HITB SECCONF 2024 BANGKOK

HTTPS://CONFERENCE.HITB.ORG/HITBSECCONF2024BKK

Exploiting the In-Vehicle Browser: A Novel Attack Vector in Autonomous Vehicles

Ravi Rajput

Manager, Exela Technologies

TRACK 1

30 AUG

#HITB2024BKK

whoami?

- Manager – Information Security @ Exela Technology
- Focusing on Binary exploitation, AI/ML Security, Automotive and Telecom exploitation.
- Ex – Null Ahmedabad chapter lead
- Author and Project Lead – AutoHackOS
- Training author – AutoSec Pro @ PenTest Magazine and Hakin9
- Speaker at BLACKHAT ASIA 2023, Nullcon, HITCON, Bsides Delhi, Bsides Maharastra, Bsides Ahmedabad, Bsides Indore, Bounty Bash, UnitedCon and Null Community
- Core team and Co-Organizer - Telecom village @DefCon
- Life time learner.
- Life mantra – “The more you try to know, you understand you don’t know anything yet” so I am too learning everyday.

#HITB2024BKK

Agenda

1) Android Auto vs Android Automotive OS
2) Browser Integration in Automotive OS
3) Attack Surface
4) Browser APIs
5) CarService and CarAPI
6) Recon methodology
7) Exploitation Methodology
8) POC
9) Flowchart of exploitation
10) What happened Next?
11) Mitigation Strategies

#HITB2024BKK

Disclaimer

1) This vulnerability was reported in 2022.

2) It was mitigated by head of technology by Vendor (Car Manufacturer) within 2 days.

3) Vendor explicitly asked to not name them in the talk and hide most technical details.

#HITB2024BKK

Difference between Android auto and Android Automotive OS

Android Auto is a platform running on the user's phone, projecting the Android Auto user experience to a compatible in-vehicle infotainment system over a USB connection. Android Auto supports apps designed for in-vehicle use.

Android Automotive is an operating system and platform running directly on the in-vehicle hardware. It is a full-stack, open source, highly customizable platform powering the infotainment experience. Android Automotive supports apps built for Android as well as those built for Android Auto

Browser Integration in automotive

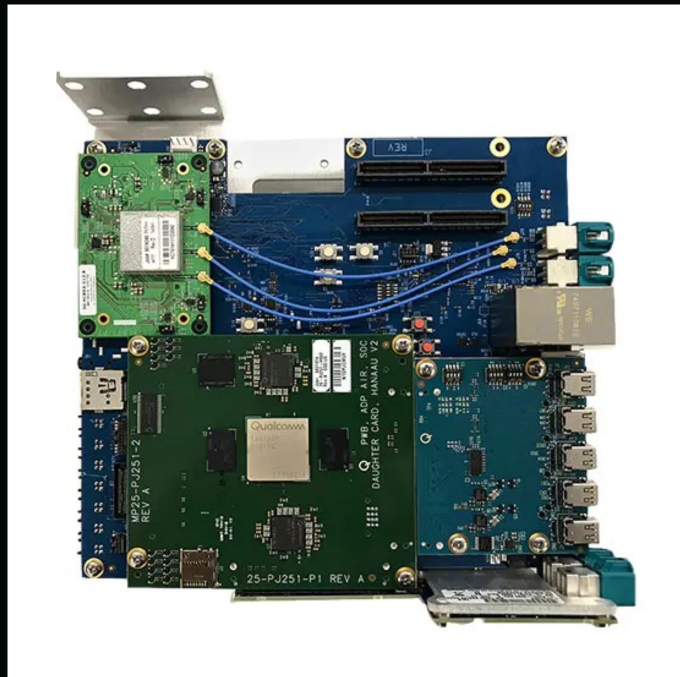
We started with this simple radio

And we are here with browser in Car's head unit

#HITB2024BKK

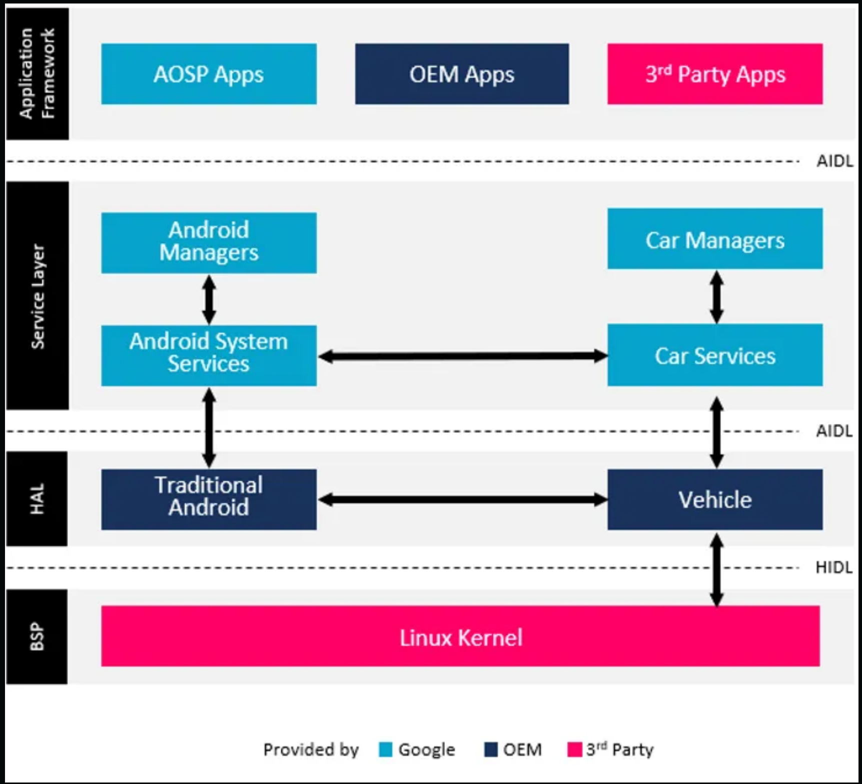
Browser Integration in automotive

Hardware making this possible – eg, SA8155P Automotive Development Platform



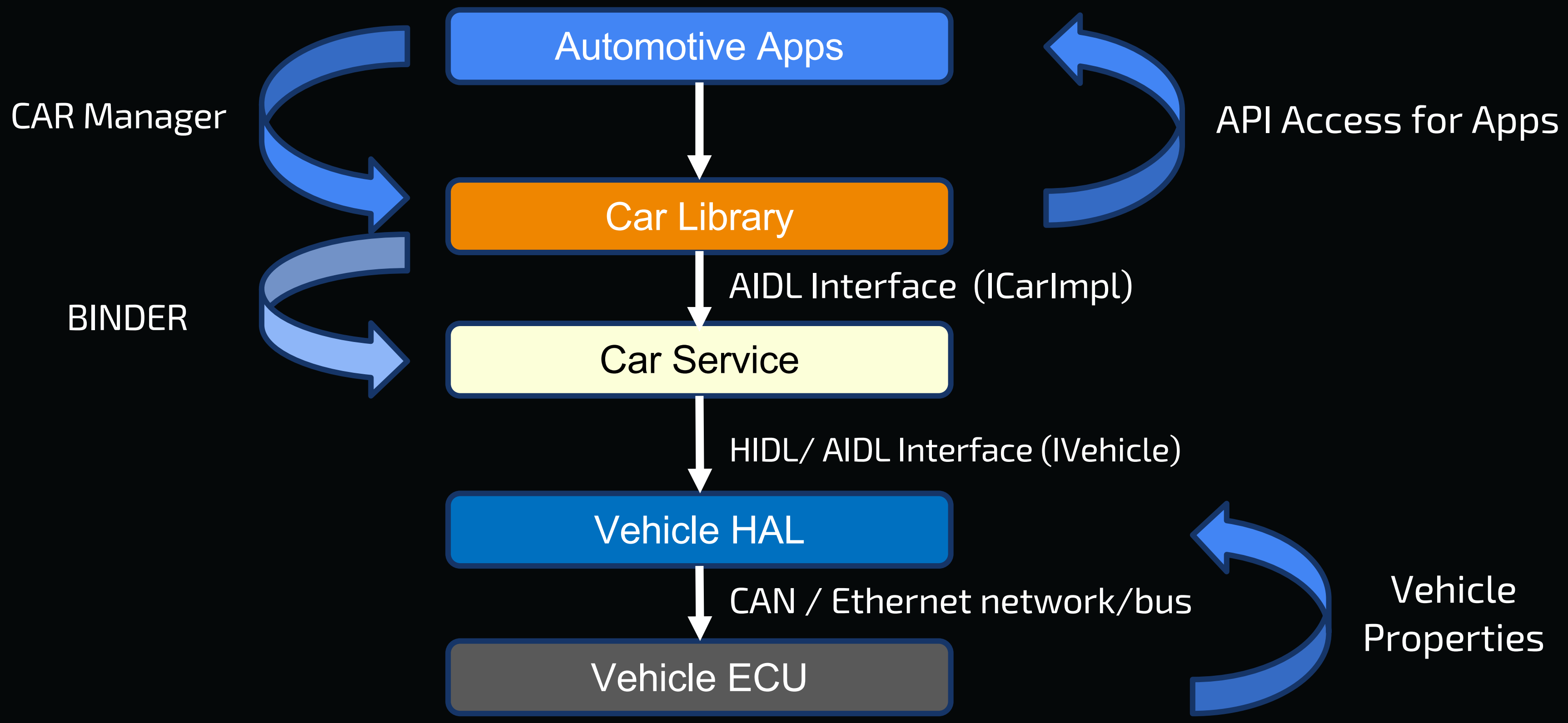
#HITB2024BKK

Browser Integration in automotive



#HITB2024BKK

Browser Integration in automotive



#HITB2024BKK

Attack Surface

1) Custom OEM Browsers because they are not fixed immediately
2) Web View based apps, examples are connected car apps, Messaging or some social media apps.
3) Music and streaming apps, E-commerce and payment apps, examples are Payment for charging station.
4) Cloud based Diagnostic apps.

#HITB2024BKK

Browser APIs

1. WebView API: Used for embedding web content within apps. Although the Chromium-based browser itself is a more advanced component than WebView, understanding WebView helps in grasping how web content is managed in Android Automotive.

2. System Services API: Provides access to various system services that the browser may interact with, such as audio, notifications, and location services.

3. Media API: Allows the browser to interact with media playback and control functionalities, which is crucial for streaming content within the browser.

4. Vehicle Context API: Provides access to vehicle-specific data such as speed, location, and battery status, which can be useful for web applications running within the browser.

5. Connectivity API: Handles network connectivity aspects, including Wi-Fi, cellular data, and Bluetooth, which the browser utilizes to access online resources and services.

6. Permissions API: Manages permissions for accessing various resources and services, ensuring the browser and web content adhere to security and privacy guidelines.

7. Application Context API: Provides context about the application environment, which can be used to manage browser settings and configurations.

#HITB2024BKK

Car Services and CarApi

1) **CarSensorManager :** Provides access to various vehicle sensors, such as speed, RPM, fuel level, and other sensor data.
2) **CarAudioManager :** Manages audio settings and streams, allowing control over different audio zones within the vehicle.
3) **CarMediaManager :** Manages media playback and controls in the vehicle, interfacing with media apps and the vehicle's display.
4) **CarNavigationManager :** Provides APIs for navigation and location services within the vehicle.
5) **CarInfoManager :** Provides access to static information about the car, such as manufacturer details, model, and software versions.
6) **CarClimateManager :** Manages climate control systems within the vehicle, including temperature, fan speed, and other HVAC settings.
7) **CarEvsManager :** Handles embedded video streams, like reversing cameras or video feeds from external sources.
8) **CarDiagnosticManager :** Provides access to vehicle diagnostics data, such as error codes and system statuses.
9) **CarPowerManager :** Manages vehicle power states, including ignition, battery levels, and power-off procedures.
10) **CarPackageManager :** Manages apps and packages within the automotive OS, including installation, updates, and permission handling.

Recon Methodology

1) Use command “lshal” to list VHAL Properties enabled by vehicle manufacturer.

2) If the scope is whitebox, take the image of android automotive os and fetch apk for browser

3) If Blackbox but you can enable developer options, pull apk of browser

4) Analyze the manifest file for following lines that indicate that it interact with VHAL

    1) Permissions: The manifest file should request specific permissions that allow the app to access vehicle data. Look for permissions such as:

        android.permission.ACCESS_VEHICLE_DATA

        android.permission.BIND_CAR_SERVICE

    2) Services: Check for services declared in the manifest that might be used to interact with the vehicle's hardware.

        <service android:name=". CarService" android:permission="android.car.permission.BIND_CAR_SERVICE" />

    3) Intent Filters: Look for any intent filters that suggest the app is set up to handle specific automotive intents or communicate with vehicle systems.

        <meta-data android:name="android.car.permission.VEHICLE_DATA" android:value="true" />

    4) Library: Some browser apps may declare the use of specific libraries that facilitate interaction with VHAL or other automotive-specific APIs.

        <uses-library android:name="com.android.car" />

#HITB2024BKK

Exploitation Methodology

1) Start Monitoring can bus before starting testing.
2) Enable Developer options/mode from head unit.
3) Enable debug shell and Diagnostic service via adb shell.
4) Enable VHAL debug and list VHAL properties available.
5) Collect logs with dumpsys.
6) Start fuzzing browser and keep watch on log on adb shell as well as can bus log.
7) Use logcat to understand what failed while fuzzing and analyze anr files:

```
adb logcat -s CarServiceHelper | fgrep "carwatchdog killed"
grep -Hn "pid 594" /data/anr/*
```

8) Keep watch on any trigger and try to reproduce.
9) Loop back

#HITB2024BKK

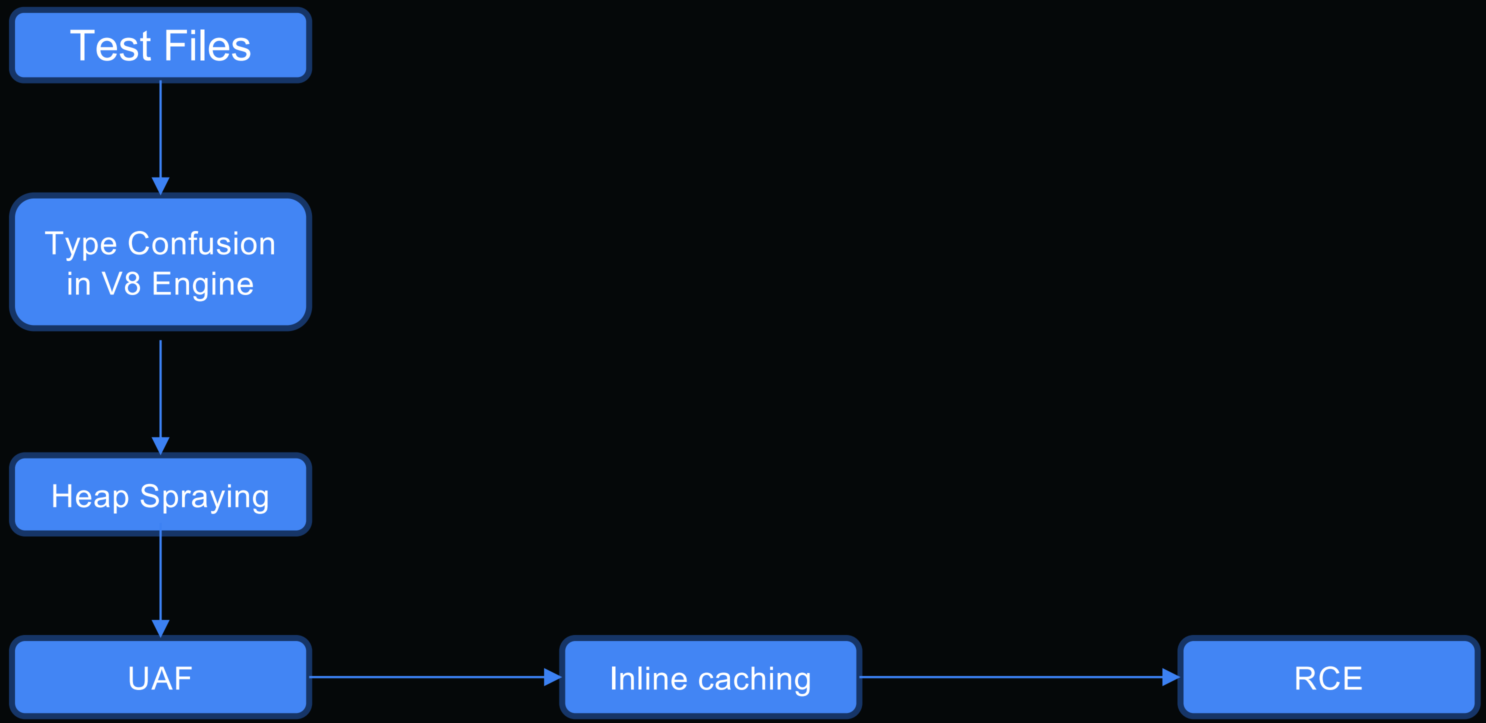
POC - DOS

Implications:

1) Airbag controls failed
2) Viper fluid system failed
3) Airbag disabled
4) SOS and other emergency warnings triggered
5) Emergency call system failed

#HITB2024BKK

Flowchart of exploitation



#HITB2024BKK

What happened next?

1) Browsers are Parked Apps, which means we can't exploit while car is accelerating or deaccelerating.
2) Standard Browsers like chrome doesn't interact with Carservice or CarApi until and unless we target other webview based applications or something, but the exploitation step will change then.
3) Browsers are typically sandboxed which will make exploitation even harder
4) Vehicle manufacturers just want compliance but sometimes ignore security lifecycle.

We Failed in exploitation !!!

#HITB2024BKK

Mitigation strategies

1) Setup code analyzers like SonarQube in CI/CD pipeline.
2) Use OWASP Dependency check or Snyk in CI/CD pipeline.
3) Deploy and test on AWS Gravition.
4) Incorporate CATBox into the CI/CD pipeline to run end-to-end tests on the browser and head unit. CATBox can be configured to validate critical functionalities, including security aspects, and ensure compliance with automotive standards.
5) Implement VSOC for complete observability.

#HITB2024BKK

Thank You

#HITB2024BKK