USENIX'24 Artifact Appendix: A Flushing Attack on the DNS Cache

Yehuda Afek
Tel-Aviv University

Anat Bremler-Barr
Tel Aviv University

Shoham Danino
Reichman University

Yuval Shavitt
Tel-Aviv University

A Abstract

To fully understand the root cause of the CacheFlushAttack and to analyze its effective flush rate, we developed a mini-lab setup, disconnected from the Internet, that contains all the components of the DNS system, clients, resolvers, and authoritative name servers. This setup is built to analyze and examine the behavior of a resolver (or any other component) in details. On the other hand it is not useful for performance analysis (stress analysis). Here we provide the code and details of this setup enabling to reproduce our analysis. Moreover, researchers may find it useful for farther behavioral analysis and examination of different components in the DNS system.

A.1 Description & Requirements

DNS-CacheFlushSimulator is an Inner-Emulator environment for DNS protocol which was built as part of CacheFlushAttack research. DNS-CacheFlushSimulator includes clients, resolver and four authoritative name servers. The resolver is a BIND9 recursive resolver with the latest version (9.18.21). The four authoritative name servers are: a 'root', an attacker, and two malicious delegation authoritative name servers.

Most of the CacheFlushAttack measurements were carried out on a BIND9 version 9.18.21 resolver compiled to work with the local 'root' authoritative name server. The authoritative name servers are implemented with Name Server Daemon (NSD) version 4.3.3. The clients are deployed on the same machine, which was configured to send DNS queries directly to the local recursive resolver. The setup configuration and environment are provided in GitHub (*https://github.com/shohamda/CacheFlushSimulator*).

In order to use DNS-CacheFlushSimulator, a docker docker is required.

A.1.1 Security, privacy, and ethical concerns

To ensure that no harm may be done outside of the setup, the environment runs locally in closed Docker container environment. It is thus important to use "–dns 127.0.0.1" flag to configure this. Changing the "resolv.conf" configuration inside the docker container is not enough (see Appendix A.2).

A.1.2 How to access

DNS-CacheFlushSimulator source code can be found at DNS-CacheFlushSimulator GitHub (1.0 Tag). The environment docker image can be accessed through Docker-Hub (1.0 Tag).

A.1.3 Hardware dependencies

There is no hardware dependencies required for using DNS-CacheFlushSimulator. During our research, we used an Ubuntu computer or Virtual Machine (we recommend using Ubuntu 20.04 or above) which is capable of running Docker images according to "Install Docker Engine on Ubuntu" specification.

A.1.4 Software dependencies

1. Docker
2. WireShark (To install WireShark on Ubuntu use: `apt install wireshark`).
3. Resperf (To install Resperf on Ubuntu use: `apt install resperf`).

A.1.5 Benchmarks

In order to conduct the experiments described in CacheFlushAttack paper (Section 5), the setup should contain a BIND resolver with a new version bind9.18.21 and at four authoritative servers (local root authoritative and three authoritative to simulate the "attack.com" and "delegation.attack" authoritative in Figure 1). For

CacheFlushNS Attack, a malicious zone file is required for the attacker authoritative (i.e., “home.lan.forward” for “home.lan” server). The malicious zone file contain malicious referral list with 1900 NS for each domain, which all delegate the resolver to a server IP address that is non-responsive to DNS queries through another authoritative server (the local root server can also be used to delegate the resolver to this IP address). For CacheFlushCNAME Attack, a malicious zone file is required for the attacker authoritative as well (i.e., “home.lan.forward_CNAME” for “home.lan” server). The malicious zone file contains a malicious CNAME chain of 1 million records, each pointing to the next domain.

For instance, if the malicious request from the client is “attacker0.home.lan”, the malicious referral response include a 1,900 list of name servers return. In order to create such referral list the “/env/nsd_attack_home/home.lan.forward” zone file needs to have 1900 records per one malicious request. That is, the malicious zone file includes 1900 records for the malicious request which leads to a none existent domain name (e.g., attacker0 IN NS auth0.fun.lan. ... auth1900.fun.lan.) and the “fun.lan” authoritative server, which does not have any record to the non-responsive domain name, includes a wildcard record delegating the resolver to a non responsive IP address (e.g., * IN A 127.0.0.212).

A.2 Set-up

The following tree structure represent relevant folders and file in the environment with description for each one of them.

```
env
├── client ... (127.0.0.1)
├── resolver ... (127.0.0.1)
├── nsd_root ... (127.0.0.2) - Root authoritative
│       server configuration folder
│   ├── lan.forward ...
│   │   Zone file for SLD server ".lan"
│   ├── lan.reverse
│   ├── net.forward ...
│   │   Zone file for root server ".net"
│   ├── net.reverse
│   ├── nsd.conf ...
│   │   Configuration file for NSD,
│   │   contains the IP address of the
│   │   root server
│   └── nsd.db ... NSD DB, for internal NSD
│           usage
├── nsd_attack_home ...
│   (127.0.0.200) "home.lan"
│   malicious authoritative server
│   configuration folder, (which
│   simulates 'attack.com" server in
│   Figures 1,2)
│   ├── home.lan.forward ...
│   │   Zone file for sld ".home.lan",
│   │   with the malicious CacheFlushNS
│   │   Attack
│   ├── home.lan.forward_cname ...
│   │   Zone file for sld ".home.lan",
│   │   with the malicious
│   │   CacheFlushCNAME Attack
│   ├── home.lan.reverse
│   ├── nsd.conf ...
│   │   Configuration file for NSD,
│   │   contains the IP address of the
│   │   malicious authoritative server
│   ├── nsd.db ... NSD DB, for internal NSD usage
│   └── named.conf ...
│       Bind9 configuration, contains
│       the IP address of the local
│       environment
├── nsd_attack_fun ...
│   (127.0.0.100) "fun.lan"
│   malicious authoritative server
│   configuration folder, (which
│   simulates 'delegation.attack"
│   server in Figure 1)
│   ├── fun.lan.forward ...
│   │   Zone file for sld ".fun.lan",
│   │   with the malicious CacheFlushNS
│   │   Attack
│   ├── fun.lan.reverse
│   ├── nsd.conf ...
│   │   Configuration file for NSD,
│   │   contains the IP address of the
│   │   malicious authoritative server
│   ├── nsd.db ... NSD DB, for internal NSD usage
│   └── named.conf ...
│       Bind9 configuration, contains
│       the IP address of the local
│       environment
├── bind9_18_21 ...
│   Bind source code with
│   modification to use local root
│   server
└── nsd ... NSD source code from https://github.com/NLnetLabs/nsd,
        this folder relevant in case
        of changes to the original NSD
        code (In our experiment we didn't
        change this code)
```

A.2.1 Installation

1. Pull the docker image from Docker Hub (`docker pull shohamd/cacheflushsimulator:1.0`).
2. Run the docker image as a container interactively so you can control the environment (`docker container run --dns 127.0.0.1 --mount type`

=**bind**,**source**=<local_folder_path>,target=/app -it shohamd/cacheflushsimulator:latest /bin/bash).
It is important to use the `dns 127.0.0.1` flag so the environment DNS will be local, changing the `resolv.conf` file inside a Docker container does not work. Note that we are mounting `<local_folder_path>` to the folder `/app` inside the docker container so it will be easier to copy files to and from the docker container.

3. Now you have a terminal inside the environment.

4. In order to open another terminal for the environment first run `sudo docker container ls`, look for `cacheflushsimulator` docker image name and copy its `<CONTAINER ID>`.
Then, run `sudo docker exec -it <CONTAINER ID> bash`.
To open more terminals into the environment, repeat this process.

To conduct the experiments described in CacheFlushAttack paper, the setup needs to include a resolver and at least three authoritative servers.

Environment IP address:

1. 127.0.0.1 – Client
2. 127.0.0.1 – Our own Resolver (The client and resolver have the same IP address)
3. 127.0.0.2 – Root authoritative
4. 127.0.0.100 – "fun.lan" TLD authoritative (which simulates "delegation.attack" server in Figure 1).
5. 127.0.0.200 – "home.lan" TLD authoritative (which simulates "attack.com" server in Figure 1).
6. 127.0.0.53 – The default resolver – DO NOT USE IT WHILE TESTING!

Authoritative Servers: Our authoritative servers are located at "/env/nsd_root", "/env/nsd_attack_home" and "/env/nsd_attack_fun". To use them, first configure their zone files which are located inside their folder and called "ZONE_NAME.forward". After changing the zone file a restart to the authoritative is required in order to apply the changes.

Resolver: Go to the resolver implementation folder "/env/bind9_18_21" and run: `make install`.

Starting the environment: Open four terminals in the Docker container: First, turn on the Resolver using the following commands:

```
cd /etc
named -g -c /etc/named.conf
```

If there is a key-error run `rndc-confgen -a` and try to start it again. If you are getting the error: "loading configuration: Permission denied", use the following commands to correct the error:

```
chmod 777 /usr/local/etc/rndc.key
chmod 777 /usr/local/etc/bind.keys
```

Now, turn on the Authoritative name servers in a different environment terminal: Navigate to the Authoritative server folders ("/env/nsd_root", "/env/nsd_attack_home" and "/env/nsd_attack_fun".), then run in each authoritative server folder: `nsd -c nsd.conf -d -f nsd.db`

If there is an error stating that the port is already in use, run `service nsd stop` and start it again.

A.2.2 Basic Test

To make sure that the setup is ready and well configured, the following steps are required:

1. Run another shell inside the docker container using `docker exec -ti <container id> bash` and run `tcpdump -i lo -s 65535 -w /app/dump.pcap`

2. Query the resolver from within the docker container `dig firewall.fun.lan` and make sure that the correct IP address is received, you should see `Address: 127.0.0.207`

3. Stop `tcpdump` (you can use `^C`), Open WireShark, load the file `<local_folder_path>/dump.pcap` and filter DNS requests. You should observe the whole DNS resolution route for the domain name requested (`firewall.fun.lan`).

   (a) `firewall.fun.lan` query from client to resolver (ip `127.0.0.1` to ip `127.0.0.1`)
   (b) Resolver query to the root server (from `127.0.0.1` to `127.0.0.2`)
   (c) Root server return the SLD address (from `127.0.0.2` to `127.0.0.1`)
   (d) Resolver query the SLD (from `127.0.0.1` to `127.0.0.100`)
   (e) SLD return the address for the domain name (`127.0.0.207`)
   (f) Resolver return the address to the client (`127.0.0.207`)

NOTE: The address `firewall.fun.lan` is configured in `/env/nsd_attack/fun.lan.forward` and by performing the above test ensures that the resolver accesses the authoritative through the root server.

A.3 Evaluation workflow

As explained in Appendix A.1.5, in order to test CacheFlushAttack using DNS-CacheFlushSimulator a client, a resolver (Bind 9.18.21 which was the version tested in CacheFlushAttack paper) and four authoritative servers with pre-configured zone files are required. See Appendix A.1.5 for detailed example of such zone file configuration.

A.3.1 Major Claims

(C1): As part of CacheFlushNS, the entire referral list is stored in the benign cache along with the IP addresses of the first 20 NS in the list. This is proven by experiment (E1) below, as described in Section 4 and whose results are reported in Figure 1. The reproduction (proof) of this claim does not require significant resources of any sort, neither compute nor memory.

(C2): As part of CacheFlusCNAME, a chain of 17 records is stored in the benign cache. This is proven by experiment (E2) below, as described in Section 4 and whose results are reported in Figure 2. The reproduction (proof) of this claim does not require significant resources of any sort, neither compute nor memory.

(C3): Each CacheFlushNS malicious request may insert at least 100KB into the benign cache (The CAF defined in Section 4). This is proven by experiment (E3) below, as described in Section 5 and whose results are reported in Figure 1. The reproduction (proof) of this claim does not require significant resources of any sort, neither compute nor memory.

(C4): The resolver exhibited a significant performance degradation in its throughput measurements during the CacheFlushAttack. This is proven by experiment (E4) below, as described in Section 5 and whose results are reported in Figure 8.

(C5): When the benign and the attacker querying rates are at a constant uniform rate. A cache miss would occur iff, the benign rate is lower than the flushing rate (Equation 1). This is proven by experiment (E5) below, as described in Section 5 and whose results are reported in Figure 5.

(C6): All benign domains with lower rate than the border rate, will suffer with high probability cache miss under attack rate (Equation 2) This is proven by experiment (E6) below, as described in Section 5 and whose results are reported in Figure 7.

(C7): The resolver average cache miss percentage can be calculated for a given distribution (Equation 3) This is proven by experiment (E7) below, as described in Section 5 and whose results are reported in Table 1.

(C8): The proposed CacheFlushAttack mitigation greatly reduce the attacks effectiveness (see Section 6). This is proven by experiment (E8) below.

A.3.2 Experiments

(E1): Fill the benign cache with one CacheFlushNS domain. (5 human minutes + 1 compute minutes):
Preparation: First, make sure that your resolver is configured to use Bind9.18.21 resolver (first run: `cd /env/bind9_18_21` and then run: `make install`). Turn on the resolver with the following commands:

```
cd /etc
named -g -c /etc/named.conf
```

In addition, the malicious referral response should include a long list of name servers, in order to create such referral list the "/env/nsd_attack_home/home.lan.forward" zone file needs to have 1900 records per one malicious request. For example, you can create one malicious request using a short script we provided (/env/reproduction/CacheFlushNS_zone_file.py) that generates the malicious request configuration and copy its output into the zone file For your convenience we uploaded our "/env/nsd_attack_home/home.lan.forward" zone file which includes our attack.
Execution: Query the resolver with a malicious query (e.g., "dig attack0.home.lan").
Results: Dump the cache using `rndc dumpdb -cache`. Open the Cache file using `less /etc/bind/named_dumpdb` and see the "attack0.home.lan" domain with 1900 NS referral list along with the IP addresses of the first 20 NS in the list at the end of the cache file.

(E2): Fill the benign cache with one CacheFlushCNAME domain. (5 human minutes + 1 compute minutes):
Preparation: First, make sure that your resolver is configured to use Bind9.18.21 resolver (first run: `cd /env/bind9_18_21` and then run: `make install`). Turn on the resolver with the following command `named -g -c /etc/named.conf`. In addition, the malicious referral response should include a long CNAME chain, in order to create such CNAME cahin the "/env/nsd_attack_home/home.lan.forward_cname" zone file needs to have a long chain per one malicious request. For example, you can create one malicious request using a short script we provided (/env/reproduction/CacheFlushCNAME_zone_file.py) that generates the malicious request configuration and copy its output into the zone file For your convenience we uploaded our

"/env/nsd_attack_home/home.lan.forward_cname" zone file which includes our attack.

Execution: Query the resolver with a malicious query (e.g., "dig attack1.home.lan").

Results: Dump the cache using `rndc dumpdb -cache`. Open the Cache file using `less /etc/bind/named_dumpdb` and a CNAME chain of 17 domains starting with the name "attack1.home.lan" is shown in the benign part of the cache file.

(E3): Insert at least 100KB into the benign cache (the CAF). (5 human minutes + 1 compute minutes):

Preparation: Same as for (E1)

Execution: Same as for (E1)

Results: Check the cache size using `ls -l /etc/bind/named_dumpdb` before and after querying the malicious domain. Cache size has been increased by 100KB. Reminder: the increase depends on the length of the domain names.

(E4): Resolver throughput. (40 human minutes + 20 compute minutes):

Preparation: First, make sure that your resolver is configured to use Bind9.18.21 resolver (first run: `cd /env/bind9_18_21` and then run: `make install`). Turn on the resolver with the following command `named -g -c /etc/named.conf`. In addition, configure the authoritative name servers zone files, for CacheFlushNS use "/env/nsd_attack_home/home.lan.forward" and for CacheFlushCNAME use "/env/nsd_attack_home/home.lan.forward_cname". In "nsd" folders, run the authoritative name servers using the command: `nsd -c nsd.conf -d -f nsd.db`. You can create a malicious domains file and a benign file using a short script we provided (/env/reproduction/Resperf_Benign_Domains.py and /env/reproduction/Resperf_Attack_Domains.py). For your convenience we uploaded an attack file at: "/env/reproduction/Attack_domains.txt" and a benign file at: "/env/reproduction/Benign_domains.txt".

Execution: Open two clients, at the attacker client run:

```
resperf -d /env/reproduction/Attack_domains.txt -s 127.0.0.1 -v -m <attack_rate> -C 1000 -q 640000 -r 0 -c 100 -R
```

At the benign client run:

```
resperf -d /env/reproduction/Benign_domains.txt -s 127.0.0.1 -v -m <attack_rate> -C 1000 -q 640000 -r 0 -c 100 -R
```

Results: Check the "Maximum throughput" in the resperf statistics results.

(E5): Equation 1. (40 human minutes + 20 compute minutes):

Preparation: In addition to the preparations described in (E4), both clients and authoritative name servers should have a WireShark installed. To utilize tshark for streamlined analysis, making data parsing and processing straightforward, use this command:(`docker container run --dns 127.0.0.1 --cap-add=NET_ADMIN --mount type=bind,source=<local_folder_path>,target=/app -it shohamd/cacheflushsimulator:latest /bin/bash`). Create a benign file with only one domain, you can use t short script we provided (/env/reproduction/Resperf_Benign_Domains.py) by set DOMAINS_NUM=1.

Execution: On the benign client, record the benign packets using this command:

```
tshark -i any -Y 'dns contains "fun"' -T fields -e frame.number -e frame.time -e ip.src -e ip.dst -e tcp.srcport -e tcp.dstport -e udp.srcport -e udp.dstport -e dns.id -e dns.qry.name -e dns.flags.rcode > /env/reproduction/tshark_benign_client.txt
```

On the authoritative, record the benign packets using this command:

```
tshark -i any -Y 'dns contains "fun"' -T fields -e frame.number -e frame.time -e ip.src -e ip.dst -e tcp.srcport -e tcp.dstport -e udp.srcport -e udp.dstport -e dns.id -e dns.qry.name -e dns.flags.rcode > /env/reproduction/tshark_auth.txt
```

At the attacker client run:

```
resperf -d /env/reproduction/Attack_domains.txt -s 127.0.0.1 -v -m <attack_rate> -C 1000 -q 640000 -r 0 -c 100 -R
```

and at the benign client run:

```
resperf -d /env/reproduction/Benign_domains.txt -s 127.0.0.1 -v -m <benign_rate> -C 1000 -q 640000 -r 0 -c 100 -R
```

Results: Analyze the output files using the script in "/env/reproduction/CacheMiss.py". There is a cache miss for each domain included in the script result.

Note: Since this is a lab environment rather than the cloud environment described in the article, the outcomes may differ as there are no delays and all virtual machines run on a single physical machine.

(E6): Equation 2. (40 human minutes + 20 compute minutes):

Preparation: Same as (E5)

Execution: Same as (E5). Using the "University-Domain.txt" file for the benign client. using this command:

```
resperf -d /env/reproduction/University_domains.txt -s 127.0.0.1 -v -m <attack_rate> -C 1000 -q 640000 -r 0 -c 100 -R
```

Results: Same as (E5)

(E7): Equation 3. (40 human minutes + 20 compute minutes):

Preparation: Same as (E6)

Execution: Same as (E6).

Results: The last row in the "/env/reproduction/CacheMiss.py" output is the average.

(E8): Mitigation. (40 human minutes + 20 compute minutes):

Preparation: First, make sure that your resolver is configured to use Bind9.18.21 resolver (first run: `cd /env/bind9_18_21` and then run: `make install`). Turn on the resolver with the following command: `named -g -c /etc/named.conf`. For CacheFlushNS attack, the mitigate referral response should include 20 records per one malicious request. You can create such a zone file using a short script we provided (/env/reproduction/CacheFlushNS_mitigation.py) For your convenience we uploaded an example in "/env/nsd_attack_home/home.lan.forward_NS_mitigation". For CacheFlushCNAME attack, the mitigate referral response should include a CNAME chain of length of 8 per one malicious request. You can create such a zone file using a short script we provided (/env/reproduction/CacheFlushCNAME_mitigation.py) For your convenience we uploaded an example in "/env/nsd_attack_home/home.lan.forward_CNAME_mitigation".

Execution: Same as (E5)

Results: Same as (E4) and (E7).

B Notes on Reusability

The simulator above was built based on a simulator that was created for NRDelegationAttack [1], and it was modified to meet our needs. Those who conduct research on DNS can use this simulator as it contains all of the necessary components, and it can be easily modified by adding authoritative name servers, updating resolver versions, modifying authoritative zone files, and modifying all component configurations.

Acknowledgements: The authors are grateful to the USENIX Security artifact referees for their dedicated careful review and discussions which have significantly improved the artifact.

B.1 Version

Based on the LaTeX template for Artifact Evaluation V20220926. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at `https://secartifacts.github.io/usenixsec2024/`.

References

[1] AFEK, Y., BREMLER-BARR, A., AND STAJNROD, S. Nrdelegationattack: Complexity ddos attack on DNS recursive resolvers. In *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023* (2023), J. A. Calandrino and C. Troncoso, Eds., USENIX Association, pp. 3187–3204.