One Size Does Not Fit All: Uncovering and Exploiting Cross Platform Discrepant APIs in WeChat

Chao Wang, Yue Zhang, and Zhiqiang Lin, *The Ohio State University*

https://www.usenix.org/conference/usenixsecurity23/presentation/wang-chao

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Open access to the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium is sponsored by USENIX.

USENIX'23 Artifact Appendix: One Size Does Not Fit All: Uncovering and Exploiting Cross Platform Discrepant APIs in WeChat

Chao Wang
The Ohio State University

Yue Zhang
The Ohio State University

Zhiqiang Lin
The Ohio State University

A Artifact Appendix

A.1 Abstract

APIDiff is an automatic tool that generates test cases for each API and identifies execution discrepancies. APIDiff consists of three key components. The Test Case Generator is responsible for creating test cases for each API, initializing the parameters correctly, resolving dependencies between APIs, and mutating parameter values to achieve high coverage. The Code Executor executes the test cases on different platforms (Windows, Android, and iOS) to generate corresponding outputs and detect discrepancies that may pose security concerns. The Discrepancies Analyzer employs differential analysis and predefined policies to examine error codes and return values of tested APIs, enabling the identification of discrepant APIs.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

To ensure compliance with community practices, it is imperative to strictly adhere to the following requirements:

- **Thorough Analysis and Controlled Environment**: All analysis and execution of attacks must be conducted within a controlled environment, utilizing personal accounts and machines.
- **Confidentiality of Attack Code and Malware**: Any developed attack code and malware must be kept private and confidential to prevent any potential harm to users, miniapp developers, and platform providers.
- **Responsible Reporting**: Any discoveries made during the analysis should be promptly and responsibly reported to Tencent, following their specified guidelines and procedures.

A.2.2 How to access

Please find our project online: https://github.com/OSUSecLab/APIDiff/tree/f65137b3f8dc037021773134db40b1d384d542b7

A.2.3 Hardware dependencies

To run the tools, you require a specific environment with the following software and hardware requirements. Firstly, you need an operating system (OS) such as Linux, macOS, or Windows. Additionally, you need Node.js, which is a JavaScript runtime environment. You also need WeChat DevTools, which is a development tool specifically for creating and debugging MiniApps. In terms of hardware, you will need devices for conducting experiments. This includes Android devices, iOS devices, and Windows devices.

A.2.4 Software dependencies

Please ensure that you have the latest version of WeChat installed on your devices. In order to create and debug a MiniApp for API testing, it is necessary to have a WeChat account and install the IDE specified by Tencent.

A.2.5 Benchmarks

None.

A.3 Set-up

A.3.1 Installation

To get started, please follow the steps below:

1. Clone the project: Use the Git command or a Git client to clone the project repository. For example, you can use the following command in your terminal:

```
git clone https://github.com/OSUSecLab/APIDiff/
```

2. Install additional dependencies:

```
# make sure node.js and npm is installed
node --version

# install for apitest-gen
cd apitest-gen && yarn && cd ..

# install for client
cd client && yarn && cd ..

# install for server
cd server && yarn && cd ..
```

A.3.2 Basic Test

You can test each component by simply executing the following commands:

- **apitest-gen**: `npx ts-node apitest-gen/src/apigen/main.ts`
- **client**: `npx ts-node client/src/index.ts`
- **server**: `npx ts-node server/agent/index.ts`

A.4 Evaluation workflow

Generate the test cases Make sure you have documents available for the test case generation. Note that the input document should be an array of API (Array in typescript) in JSON format. You can refer to Pre-processing for more information.

You can specify the input document file via `apitest-gen/src/apigen/config.ts` or via the environment variable `INPUT_DOC`.

You can specify the output directory via the same config file or via the environment variable `OUTPUT_DOC`.

Then, execute the `apitest-gen/src/apigen/main.ts` to generate test cases. (i.e., `ts-nodeapitest-gen/src/apigen/main.ts`)

Find the debug URL The debug URL is embedded in WeChat DevTools. You need a WeChat account to create and test a MiniApp, and via initiating a remote debug session you can obtain such a debug URL.

1. **Tweak the WeChat DevTools** Locate your WeChat DevTools installation directory, look for package.json inside package.nw (in macOS the location is `/Applications/wechatwebdevtools.app/Contents/Resources/package.nw/package.json`). Then, find the `-disable-devtools` flag, remove it and save the file.
2. **Initiate a remote debug session.** Choose a device target to start a remote debug session. Make sure you are using remote-debug 2.0 and enable LAN mode for best network latency. Make sure the remote debugger window is popped.
3. **Find the debug URL** Make sure the current focused window is the remote debugger, press F12 to open the chrome devtools. Switch to the Elements tab and find the webview tag (the selector is `body > div:nth-child(1) > div > div > div.debugger > webview`). Now you can find the debug string is inside the src property starting with `ws=`. An example debug string is `ws=127.0.0.1:40204`. You can now transform this string into the debug URL: `ws://127.0.0.1:40204`.

Run the server Make sure you have all dependencies installed. Then, you can start a server directly by the following command:

```
cd server
REMOTE_DEBUG_WS=<your debug URL> ts-node agent/index.ts
```

Make sure the *[evaluator init] global* message is prompted after running the server. If you did not see this message, the debug URL might be invalid due to timeout. You need to redo the previous steps. Note that the debug URL won't change as long as the WeChat DevTools is not closed or restarted.

You can also interact with the debugger via specifying *ENABLE_NODE_REPL* environment variable. The Node.JS REPL contains global objects for debugging. Please refer to *server/agent/index.ts* for more information.

The server will open a port for the incoming requests from the client. You can specify the port in server/agent/listener/config.ts

Run the client You can now run the client to start testing. You need to change the config file defined in `client/src/config.ts` based on your previous configuration. After that, you can run the client by the following command:

```
cd client && ts-node src/index.ts
```

A.5 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/`.