ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE
ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL
ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

USENIX Security '25 Artifact Appendix: X.509DoS: Exploiting and Detecting Denial-of-Service Vulnerabilities in Cryptographic Libraries using Crafted X.509 Certificates

Bing Shi[1], Wenchao Li[1], Yuchen Wang[1], Xiaolong Bai[1]*, Luyi Xing[2]*

[1]*Alibaba Group*, [2]*Indiana University Bloomington*
{*shibing.shi*, *huiren.lwc*, *tianwu.wyc*}*@alibaba-inc.com*, *bxl1989@gmail.com*, *luyixing@iu.edu*

A Artifact Appendix

This artifact appendix is meant to be a self-contained document which describes a roadmap for the evaluation of our artifact.

A.1 Abstract

X.509DoSTool is developed to facilitate the rapid generation of crafted certificates and the automatic detection of DoS risks in cryptographic implementations. The tool primarily provides three commands for users: the `generate` command creates crafted certificates that can be used to identify specific DoS risks discussed in our paper; the `edit` command allows modification of ASN.1 objects within a certificate without violating DER encoding rules; and the `detect` command identifies specific issues in the implementations of the cryptographic libraries mentioned in our paper. The artifact evaluation should primarily focus on these three commands. We provide the source code of the tool and the necessary test scripts to facilitate the evaluation process.

A.2 Description & Requirements

This section lists the information necessary to recreate the same experimental setup we have used to run our artifact.

A.2.1 Security, privacy, and ethical concerns

Conducting the artifact evaluation for *X.509DoSTool* does not raise any security, privacy, or ethical concerns.

A.2.2 How to access

The artifact is available at our Zenodo repository (please refer to v1.0.2 and later): `https://zenodo.org/records/14726326`.

*Xiaolong Bai and Luyi Xing are corresponding authors.

A.2.3 Hardware dependencies

No special hardware dependencies are required for our artifact. Our testing environment is built on a Linux server equipped with 4GB of RAM and a 4-core CPU, running Ubuntu v22.04.

A.2.4 Software dependencies

The software dependencies required to run this tool primarily include the following:

- Python v3.10.12
- pip v22.0.2
- setuptools v75.8.0 (Python package)
- psutil v6.0.0 (Python package)
- pyasn1 v0.6.0 (Python package)
- pyasn1_modules v0.4.0 (Python package)
- pycryptodome v3.20.0 (Python package)
- OpenSSL v3.0.2

The versions used during testing should match or closely align with those mentioned above, as significant version discrepancies may introduce potential untested issues.

Additionally, before evaluators attempt to use the tool to detect issues in a specific library, the corresponding version of the library (as listed in Table 3 and Table 4 of our paper) needs to be installed first. The libraries and their version information involved include:

- OpenSSL v3.0.0, v1.0.2
- Botan v1.11.21, v1.11.26, v3.2.0, v3.4.0
- Bouncy Castle v1.70, v1.77
- Crypto++ v5.6.4, v8.9
- GnuTLS v3.7.10, v3.7.11
- phpseclib v3.0.18, v3.0.33, v3.0.35

A.2.5 Benchmarks

None.

A.3 Set-up

This section includes the installation and configuration steps required to prepare the environment to be used for the evaluation of our artifact.

A.3.1 Installation

To install, run `pip install .` in the root directory of the repository. This will package the necessary files and create a command named `x509dostool`.

Upon installation, the command is readily available for the root user. However, for non-root users, it is necessary to first run `export PATH=$PATH:~/.local/bin` before the command can be used.

A.3.2 Basic Test

To execute, run `x509dostool`. If the installation is successful, it will display the tool's version number along with a help page containing the three basic subcommands: `generate`, `edit`, and `detect`.

A.4 Evaluation workflow

This section includes the operational steps and experiments required to evaluate the functionality of our artifact and validate the key results and claims of our paper.

A.4.1 Major Claims

Our major claims include the following:

(C1): *X.509DoSTool* can be used to rapidly generate crafted certificates that facilitate the identification of specific DoS risks discussed in our paper.

(C2): *X.509DoSTool* can be used to modify ASN.1 objects involved in a certificate without violating DER encoding rules.

(C3): *X.509DoSTool* can be used to identify specific issues in the implementations of the cryptographic libraries mentioned in our paper.

A.4.2 Experiments

The corresponding experimental steps are as follows:

(E1): Test the `generate` command of `x509dostool`, and use `openssl` to examine the generated certificate.
Preparation: Enter the `test/generate/` directory.
Execution: Run `./test_generate.sh`.
Results: The script prints each executed `generate` command and checks the execution status, as well as the parsing status of the generated certificate using `openssl`. If all commands pass the aforementioned checks, the message *all tests passed* will be displayed. Additionally, running `./test_generate.sh --verbose` will output the intermediate results of the `generate` and `openssl` commands, providing evaluators with further details for evaluation.

(E2): Test the `edit` command of `x509dostool`, and use `openssl` to examine the edited certificate.
Preparation: Enter the `test/edit/` directory.
Execution: Run `./test_edit.sh`.
Results: Upon initial execution, a directory named `certs/` will be created, containing the certificates to be utilized. The script then prints each executed `edit` command and checks the execution status, as well as the parsing status of the edited certificate using `openssl`. If all commands pass the aforementioned checks, the message *all tests passed* will be displayed. Additionally, running `./test_edit.sh --verbose` will output the intermediate results of the `edit` and `openssl` commands, providing evaluators with further details for evaluation.

(E3): Test the `detect` command of `x509dostool`, and compare it with the experimental results presented in Section 7 of our paper.
Preparation: Enter the `test/detect/` directory.
Execution: Run `./test_detect.sh`.
Results: Upon initial execution, a directory named `certs/` will be created, containing the certificates to be utilized. The script then prints the cryptographic libraries present in the current environment along with their respective versions, and outputs the results of the detection of issues within these libraries. Note that the script `test_detect.sh` performs detection against the cryptographic libraries present in the evaluator's current environment. Given the tool's requirements, at least one OpenSSL library will be present in the environment. The evaluator can install the corresponding library of interest in the environment for detection.

A.5 Notes on Reusability

Note that the version numbers of the cryptographic libraries tested, as indicated in our paper, are not strictly required for reproducing the experimental results. In most cases, installing an earlier version to test as many cases as possible for detection is also feasible. However, this does not imply that any version earlier than these can be used, as some issues were introduced only after specific version updates. Additionally, if users wish to apply this tool to libraries beyond those discussed in this paper, they can refer to the scripts in `test/scripts/` for creating corresponding scripts. Users can also specify other parameters during the execution of the `generate` command or use the `edit` command to enrich the contents of the `test/detect/certs/` directory.

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/.