Explot DNSSEC

发掘利用DNSSEC的潜在攻击。

# Large Answer

## Intro.

RRSIG通过（Owner name, Class, Type）来标识所覆盖的资源记录(Resource Record, RR)；而具有相同(Owner name, Class, Type)的RR被称为资源记录集合（Resource Record Set, RRSET)。

RRSIG对应着一个RRSET，解析器通过验证RRSIG从而确保RRSET的数据完整性。

RRSIG记录中包含着对RRSET的签名，其签名signature的计算方式如下：

Signature = sign(RRSIG\_RDATA | RR(1) | RR(2)...)  
 RR(i) = owner | type | class | TTL | RDATA lenth | RDATA

RR(i)中的RDATA以canonical form表示

Canonical form: (Modified wire format)

Domain name fully expanded, fully qualified.

All uppercase to lowercase.

RRSET中的RR以canonical order排列

Canonical Order:

Treating RDATA as octets sequence and compared.

解析器在验证RRSET前，首先需要验证其RRSIG的有效性，并随后重建其所签名的内容，即构建(RRSIG\_RDATA | RR(1) | RR(2)...))，并在此之后，利用对应RRSIG记录中的签名和DNSKEY记录中的公钥进行签名校验。

可以将解析器的操作分为以下几步：

1. 明文重建
2. 明文摘要计算
3. 签名校验

往RRSET中加入大量RR，或增加每个RR的RDATA长度的方式，可以大幅增加签名内容的长度，并从而增加解析器的计算消耗。

通过增加明文，可以增大解析器在第一步1.明文重建及第二步2.明文摘要计算的资源消耗，从而起到潜在的攻击效果。

## Test

为了测试增加明文长度是否会对解析器处理时间造成显著影响，采用控制变量法，将无效的多余碰撞密钥删除，确保RRSIG验证仅需一次即可成功。

Large Answer通过增加明文长度，扩大解析器在明文重建和明文摘要计算两个操作上的资源消耗来进行攻击。

根据RFC4035的描述，解析器会从RRSIG开始遍历，确定该RRSIG对应的DNSKEY，并随后进行明文重建的验证操作，由此每一个RRSIG都会触发一次明文重建和明文摘要计算过程。

不难想到Large Answer可以与KeyTrap中的SigJam结合，从而进一步扩大计算资源消耗。

在本次测试中将关注于Large Answer的资源消耗能力，而暂不考虑进一步结合的效果。

### One Answer

首先对正常情况，即回复中仅仅包含一条Answer进行测试。

对one.keytrap.test发送解析请求，该域名仅包含一条A记录。

进行五次独立测试，每次测试后均重启解析器，已清除缓存的影响，每次测试的处理时间如下(单位msec)：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试次数 | 1 | 2 | 3 | 4 | 5 | Avg. |
| 耗时(msec) | 1.165 | 1.297 | 1.239 | 1.190 | 1.288 | 1.2358 |

### 256 Answer

对answer.keytrap.test发送解析请求，该域名包含256条A记录（A记录由编写的Python脚本generate\_A\_record.py批量生成）。

测试方法与1.2.1节相同，进行五次测试，256条记录全被回复，测试结果如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试次数 | 1 | 2 | 3 | 4 | 5 | Avg. |
| 耗时(msec) | 1.976 | 2.226 | 2.047 | 2.994 | 2.446 | 2.3378 |

可以看到Large Answer确实能明显增加解析器的处理时间，测试结果显示Large Answer的处理耗时约是正常请求的189.2%( 2.3378 / 1.2358 \* 100% )，耗时时扩大近一倍。

### 1024 Answer

对happy.keytrap.test发送解析请求，该域名包含1024条A记录。

所有1024条记录均被正常回复，五次测试结果如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试次数 | 1 | 2 | 3 | 4 | 5 | Avg. |
| 耗时(msec) | 4.194 | 4.332 | 4.233 | 4.216 | 4.166 | 4.2282 |

将A记录数量设置为1024条后，解析器的处理耗时进一步增加，处理耗时约是正常请求的342.1%，耗时扩大约2.4倍。

### Limitations?

#### 65536 (256 \* 256) Answer

运行dnssec-signzone对区域文件签名，持续运行数秒后报错：

无法对区域文件进行签名：（ran out of space）

aspera@TochusCRevo:~/Sign/keys$ dnssec-signzone -S -K . -o keytrap.test. keytrap.test.zone

dnssec-signzone: error: dns\_master\_load: keytrap.test.zone:65549: wow.keytrap.test: ran out of space

dnssec-signzone: fatal: failed loading zone from 'keytrap.test.zone': ran out of space

后续实验证明即时签名成果，对应的回复也会因数据包过大而被过滤剔除。

#### 16384 (64 \* 256) Answer

签名成功。

对域名okay.keytrap.test发送解析请求，该域名包含16384（64\*256）条A记录。

返回结果SERVFAIL。

**观察Unbound日志输出：**

Unbound从BIND9名称服务器中收到的回复为空（Non-answer），Validator模组无法对Non-answer回复进行验证，于是报错返回SERVFAIL。

Unbound继续进行了五次尝试后终止查询，向客户端返回了SERVFAIL结果，查询耗时14.589msec。

**观察BIND9名称服务器日志输出：**

未发现任何记录。

**对区域文件中的其他域名进行测试：**

发送对www.keytrap.test的解析请求，请求被正常处理并返回对应A记录。

发送对what.keytrap.test的解析请求，请求被正常处理并返回NXDomain。

由此可推断配置文件被BIND9正常载入，名称服务器正常工作中。

猜测域名okay.keytrap.test的回复数据包因为过大，而在某个环节中被过滤抛弃了。

#### 4096（16 \* 256）Answer

对compromise.keytrap.test发送解析请求，该域名包含4096条A记录。

返回SERVFAIL。

由此推测回复中所能加入的A记录最大值位于(1024, 4096)区间内。

#### TEXT RR

根据RFC4034所描述内容来看，使用TEXT 记录和A记录所能达到的效果应并无区别。TEXT类型的RDATA大小可以大大超过A记录，但据测试结果来看，目前并没有对返回记录数目的相关限制，而涉及到的是对数据包大小的限制。

使用大量A记录与使用数量相对较少的TEXT记录所产生效果应该类似，同时它们也均收到解析器对数据包大小的限制。

### Further Exploration

通过为RRSET设置多个签名，每个签名将会导致解析器重新进行1.明文重建，2.明文摘要计算两步，从而多次触发Large Answer的效果，能进一步对计算资源消耗效果进行增幅，因为Unbound对验证失败次数限制为8次，Large Answer在Unbound实现中可以再被增幅1~9x。

# Large NSEC（NXDOMAIN）

## Intro.

为了避免DNS缓存污染，DNSSEC使用NSEC记录按照Canonical Name Order将所有权威下的所有名称（Owner name）串联起来。

每个NSEC记录都有着对应的RRSIG签名，DNSSEC要求解析器需要通过RRSIG签名验证NSEC记录的有效性，当区域中包含大量的NSEC记录时，解析器理应需要对NSEC链条上每一个记录进行一次验证，产生大量的RRSIG验证操作，进而消耗解析器资源。

## Test

在使用dnssec-signzone对区域配置文件进行签名时，会自动生成每个名称对应的NSEC记录，因此通过往区域配置文件中加入大量不同名称的记录，即可生成大量的对应NSEC记录，构成较长的NSEC链条。

在测试中所有RRSIG都由同一个正确密钥生成，确保每次实验中每个RRSIG验证次数均为一次。

### 64NSEC

向区域配置文件中加入64个名称，分别从name0001, name0002，......，至name0063，每个名称均有1个A记录。

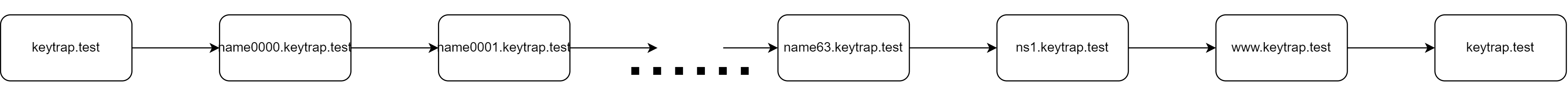
发送对域名name0064.keytrap.test的解析请求。

请求返回NXDomain，耗时1.408msec，想象中的请求耗时大幅增加情况并未出现。

观察Unbound日志输出可以发现：

BIND9名称服务器直接返回了名称为name0063的NSEC记录，该NSEC记录中表明下一个名称为ns1，而并不是name0064，从而间接告诉了解析器name0064并不存在。

观察BIND9区域配置文件可以发现配置中的NSEC链条如下图所示：



根据Unbound日志输出可以得出结论：对NSEC链条的遍历过程在BIND9名称服务器中完成，而并不是在解析器侧，因此Unbound并不会对链条中的每个NSEC记录进行RRSIG验证，而仅需验证所接收到的NSEC记录。

能否或如何通过Large NSEC来扩大DNSSEC验证计算仍需进一步考虑。

# Large NSEC3 + High Iteration

(Suspend)

## Intro.

NSEC3为了解决NSEC存在的两个问题而产生：

1. Zone walk
2. High cost of resigning.

第一个问题源于NSEC记录中区域的所有名称均以明文形式显示，攻击者可以通过遍历NSEC记录来获取到区域中的所有名称。

第二个问题则源于DNSSEC要求区域中所有名称必须拥有NSEC记录，因此区域配置中每加入一个新名称，管理者就需要对区域配置文件进行重新签名。

为了解决第一个问题NSEC3使用名称的哈希摘要来记录区域中的名称，从而防止Zone walk。为了解决第二个问题NSEC3引入了OPT-OUT标志，从而标识某个特定区域不需要启用DNSSEC，也不需要为其配置NSEC3记录。

NSEC3记录使用哈希摘要储存名称，哈希摘要的计算参数被储存在NSEC3PARAM资源记录中，这些参数包含算法名，迭代次数和随机值（salt）。

这些参数被用来计算每个名称所对应的哈希摘要，算法名确定摘要所使用的哈希算法类型，迭代次数确定哈希计算的重复次数，而随机值则会被加入在每次计算前输入尾部。

CVE-2023-50868指出较高的迭代次数会消耗解析器的计算资源，相关讨论也说明随机值和迭代次数也并不能很好的保护名称被攻击者解开。

RFC 9276文档中进而对NSEC3所使用的参数进行了建议，建议迭代次数为0，不使用随机值，从而缓解利用NSEC3消耗解析器资源的潜在能力，但目前业界仍未广泛采纳RFC 9276中设立的相关标准，NSEC3仍存在被攻击利用的可能性。

## Test

Iteration参数决定了名称摘要计算时哈希算法的重复次数，预期越大的迭代次数会使解析器进行越多次数的哈希计算，从而消耗计算资源，增大解析处理耗时，并有可能起到资源耗尽效果。

测试中设置对应nsec3xxx.keytrap.test域名，并对nsec3(xxx-1).keytrap.test进行请求，收到NXDOMAIN回复及相应的NSEC3记录，触发解析器对NSEC3记录的验证和比较，以测试NSEC3的迭代次数设置对解析器查询处理时间影响情况。

### 1 Iteration

将NSEC3PARM的参数设置为迭代次数0，随机值为空，加密算法使用默认算法。

设置域名nsec3.keytrap.test，发送对域名nsec2.keytrap.test的解析请求，收到NXDOMAIN回复，因为回复为NXDOMAIN，Unbound会进行重复尝试五次查询，五次查询耗时结果如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | Avg. |
| 1.167 | 1.183 | 1.352 | 1.209 | 1.276 | 1.2374 |

观察Unbound日志输出，发现Unbound支持的NSEC3最大迭代次数为150。

### 10 Iteration(Default)

将NSEC3PARM的参数设置为迭代次数10，随机值为空，加密算法使用默认算法。

设置域名nsec310.keytrap.test，发送对域名nsec309.keytrap.test的解析请求，收到NXDOMAIN回复，五次测试中解析耗时结果如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试次数 | 1 | 2 | 3 | 4 | 5 | Avg. |
| 耗时(msec) | 1.205 | 1.431 | 1.263 | 1.215 | 1.311 | 1.285 |

将Iteration设为10并未造成解析耗时的明显变化。

### 50 Iteration

将NSEC3PARM的参数设置为迭代次数50，随机值为空，加密算法使用默认算法。

设置域名nsec350.keytrap.test，发送对域名nsec349.keytrap.test的解析请求，收到NXDOMAIN回复，五次测试中解析耗时结果如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | Avg. |
| 1.222 | 1.054 | 1.254 | 1.218 | 1.100 | 1.1696 |

将Iteration设为10仍未造成解析耗时的明显变化。

### 150 Iteration

将NSEC3PARM的参数设置为迭代次数150，随机值为空，加密算法使用默认算法。

设置域名nsec3150.keytrap.test，发送对域名nsec3149.keytrap.test的解析请求，收到NXDOMAIN回复，五次测试中解析耗时结果如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | Avg. |
| 1.299 | 1.065 | 1.157 | 1.256 | 1.379 | 1.2312 |

将Iteration设为150仍未造成解析耗时的明显变化。

### 151 Iteration

dnssec-signzone报错，提示迭代数最大为150。

aspera@TochusCRevo:~/Sign/keys$ dnssec-signzone -S -K . -3- -H 151 -o keytrap.test. keytra

p.test.zone

dnssec-signzone: fatal: NSEC3 iterations too big. Maximum iterations allowed 150.

搜索资料发现isc.org在2023-02-14的更新中使得dnssec-signzone不会接收迭代次数超过150的签名请求，猜测要达到更高的迭代次数需要手动实现NSEC3名称哈希摘要的计算算法或使用低版本dnssec-signzone等。

KeyTrap || NSEC3 Validate

# Long CNAME Chain √

# Long Delegation Chain √

# Validation Cache Bypass ❓

# Different RCODE

# 99\*. Combinations

# 999\*. Ref