USENIX'23 Artifact Appendix: NRDelegationAttack: Complexity DDoS attack on DNS Recursive Resolvers

Yehuda Afek [*]
Tel-Aviv University
afek@tauex.tau.ac.il

Anat Bremler-Barr [†]
Tel-Aviv University
anatbr@tauex.tau.ac.il

Shani Stajnrod
Reichman University
shaaniba93@gmail.com

A Abstract

To fully understand the root cause of the NRDelegationAttack and to analyze its amplification factor, we developed mini-lab setup, disconnected from the Internet, that contains all the components of the DNS system, a client, a resolver, and authoritative name servers. This setup is built to analyze and examine the behavior of a resolver (or any other component) under the microscope. 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-FullProtocolSimulator is an Inner-Emulator environment for DNS protocol which was built as part of NRDelegationAttack research. DNS-FullProtocolSimulator includes a client, resolver and three authoritative name servers. The resolver is a BIND9 recursive resolver with both the NXNSAttack patched version (BIND9 version 9.16.6) and a pre-NXNS version (BIND9 version 9.16.2). The three authoritative name servers are: a 'root', an attacker, and a malicious delegation authoritative name servers.

Most of the NRDelegationAttack measurements were carried out on a BIND9 version 9.16.6 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/ShaniBenAtya/dnssim*).

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

[*]Member of the Checkpoint Institute of Information Security.

[†]The majority of this research was carried out while the author was at Reichman University. Member of the Checkpoint Institute of Information Security.

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-FullProtocolSimulator source code can be found at DNS-FullProtocolSimulator GitHub (1.4 Tag). The environment docker image can be accessed through DockerHub (1.7 Tag).

A.1.3 Hardware dependencies

There is no hardware dependencies required for using DNS-FullProtocolSimulator. 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. Kcachegrind (To install Kcachegrind on Ubuntu use: `apt install kcachegrind`).

A.1.5 Benchmarks

In order to conduct the experiments described in NRDelegationAttack paper (Section 5), the setup should contain a resolver with a non vulnerable to NXNSAttack version (e.g. bind-9.16.6) and at least two authoritative servers (local root authoritative and at least one more authoritative to simulate the "referral.com" authoritative in Figure 1). In addition, a malicious zone file is required for the attacker authoritative (i.e., "home.lan" server). The malicious zone file may contain

as many malicious name servers as necessary for the specific test, we used malicious referral list with 1500 NRDelegation name servers 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 instance, if the malicious request from the client is "attacker.home.lan", the malicious referral response should include a long list of name servers. In order to create such referral list the "/env/nsd_attack/home.lan.forward" zone file needs to have 1500 records per one malicious request. That is, the malicious zone file includes 1500 records for the malicious request which leads to a none existent domain name (e.g., attacker IN NS ns0...1500.fake0...1500.fake) and the root 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 127.0.0.89).

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 ... (127.0.0.200) "home.lan"
    │              malicious authoritative
    │              server configuration
    │              folder, (which simulates
    │              'referral.com" server in
    │              Figure 1).
    ├── home.lan.forward ...
    │   Zone file for sld
    │   ".home.lan", this sld
    │   represents the malicious
    │   authoritative.
    ├── 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.
    .3 named.conf ...
    Bind9 configuration,
    contains the IP address
    of the local environment.
├── bind9_16_6, bind9_16_2 OR bind9_16_33 ...
│   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 shanist/dnssim:1.7`).
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 shanist/dnssim:1.7 /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 `dnssim` 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 NRDelegationAttack paper, the setup needs to include a resolver and at least two 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. 127.0.0.200 – "home.lan" TLD authoritative (which simulates "referral.com" server in Figure 1).
5. 127.0.0.53 – The default resolver – DO NOT USE IT WHILE TESTING!

Authoritative Servers: Our authoritative servers are located at "/env/nsd_root" and "/env/nsd_attack". 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: First, go to the resolver implementation folder (We have bind-9.16.2 (Which is vulnerable to NXNSAttack), Bind-9.16.6 (Which is non-vulnerable to NXNSAttack and vulnerable to NRDelegationAttack) and bind-9.16.33 (Which is non-vulnerable to both attacks)). You can easily replace the Bind9 version by going to the correct Bind9 version folder (e.g.,"/env/bind9_16_6", "/env/bind9_16_2" or "/env/bind9_16_33") and run: `make install`. NOTE: The environment is pre-installed with Bind 9.16.6 which was the main Bind9 resolver version tested in NRDelegationAttack paper.

Starting the environment: Open three 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 servers in a different environment terminal: Navigate to the Authoritative server folder (/env/nsd_attack and /env/nsd_root), 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`
2. Query the resolver from within the docker container `dig firewall.home.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` and filter DNS requests. You should observe the whole DNS resolution route for the domain name requested (`firewall.home.lan`).
   (a) `firewall.home.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 `127.0.0.200`)
   (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.home.lan` is configured in `/env/nsd_attack/home.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 NRDelegationAttack using DNS-FullProtocolSimulator a client, a resolver (The environment is pre-installed with Bind 9.16.6 which was the main Bind9 resolver version tested in NRDelegationAttack paper) and at least two 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): If the number of names in the referral list is large, e.g., 1,500, then each NRDelegationAttack malicious packet costs at least 5,600 times more CPU instructions relative to a benign query. This is proven by experiment (E1) below, as described in Section 5.2 and whose results are reported in Figure 3. The reproduction (proof) of this claim does not require significant resources of any sort, neither compute nor memory (10 human minutes + 2 compute minutes).

(C2): The resolver exhibited a significant performance degradation in its throughput measurements during the NRDelegationAttack. This is may be proven by experiment (E2) below, as described in Section 6.2 and whose results are reported in Figure 5 in the paper. While in the paper this test and measurements were done on a cloud setup with each DNS component implemented on a separate server, here we show that the same test can be carried out on the closed virtual setup. The results on this isolated virtual setup provide an indication of the phenomena and are

not reliable as those presented in Section 6 in the paper. The reproduction (proof) of this claim does not require significant resources of any sort, neither compute nor memory (20 human minutes + 4 compute minutes).

(C3): The attack is empowered mainly due to the NXNSAttack mitigations: NRDelegationAttack is much worse for NXNS-patched servers than unpatched servers (See Section 4). This is proven by experiment (E3) below. The reproduction (proof) of this claim does not require significant resources of any sort, neither compute nor memory (10 human minutes + 2 compute minutes).

(C4): The proposed NRDelegationAttack mitigation greatly reduces the attacks effectiveness (see Section 8). This is proven by experiment (E4) below. The reproduction (proof) of this claim does not require significant resources of any sort, neither compute nor memory (10 human minutes + 2 compute minutes).

(C5): NRDelegationAttack affects open resolvers as well as the vendors. This claim is problematic to reproduce due to ethical considerations, in addition, most of the open resolvers patched their implementations to NRDelegationAttack as part of the responsible disclosure procedure (see Section 7 in the paper).

A.3.2 Experiments

For the following experiments, Resperf, Valgrind and Kcachegrind tools are needed.

(E1): Instructions measurement experiment

Preparation: For this experiment Valgrind and Kcachegrind are required:
First, make sure that your resolver is configured to use Bind9.16.6 resolver which is patched to NXNSAttack (first run: `cd /env/bind9_16_6` and then run: `make install`). Turn on the resolver with the Valgrind tool with the following command (make sure to run the resolver from "/etc") `valgrind --tool=callgrind 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.lan.forward" zone file needs to have 1500 records per one malicious request. For example, you can create one malicious request using a short script we provided (`python /env/reproduction/genAttackers.py`) that generates the malicious request configuration and copy its output from the "attackerNameServers.txt" output file into the zone file For your convenience we uploaded our "/env/nsd_attack/home.lan.forward" zone file which includes our attackers to "/env/reproduction" folder.

Execution: Query the resolver with a malicious query (e.g., "dig attack0.home.lan"). Stop the resolver and restart it using with Valgrind as explained before. Query the resolver with a legitimate query (e.g., "dig test.home.lan").

Results: Copy the results file from the docker container `/etc/callgrind.VALGRIND_TEST_NUMBER` (which is the folder from which the resolver is executed) to the host, `cp /path/in/docker/callgrind.VALGRIND_TEST_NUMBER /app/` so you could access the file in `<local_folder_path>` alternatively you can use (`docker cp <CONTAINER_ID>:/path/in/docker/callgrind.VALGRIND_TEST_NUMBER /path/in/host`.
Note that the `VALGRIND_TEST_NUMBER` is a number given by Valgrind. Open the results files in Kcachgrind: first add permissions to open the file (`sudo chown USERNAME:USERNAME OUTFILE_NAME`) and then open the file: (`kcachgrind ./OUTFILE_NAME`).
In the tool, choose `Instructions Fetch` tab and record the `Incl.` value of `fctx_getaddresses` function.
Please make sure that the "relative" button is unchecked.
Repeat this step with each file and compare the results.
Benign query results should be around 200,000 instructions, while the malicious query should have more than 2,000,000,000.

(E2): Throughput measurement experiment

Preparation: For this experiment Resperf is required:
To configure the malicious authoritative zone file ("/env/nsd_attack/home.lan.forward" file) with multiple malicious domain names (multiple attackers, each of them configured as explained in E1 and multiple benign domain names, you can use the script we provided (`python /env/reproduction/genAttackers.py`) and change the number of attackers generated by changing the ATTACKERS_NUM variable (e.g. ATTACKERS_NUM = 50). Note that the zone file length is bounded by the file size, therefore we used only 50 different attacker malicious requests in our measurements. In addition, we uploaded a script that generates a list of names for the Resperf tool (`python /env/reproduction/genNamesToCheck.py`), the script creates two output files: "benignNamesE2.txt" for benign user domain names and "attackerNamesE2.txt" for that attacker.

Execution: First make sure that the resolver is using Bind 9.16.6 version (checking Bind9 version can be done by running `named -v` on the resolver terminal and changing its version can be done by running: `cd /env/bind9_16_2` and then: `make install`) and that both the resolver and authoritative servers are running.
Benign and malicious users commands are being executed from two terminals inside the docker container, so two instances of Resperf tool are required:
The first simulates the attacker and issues queries each time at a fixed rate, and the second tool ramps up the benign user requests until things start to fail.
The malicious user command should be run first but ultimately in parallel to the benign user command.
In your benign user run: `resperf -d INPUT_FILE -s`

`127.0.0.1 -v -R -P OUTPUT_NAME`.
And from the malicious user run: `resperf -d INPUT_FILE -s 127.0.0.1 -v -m 15000 -c 60 -r 0 -R -P OUTPUT_NAME`
Where: -m is the number of QPS that are sent, -c is the duration of time in which Resperf tries to send the queries, -r is the duration of time in which Resperf ramps-up before sending the packets in a constant time, we want the ramp-up to be zero and OUTPUT_NAME is the output file name of your choice (make sure to use different file names for each test).
The benign and malicious input files should include only benign or malicious domain names respectively.
You should run two "sub"-experiments: First, you need to measure the effect of the attack on benign users throughput, in this experiment "INPUT_FILE" = "benignNamesE2.txt" for benign user, and "INPUT_FILE" = "attackerNamesE2.txt" for attacker user. Then, to measure the resolver throughput without any attack restart the resolver (in order to test with clean cache), and run the experiment using the "benignNamesE2.txt" file created using the "genNamesToCheck.py" script as the "INPUT_FILE" to both Resperf commands.

Results: Open only the benign output files from both "sub"-experiments using a text editor (from both the benign and attacker terminal) and compare the benign user throughput presented in the "responses_per_sec" column. A major obstacle in reproducing the experiment was the use of docker instead of multiple clients. The experiment required different computers to send attack and benign queries simultaneously and was originally done using our cloud setup environment as described in Section 6.2 of, but using the docker forced us to use only one machine for all the tests. Nevertheless when performing the test within the docker, we are still able to observe that the resolver throughput for benign users while the attack takes place is degraded. The difference was much smaller than in the original experiment (Sec 6.2), but it was statistically significant and can be mainly seen at the first 10 lines of the output file (in which the attack QPS is high and the resolver throughput is degraded by more than 90% (actual_qps are around 5000 but the responses_per_sec are around 100) or even complete denial of service).
Nevertheless, we are still able to observe that the resolver throughput for benign users while NRDelegationAttack takes place is degraded. The difference was much smaller than in the original experiment, but it was statistically significant and can be mainly seen at the first 10 lines of the file (in which the attack QPS is high and the resolver throughput is degraded by more than 90% ("actual_qps" are around 5000 but the "responses_per_sec" are around 100) or even complete denial of service).

(E3): Instructions measurement experiment - NXNSAttack unpatched server
Preparation: For this experiment follow E1 instructions using Bind9.16.2 resolver (which is not patched to NXNSAttack) instead of Bind9.16.6 resolver (which is patched to NXNSAttack). Checking Bind9 version can be done by running `named -v` on the resolver terminal and changing its version can be done by running: **`cd /env`**`/bind9_16_2` and then: `make install`).
Execution: Follow E1 instructions.
Results: Follow E1 instructions. Benign query results should be around 200,000 instructions, while the malicious query should be around 200,000,000.

(E4): Instructions measurement experiment - NRDelegationAttack mitigation
Preparation: For this experiment follow E1 instructions using Bind9.16.33 resolver (which is not patched to NRDelegationAttack). Checking Bind9 version can be done by running `named -v` on the resolver terminal and changing its version can be done by running: **`cd /env`**`/bind9_16_2` and then: `make install`).
Execution: Follow E1 instructions.
Results: Follow E1 instructions. Benign query results should be around 200,000 instructions, while the malicious query should less than 10,000,000.

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

A.4 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/usenixsec2023/.