Proof of Concept (PoC)

Automotive Threat Matrix (ATM)

🔍 Objective

To understand, analyze, and demonstrate how modern vehicles are exposed to cybersecurity threats by using the Automotive Threat Matrix (ATM) **developed by** Auto-ISAC**. This PoC aligns the MITRE-style tactics with** real-world automotive vulnerabilities**, supported by** techniques **and** procedures **observed in real attacks.**

All Tactics in the Automotive Threat Matrix

🚗 **1.** Reconnaissance

Goal: Gather information about the vehicle system before launching an attack.

Example Activities:

- Scanning Bluetooth/Wi-Fi signals emitted by the vehicle.
- Reading public documentation on the car's ECUs or infotainment system.
- Observing vehicle behavior or firmware version via diagnostic ports.

Use Case:
A hacker captures broadcasted telematics traffic or sniff signals near a car dealership to identify exploitable systems.

🌍 **2.** Manipulate Environment

Goal: Deceive the vehicle’s sensors or environment-based decision systems.

Example Activities:

- Spoofing GPS to trick the car into believing it's in a different location.
- Shining a laser at LIDAR to simulate obstacles.
- Changing road signs or painting fake lanes to confuse autonomous driving logic.

Use Case:
Attacker sets up a fake GPS transmitter near a parking area to redirect the car's navigation route.

🔒 **3.** Initial Access

Goal: Gain entry into the vehicle’s network or operating system.

Example Activities:

- Exploiting a Bluetooth vulnerability in the infotainment system.
- Plugging a malicious USB into a dashboard port.
- Using a rogue cellular base station to intercept OTA (over-the-air) updates.

Use Case:
Hacker uses a Bluetooth-based exploit to remotely compromise a Jeep Grand Cherokee’s IVI system.

⚙️ **4.** Execution

Goal: Run malicious code or commands on the vehicle’s systems.

Example Activities:

- Executing a shell command in the IVI OS after gaining access.
- Triggering a payload delivered through malicious firmware.
- Launching scripts that interact with CAN bus messages.

Use Case:
After gaining access via Wi-Fi, attacker executes code to dump logs and open CAN access.

♻️ **5.** Persistence

Goal: Maintain access even after the vehicle is rebooted or repaired.

Example Activities:

- Installing a boot-level implant in an ECU’s firmware.
- Modifying startup scripts in the OS to load attacker tools.
- Exploiting UDS (Unified Diagnostic Services) to permanently reprogram an ECU.

Use Case:
Attacker modifies the IVI firmware so that even after a factory reset, the backdoor remains active.

🗝️ **6.** Privilege Escalation

Goal: Move from low-privilege access to full control (e.g., from user to root).

Example Activities:

- Exploiting OS kernel vulnerability to gain root.
- Cracking developer/debug backdoors.
- Sending diagnostic messages that unlock privileged ECU modes.

Use Case:
Attacker escalates from normal IVI user to root by exploiting a known Linux vulnerability (e.g., Dirty COW).

🕵️ **7.** Defense Evasion

Goal: Avoid detection by system defenses, logs, or diagnostic tools.

Example Activities:

- Disabling logging mechanisms in IVI.
- Spoofing CAN timestamps to appear normal.
- Encrypting malicious payloads to avoid detection during OTA scanning.

Use Case:
Attacker patches the system log daemon to stop recording CAN injections.

🔐 **8.** Credential Access

Goal: Obtain usernames, passwords, keys, or tokens for authentication.

Example Activities:

- Dumping memory to extract Wi-Fi keys.
- Capturing hashed PINs sent over CAN.
- Brute-forcing debug access credentials on IVI systems.

Use Case:
Attacker recovers Wi-Fi credentials from infotainment memory and uses them for remote attacks.

🔍 **9.** Discovery

Goal: Map out internal vehicle systems and services.

Example Activities:

- Scanning CAN or Ethernet buses for active ECUs.
- Sending UDS requests to identify software versions or diagnostics.
- Enumerating connected control units using tools like CANalyse.

Use Case:
After access, attacker discovers brake and steering ECUs by scanning CAN services.

🔄 **10.** Lateral Movement

Goal: Move from one part of the system to another (e.g., IVI → Brake ECU).

Example Activities:

- Using CAN messages to pivot to other ECUs.
- Exploiting trusted communication between telematics and control systems.
- Injecting messages via Ethernet gateway.

Use Case:
Hacker uses the IVI system to send crafted messages to the TCU (Telematics Control Unit) to bridge to the CAN network.

📥 **11.** Collection

Goal: Gather sensitive data from the vehicle.

Example Activities:

- Capturing GPS logs, microphone recordings, or dashcam images.
- Logging driver behavior (speeding, braking).
- Reading ECU logs for sensitive telemetry.

Use Case:
Malware logs driver’s location history and sends it to attacker over C2.

📡 **12.** Command and Control (C2)

Goal: Establish a channel for the attacker to control the vehicle remotely.

Example Activities:

- Reverse shell from IVI to attacker server.
- Covert DNS or SMS-based command system.
- Using LTE or Wi-Fi to maintain persistent communication.

Use Case:
Attacker installs script that checks every hour for commands from an external IP.

📤 **13.** Exfiltration

Goal: Steal and transfer data out of the vehicle system.

Example Activities:

- Sending GPS, audio, or logs to remote server.
- Uploading stolen credentials.
- Exporting firmware dumps or camera images.

Use Case:
Attacker transfers log files from IVI to their FTP server using a covert channel.

🛑 **14.** Affect Vehicle Function

Goal: Directly control or sabotage the vehicle’s physical systems.

Example Activities:

- Sending spoofed CAN messages to disable brakes or honk the horn.
- Overriding lane assist or steering control.
- Disabling airbags or safety systems.

Use Case:
Attacker injects CAN frames to cut engine while driving — demonstrated in Jeep 2015 exploit.

⚙️ Three Techniques with Technique Codes

1. ATM-T0012 – Exploit via Radio Interface

- **Tactic**: Initial Access
- **Technique Description**: Attackers exploit weaknesses in Bluetooth, cellular telematics, or V2X radio interfaces to gain code execution access in vehicle systems (usually infotainment or telematics modules).
- **Real-World Context**: Used in the “PerfektBlue” attack chain where researchers achieved remote control by chaining Bluetooth vulnerabilities in major OEM infotainment stacks.
- **Typical Procedure**:
  1. Scan for the vehicle’s Bluetooth or telematics module broadcast.
  2. Send crafted protocol packets to trigger a stack overflow or logic bug.
  3. Pivot into the infotainment OS and achieve remote shell access or payload execution.

---

2. ATM-T0044 – Network Service Scanning

- **Tactic**: Discovery
- **Technique Description**: After gaining initial access, attackers probe the vehicle’s internal CAN or Ethernet networks to discover ECU services, ports, and diagnostic interfaces such as UDS (Unified Diagnostic Services).
- **Real-World Context**: Penetration testers routinely use tools like CANalyse or UDSim to enumerate firmware versions, diagnostic services, and communication bite IDs in ECUs.
- **Typical Procedure**:
  1. Inject scan frames or diagnostics requests into CAN or IP network.
  2. Log responses from ECUs such as brake, steering, or powertrain modules.
  3. Determine services like UDS support or firmware version to locate privilege escalation points.

---

3. ATM-T0009 – Rogue Wi-Fi Access Point

- **Tactic**: Manipulate Environment
- **Technique Description**: Attackers set a spoofed Wi-Fi hotspot to trick vehicle systems (infotainment or connected devices) into connecting, thereby enabling phishing, payload delivery, or telemetry interception.
- **Real-World Context**: Demonstrated in PoCs and testing of telematics interfaces, where a spoofed SSID (e.g. “Car-OTA_Update”) is used to deliver malicious firmware or harvest traffic.

- **Typical Procedure**:
  1. Configure a Wi-Fi AP with a spoofed SSID near a vehicle.
  2. Wait for the vehicle system to auto-connect.
  3. Serve a malicious payload (e.g. OTA update or application package) under guise of legitimate update or map patch.

📋 Summary Table

Technique Code	Name	Tactic	Real-World Use
ATM-T0012	Exploit via Radio Interface	Initial Access	Bluetooth vulnerability in IVI, remote code execution
ATM-T0044	Network Service Scanning	Discovery	CAN/UDS probing via CANalyse, firmware enumeration
ATM-T0009	Rogue Wi-Fi Access Point	Manipulate Environment	Spoofed Wi-Fi used to deliver OTA malware or intercept data

Procedure 1: Gaining Initial Access via Bluetooth Exploit

- **Tactic**: Initial Access
- **Technique**: **Exploit via Radio Interface** (ATM-T0012)
- **Goal**: Remotely access the infotainment system using a Bluetooth vulnerability.

🔧 Step-by-Step:

1. **Reconnaissance**: Scan for nearby vehicle Bluetooth devices using tools like hcitool or btmon.
2. **Target Selection**: Identify a known vulnerable device name or MAC address used by the target vehicle's infotainment.
3. **Exploit**: Send specially crafted L2CAP or SDP packets using tools like BTstack or a custom script to exploit a stack overflow or logic flaw.
4. **Shell Access**: Once exploited, gain access to the Linux-based OS running on the IVI (infotainment head unit).
5. **Persistence (optional)**: Drop a payload or create a cron job for re-entry.

Procedure 2: Internal Network Scanning using UDS to Discover ECU Functions

- **Tactic**: Discovery
- **Technique**: **Network Service Scanning** (`ATM-T0044`)
- **Goal**: Map the internal CAN network and discover active Electronic Control Units (ECUs) and their services.

🔧 Step-by-Step:

1. **Physical Access**: Connect a diagnostic tool or a CAN adapter (e.g., USB2CAN or Vector VN1610) to the OBD-II port or backend Ethernet/CAN port inside the car.
2. **Interface Setup**: Use tools like can-utils on Linux:
   sudo ip link set can0 up type can bitrate 500000
3. **Send UDS Requests**: Use isotpsend or custom UDS script to send service discovery commands: echo -n "1001" | isotpsend -s 0x7df -d 0x7e0 can0
4. **Log Responses**: Capture responses using candump: candump can0
   This helps identify modules like the Engine Control Unit, Brake ECU, or Telematics Control Unit and their firmware/software versions.

5. **Map ECUs**: Based on positive responses and their CAN IDs, draw a logical map of connected modules and their capabilities.

📚 References:

1. https://atm.automotiveisac.com/matrix

2. https://ioactive.com/pdfs/IOActive_Remote_Car_Hacking.pdf

3. https://www.nostarch.com/carhacking

4. https://www.armis.com/research/blueborne/

5. https://github.com/linux-can/can-utils

6. https://www.blackhat.com/docs/us-15/materials/us-15-Rogers-Look-No-Hands-Hacking-Remote-Vehicle-Systems.pdf