必须对全世界的所有主机文件都进行更新，使用一个文件的方式维护 IP 地址与名称之间的转换将不再适合，于是 DNS 就这样诞生了。

DNS 是用于管理主机名称和地址信息映射的分布式数据库系统，将这个巨大的映射信息划分成许多较小的部分，并把每一部分存储在不同的计算机上，需要映射的计算机可以查询一个最近的持有所需要信息的计算机。DNS 目前已经成为大部分网络应用的基础了。一旦遭受攻击，用户将不能进行正常的网络访问，因此 DNS 的安全影响巨大。

在 DNS 攻击中危害比较大的当属 DNS 缓存中毒。接下来我们将详细介绍缓存中毒的攻击原理和过程,以及如何防护。

1　DNS 缓存中毒攻击原理

在介绍攻击原理和过程之前首先来介绍一下 DNS 工作的过程。

DNS 被设计成客户 - 服务器应用程序，需要把地址映射为名字或把名字映射为地址的主机需要调用 DNS 解析程序，向最近的 DNS 服务器发出查询请求，此时，DNS 服务器可以有两种方式来响应客户的请求，一种叫递归解析，另一种叫迭代解析。DNS 缓存中毒攻击主要是针对递归解析方式工作并缓存非本域的解析结果的 DNS 服务器。下面就主要介绍下递归解析过程。

DNS 递归解析过程如图 1 所示。

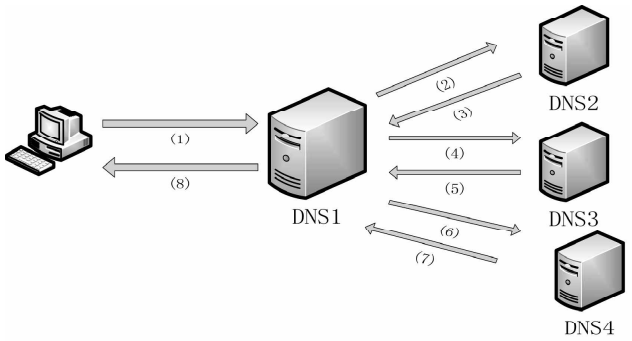

图 1　DNS 递归解析示意图

（1）应用程序需要解析一个域名时，会向本机上的 DNS 客户端 -- 解析程序发起域名解析的请求，解析程序收到应用程序的请求后会代表应用程序向首选 DNS 服务器 DNS1 发起域名解析请求；

（2）DNS1 收到解析请求后会去查询自己的管辖域，如果请求的是自己管辖域的域名就会直接将查询结果返回给解析程序，进而传递给应用程序，如果不在 DNS1 的管辖域内，DNS1 就向它的上级服务器 DNS2 发起解析请求，等待 DNS2 的查询结果；

（3）如果 DNS2 解析不了，就会告诉 DNS1，我解析不了，但是我知道 DNS3 可能会知道，你去问问 DNS3 看看；

（4）DNS1 就会给 DNS3 发出解析请求，等到 DNS3 的查询结果；

（5）如果 DNS3 可以解析，那么就告诉 DNS1 解析结果，如果 DNS3 也解析不了，那么 DNS3 也会像 DNS2 那样，告诉 DNS 它所知道的某个 DNS 可能会知道；

（6）最后，DNS1 又向 DNS4 发出查询请求，等待查询结果；

（7）DNS4 可以解析该域名，就发送一个相应包给 DNS1 解析结果；

（8）DNS1 收到解析结果后就会将结果发送给客户端的解析程序，并将解析结果保存在自己的缓存中，以便以后再有

请求解析该域名时，就可以直接从缓存中得到解析的结果，提高效率。至此，整个解析过程就结束了。

在DNS1发出的查询请求中会包含一个TID，用来表示某个查询，在收到的响应数据包中也同样会包括TID，表示是对某个TID查询请求的结果响应，DNS1会根据TID来判断是它发出的某个请求的响应。

DNS服务器不是收到的任何DNS响应数据包都会接受的，它会解码数据包，检查数据包是否符合接受的标准，那DNS服务器检查数据包中的哪些数据来判断该数据包是不是它所希望的包呢？

（1）响应数据包中目的端口

在DNS服务器发出的查询请求包中包含了它的UDP端口号，对方收到该数据包后返回的响应数据包中的目的端口号就应该为DNS服务器发出包的源UDP端口号，如果收到数据包中的目的端口号不对，就说明不是对DNS服务器发出的请求的响应，因此，网络协议栈就会直接丢弃该数据包，而不会将数据包传递到DNS服务器。

（2）响应数据包中的问题域

当给某个查询请求数据包发回响应数据包时，DNS服务器会拷贝请求数据包中的问题域，因此DNS服务器收到响应数据包后就可以通过检查问题域是否与发出的查询包中的问题域是否一致来判断响应数据包的合法性。举个例子，DNS1向DNS2发出问题域是www.nsfocus.com的请求数据包，询问www.nsfocus.com的IP地址,DNS1不能将问题域为www.baidu.com的响应数据包中的地址做为查询域名为www.nsfocus.com的地址的回答。当出现这样的情况服务器就会丢弃该数据包。

（3）响应数据包中的查询ID(TID)

DNS服务器没发出一个查询请求都会分配对应的一个查询ID号，在内部对应是某一个域名查询请求，在DNS服务器内部区分不同的查询就要TID号，如果DNS服务器收到的响应数据包中的TID号不在DNS服务器等待响应数据的TID中，就表示DNS服务器收到的响应数据包不是回答自己发出的查询请求的，服务器就会丢弃该数据包。

（4）响应数据包中的授权域和附加域

授权域和附加域中的域名必须和问题域中的域名是同属于某个域名下的子域名。

如果上述所有条件都满足了，DNS域名服务器就会接受该响应数据包是对它发出的某个查询请求的响应，使用相应数据包中的数据，并缓存结果。

如果某个攻击者可以预测和伪造响应数据包，他就可以对DNS服务器发起缓存中毒攻击了。由于DNS服务器发送的数据包中的TID只有16位，而且大部分的DNS实现发出的请求中的源端口都是固定不变的。因此，攻击者就可以伪造DNS响应包，DNS服务器在收到响应包后就会将解析结果保存在自己的缓存中，进而实现了缓存中毒攻击。

值得注意的是当DNS服务器收到了不是它所希望的的数据包时，它就会简单的丢弃数据包，因此，攻击者就不必每次都需要猜对所有的数据，可以发送许多包猜测一些关键的参数。

由于DNS服务器发出的请求包中的源端口都是固定的，因此，攻击者就可以向攻击目标发送一个精心构造的域名请求，通过上述的递归解析，最终攻击目标就会向攻击者控制的DNS服务器发出查询请求，此时，攻击者就可以通过抓包来得到攻击目标发出查询请求时所用的UDP端口是多少。接下来就是猜测目标DNS服务器查询包中的TID了。由于一些DNS实现使用了简单的TID生成算法，如，简单的加一，这样就可以预测DNS服务器下一个查询请求会使用的TID了，攻击者就可以向目标服务器发送大量的带预测TID的请求响应包来命中目标服务器的查询请求，在目标服务器接收了伪造的响应包后，就会将伪造包中的响应数据存储在自己的缓存中，到此DNS缓存中毒就算成功了。

2 DNS缓存中毒的防护

及时检查自己的DNS服务器是否存在DNS缓存中毒漏洞，如果发现了自己的DNS服务器存在该漏洞就可以在被攻击前采取措施修补，避免攻击事件的发生。

如何检查DNS服务器是否存在DNS缓存中毒漏洞的？

目前检查DNS服务器是否存在缓存中毒漏洞的主要方法是模拟攻击过程中的一部分，对客户的DNS服务器不会造成任何损害。检查的具体过程是首先需要搭建一个自己可以控制DNS服务器，与扫描器协同工作，让扫描器向要检查的DNS服务器查询只有控制的DNS服务器才能解析的域名请求，被检查的DNS服务器需要解析该域名就会向别的DNS服务器请求解析，根据前文介绍的DNS服务器解析原理，最终被检查的服务器发出的查询请求可以被导向到我们控制的DNS服务器，这样，控制的服务器就可收到检查服务器的请求，进而可以收集到检查服务器发出请求包的特征，被控服务器就可以将收集到的特征信息作为查询的响应数据返回给检查的DNS服务器，进而可以返回给扫描器，这样扫描器就可以根据收集的特征来判断检查的DNS服务器是否存在缓存中毒漏洞。

绿盟科技的远程安全评估系统使用了类似技术进行相关

漏洞的扫描并提供了解决方案。

详细解决方案可以参考绿盟科技漏洞库中相关描述 :http://www.nsfocus.net/index.php?act=sec_bug&do=view&bug_id=12124

技术层面的防护措施如下：

（1）UDP 源端口随机化选择是一种比较有用的防护举措，不应该再是默认的 53，而是应该在 UDP 端口范围内随机选择，使得投毒命中的概率极大的降低。

（2）加密所有对外的数据流，对服务器来说就是尽量使用 SSH 之类的有加密支持的协议，对一般用户应该用 PGP 之类的软件加密所有发到网络上的数据。

（3）禁用 DNS 缓存，在注册表中找到对应的键值，进行修改并保存。

（4）及时刷新 DNS，DNS 缓存会被重建，再次用域名访问 IP 服务器，故障消失。

（5）修改 TTL 值可以有效防止缓存中毒，但较小的 TTL 值会增加服务器的负担，应根据服务器的性能和网络情况合理地选择。

（6）采用 DNSSEC 机制，依靠公钥技术对于包含在 DNS 中的信息创建密码签名。密码签名通过计算出一个密码 hash 数来提供 DNS 中数据的完整性，并将该 hash 数封装进行保护。私/公钥对中的私钥用来封装hash数，然后可以用公钥把hash数译出来。如果这个译出的hash值匹配接收者刚刚计算出来的hash数，那么表明数据是完整的。

3 总结

最近几次大规模的 DNS 攻击事件，充分暴露了 DNS 协议的安全隐患。DNS 的安全脆弱性和 DNSSEC 强大的优越性促使我们要抓紧全面实施 DNSSEC，但是 DNSSEC 还不能完全解决 DNS 安全问题，而且很多技术问题还在研究过程中，所以在彻底实施 DNSSEC 之前要考虑别的有效防范措施！不过，有理由相信有了不断完善的 DNSSEC 技术 DNS 安全问题将有望获得全面解决。目前我们的扫描器已经实现了对查询包中端口固定的 DNS 服务器的缓存中毒扫描的功能。

参考文献

[1]W.Richard Stevens.[M] TCP/IP协议详解.2006.

[2]http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html#poisoning.2008.

[上接 54 页]

者Observer则提供一个Update操作，注意这里的CBaseChart的Update操作并不在CBaseChart改变了CChartModel目标状态的时候就对自己进行更新，这个更新操作要延迟到CChartModel对象发出Notify通知所有CBaseChart进行修改(调用Update)。CChartModel维护一个list作为存储其所有观察者的容器。每当调用Notify操作就遍历list中的Observer对象，并广播通知改变状态(调用CBaseChart的Update操作)。目标的状态 state 可以由 CChartModel 自己改变，也可以由CBaseChart的某个操作引起state的改变(可调用CChartModel的SetState操作)。Notify操作可以CChartModel主动广播，也可以由观察者来调用(因为 CBaseChart 维护一个指向CChartModel的指针)。Observer模式实现示意图见图5。

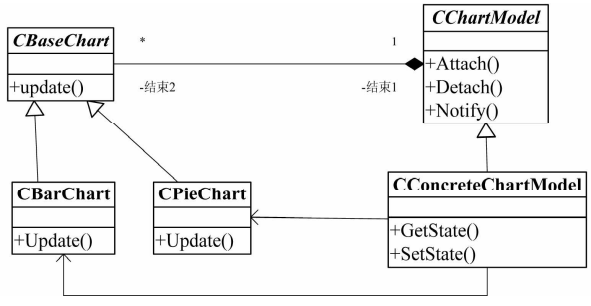

图5 Observer 模式实现示意图

通过上述三种设计模式的应用，解决了通用数据图控件在软件整体架构、访问接口等主要问题，使得该控件在结构上体现出一定的灵活性，以适应可能的变化和需求；同时将软件内部的复杂性封装起来，为用户提供一个简单而稳定的访问接口，从而在实现功能的同时，提高了软件的重用性。下面给出通用数据图控件的两个具体应用。

3 结论

本文归纳了三个在开发中涉及到的软件设计模式，根据各种设计模式适用性不同及各自特点，将设计模式的思想灵活地应用到通用数据图控件的设计和开发中，在满足需求的同时，使软件具有良好的结构，并可以增强软件的兼容性、稳定性和可扩充性。

参考文献

[1]Erich Gamma,李英军.设计模式-可复用面向对象软件的基础.北京:机械工业出版社.2000.

[2]张邵兰.MVC设计模式研究.山东建筑工程学院学报.2004.