USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Your Shield is My Sword: A Persistent Denial-of-Service Attack via the Reuse of Unvalidated Caches in DNSSEC Validation

Shuhan Zhang, *Tsinghua University and Tsinghua Shenzhen International Graduate School;* Shuai Wang and Li Chen, *Zhongguancun Laboratory;* Dan Li and Baojun Liu, *Tsinghua University*

https://www.usenix.org/conference/usenixsecurity25/presentation/zhang-shuhan

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium is sponsored by USENIX.

USENIX Security '25 Artifact Appendix: Your Shield is My Sword: A Persistent Denial-of-Service Attack via the Reuse of Unvalidated Caches in DNSSEC Validation

Shuhan Zhang*‡, Shuai Wang†, Li Chen†, Dan Li*, Baojun Liu*
*Tsinghua University, †Zhongguancun Laboratory
‡Tsinghua Shenzhen International Graduate School

A Artifact Appendix

A.1 Abstract

This artifact is the Proof-of-Concept of the RUC attack. It includes the Dockerfiles of the tested DNS software and the victim domains' nameserver, scripts for the three RUC attack variants, and detailed instructions to facilitate the vulnerability reproduction within a controlled environment.

A.2 Description & Requirements

In this section, we first discuss the security, privacy and ethical concerns of our artifact. Then, we provide its access link, and demonstrate its hardware and software dependencies.

Specifically, our artifact consists of the following components:

- Dockerfiles of the tested DNS software and authoritative nameservers of the victim domains (see "dockers/").
- Scripts to launch each of the three RUC attack variants within the controlled environment (see "poc_scripts/").
- Detailed instructions on the setup of the test environment (see "environment_setup.md") and steps to reproduce three RUC variants (see "ruc_reproduction.md").

Details about the artifact composition and the file structure are demonstrated in the file "README.md".

A.2.1 Security, privacy, and ethical concerns

First, for the tested DNS resolver software, we collect the publicly available version of each software from the vendor's official website. All the tests are performed within a controlled environment set up on the local machine. Specifically, for open-source software, we build Docker images and run them in containers. For closed-source software, *e.g.*, Microsoft DNS, we provide a virtual machine with the DNS service installed. Second, for the victims of the RUC attack, the tests in our artifact target the victim domains under our controlled apex zone. We simulate an on-path attacker located with the victim domains' nameserver in a local Docker container, ensuring no harm to DNS domains in the real world. Third, we have excluded any scripts or datasets that could potentially be exploited by adversaries to launch real-world attacks, *e.g.*, the scripts for large-scale open resolver measurement. Moreover, this artifact does not involve any sensitive data such as personal information. Therefore, we believe that this artifact does not raise ethical issues.

A.2.2 How to access

We have released our artifact on Zenodo, which can be accessed at https://doi.org/10.5281/zenodo.15543846. Please refer to the latest version of our artifact as automatically directed by this concept DOI.

A.2.3 Hardware dependencies

This artifact can run in standard CPU-based environments, and no specialized hardware is required. Nevertheless, to ensure the successful build of the DNS software docker images, we recommend that the host machine for setting up the Linux-based test environment is equipped with a number of CPU cores ≥ 4 and RAM $\geq$ 8GB.

For Microsoft DNS, we provide a virtual machine .ova file that can be imported into a virtual machine tool, *e.g.*, VMware Workstation. Note that this virtual machine is only compatible with host machines based on x86 architecture.

A.2.4 Software dependencies

All the Dockerfiles in this artifact are built on Ubuntu 22.04 with Docker Engine. The Docker Engine can be installed on the host machine using the following command:

```
apt install docker.io
```

The scripts for testing RUC vulnerabilities are executed in Python environment, *e.g.*, Python 3.10.12. All the required Python packages are listed in "poc_scripts/requirement.txt", which can be installed using the `pip` command.

A.2.5 Benchmarks

None.

A.3 Set-up

This section describes the environment setup for the RUC vulnerability reproduction. For detailed, specific explanations of the set-up steps, please refer to "environment_setup.md".

A.3.1 Installation

Resolvers running on Linux. A Docker environment is set up to test the five Linux-based DNS resolvers.

First, create a Docker network named `ruc-test-net` on the local host machine:

```
docker network create --subnet "172.22.0.0/16"
                              ruc-test-net
```

This network is dedicated to the RUC vulnerability test. All DNS components, *i.e.*, the tested resolvers, the authoritative nameserver of the victim domains and the client, will be assigned with an IP address in the subnet `172.22.0.0/16`.

Next, under the base folder, run the following commands to build Docker images and run containers for all the tested Linux-based resolvers:

```
bash start_resolver.sh
```

Microsoft DNS resolver. For Microsoft DNS, we offer a ready-made Windows Server virtual machine (.ova file[1]) with Microsoft DNS installed and DNSSEC validation enabled. After importing it into a virtual machine tool such as VMware Workstation, you can log in to the server as `Administrator` with password `RUC@Sec25`. We have put the test scripts on the Desktop of the virtual machine, *i.e.*, under "C:\Users\Administrator\Desktop\poc_scripts\". All necessary python packages have been installed in advance.

If you intend to manually install Microsoft DNS on a Windows Server virtual machine and configure DNSSEC validation, please refer to the detailed instructions in the manual "dockers/resolver_software/microsoft/microsoft_setup.pdf".

Victim domains and nameserver. For Linux-based resolvers, we support the vulnerability test using our controlled apex zone, `dnssec-ruc.xyz`. Each RUC attack variant corresponds to a dedicated victim subdomain under `dnssec-ruc.xyz`. The authoritative nameserver of these subdomains will be automatically resolved to `172.22.2.1` as configured at our apex zone, *i.e.*, the IP address of the local nameserver container.

Run the following command to build the Docker image for the local nameserver and start a container:

```
bash start_nameserver.sh
```

[1] https://doi.org/10.5281/zenodo.15509714

The container runs the latest version of BIND to provide authoritative DNS service. It also runs an API for the RUC attack, which is invoked by the attack script to simulate on-path response manipulations. Notably, the RUC vulnerabilities of DNS resolvers are independent of specific victim domains, and all resource records of the victim domains under our apex zone are transparent. Please refer to the apex zone file "victim_config/apex_zone.txt" for their referral delegation records (*e.g.*, `NS`, `DS` records) and zone files under "victim_config/bind/" for their authoritative records.

For Microsoft DNS, the victim subdomains are under our controlled zone `dnssec-ruc-ms.xyz`. Their nameserver will be automatically resolved to `47.251.171.85`, *i.e.*, our own DNS server dedicated to testing Microsoft DNS.

RUC attacker. Use the following command to build a Docker image and start a container for the RUC attacker:

```
bash start_attacker.sh
```

This command will mount the PoC scripts ("poc_scripts/") under the "/root/" directory in the container. For simplicity, we simulate DNS queries of ordinary clients to verify the RUC attack within the same container as the attacker.

Pull ready-made docker images from DockerHub. Alternatively, you can pull the ready-made Docker images of the resolver software (except Microsoft DNS), nameserver and RUC attacker from DockerHub (see "dockers/dockerhub.md").

A.3.2 Basic Test

To verify the environment setup, we issue DNS queries from the RUC attacker container to each tested resolver, requesting a DNSSEC-signed domain hosted on the local nameserver. Run the following commands for this basic test:

```
cd poc_scripts
bash test_basic.sh
```

The outputs of the basic test will be stored in "poc_scripts/basic_test_result/log_basic_test.csv". All entries should be tagged as "succeed" if the setup is correct, *i.e.*, the tested resolvers are running with DNSSEC validation enabled, and the nameserver is also functioning properly.

For Microsoft DNS, on the Windows Server virtual machine, enter the folder "poc_scripts" located on the Desktop and run the script "test_basic_microsoft.ps1" using PowerShell. The outputs will be stored in "log_basic_test-microsoft.csv" under the folder "basic_test_result". Similarly, there should be "succeed" in the output, indicating a successful environment setup.

A.4 Evaluation workflow

This section illustrates the steps to reproduce the three RUC attack variants (see more details in "ruc_reproduction.md").

A.4.1 Major Claims

(C1): The vulnerabilities of the tested DNS resolver software against each RUC variant are consistent with those detailed in Section 5.1, Table 2 of the main paper. For Linux-based resolvers, the reproduction of attack variants $\mathsf{RUC}_{\mathsf{SEC}}$ ($\mathsf{RUC}_{\mathsf{DNSKEY}}$ and $\mathsf{RUC}_{\mathsf{DS}}$), $\mathsf{RUC}_{\mathsf{NSIP}}$, and $\mathsf{RUC}_{\mathsf{EDNS0}}$ corresponds to experiment **E1**, **E2**, **E3** and **E4**, respectively. For Microsoft DNS, the reproduction of all attack variants corresponds to experiment **E5**.

A.4.2 Experiments

The PoC script of the RUC attack is "poc_scripts/ruc_poc.py". The workflow of this script is as follows:

1. `config_authns('inject')`: A response manipulation program is started by invoking an API (see "victim_config/attack_api/") on the victim domain's nameserver, simulating a (short-lived) on-path attacker.
2. `inject_cache()`: The attacker injects forged, unvalidated data into the cache of the target resolver, *e.g.*, `DNSKEY` without `RRSIG` for an $\mathsf{RUC}_{\mathsf{DNSKEY}}$ attack.
3. `config_authns('resume')`: The manipulation program at the nameserver side is stopped, and the original, benign service of the nameserver is resumed.
4. `verify_dos()`: The ordinary client sends routine DNS queries demanding DNSSEC validation to the target resolver. Responses from the resolver are recorded and checked if they carry valid answers. If no answer is returned (*i.e.*, `SERVFAIL` or timeout), the resolver is vulnerable to the corresponding RUC variant.

The test results of the RUC vulnerabilities are logged in "poc_scripts/ruc_test_result/log_ruc_test.csv" (for Microsoft DNS, the file name is "log_ruc_test-microsoft.csv"). Each line of the file is in the following format:

```
{resolver software}|{resolver IP}|{RUC variant}|
{with RRSIG}|{vulnerable}|{query failure rate}
```

The field "**vulnerable**" indicates the vulnerability of the tested resolver against the specific RUC attack variant, which should be consistent with Table 2 of the main paper.

For testing Linux-based resolvers in **E1-E4**, first enter the folder "poc_scripts":

```
cd poc_scripts
```

Then, run the corresponding test script for each RUC attack variant as follows.

(E1): [Test $\mathsf{RUC}_{\mathsf{DNSKEY}}$ of $\mathsf{RUC}_{\mathsf{SEC}}$] [10 human-minutes]:
Execution:

```
bash test_ruc_dnskey.sh
```

Results: For vulnerable resolvers, the "vulnerable" field in the test result output should be "True", *i.e.*, all responses of the verifying queries should be `SERVFAIL`.

(E2): [Test $\mathsf{RUC}_{\mathsf{DS}}$ of $\mathsf{RUC}_{\mathsf{SEC}}$] [10 human-minutes]:
Execution:

```
bash test_ruc_ds.sh
```

Results: For vulnerable resolvers, the "vulnerable" field in the test result output should be "True", *i.e.*, all responses of the verifying queries should be `SERVFAIL`.

(E3): [Test $\mathsf{RUC}_{\mathsf{NSIP}}$] [5 human-minutes]:
Execution:

```
bash test_ruc_nsip.sh
```

Results: For vulnerable resolvers, the "vulnerable" field in the test result output should be "True", *i.e.*, all responses of the verifying queries should be `SERVFAIL` or time out.

(E4): [Test $\mathsf{RUC}_{\mathsf{EDNS0}}$] [5 human-minutes]:
Execution:

```
bash test_ruc_edns0.sh
```

Results: For vulnerable resolvers, the "vulnerable" field in the test result output should be "True", *i.e.*, all responses of the verifying queries should be `SERVFAIL`.

(E5): [Test Microsoft DNS] [5 human-minutes]:
Execution: Run "C:\Users\Administrator\Desktop\poc_scripts\test_microsoft.ps1" using PowerShell on the Windows Server virtual machine.
Results: For attack variants to which Microsoft DNS is vulnerable resolvers, the "vulnerable" field in the test result output should be "True", *i.e.*, all the response status codes of the verifying queries should be `SERVFAIL`.

If you are interested in a specific resolver, please refer to the commands in "ruc_reproduction.md" to specify the tested resolver and inspect its behavior under a given RUC variant. This document also demonstrates the detailed workflow and the input options of the PoC script.

A.5 Notes on Reusability

Apart from reproducing the proposed RUC vulnerability, we believe that the Dockerfiles and virtual machine of DNS resolvers can also facilitate the testing or reproduction of other DNS vulnerabilities within the controlled environment. Moreover, researchers can reuse the Dockerfiles to install arbitrary versions of the resolver software by simply changing the version number in the corresponding file.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2025/.