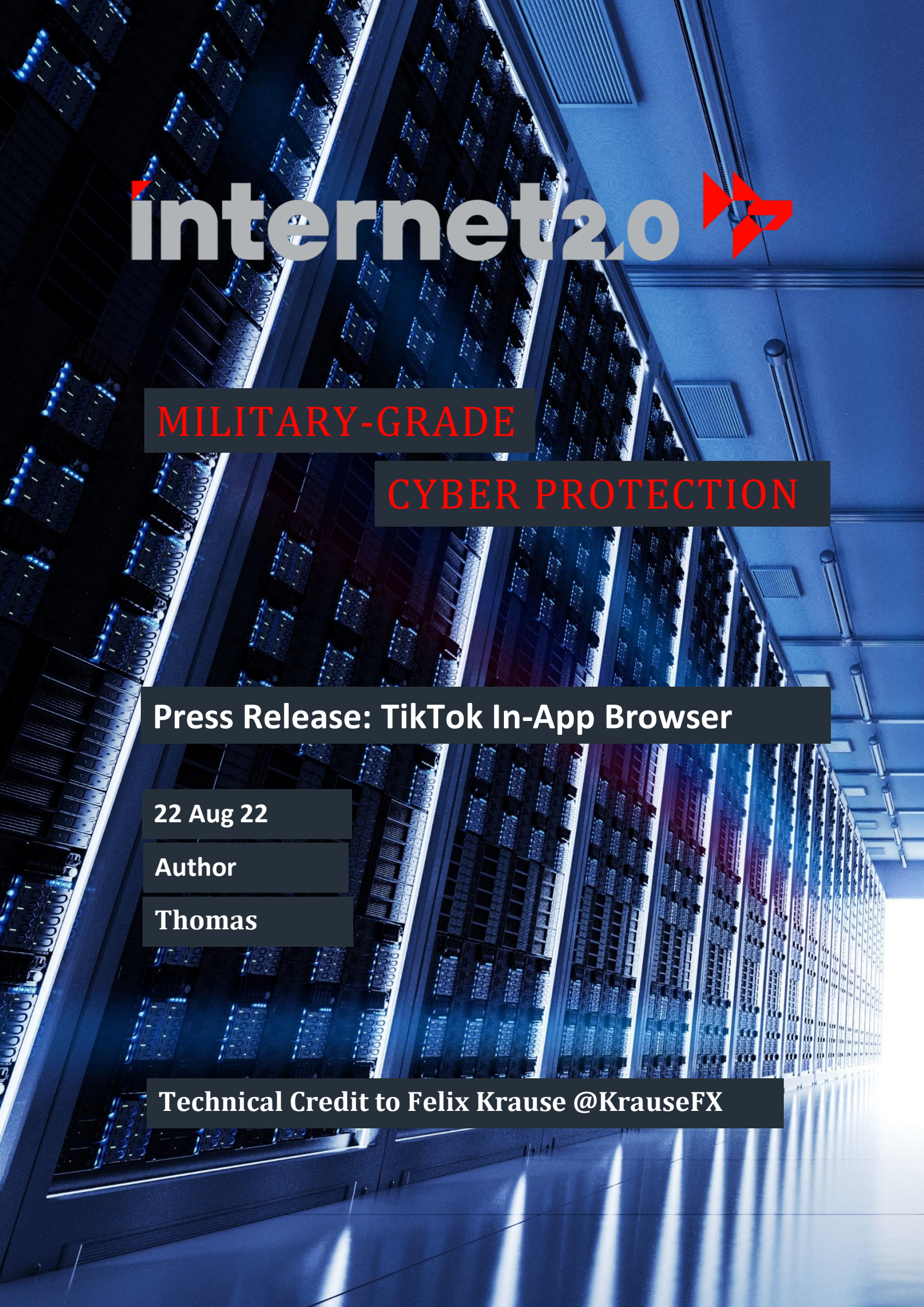


Press Release: TikTok In-App Browser

22 Aug 22

Author

Thomas

Technical Credit to Felix Krause @KrauseFX

Executive Summary

As first mentioned by the researcher Felix Krause (@KrauseFx) the TikTok IOS application version 25.8.3 at 20 Aug 2022 also in our opinion has the capability to inject potentially malicious JavaScript into third party websites while using the In-App Browser inside of the TikTok application. It must be noted that we cannot verify if TikTok uses the functionality in this software development kit (SDK) data harvesting. What we can verify based on the work done by Felix is that the code base TikTok uses has the capacity to inject malicious JavaScript into third party websites while using the In-App Browser inside of the TikTok application.

We spoke to Felix and he also noted that he was misquoted in the media and by TikTok on what TikTok uses this codebase for. He wrote the balanced statement on TikTok at figure 1.

FAQs for non-tech readers

- **Can in-app browsers read everything I do online?** No! They are only able to read and watch your online activities when you open a link or ad from within their apps.
- **Do the apps above actually steal my passwords, address and credit card numbers?** No! I wanted to showcase that bad actors could get access to this data with this approach. As shown in the past, if it's possible for a company to get access to data legally and for free, without asking the user for permission, they will track it.
- **How can I protect myself?** Whenever you open a link from any app, see if the app offers a way to open the currently shown website in your default browser. During this analysis, every app besides TikTok offered a way to do this.
- **Are companies doing this on purpose?** Building your own in-app browser takes a non-trivial time to program and maintain, significantly more than just using the privacy and user-friendly alternative that's already been built into the iPhone for the past 7 years. Most likely there is some motivation there for the company to track your activities on those websites.
- **I opened InAppBrowser.com inside an app, and it doesn't show any commands. Am I safe?** No! First of all, the website only checks for one of many hundreds of attack vectors: JavaScript injection from the app itself. And even for those, as of December 2020, app developers can completely hide the JavaScript commands they execute, therefore there is no way for us to verify what is actually happening under the hood.

Figure 1. Felix Statement[1]

Of note Felix commented here on the use of In-App Browser across the entire industry.

[1] https://twitter.com/KrauseFx/status/1560374054216941569/photo/1

iOS Apps that have their own In-App Browser

- **Option to open in default browser**: Does the app provide a button to open the currently shown link in the default browser?
- **Modify page**: Does the app inject JavaScript code into third party websites to modify its content? This includes adding tracking code (like inputs, text selections, taps, etc.), injecting external JavaScript files, as well as creating new HTML elements.
- **Fetch metadata**: Does the app run JavaScript code to fetch website metadata? This is a harmless thing to do, and doesn't cause any real security or privacy risks.
- **JS**: A link to the JavaScript code that I was able to detect. Disclaimer: There might be other code executed. The code might not be a 100% accurate representation of all JS commands.

App	Option to open in default browser	Modify page	Fetch metadata	JS	Updated
TikTok	⛔	Yes	Yes	.js	2022-08-18
Instagram	✅	Yes	Yes	.js	2022-08-18
FB Messenger	✅	Yes	Yes	.js	2022-08-18
Facebook	✅	Yes	Yes	.js	2022-08-18
Amazon	✅	None	Yes	.js	2022-08-18
Snapchat	✅	None	None		2022-08-18
Robinhood	✅	None	None		2022-08-18

Click on the `Yes` or `None` on the above table to see a screenshot of the app.

Important: Just because an app injects JavaScript into external websites, doesn't mean the app is doing anything malicious. There is no way for us to know the full details on what kind of data each in-app browser collects, or how or if the data is being transferred or used. This publication is stating the JavaScript commands that get executed by each app, as well as describing what effect each of those commands might have. For more background on the risks of in-app browsers, check out last week's publication.

Even if some of the apps above have green checkmarks, they might use the new Isolated World JavaScript, which I'll describe below.

Figure 2. Felix Statement[2]

[2] https://twitter.com/KrauseFx/status/1560372215048175617/photo/1

Verification

We took a third party look at Felix's research and have chosen to release this press statement in support of his findings.

We found that inside of TikTok's code there are two folders labeled as "webview_monitor_js_file*", see image 1, these folders are responsible for housing version 1 and version 2 of the SDK responsible for the In-App Browser. In each folder there are two files labeled as "slardar_bridge.js" and "slardar_sdk.js" respectively. The "slardar_sdk.js" file is responsible for creating all the hooks and functions that will be required and the "slardar_bridge.js" file is responsible for calling those hooks and functions.

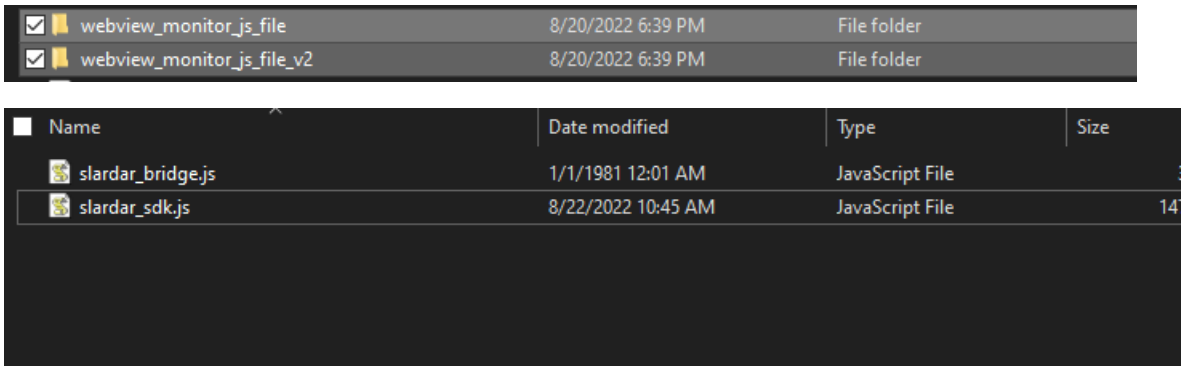
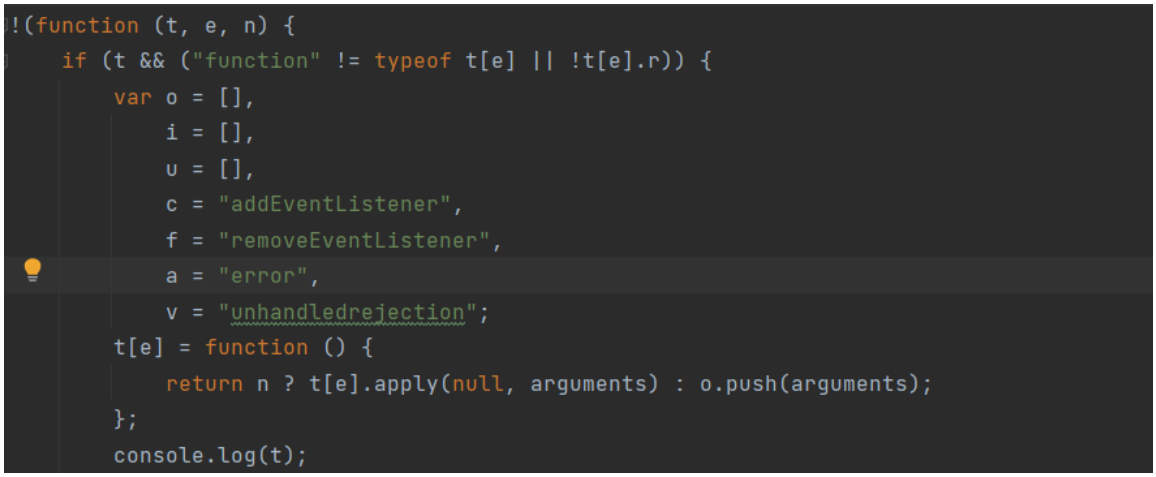


Figure 3. folders inside TikTok Source code

We have taken the liberty to upload the files as is to VirusTotal for researchers to analyze themselves. You can find a copy of version 1 here: https://www.virustotal.com/gui/file/97f24566e7dbc114a47c101c5600471192b98e832b06f3871fd23fc9e904d631 and a copy of version 2 here: https://www.virustotal.com/gui/file/3e58b0c33d2eb5d6b752de3c50255653d10b6c54efc05d839f0e039aa39b0462.

We will be focusing on the version 2 SDK for this analysis. By adding a "console.log" in the JavaScript and running the JavaScript in JSFiddle we can output the data object that is created during initialization of the JavaScript. This allows us to verify what hooks, event listeners, and views are being initialized by the SDK file:



```
!(function (t, e, n) {
    if (t && ("function" != typeof t[e] || !t[e].r)) {
        var o = [],
            i = [],
            u = [],
            c = "addEventListener",
            f = "removeEventListener",
            a = "error",
            v = "unhandledrejection";
        t[e] = function () {
            return n ? t[e].apply(null, arguments) : o.push(arguments);
        };
        console.log(t);
```

Figure 4. Code inside version 2 of the WebView monitor JS file with conolse.log location

From here we can get all the information that is being passed to this function some interesting aspects that are initialized by this function are:

- Screen resolution
- Event listeners for
    - Onclick – waits for the user to click and reacts to the click
    - Ondblclick – waits for the user to double click and reacts to the click
    - Onmouseevent – reacts to any event that is done by the mouse or screen tap
    - Onkeypress – reacts to any event that is done by pressing a key

These are interesting in the fact of how intrusive they are. For example, "onkeypress" will react to any event from pressing a key and if used correctly can essentially be a keylogger. It is also worth mentioning that there is a hook enabled for all XHR requests as well. This means that any ajax request passed through this window is hooked by these functions.

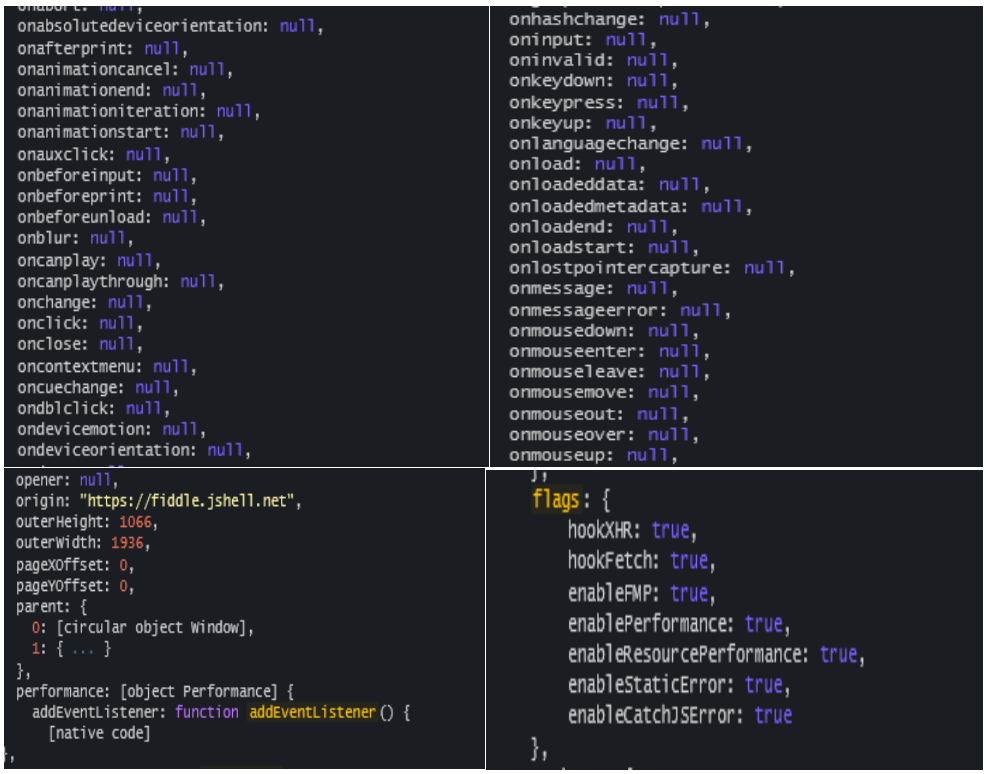


Figure 5. Event listeners and hooks initialized by the WebView monitor JS file

As you can notice all the “on” listeners are set to null, meaning that there is nothing associated with these hooks at the time of the screenshot. To determine what hooks are enabled we need to analyze the JavaScript code while using the application itself, by doing so we can determine the SDK behavior dynamically and get a better understanding of what it does. We were able to do this using a Samsung phone on the following system:

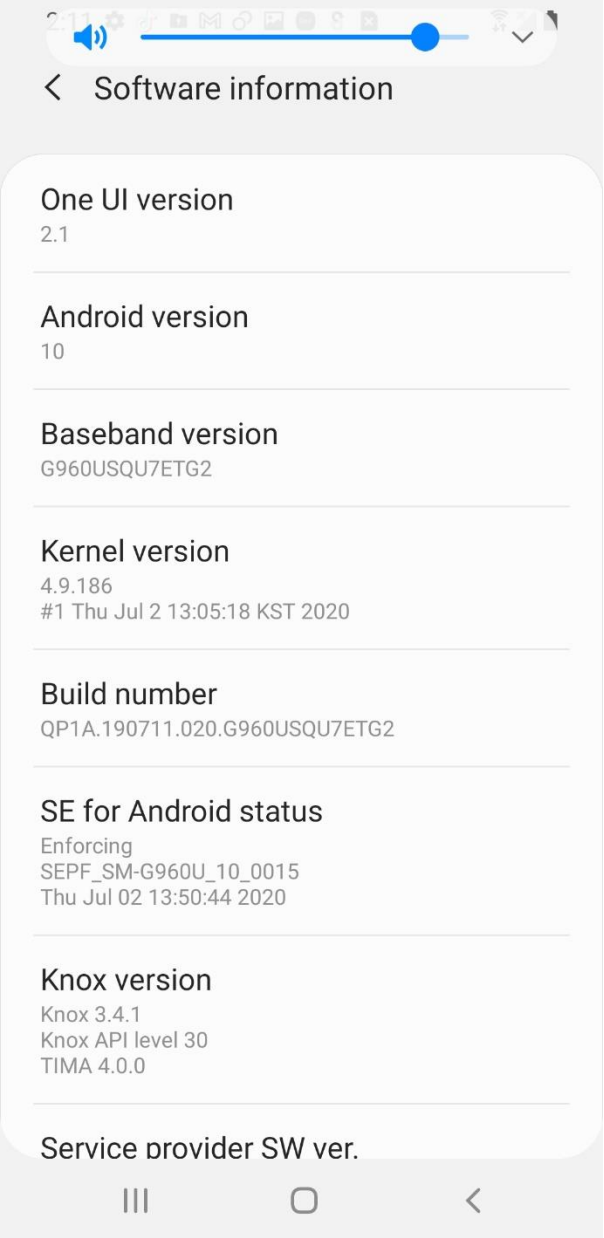


Figure 6. Build information of Android device used

From here we were able to trick the TikTok application into visiting a link that would allow us to view what JavaScript is being injected into the third party website content (this is the same link that was used by Felix originally), the results are as follows:

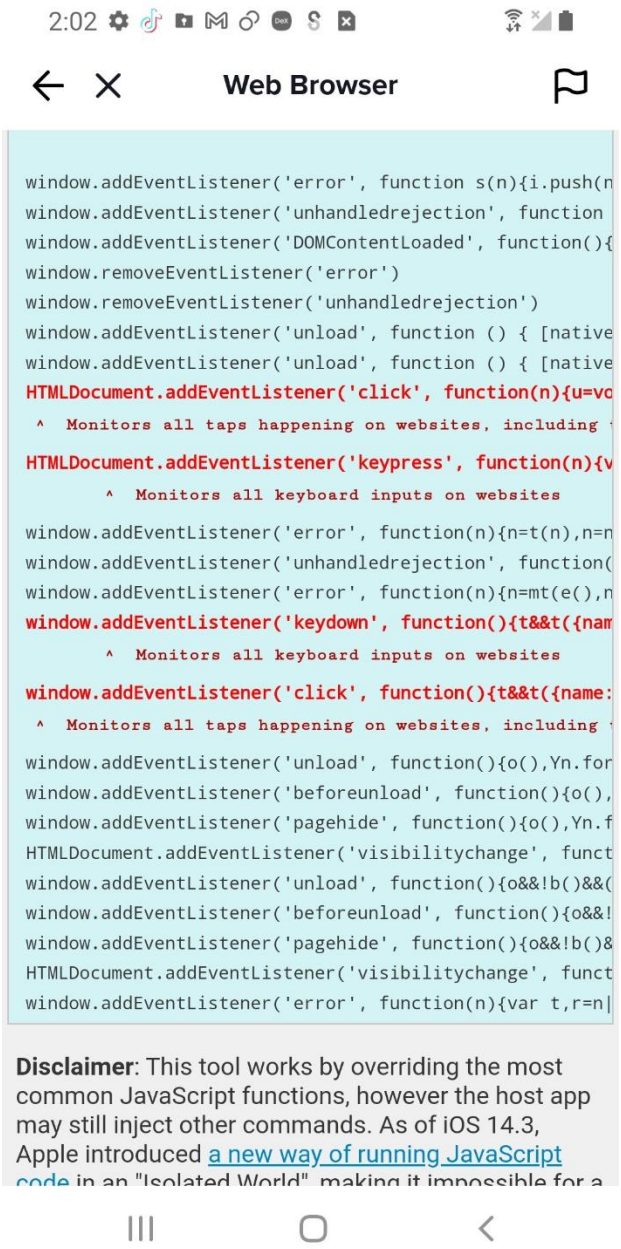


Figure 7. 'Tricking' TikTok into visiting https://inappbrowser.com to verify event listeners

As previously stated by Felix Krause in a twitter post here: https://twitter.com/KrauseFx/status/1560370732705742848 figure 7 confirms Felix's suspicions that in fact TikTok is injecting potentially malicious JavaScript into third party websites. We are unable to confirm if any of this information is retrieved from the device but can confirm that there is potential for the data to be stored on the device via two strings that may be indicating logging in the SDK source code:

Figure 8. Possible logging inside the TikTok SDK

Credit of research

Internet 2.0 has a strong support for third-party researchers. The Security researcher community is fragile, and we must give credit where it is due to foster more security researchers to make public statements to support consumer awareness. In our opinion Felix has taken a balanced and truthful statement and we have written this with his permission in support of his work. For any reference for this research please credit Felix Krause https://twitter.com/KrauseFx.

Values on Security Research

Internet 2.0 has dealt with many credible but also negative people in the security researcher community. We will actively support and give credit to any researcher that

- Uses technical documentation to back up their claims.
- Make balanced and truthful statements.
- Does not exaggerate.
- Does not make partisan political statements. And
- Does not criticize another researcher while first doing no due diligence themselves. In our opinion there are too many ill-informed people on twitter trying to get famous on doing a takedown of a person's legitimate research while first doing no work themselves.

if you want to work and publish research in partnership with Internet 2.0 please contact us at: contact@internet2-0.com. If your technical research is of the public interest we will support your work.


internet2o
MILITARY-GRADE
CYBER PROTECTION
Australia
Level
18 National
Barton ACT 2600
ABN: 17 632 726
United States
Suite
211 N Union
Alexandria
EIN: 86-
contact@internet2-