Diffie-Hellman Picture Show: Key Exchange Stories from Commercial VoWiFi Deployments

Gabriel K. Gegenhuber and Florian Holzbauer, *University of Vienna;* Philipp É. Frenzel, *SBA Research;* Edgar Weippl, *University of Vienna and Christian Doppler Laboratory for Security and Quality Improvement in the Production System Lifecycle (CDL-SQI);* Adrian Dabrowski, *CISPA Helmholtz Center for Information Security*

https://www.usenix.org/conference/usenixsecurity24/presentation/gegenhuber

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Open access to the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium is sponsored by USENIX.

USENIX Security '24 Artifact Appendix: Diffie-Hellman Picture Show: Key Exchange Stories from Commercial VoWiFi Deployments

Gabriel K. Gegenhuber[1,2], Florian Holzbauer[1,2], Philipp É. Frenzel[3], Edgar Weippl[1,4], and Adrian Dabrowski[5]

[1]University of Vienna, Faculty of Computer Science, [2]UniVie Doctoral School Computer Science, [3]SBA Research, [4]Christian Doppler Laboratory for Security and Quality Improvement in the Production System Lifecycle (CDL-SQI), [5]CISPA Helmholtz Center for Information Security

A Artifact Appendix

A.1 Abstract

This artifact contains the source code necessary to run client- and server-side evaluation scripts for our VoWiFi security analysis. While the scripts can be used to scan for various security parameters (e.g., ciphers), our evaluation focuses on the key exchange (i.e., the supported Diffie-Hellman (DH) groups and the rekey-timings) that are used for the first (i.e., *phase 1*) VoWiFi tunnel that is essential to the security of the overall communication.

A.2 Description & Requirements

The artifact includes the client-configurations (extracted from smartphone ROMs) that were used for the paper in the dumps.zip file. Our client-side analysis then shows how we extracted and harmonized the VoWiFi parameters from these files. To make it more convenient to repeat the analysis for other ROMs, we also list our approach for extracting the configuration databases from smartphone ROMs of different manufacturers. To extract the used configurations on the server side we provide the source code to actively probe the operators by initiating an IKE (Internet Key Exchange) handshake.

A.2.1 Security, privacy, and ethical concerns

Our server-side scripts actively probe operator's VoWiFi infrastructure and negotiate security associations (SAs) with their their ePDGs (evolved Packet Data Gateways). We tried to reflect normal user behavior whenever possible (e.g., by only sending well-formed requests) and took care not to overstress any mobile operator's infrastructure. Since those systems (e.g., ePDGs) are made to handle traffic for (millions) of customers, we are confident that our probing does no harm.

The server-side scans use *scapy* to send and receive packets and thus require root privileges.

A.2.2 How to access

The artifact's source code is accessible at `https://github.com/sbaresearch/vowifi-epdg-scanning/releases/tag/usenix-sec24-ae`. The repository and sub repositories contain documentation to make the artifact evaluation process as easy as possible.

A.2.3 Hardware dependencies

None. Our scripts can be run on any state-of-the-art computer system (e.g., a personal laptop or a server).

A.2.4 Software dependencies

During our analysis, we used an Ubuntu 22.04.2 LTS system with dual-stack Internet connectivity (to scan for both IPv4 and IPv6). Nevertheless, due to limited IPv6 support on the operator side, running it on an IPv4 only host should work equally well and will not (considerably) alter the results.

All required dependencies for our python scripts are listed in the corresponding `requirements.txt` files. Since we use scapy for sending and receiving packets for the server side probing, some platform-specific files might be necessary. [1]. Moreover, *tcpdump* is required to record the relevant packets for subsequent manual analysis.

A.2.5 Benchmarks

None.

[1] `https://scapy.readthedocs.io/en/latest/installation.html#platform-specific-instructions`

A.3 Set-up

A.3.1 Installation

The installation steps are explained in detail in the repository's README.md file.

Cloning the Repository We use Git LFS (Git Large File Storage) for the dumps.zip file containing client-side operator configurations. To successfully clone the repository including the 400MB dumps.zip file, Git LFS hooks are required.
Thefore:

- Make sure to have Git LFS installed.
- Setup the LFS hooks by running `git lfs install`.
- Clone the repository using `git clone`.

Alternatively, if you do not want to use Git LFS the repository also contains instructions to download the dump.zip file via a mirror at our scientific artifact storage.

Client Side Analysis The Python environment for the client side analysis can be setup as follows:

```
cd client-side
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

Server Side Analysis The Python environment for the server side analysis can be setup as follows:

```
cd server-side
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

A.3.2 Basic Test

The execution steps are explained in detail in the repository's README.md file.

Client Side Analysis The the client side analysis can be easily reproduced using the provided jupyter notebook. It can be run by executing:

```
jupyter notebook client_side_evaluation.ipynb
```

The jupiter notebook extracts the configuration files from the different device databases and saves them in a harmonized json format. Based on the json files, the jupyter notebook allows to generate the graphs that were included into our paper.

Server Side Analysis The serverside scans can be initiated using the following commands:

```
sudo su
source venv/bin/activate
./epdg_scanner.py --testcase SUPPORT_DH_768MODP
./epdg_scanner.py --testcase SUPPORT_DH_1024MODP
./epdg_scanner.py --testcase SUPPORT_DH_1536MODP
```

The probing results can then be found in the results directory. The .txt file contains the security associations that were negotiated with each server. The .pcap file can be used for further (more precise) analysis with Wireshark.

The repository contains simple command line examples that can be used to filter the .txt files containing the results.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): Deprecated DH Groups on the Client Side This corresponds to Section 5 in the paper. The provided scripts can be used to extract the prevalent DH groups (and the corresponding rekey timers) that are preloaded/provisioned to commodity smartphones. The evaluation shows that a huge share of operators rely on deprecated settings.

(C2): Deprecated DH Groups on the Server Side This corresponds to Section 6 in the paper. Our scripts probe the operators live infrastructure, i.e., they try to negotiate a security association using the selected DH group. The evaluation shows that a considerable share of operators still support deprecated DH groups (i.e., 768, 1024, 1536-bit MODP).

A.4.2 Experiments

(E1): *[Client Side Analysis] [1 human-hour + 5GB disk space]: The *dumps.zip* file contains the configuration databases that were extracted from commodity smartphones (more details can be found in the README.md file). The repository provides scripts to extract and evaluate the preloaded settings.*

(E2): *[Server Side Analysis] [3 x 1 hour scanning time (1 hour per group) + 1 hour for evaluation (all groups)]: The server-side probing can be executed for any specific set of security associations (i.e., for a selected DH group). After completing a scan round, the operators supporting the corresponding DH group can be filtered using the provided evaluation scripts.*

A.5 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/usenixsec2024/.