Anatsa Campaign Technical Analysis | ThreatLabz

By Himanshu Sharma, Gajanan Khond

Published: 2024-05-27 · Archived: 2026-04-02 11:50:37 UTC

As mentioned previously, Anatsa utilizes remote payloads retrieved from C2 servers to carry out further malicious activity.

In the figure below, the dropper application is shown with encoded links to remote servers, from which the next stage payload will be downloaded. In addition to downloading the payload, the malware also retrieves a configuration file from the remote server to execute the next stage payload.

Figure 3: Anatsa dropper's payload and configuration URLs.

In the figure below, the DEX file is downloaded and will be loaded by the parent fake QR code application.

Figure 4: Anatsa dropper's network request to download the DEX file for the next stage payload.

The application utilizes reflection to invoke code from a loaded DEX file. The necessary configuration to load the DEX file is downloaded from the control server, as depicted in the network response shown below.

Figure 5: Anatsa dropper's configuration to run the downloaded DEX file.

After the next stage payload is downloaded, Anatsa performs a series of checks for the device environment and device type. This is likely designed to detect analysis environments and malware sandboxes. Upon successful verification, it proceeds to download the third stage and final payload from the remote server, as depicted in the figure below.

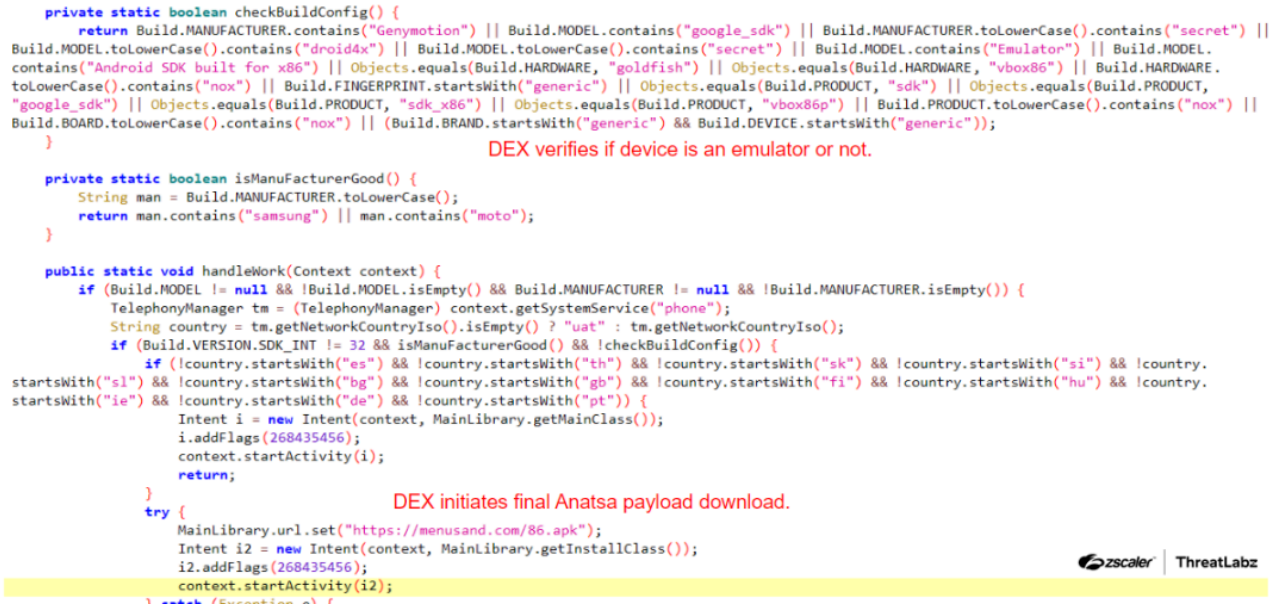

Figure 6: Code that checks the device environment and downloads final stage Anatsa payload.

In this particular campaign, the Anatsa malware injected uncompressed raw manifest data into the APK. The threat actors also intentionally corrupted the compression parameters in the manifest file to hinder analysis. The figure below depicts the corrupted ZIP headers.

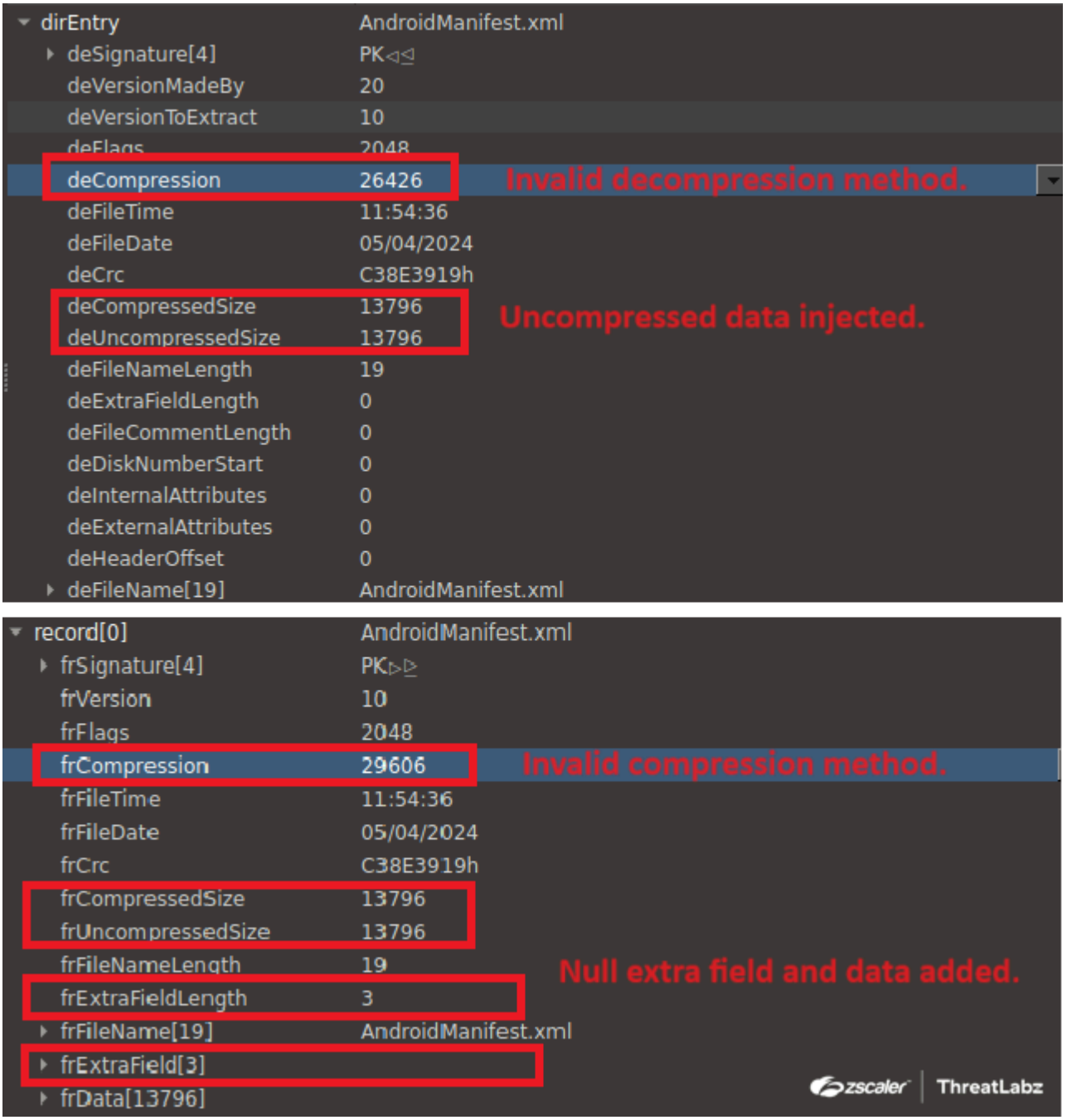

Figure 7: Anti-analysis technique utilized by Anatsa with malformed ZIP parameters.

In order to statically analyze the payload, the headers of the ZIP file must be fixed alongside the compressed data.

After the APK is loaded, the malware requests various permissions, including the SMS and accessibility options, which are commonly associated with mobile banking trojans. The malware conceals the final DEX payload within the asset files. During runtime, the payload decrypts the DEX file using a static key embedded within the code.

Figure 8: Anatsa malware with the correct manifest.

Upon execution, the malware decodes all encoded strings, including the C2 communication. The malware establishes communication with the C2 server to carry out various activities, such as registering the infected device and retrieving a list of targeted applications for code injections.

In order to steal data from financial applications, Anatsa downloads a target list. The figure below shows the Anatsa configuration request and response.

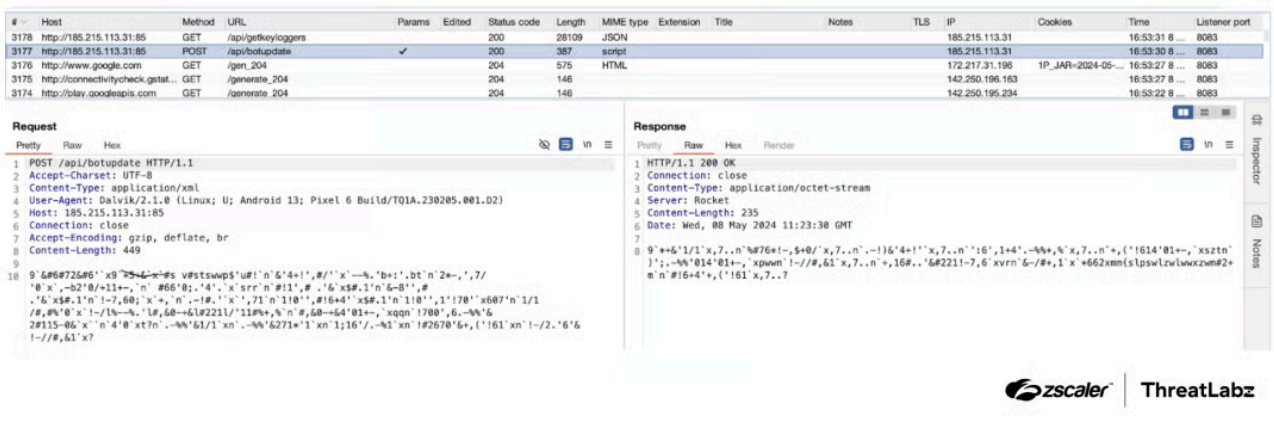

Figure 9: Anatsa configuration request and response being intercepted.

The figure below shows the request and response data being decoded with an XOR key.

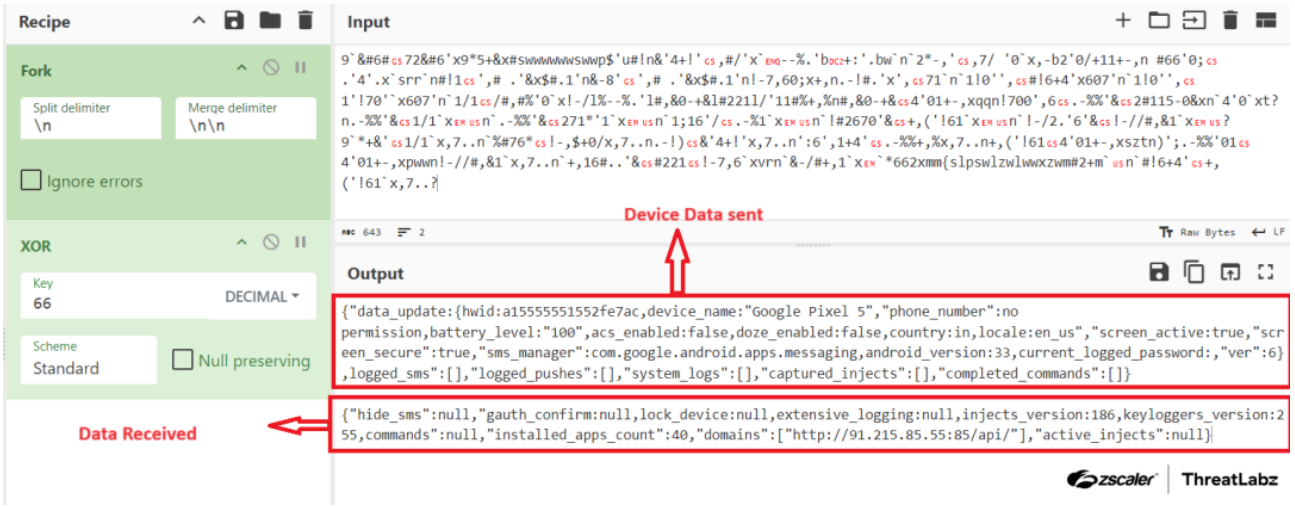

Figure 10: Example decrypted Anatsa request and response data using an XOR key.

Upon receiving a list of financial application package names, the malware scans the victim's device to check if any of these targeted applications are installed. Once the malware identifies the presence of a targeted application, Anatsa communicates this information to the C2 server. In response, the C2 server provides a fake login page for the banking application. This activity is illustrated in the figure below.

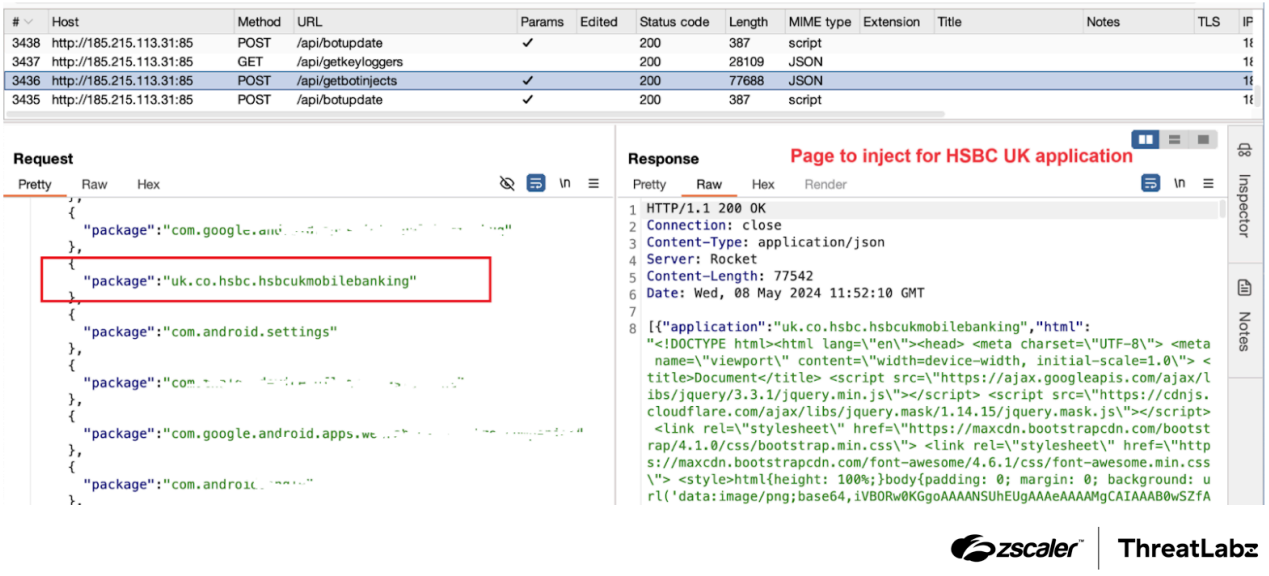

Figure 11: Anatsa injection configuration request based on the presence of a specific financial application.

The fake login page is loaded within a JavaScript Interface (JSI) enabled webview, which is designed to deceive the user into providing their banking credentials. Once the victim enters their credentials that data is sent back to the C2 server.

Explore more Zscaler blogs