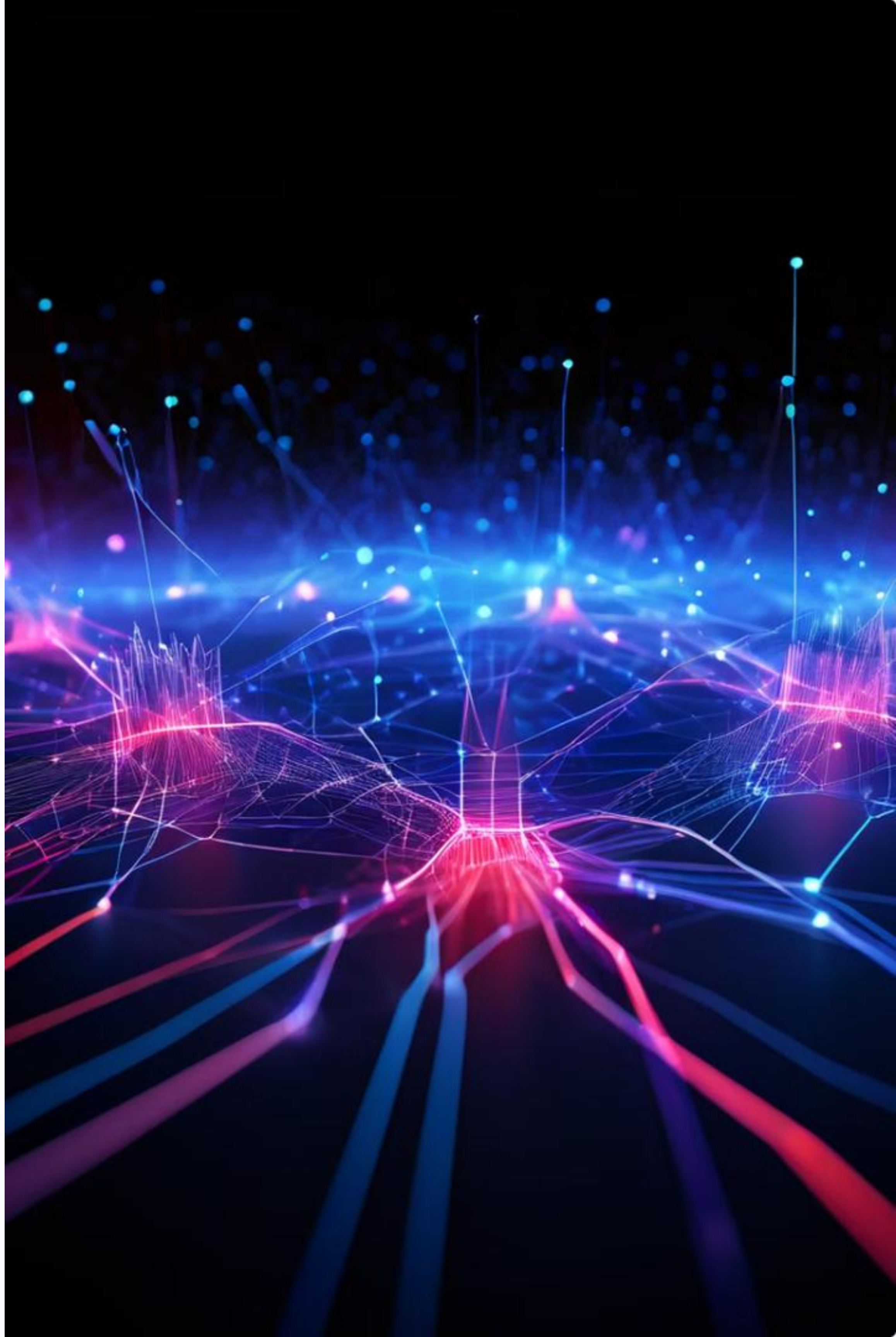
# A Flushing Attack on the DNS Cache

## Bind9 vs. Knot Resolver

Presented by **Tomer Yihya** and **Or Asheri** on 29.09.2024



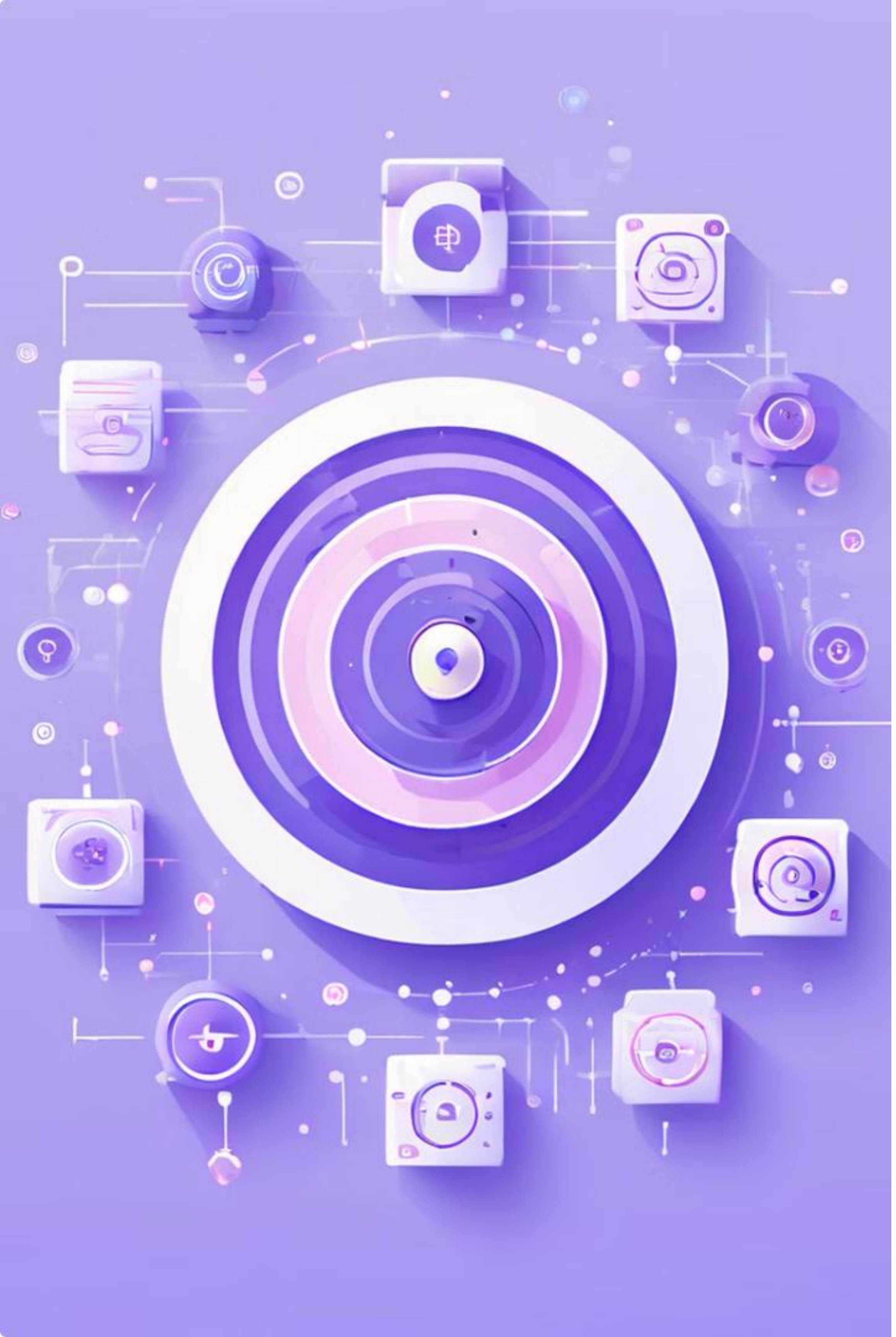
# Background on the Problem

## Bind9 Resolver

- **Common/Uncommon:** widely used, one of the most common DNS resolvers.
- **How it Works:** Receives a query, checks the cache and returns a valid response if it exists, otherwise it queries external DNS servers and caches the response.
- **Immunity/Vulnerabilities:** exposed to attacks like cache poisoning, cache flush attacks, and DNS amplification attacks.

## Knot Resolver

- **Common/Uncommon:** Less common but gaining popularity due to advanced features.
- **How it Works:** Similar query process but with advanced caching mechanisms, DNSSEC validation, and customizable policies using Lua scripting.
- **Immunity/Vulnerabilities:** Highly reliable with advanced security features like sophisticated caching strategies, strict DNSSEC enforcement, and custom policy capabilities that mitigate various attacks.



# Goals and Objectives

1

## Purpose

To examine the functionality of Knot Resolver and assess its immunity to flushing attacks.

2

## Scope

Knot Resolver's immunity to NS and A cache flushing attacks.

3

## Examples

Compare NS and A Cache Flush attacks on Bind9 and Knot Resolver.

# Project Plan

- Reproduction of the attacks on Bind9.
- Build docker image with Knot resolver.
- Examine Knot Resolver and trying to attack it.
- Taking traffic captures.
- Extracting the data to a CSV from the cache of Knot after the cache flush attacks.
- Examine the Cache contents of each attack (A,NS), the responses, and gathering conclusions.

# Knot Resolver Features

| Feature | Bind9 | Knot |
| --- | --- | --- |
| **Implementation** | Traditional caching | Advanced caching with metadata |
| **Flexibility** | Limited customization | Custom Lua scripting |
| **Performance** | Vulnerable under attack | Efficient resource usage |
| Use **Cases** | General DNS resolution | High customization and security |

# Research insights

## Summary of findings

- **Bind9** is vulnerable to cache flushing attacks due to its **traditional storage mechanisms**.
- **Knot Resolver** remains resistant to NS flush attack attacks thanks to its **advanced features**.

## Knot resolver defenses

- **Lua scripting** provide **flexibility** for efficient cache **management**.
- **Rate limiting** and anomaly detection.
- Strict **DNSSEC validation** maintains the integrity and security of cached responses.
- **Advanced caching strategies** prevent excessive NS and A records from overwhelming the cache.

# Knot Configuration (Kresd.conf)

```
-- Load useful modules
modules = {
    'hints',       -- Load /etc/hosts and allow custom root hints
    'policy',      -- User-defined forwarding/routing policies
    'stats',       -- Statistics module
}

-- Set root hints for the resolver using the hints.root() function
hints.root({'127.0.0.2'})  -- Root authoritative server

-- Define policy rules for specific TLDs
policy.add(policy.suffix(policy.STUB({'127.0.0.100'}), { todname('fun.lan') }))
policy.add(policy.suffix(policy.STUB({'127.0.0.200'}), { todname('home.lan') }))

-- Optionally, add a global policy to forward all other queries to the root server
policy.add(policy.all(policy.STUB({'127.0.0.2'})))

-- Cache size
cache.open(10 * MB, '/var/cache/knot-resolver/cache.lmdb')

-- Listen on loopback interface
net.listen('127.0.0.1', 53)

-- Logging
log = {
    { 'stdout', 'debug' }  -- Log to stdout at debug level
}

-- Control interface
control = { socket = '/run/knot-resolver/control.sock' }

-- Disable DNSSEC (if necessary)
-- trust_anchors.remove('.')
```

We configured Knot with the auth servers we are using to perform the attack.

In addition, we configured the resolver's cache in a specific path (cache.lmdb) with 10 Mbs size.

We added logging,
all the rest configured as default.

# Dump cache to CSV script

dump_cache_to_csv.py > ...

```python
import lmdb
import pandas as pd
import os

# Path to your LMDB file (assuming it's in the same directory as the script)
current_dir = os.path.dirname(os.path.abspath(__file__))
lmdb_path = os.path.join(current_dir, 'cache.lmdb')

# Open the LMDB environment
env = lmdb.open(lmdb_path, readonly=True)

# Initialize an empty list to store the data
data = []

# Read the data from the LMDB file
with env.begin() as txn:
    cursor = txn.cursor()
    for key, value in cursor:
        # Decode key (assuming keys are strings)
        decoded_key = key.decode('utf-8', errors='ignore')
        # Convert value to a hexadecimal string or handle it as binary
        hex_value = value.hex()
        data.append((decoded_key, hex_value))

# Convert the data to a pandas DataFrame
df = pd.DataFrame(data, columns=['Key', 'Value'])

# Write the DataFrame to a CSV file with escape character
df.to_csv('output.csv', index=False, escapechar='\\')

# Write the DataFrame to an Excel file
# df.to_excel('output.xlsx', index=False)

print("Data has been successfully exported to 'output.csv'")

```

*Since the cache is in “cache.lmdb/data.mdb”*

*and there is no app to open it and observe,*

*we wrote a script to dump the data into a CSV file.*

# Snippets and Insights

NS CacheFlushAttack /Knot

## *Dig response*

```
root@612e0d275b27:/env# dig attack99.home.lan
;; Truncated, retrying in TCP mode.

; <<>> DiG 9.18.21 <<>> attack99.home.lan
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30856
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1900, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;attack99.home.lan.            IN      A

;; AUTHORITY SECTION:
attack99.home.lan.      8600    IN      NS      auth596.fun.lan.
attack99.home.lan.      8600    IN      NS      auth597.fun.lan.
attack99.home.lan.      8600    IN      NS      auth598.fun.lan.
attack99.home.lan.      8600    IN      NS      auth599.fun.lan.
attack99.home.lan.      8600    IN      NS      auth600.fun.lan.
attack99.home.lan.      8600    IN      NS      auth601.fun.lan.
attack99.home.lan.      8600    IN      NS      auth602.fun.lan.
attack99.home.lan.      8600    IN      NS      auth603.fun.lan.
attack99.home.lan.      8600    IN      NS      auth604.fun.lan.
attack99.home.lan.      8600    IN      NS      auth605.fun.lan.
attack99.home.lan.      8600    IN      NS      auth606.fun.lan.
attack99.home.lan.      8600    IN      NS      auth607.fun.lan.
attack99.home.lan.      8600    IN      NS      auth608.fun.lan.
attack99.home.lan.      8600    IN      NS      auth609.fun.lan.
```

## *The cache.*

| Key | Value |
|---|---|
| E | 01000000b897ec660500000051004b00ab888500000100010000000200000 |
| E0 | b897ec660500000051005100190085000001000000010001000030000100 |
| S | 1E+46 |
| S | 0300000001000000000000000157f00000000000000000000 |
| VERS | 700 |
| | b897ec660500000051005f00782b850000010000000100010d5f74612d3461 |
| lanhomeattack0E | a198ec660500000051008fa66772850000010000076c00010761747461636 |
| 00002198000a0761757468373437c034c00c0002000100002198000a0761757468373438c034c00c00020001000021 | |
| 000b086175746831343730c034c00c0002000100002198000b086175746831343731c034c00c000200010000219800 | |
| lanhomeattack99E | 6399ec6605000000510090a63231850000010000076c00010861747461636 |
| 0100002198000a0761757468373437c035c00c0002000100002198000a0761757468373438c035c00c000200010000 | |
| 98000b086175746831343730c035c00c0002000100002198000b086175746831343731c035c00c000200010000219 | |
| netroot-serversaE | b897ec660500000051004c003ea98500000100010001000101610c726f6f74 |
| netroot-serversaE | b897ec660500000051005200c1f98500000100000001000101610c726f6f74 |

Permanent keys in the cache regardless of the queries.

Seems Knot is immune to the attack,
We will see a proof in the next slide.

# Snippets and Insights

NS CacheFlushAttack /Knot

***Script we wrote to create a zonefile which we use to analyze the cache key:value insertions.***

```
with open('NSCacheFlush_zone_file.txt', 'w') as f:
    DOMAINS_NUM = 16

    for i in range(1, DOMAINS_NUM):
        for j in range(2**i):
            print(f'attack{i}   3600    IN  NS  auth{j}.fun.lan.', file=f)
```

***We crafted it to generate NS Records number as a exponential function of the domain number.***

***(we made a barrier at domain num = 16, since NSD refused to load a larger number of RR set per domain).***

# Snippets and Insights

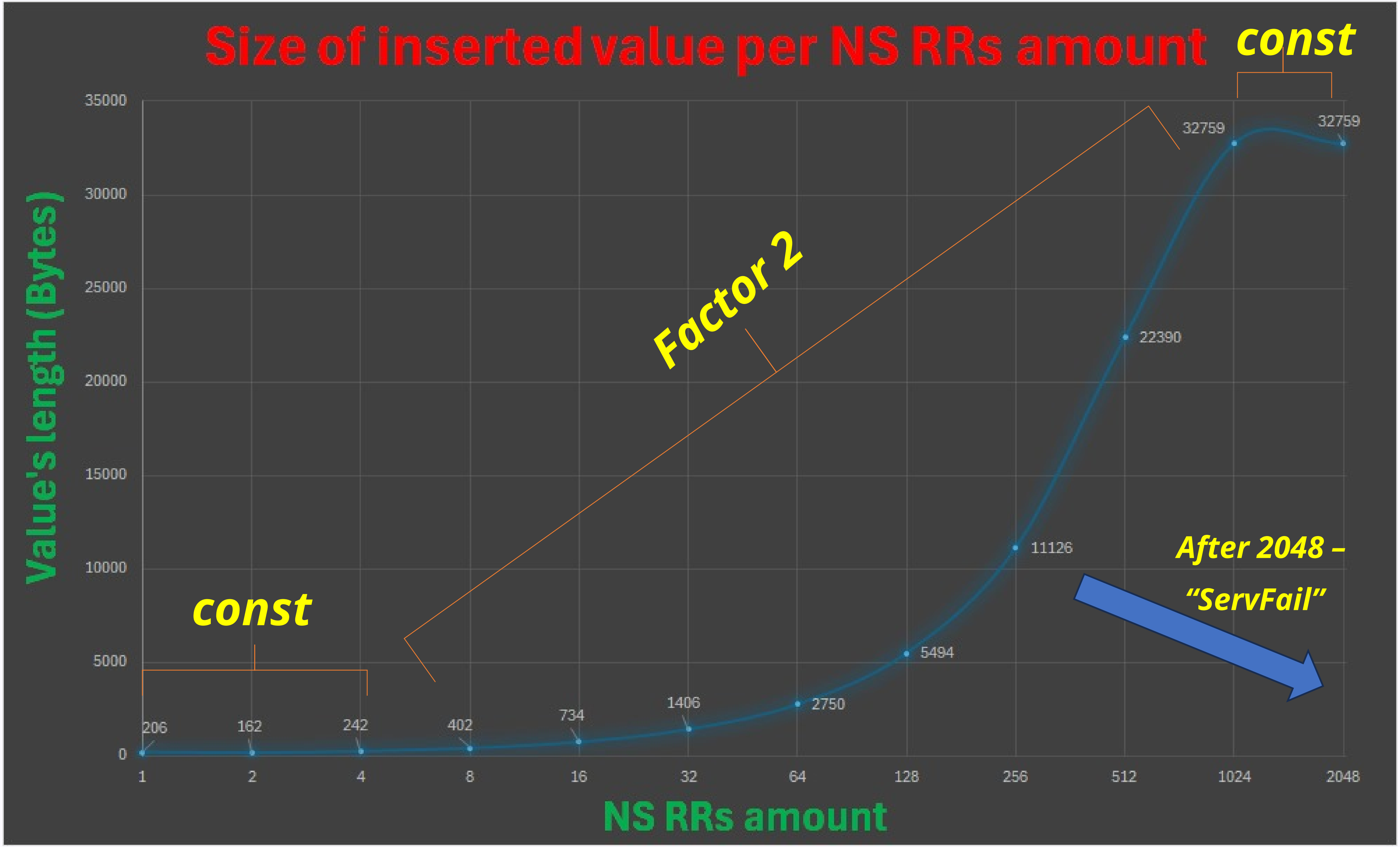
## NS CacheFlushAttack /Knot

***After extraction of the Cache's content,***
***and ordering it by number of RRs per domain in the Zonefile.***

| Key | NS RRs | Value Size | Value |
|---|---|---|---|
| lanhomeattack0E▯(2^0) | 1 | 206 | 8240026705000000051005b0028e3850300010000000100010761747461636b3004686f6d65036c |
| lanhomeattack1E▯ (2^1) | 2 | 162 | 86400267050000000510045 00fc0085000001000000010001076174746163 6b3104686f6d65036c6 |
| lanhomeattack2E▯ (2^2) | 4 | 242 | 8a4002670500000051006d001df68500000100000003000107617474616 36b3204686f6d65036c6 |
| lanhomeattack3E▯ (2^3) | 8 | 402 | 8f40026705000000 5100bd0003c1850000010000000700010761747461636b3304686f6d65036c6 |
| lanhomeattack4E▯ (2^4) | 16 | 734 | 9240026705000000051006301 6d958500000100000 00f00010761747461636b3404686f6d65036c6 |
| lanhomeattack5E▯ (2^5) | 32 | 1406 | 954002670500000 05100b302e62885000001000 0001f00010761747461636b3504686f6d65036c6 |
| lanhomeattack6E▯ (2^6) | 64 | 2750 | 98400267050000000510053 05825e850000010000003f00010761747461636b3604686f6d65036c6 |
| lanhomeattack7E▯ (2^7) | 128 | 5494 | 9b40026705000000 5100af0a1d10850000010000007f00010761747461636b3704686f6d65036c6 |
| lanhomeattack8E▯ (2^8) | 256 | 11126 | 084102670500000 05100af15586c8500000100 0000ff00010761747461636b3804686f6d65036c61 |
| lanhomeattack9E▯ (2^9) | 512 | 22390 | a44002670500000 05100af2b222585000001000001ff00010761747461636b3904686f6d65036c61 |
| lanhomeattack10E▯ (2^10) | 1024 | 32759 | a8400267050000005100c8570c6085000001000003ff00010861747461636b313004686f6d65036 |
| lanhomeattack11E▯ (2^11) | 2048 | 32759 | 0f41026705000000 5100c8b395e485000001000007ff00010861747461636b313104686f6d65036 |

# Snippets and Insights

NS CacheFlushAttack /Knot




***Graph of the presented in the previous slide***

***Conclusions in the next slide.***

# Snippets and Insights

NS CacheFlushAttack /Knot

***Query response For a domain which Isn't in the zone file. (Attack20)***

```
root@612e0d275b27:/env# dig attack20.home.lan

; <<>> DiG 9.18.21 <<>> attack20.home.lan
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 35757
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;attack20.home.lan.     IN   A

;; AUTHORITY SECTION:
home.lan.       86400   IN  SOA ns1.home.lan. admin.home.lan. 3311010299 28800 7200 864000 86400
```

***Query response For a domain which exists in the zone file. (attack 8).***

```
; <<>> DiG 9.18.21 <<>> attack8.home.lan
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42186
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 255, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;attack8.home.lan.      IN   A

;; AUTHORITY SECTION:
attack8.home.lan.   3600    IN  NS  auth152.fun.lan.
attack8.home.lan.   3600    IN  NS  auth153.fun.lan.
attack8.home.lan.   3600    IN  NS  auth154.fun.lan.
attack8.home.lan.   3600    IN  NS  auth155.fun.lan.
```

# Snippets and Insights NS CacheFlushAttack /Knot

1. *from 1 to 8 number of RRs per domain, we can see the value size is remain approximately constant.*

2. *from 16 to 1024 RRs per domain, we see an increasing by factor 2 gradient in the amount of bytes In the values which were inserted to the cache.*

3. *from 1024 to 2048 number of RRs per domain, we see that the value size in bytes has remained 32759. Therefor we believe it's the upper bound of the value size.*

4. *for 4096 NS Records and beyond, the dig response status is "Servfail" (shown in the next slide) and the key:value pair doesn't enter the cache.*

# Snippets and Insights

NS CacheFlushAttack /Knot

***Query response for domain with 4096 NS record names.***

***The answer is the same for higher number of RR names per domain.***

```
root@612e0d275b27:/env# dig attack12.home.lan

; <<>> DiG 9.18.21 <<>> attack12.home.lan
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 8039
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;attack12.home.lan.     IN   A

;; Query time: 3 msec
;; SERVER: 127.0.0.1#53(127.0.0.1) (UDP)
;; WHEN: Sun Oct 06 07:49:39 UTC 2024
;; MSG SIZE  rcvd: 46
```

# Snippets and Insights

NS CacheFlushAttack / Knot

***Malicious Queries***
***Attack rate: 10000 qps***

```
Statistics:

  Queries sent:         999999
  Queries completed:    926246
  Queries lost:         73753
  Response codes:       NOERROR 6622 (0.71%), SERVFAIL 45974 (4.96%), NXDOMAIN 873650 (94.32%)
  Reconnection(s):      0
  Run time (s):         136.337502
  Maximum throughput:   10008.000000 qps
  Lost at that point:   0.00%
```

***Benign Queries***
***Rate: 100 qps***

```
Statistics:

  Queries sent:         9999
  Queries completed:    9281
  Queries lost:         718
  Response codes:       NOERROR 9281 (100.00%)
  Reconnection(s):      0
  Run time (s):         130.020002
  Maximum throughput:   100.000000 qps
  Lost at that point:   0.00%
```

We can see that the flooding with malicious requests didn't manage to cause DoS.

# Snippets and Insights

NS CacheFlushAttack / Knot

***We have also checked Knot resolver cache after running the resperf experiment.***

| Key | Value |
|---|---|
| S▯d | 04000000030000000000000000157f0000000000000000000 |
| S▯ | 04000000030000000000000000157f0000000000000000000 |
| VERS | 700 |
| lanfunbenign8472E▯ | 8f80f6660500000051008300fc31850000010001000200030a62656e69676e383437320366 |
| lanfunbenign8477E▯ | 8f80f66605000000510083000012850000010001000200030a62656e69676e383437370366 |
| lanfunbenign8481E▯ | 8f80f6660500000051008300802c850000010001000200030a62656e69676e383438310366 |
| lanfunbenign8491E▯ | 8f80f66605000000510083000633850000010001000200030a62656e69676e383439310366 |
| lanfunbenign8497E▯ | 8f80f6660500000051008300901f850000010001000200030a62656e69676e383439370366 |
| lanfunbenign8502E▯ | 8f80f6660500000051008300ec13850000010001000200030a62656e69676e383530320366 |
| lanfunbenign8503E▯ | 8f80f66605000000510083002b53850000010001000200030a62656e69676e383530330366 |
| lanfunbenign8504E▯ | 8f80f66605000000510083005acd850000010001000200030a62656e69676e383530340366 |
| lanfunbenign8505E▯ | 8f80f66605000000510083007ddb850000010001000200030a62656e69676e383530350366 |
| lanfunbenign8506E▯ | 8f80f666050000005100830010b7850000010001000200030a62656e69676e383530360366 |
| lanfunbenign8507E▯ | 8f80f666050000005100830020c8850000010001000200030a62656e69676e383530370366 |
| lanfunbenign8508E▯ | 8f80f66605000000510083002fa4850000010001000200030a62656e69676e383530380366 |
| lanfunbenign8509E▯ | 8f80f6660500000051008300af58850000010001000200030a62656e69676e383530390366 |
| lanfunbenign8510E▯ | 8f80f66605000000510083008d23850000010001000200030a62656e69676e383531300366 |

…

| | |
|---|---|
| lanhomeattack26271E▯ | 9080f66605000000051005f00ffcb850300010000000100010b61747461636b3236323 |
| lanhomeattack26272E▯ | 9080f66605000000051005f004d86850300010000000100010b61747461636b3236323 |
| lanhomeattack26273E▯ | 9080f66605000000051005f00e579850300010000000100010b61747461636b3236323 |
| lanhomeattack26274E▯ | 9080f66605000000051005f004902850300010000000100010b61747461636b3236323 |
| lanhomeattack26275E▯ | 9080f66605000000051005f001fa8850300010000000100010b61747461636b3236323 |
| lanhomeattack26276E▯ | 9080f66605000000051005f00fef7850300010000000100010b61747461636b3236323 |
| lanhomeattack26277E▯ | 9080f66605000000051005f00c532850300010000000100010b61747461636b3236323 |
| lanhomeattack26278E▯ | 9080f66605000000051005f00d8f8850300010000000100010b61747461636b3236323 |
| lanhomeattack26279E▯ | 9080f66605000000051005f0088d4850300010000000100010b61747461636b3236323 |
| lanhomeattack26280E▯ | 9080f66605000000051005f00ae0c850300010000000100010b61747461636b3236323 |
| lanhomeattack26281E▯ | 9080f66605000000051005f00b012850300010000000100010b61747461636b3236323 |
| lanhomeattack26282E▯ | 9080f66605000000051005f00f466850300010000000100010b61747461636b3236323 |
| lanhomeattack26283E▯ | 9080f66605000000051005f006ebe850300010000000100010b61747461636b3236323 |
| lanhomeattack26284E▯ | 9080f66605000000051005f00c694850300010000000100010b61747461636b3236323 |

***Seems the LRU policy works as expected, as values insertion and eviction takes place without issues.***

***We assume that its due the fact that Knot Keeps only 1 key:value pair per query.***

# Snippets and Insights

NS CacheFlushAttack /Bind vs. Knot

## Dig response

### Bind

```
root@e55a9d18506b:/env# dig attack0.home.lan

; <<>> DiG 9.18.21 <<>> attack0.home.lan
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 60124
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 0e53cf4bd0d6397f0100000066f69b1376b0b4fdb37c32cc (good)
;; QUESTION SECTION:
;attack0.home.lan.              IN      A

;; Query time: 23 msec
;; SERVER: 127.0.0.1#53(127.0.0.1) (UDP)
;; WHEN: Fri Sep 27 11:46:27 UTC 2024
;; MSG SIZE  rcvd: 73
```

### Knot

```
root@612e0d275b27:/env# dig attack99.home.lan
;; Truncated, retrying in TCP mode.

; <<>> DiG 9.18.21 <<>> attack99.home.lan
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30856
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1900, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;attack99.home.lan.             IN      A

;; AUTHORITY SECTION:
attack99.home.lan.      8600    IN      NS      auth596.fun.lan.
attack99.home.lan.      8600    IN      NS      auth597.fun.lan.
```

## Resperf results

### Bind

```
Statistics:

  Queries sent:         9999
  Queries completed:    9672
  Queries lost:         327
  Response codes:       NOERROR 2018 (20.86%), SERVFAIL 7654 (79.14%)
  Reconnection(s):      0
  Run time (s):         100.000785
  Maximum throughput:   154.000000 qps
  Lost at that point:   0.00%
```

### Knot

```
Statistics:

  Queries sent:         9999
  Queries completed:    9281
  Queries lost:         718
  Response codes:       NOERROR 9281 (100.00%)
  Reconnection(s):      0
  Run time (s):         130.020002
  Maximum throughput:   100.000000 qps
  Lost at that point:   0.00%
```

# Snippets and Insights

A CacheFlushAttack

**We have written a script which will generate 2048 different A records for each of 100 domains and integrated it in a designated zonefile.**

```python
# Script to generate a large number of A records (2048 per attack domain) - for cache flush.
with open('ACacheFlush_zone_file.txt', 'w') as f:
    DOMAINS_NUM = 10000  # Number of attack domains

    for i in range(DOMAINS_NUM):
        for ip_octet1 in range(8): # Generate 8*256 A records per domain
            for ip_octet2 in range(256):
                print(f'attack{i}.home.lan. 3600 IN A 127.0.{ip_octet1}.{ip_octet2}', file=f)
```

**To test if a resolver immunity to the attack, we implemented an A records zonefile - then loaded it using nsd into nsd_attack_home directory and made the experiments.**

# Snippets and Insights

A CacheFlushAttack / Bind

## *Dig response.*

```
root@9fbfd7b484c8:/env# dig attack0.home.lan
;; Truncated, retrying in TCP mode.

; <<>> DiG 9.18.21 <<>> attack0.home.lan
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33783
;; flags: qr rd ra; QUERY: 1, ANSWER: 2048, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 6553423e2cc01ca00100000066f5d14a8cb56c02c30c284d (good)
;; QUESTION SECTION:
;attack0.home.lan.              IN      A

;; ANSWER SECTION:
attack0.home.lan.       3600    IN      A       127.0.3.162
attack0.home.lan.       3600    IN      A       127.0.1.245
attack0.home.lan.       3600    IN      A       127.0.7.74
attack0.home.lan.       3600    IN      A       127.0.3.167
attack0.home.lan.       3600    IN      A       127.0.7.10
attack0.home.lan.       3600    IN      A       127.0.1.125
attack0.home.lan.       3600    IN      A       127.0.2.250
attack0.home.lan.       3600    IN      A       127.0.3.136
attack0.home.lan.       3600    IN      A       127.0.3.42
attack0.home.lan.       3600    IN      A       127.0.1.227
attack0.home.lan.       3600    IN      A       127.0.7.172
attack0.home.lan.       3600    IN      A       127.0.4.155
attack0.home.lan.       3600    IN      A       127.0.0.170
attack0.home.lan.       3600    IN      A       127.0.3.207
```

## *The cache.*

```
; Start view _default
;
;
; Cache dump of view '_default' (cache _default)
;
; using a 0 second stale ttl
$DATE 20240909144439
; authanswer
.                       3424    IN NS   a.root-servers.net.
; answer
lan.                    3424    \-NS    ;-$NXRRSET
; . SOA root-servers.net. a.root-servers.net. 2011010203 28800 7200 864000 86400
; authauthority
home.lan.               86224   NS      ns1.home.lan.
                        86224   NS      ns2.home.lan.
; authanswer
attack0.home.lan.       3424    A       127.0.0.0
                        3424    A       127.0.0.1
                        3424    A       127.0.0.2
                        3424    A       127.0.0.3
                        3424    A       127.0.0.4
                        3424    A       127.0.0.5
                        3424    A       127.0.0.6
                        3424    A       127.0.0.7
                        3424    A       127.0.0.8
                        3424    A       127.0.0.9
                        3424    A       127.0.0.10
                        3424    A       127.0.0.11
                        3424    A       127.0.0.12
                        3424    A       127.0.0.13
                        3424    A       127.0.0.14
                        3424    A       127.0.0.15
```

Seems Bind is vulnerable to A cache flush attack as well,
We will see a proof in the next slide.

# Snippets and Insights

A CacheFlushAttack /Bind

***Malicious Queries***
***Attack rate: 10000 qps***

```
Statistics:

  Queries sent:         250676
  Queries completed:    158878
  Queries lost:         91798
  Response codes:       NOERROR 20 (0.01%), SERVFAIL 99502 (62.63%), NXDOMAIN 59356 (37.36%)
  Reconnection(s):      0
  Run time (s):         69.042792
  Maximum throughput:   10066.000000 qps
  Lost at that point:   0.00%
```

***Benign Queries***
***Rate: 100 qps***

```
Statistics:

  Queries sent:         9999
  Queries completed:    9999
  Queries lost:         0
  Response codes:       NOERROR 1436 (14.36%), SERVFAIL 8563 (85.64%)
  Reconnection(s):      0
  Run time (s):         100.000422
  Maximum throughput:   100.000000 qps
  Lost at that point:   0.00%
```

We can see that the flooding with malicious requests has indeed resulted with DoS, as in NS cache flush attack.

# Snippets and Insights

A CacheFlushAttack / Knot

## *Dig response*

```
root@a10daf5e9123:/env# dig attack42.home.lan
;; Truncated, retrying in TCP mode.

; <<>> DiG 9.18.21 <<>> attack42.home.lan
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63790
;; flags: qr rd ra; QUERY: 1, ANSWER: 2048, AUTHORITY: 0, ADDITIONAL:

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;attack42.home.lan.             IN      A

;; ANSWER SECTION:
attack42.home.lan.      3600    IN      A       127.0.0.31
attack42.home.lan.      3600    IN      A       127.0.0.32
attack42.home.lan.      3600    IN      A       127.0.0.33
attack42.home.lan.      3600    IN      A       127.0.0.34
attack42.home.lan.      3600    IN      A       127.0.0.35
attack42.home.lan.      3600    IN      A       127.0.0.36
attack42.home.lan.      3600    IN      A       127.0.0.37
attack42.home.lan.      3600    IN      A       127.0.0.38
```

## *The cache after 2 queries.*

| Key | Value |
|---|---|
| E | 01000000fe8bec660500000051004b00f3a18500000100010000000200000200010 |
| S | 0100000000000000000000005356d3da0000000000000000 |
| S | 020000000100000000000000ea7f00000000000000000000 |
| VERS | |
| lanhomeattack0E | f392ec660500000051006580975585000001080000020002076174746163 6b30046 |
| 03fcc00c0001000100000e1000047f0003fdc00c0001000100000e1000047f0003fec00c0001000100000e1000047f0003ffc00c | |
| 00047f0007fcc00c0001000100000e1000047f0007fdc00c0001000100000e1000047f0007fec00c0001000100000e1000047f00 | |
| lanhomeattack42E | de94ec66050000005100668025fb85000001080000020002086174746163 6b34320 |
| 0003fcc00c0001000100000e1000047f0003fdc00c0001000100000e1000047f0003fec00c0001000100000e1000047f0003ffc00 | |
| 1000047f0007fcc00c0001000100000e1000047f0007fdc00c0001000100000e1000047f0007fec00c0001000100000e1000047f | |
| netroot-serversaE | fe8bec660500000051004c000aaa850000010001000100010161 0c726f6f742d7365 |
| netroot-serversaE | fe8bec66050000005100520070 0a850000010000000100010161 0c726f6f742d7365 |

Seems Knot is immune to the attack,
We will see a proof in the next slide.

# Snippets and Insights

A CacheFlushAttack / Knot

***Malicious Queries***
***Attack rate: 10000 qps***

```
Statistics:

  Queries sent:          134656
  Queries completed:     111426
  Queries lost:          23230
  Response codes:        NOERROR 2419 (2.17%), SERVFAIL 2232 (2.00%), NXDOMAIN 106775 (95.83%)
  Reconnection(s):       0
  Run time (s):          58.574537
  Maximum throughput:    10018.000000 qps
  Lost at that point:    0.00%
```

***Benign Queries***
***Rate: 100 qps***

```
Statistics:

  Queries sent:          9999
  Queries completed:     9813
  Queries lost:          186
  Response codes:        NOERROR 9813 (100.00%)
  Reconnection(s):       0
  Run time (s):          100.000076
  Maximum throughput:    244.000000 qps
  Lost at that point:    0.00%
```

We can see that the flooding with malicious requests didn’t damage the responses as it did while using Bind.

# Conclusion

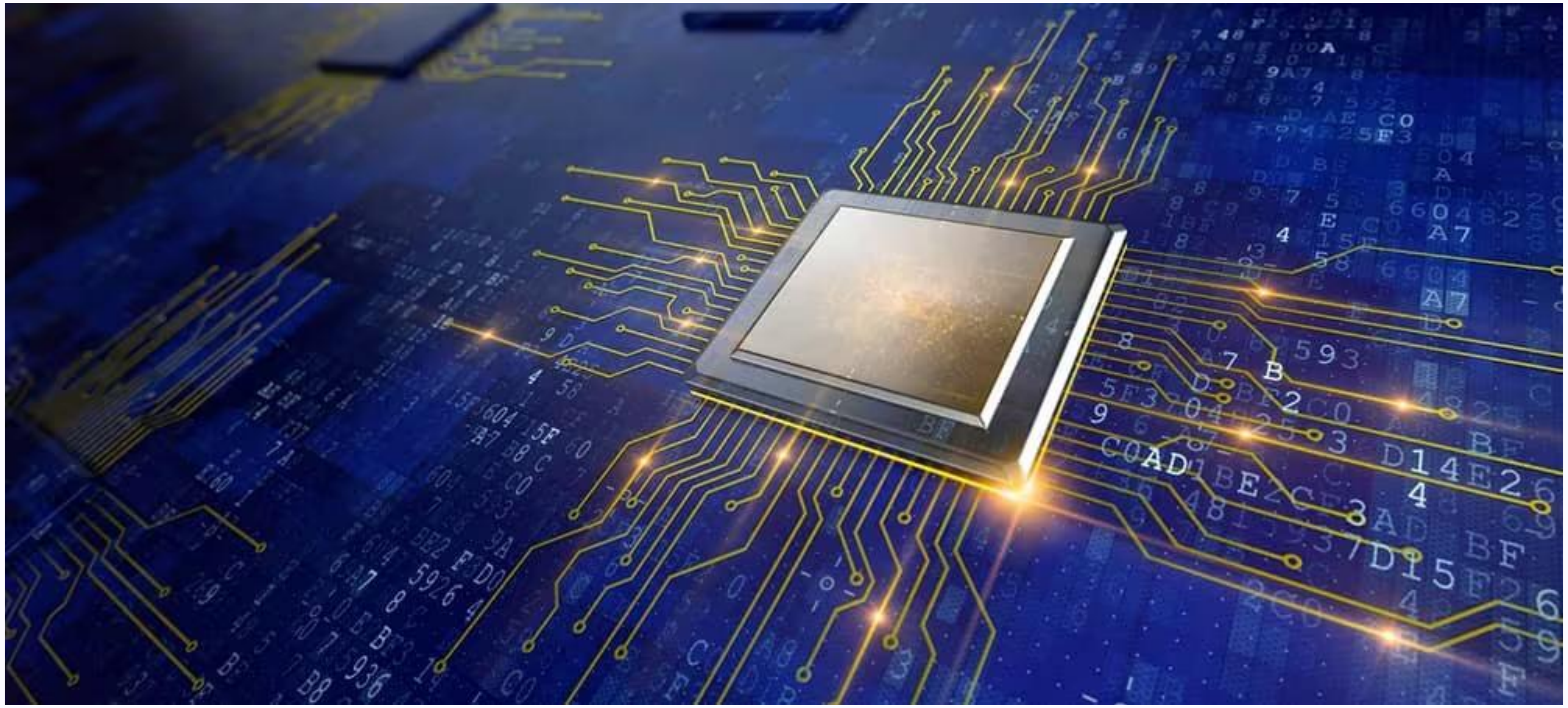
***In contrast to Bind which keeps all the results from each query in its benign cache, Knot keeps results in only one pair of Key:Value.***

***Therefor, seems to be immune to the cache flush attacks.***

# Results, documentation and reproduction.

***All resources and findings are in Github***

*DNS_CacheFlushAttack_workshop (github.com)*



# Thank you for listening !