

USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Plain Text, Plain Risks: Measuring HTTP Inclusion in Android WebViews at Scale

Philipp Beer, *TU Wien;* Sebastian Roth, *University of Bayreuth;*
Martina Lindorfer and Marco Squarcina, *TU Wien*

https://www.usenix.org/conference/usenixsecurity26/presentation/beer

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 35th USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Artifact Appendices to the Proceedings of the 35th USENIX Security Symposium is sponsored by



جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology



USENIX Security '26 Artifact Appendix: Plain Text, Plain Risks: Measuring and Understanding HTTP Inclusion in Android WebViews at Scale

Philipp Beer, Sebastian Roth, Martina Lindorfer, and Marco Squarcina

A Artifact Appendix

A.1 Abstract

We provide multiple artifacts to reproduce the results presented in the paper and support future work that builds on our findings. Specifically, we include the following artifacts:

- **APK Download:** Scripts for downloading apps (`/apk-download`). Due to the dataset size, we cannot publicly release the full set of apps used. Reviewers are granted access to it as outlined in Section A.3.1.
- **HTTP Content Behavior Comparison:** Scripts to test the HTTP content behavior in WebViews and mobile browsers (`/http-content-comparison`).
- **Static Analysis Pipeline:** Code to run the static analysis, including the manifest analysis and the bytecode analysis (`/static-analyzer`).
- **Dynamic Analysis Pipeline:** Code to run the dynamic analysis, including the custom WebView provider and modified Monkey (`/ui-interactor` and `/webview-provider`).
- **Developer Survey:** The survey questionnaire (`/survey`).
- **Reproducibility Scripts:** Scripts to reproduce the results presented in the paper (`/reproducibility`).
- **Analysis Results:** (Intermediate) results (`/results`).

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

None.

A.2.2 How to access

The artifacts accompanying the paper (excluding the results) are accessible at https://github.com/beerphilipp/plaintext-artifacts. The complete artifact is archived at https://doi.org/10.5281/zenodo.20393171.

A.2.3 Hardware dependencies

Access to the APKs. Due to the size of the APK dataset used in the paper, we cannot distribute it directly. Reviewers are granted access as outlined in Section A.3.1.

System Requirements. We require an Android emulator to run. As our APKs are built for ARM devices, we only support ARM architectures, specifically, a Mac with Apple Silicon. A minimum of 16 GB RAM and 250 GB available disk space are suggested.

A.2.4 Software dependencies

Operating System. We have tested and support macOS 26.5.

Docker. Install Docker (see https://docker.com/get-started).

Java. To be able to use the necessary Android dependencies, install a recent version of Java (see https://www.java.com/en/download/manual.jsp).

Android Dependencies. Install Android Studio (see https://developer.android.com/studio):

- Download Android Studio.
- Go to *Tools* > *SDK Manager* > *SDK Tools* and make sure *Android SDK Command-line Tools*, *Android Emulator*, and *Android SDK Platform-Tools* are installed. If not, install them.

A.2.5 Benchmarks

We require access to the APK dataset and the database snapshot (see Section A.3.1).

A.3 Set-up

A.3.1 Installation

Download the Artifact. Download the artifact from https://doi.org/10.5281/zenodo.20393171 and extract it in a directory of your choice.

Set Up Access to APK Dataset. To access the dataset, save the private SSH key that we provide for reviewers to a file named `~/.ssh/webview_key` and give it the correct permissions with `chmod 600 ~/.ssh/webview_key`. Researchers may request access to the dataset by contacting the authors.

Install Dependencies. See Section A.2.4 for instructions.

A.3.2 Basic Test

To verify correct installation and setup, first run `reproducibility/e1/start_emulator.sh` to set up and start an Android emulator. Then, run `reproducibility/basic_test.sh` in the repository's root directory. This command performs the following steps:

- Builds all required Docker images.
- Checks if the emulator is running.
- Checks if the SSH key is correctly saved.

The script should print *OK* to the console. This process may take a few minutes.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): HTTP Behavior of WebViews
WebView handles HTTP resources as described in the paper in terms of lack of auto-upgrading and blocking (cf. Section 3 and Table 1). This is proven by (E1).

(C2): Pipeline Functionality
The static analysis pipeline (cf. Section 4.2) is functional. This is proven by the experiment (E2).

(C3): Reproducibility of Paper Numbers
The measurement results in the paper (cf. Section 5) are reproducible. This is proven by experiment (E3).

A.4.2 Experiments

(E1): HTTP Content Behavior [30 human-minutes + 20 compute-minutes + 20 GB disk]
Preparation: Follow the steps in Section A.3.1 to set up the environment.
Execution: Run `reproducibility/e1/e1.sh` in the repository's root directory. The script builds the test app and starts the mixed content test server that is reachable via `mixed-content.test` on the Android emulator. Follow the output of the script to verify this experiment. To continue to the next (sub)experiment, press `Enter`, as prompted by the script.
Results: The server serves all resources over HTTP and HTTPS. Observe the device screen and whether HTTP content or HTTPS content was loaded. The script provides the expected outcome.

(E2): Analysis Pipeline [10 human-minutes + 20 compute-minutes + 20 GB disk]
Preparation: Follow the steps in Section A.3.1 to set up the environment.
Execution: Run `reproducibility/e2/e2.sh` in the repository. The script downloads a random subset of 10 apps from our dataset and runs the manifest and bytecode analyzer on these apps.
Results: The directory `reproducibility/e2/out` should contain subdirectories `bytecode` and `manifest`. The `bytecode` and `manifest` directories should contain 10 subdirectories corresponding to the APKs downloaded. For `bytecode`, each app directory should contain a result file for the bytecode analysis (`<package_name>.json`). For `manifest`, each app directory should contain the files `<package_name>-info.json`, `<package_name>-cleartext-config.json`, `<package_name>-fastbot-config.json`, and `<package_name>-manifest.xml`.

(E3): Measurement Results [1.5 human-hours + 5 compute-hours + 200 GB disk]
Preparation: Follow the steps in Section A.3.1 to set up the environment.
Execution: Run `reproducibility/e3/e3.sh <mongodb_snapshot> <gplay_db>` in the repository's root, where `<mongodb_snapshot>` refers to the path of the downloaded and extracted database snapshot and `<gplay_db>` to the path of the downloaded and extracted Google Play Crawl database. The script runs a MongoDB and Jupyter Notebook server in Docker. After the script finishes, open `http://localhost:8888` and run the notebooks (`*.ipynb`) in ascending order.
Results: Compare the results obtained from the notebooks with the results in the paper. Table 1 provides a mapping of notebooks to sections in the paper.

A.5 Notes on Reusability

To foster future research and make it easier for others to build on our work, we provide documentation in the `README.md` files included in each subdirectory.

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/usenixsec2026/.

Notebook	Notebook Cells	Paper Section	Notes
`1_generate_macros_dataset.ipynb`	Cells 2, 3	Section 4.1	
`2_generate_macros_success.ipynb`	Cells 2, 3, 4	Section 5.1	
`3_generate_macros_network_policy.ipynb`	Cells 2, 3	Section 5.2	
	Cells 4, 5, 8	Section 5.2, *Adoption over Time*	
	Cell 6	Section 5.2, *Allowed Domains*	
	Cell 7	Section 5.2, *Configuration Issues*	
`4_generate_cleartext_figure.ipynb`	Cell 4	Section 5.2, *Figure 2*	
`5_generate_macros_webview_usage.ipynb`	Cell 2	Section 5.3	*34,200* as per Section 5.1
	Cell 5	Section 5.3, *Figure 3*	
	Cell 7	Section 5.3, *Unique WebViews*	
	Cell 8	Section 5.3, *Origins of WebViews*	
`6_generate_macros_top_level.ipynb`	Cells 2, 3	Section 5.4, *Usage and Loaded Schemes*	
	Cells 3, 6	Section 5.4, *Local Address HTTP URLs*	
	Cell 4	Section 5.4, *Table 2*	
	Cell 5	Section 5.4, *Missing Protocol*	*21* may differ due to network changes
`7_generate_macros_enforcement_gaps.ipynb`	Cell 2	Section 5.5, *API Usage*	
	Cell 3	Section 5.5, *Commonly Used Base URLs*	
`8_generate_macros_policy_weakening.ipynb`	Cell 2	Section 5.6, *Prevalence*	
	Cell 5	Section 5.6, *Table 3*	
	Cell 6	Section 5.6, *Recorded Mixed Content*	
`9_generate_table_webview_capabilities.ipynb`	Cell 5	Section 5.7, *Table 4 + Risk*	*0.52%* located in caption of Table output, numbers in *Risk* extracted from Table
`10_generate_macros_loaded_content.ipynb`	Cell 2	Section 5.8	
	Cell 3	Section 5.8, *Resource Types*	only produces number for AIA Fetching
	Cell 8	Section 5.8, *Table 5 + Resource Types*	The numbers may differ, as Content-Type sniffing was performed (over the network) for URLs without an extension. For simplicity, the script skips this step, thus, the number in "Unknown" will be higher and for other resources, lower; numbers in *Resource Types* (except 80 AIA Fetching) extracted from Table
`11_generate_fyber_apps.ipynb`	Cell 2	Section 6.2, *Fyber SDK*	

Table 1: Mapping of evaluation notebooks and cells to main paper sections.