Finding, Managing, Preventing Vulnerabilities: An Automotive Perspective

Andreas Weichslgartner, Joyabrata Ghosh, Vineeth Prasanna
Who are we?
The Speakers

Dr.-Ing. Weichslgartner, Andreas

- IDS Dev, CI/CD Tester, Fuzzer, Vuln Management
- Dipl.-Ing. in ICT@FAU (2010), Dr.-Ing. in CS@FAU (2017), AEV/AUDI since 2017-2020, at CARIAD since 2020

Ghosh, Joyabrata

- Automotive Cybersecurity Management
- SBOM Security and Legal aspects
- Open-source enthusiast

Prasanna, Vineeth Bharadwaj

- Offensive Sec., Vuln Management
- China-GB/T
Revolutionizing automotive mobility requires tech expertise and scale. CARIAD and Volkswagen Group have it both.

9 leading car brands
7 million km per day of ADAS/AD data collected
14 million connected vehicles today and counting
1 consolidated software platform for the entire group
Founded in 2020, we have built a corporate startup from scratch attracting the best Tech Talents worldwide.

6,000+ CARIAD employees worldwide today

88 Nationalities working at CARIAD

360 teams at CARIAD

Join our mission and become part of one of the biggest endeavors in the automotive industry.
Agenda

• Introduction Automotive Software/Architecture
• Empirical Analysis of Automotive Software Vulnerabilities
• Automotive Supply-Chain
• SBOM/VDR/VEX/CBOM
• Vulnerability Management
• Lessons Learned
Disclaimer

All contents expressed in the following presentation are publicly known knowledge and only represent speakers’ personal opinions without any past, present and future employer viewpoints.
Automotive Architecture and Software
CAN Bus

- Popular field bus used in automotive and industrial automation
- Developed by Bosch in 1986
- Twisted pair wire (cheap and simple)
- Priority-based arbitration (0 is the highest priority)
- ID used for receiver
- CRC for error detection

```c
struct can_frame {
    canid_t can_id; /* 32 bit CAN_ID + flags */
    __u8 can_dlc; /* length in byte (0 .. 8) */
    __u8 __pad; /* padding */
    __u8 __res0; /* padding */
    __u8 __res1; /* padding */
    __u8 data[8];
};
```

linux/can.h @ torvalds/linux · GitHub (2023)
Automotive Architectures

Topology with central gateway

Automotive Architectures

Domain Model

Automotive Architectures

Zone Model

Automotive Architectures

Automotive Software

Commonly used operating systems:

- Android / Android Automotive
- Linux (e.g., Automotive Grade Linux, Yocto)
- Autosar (Adaptive/Classic)
- QNX
- VxWorks
Open Source SDV Initiatives

- **Automotive Grade Linux** (AWS, Toyota, Mazda, VW, Mercedes,...)
- **COVESA** (BMW, Ford, Bosch...)
- **The Eclipse Foundation: SDV** (Bosch, Mercedes, CARIAD, ZF, Microsoft...)
- **SOAFEE** (arm, CARIAD, Bosch, Microsoft, RedHat...)
In Other News
2-bug chain against the Alpine Halo9 (Twitter 2024-01-24)
Figure 6. Displaying an arbitrary message and a false speedometer reading on the Driver Information Center. Note that the car is in Park.

Koscher et al., Experimental Security Analysis of a Modern Automobile (2010)

CAN-Hack: Diebe klauen Autos über Netzwerkprotokoll ohne Schlüssel

Bei einem CAN-Injection-Angriff auf das Bussystem Controller Area Network reicht ein umgebauter Bluetooth-Lautsprecher, um das "Smart Key"-System auszutricksten.

heise.de

CAN do attitude: How thieves steal cars using network bus

It starts with a headlamp and fake smart speaker, and ends in an injection attack and a vanished motor

The Register

CAN Injection: keyless car theft

CANIS Blog

Car Thieves Hacking the CAN Bus

Car thieves are injecting malicious software into a car’s network through wires in the headlights (or taillights) that fool the car into believing that the electronic key is nearby.

Schneier on Security
**CVE-2023-6073 Detail**

**Description**
Attacker can perform a Denial of Service attack to crash the ICAS 3 IVI ECU in a Volkswagen ID.3 (and other vehicles of the VW Group with the same hardware) and spoof volume setting commands to irreversibly turn on audio volume to maximum via REST API calls.

**Severity**

<table>
<thead>
<tr>
<th>CVSS Version 3.x</th>
<th>CVSS Version 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CVSS 3.1 Severity and Metrics:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>NIST:</strong> NVD</td>
<td><strong>Base Score:</strong> 6.3 MEDIUM</td>
</tr>
<tr>
<td><strong>CNA:</strong> Automotive Security Research Group (ASRG)</td>
<td><strong>Base Score:</strong> 5.7 MEDIUM</td>
</tr>
</tbody>
</table>
CVE-2023-28897 Detail

Description
The secret value used for access to critical UDS services of the MIB3 infotainment is hardcoded in the firmware. Vulnerability discovered on Škoda Superb III (3V3) - 2.0 TDI manufactured in 2022.

Severity

<table>
<thead>
<tr>
<th>CVSS 3.x Severity and Metrics:</th>
</tr>
</thead>
</table>
Including the latest two CVEs reported for curl 8.4.0, the accumulated total says that **41%** of the security vulnerabilities ever found in curl would likely not have happened should we have used a memory-safe language.
A note on fuzzing: although we discovered this buffer overflow manually, we later tried to fuzz the vulnerable function, parse_tunables(); both AFL++ and libFuzzer re-discovered this overflow in less than a second, when provided with a dictionary of tunables (which can be compiled by running "ld.so --list-tunables").
What are the root causes of automotive vulnerabilities?
## CWE Top 25
Common Weakness Enumeration

<table>
<thead>
<tr>
<th>Number</th>
<th>CWE</th>
<th>Description</th>
<th>CVEs in KEV</th>
<th>Memory Safety</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CWE-787</td>
<td>Out-of-bounds Write</td>
<td>70</td>
<td>x</td>
</tr>
<tr>
<td>4</td>
<td>CWE-416</td>
<td>Use After Free</td>
<td>44</td>
<td>x</td>
</tr>
<tr>
<td>5</td>
<td>CWE-78</td>
<td>Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')</td>
<td>23</td>
<td>x</td>
</tr>
<tr>
<td>6</td>
<td>CWE-20</td>
<td>Improper Input Validation</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>CWE-125</td>
<td>Out-of-bounds Read</td>
<td>2</td>
<td>x</td>
</tr>
<tr>
<td>12</td>
<td>CWE-476</td>
<td>NULL Pointer Dereference</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>18</td>
<td>CWE-287</td>
<td>Use of Hard-coded Credentials</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>CWE-119</td>
<td>Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</td>
<td>8</td>
<td>x</td>
</tr>
</tbody>
</table>

CWE - 2023 CWE Top 25 Most Dangerous Software Weaknesses (mitre.org)
Methodology

- Based on Xiong et al. (2019)
- Published in Weichslgartner (2023)
- Query NVD for certain search terms
- Filter out false-positives

<table>
<thead>
<tr>
<th>Category</th>
<th>Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chipssets</td>
<td>snapdragon automobile, exynos auto</td>
</tr>
<tr>
<td>OEMs</td>
<td>acura, alfa romeo, aston martin, audi, bentley, bmw, bugatti, buick, cadillac, changan,</td>
</tr>
<tr>
<td></td>
<td>chevrolet, chrysler, citroën, dacia, daimler, dodge, dongfeng, ferrari, fiat, fisker,</td>
</tr>
<tr>
<td></td>
<td>ford, geely, general motors, gmc, great wall, honda, hyundai, infiniti, jaguar, jeep,</td>
</tr>
<tr>
<td></td>
<td>kia, lamborghini, lancia, land rover, lexus, maclaren, maserati, mazda, mercedes-benz,</td>
</tr>
<tr>
<td></td>
<td>mitsubishi, nissan, opel, pagani, peugeot, porsche, renault, rolls royce, saab, seat,</td>
</tr>
<tr>
<td></td>
<td>skoda, ssangyong, subaru, suzuki, tata motors, tesla, toyota, volkswagen</td>
</tr>
<tr>
<td>Automotive Technology</td>
<td>adaptive cruise control, adas, airbag, airbiquity, android auto, autoliv, bluetooth,</td>
</tr>
<tr>
<td></td>
<td>braking system, carlink, carplay, collision prevention, control unit, cruise, drivesync,</td>
</tr>
<tr>
<td></td>
<td>engine control, infotainment, keyless entry, lane keep assist, park assist, lidar,</td>
</tr>
<tr>
<td></td>
<td>controller area network, local interconnect network, media oriented systems transport,</td>
</tr>
<tr>
<td></td>
<td>flexray, obd-ii, passive anti-theft system, radio data system, steering control, telematics,</td>
</tr>
<tr>
<td></td>
<td>tire pressure</td>
</tr>
<tr>
<td>General Automotive Keywords</td>
<td>vehicle, car, automotive</td>
</tr>
</tbody>
</table>
CVEs by Category

Data from [https://nvd.nist.gov/](https://nvd.nist.gov/) (2023)
## Bounds Checks Are Hard!

<table>
<thead>
<tr>
<th>cve_number</th>
<th>impact</th>
<th>cwe</th>
<th>description</th>
<th>term</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2018-8260</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to an incorrect bounds check. This...</td>
</tr>
<tr>
<td>CVE-2018-8265</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to an incorrect bounds check. This...</td>
</tr>
<tr>
<td>CVE-2018-8268</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2018-8278</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2018-8293</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2018-8291</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-774</td>
<td>In Bluetooth, there is a possible remote code execution due to an improper memory allo...</td>
</tr>
<tr>
<td>CVE-2019-9311</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-190</td>
<td>In Bluetooth, there is a possible crash due to an integer overflow. This could lead to rem...</td>
</tr>
<tr>
<td>CVE-2019-9312</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2019-9320</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2019-9327</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2019-9328</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2019-9329</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-608</td>
<td>In Bluetooth, there is a possible out of bounds read due to uninitialized data. This could l...</td>
</tr>
<tr>
<td>CVE-2019-9330</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2019-9331</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2019-9332</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2019-9333</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2019-9341</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2019-9342</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2019-9343</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2019-9353</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2019-9355</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2019-9366</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-787</td>
<td>In Bluetooth, there is a possible out of bounds write due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2019-9365</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-502</td>
<td>In Bluetooth, there is a possible deserialization error due to missing string validation. Thi...</td>
</tr>
<tr>
<td>CVE-2019-9377</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2019-9388</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2019-9389</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-608</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2019-9387</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
<tr>
<td>CVE-2019-9388</td>
<td>CVSS:3.1</td>
<td>LAV/LAC/PR/RU/NI/N S</td>
<td>CWE-125</td>
<td>In Bluetooth, there is a possible out of bounds read due to a missing bounds check. This...</td>
</tr>
</tbody>
</table>
CVEs over Time

Data from https://nvd.nist.gov/ (2023)

- CWE-125: OOB Read
- CWE-787: OOB Write
- CWE-416: UAF
- CWE-476: Null Pointer Deref.
- CWE-120: Buf. Overfl.
- CWE-190: Integer Overflow
- CWE-862: Missing Auth.
- CWE-294: Auth Bypass
- CWE-20: Im. Inp. Val.
- CWE-200: Expose sens. Info
CVEs by CWE class

Data from https://nvd.nist.gov/(2023)
Memory Safety

- Memory safety:
  - All memory access adhere to semantics defined by language
  - E.g., runtime bounds checks, dereference checks, lifetime checks
- Memory Unsafe: C and C++ (unchecked pointer arithmetic)

- Spatial Memory Safety (still dominate CWE Top 25):
  - Only access within bounds of allocated object (CWE-787, CWE-125, CWE-119, CWE-476)
- Temporal Memory Safety:
  - Only access memory which is still valid (CWE-416)
Memory Safety CVEs

Data from https://nvd.nist.gov/ (2023)
Exploited CVEs according to Google Project Zero

- Memory Corruption: 68%
- Logic/Design Flaw: 20%
- Information Leak: 4%
- Use-after-free: 2%
- Other: 6%

Data from 0-days in the Wild (2023)
Eliminate Weakness Classes

- **Preventing**
  - Memory safe languages
  - Safe APIs/libraries
- **Mitigating**
  - Compiler/OS options
  - Sandboxing
- **Detecting** as early as possible
  - Fuzzing/Testing as part of development

A. Rebert, C. Kern: Secure by Design: Google’s Perspective on Memory Safety 2024
Training: Awareness, Tools

Requirements: KGAS, Coding Guidelines

Design: Security Risk Analysis

Implementation: SAST, SCA, Secret Scanning, Packet Manager, SBOM, CBOMs

Verification: DAST (Sanitizers, Fuzzers, etc.) Runtime CBOM

Release: Binary Hardening, Review, Pentest, Test SecTec

Response: Cont. Fuzzing, VulnScan
Supply Chain Security

Global Challenges

Australia: Critical Technology Supply Chain Principles, Security of Critical Infrastructure Act 2018


EU: GDPR, Cybersecurity Act, Cyber Resilience Act, Council conclusions on ICT supply chain security, NIS2, Chips Act (proposed).

Ireland: ECSM 009: Supply Chain Security.

New Zealand: NCSC Cyber Security Framework, Supply Chain Cyber Security

UK: Supply Chain Security Guidance, Supplier Assurance Framework, Secure development and deployment guidance, Supply Chain Guidance, How to Assess and Gain Confidence in Your Supply Chain Cybersecurity

UNECE Countries Automotive regulations: R155, R156, R157


The Minimum Elements For a Software Bill of Materials (SBOM) from NTIA
What is SBOM

A Software Bill of Materials (SBOM) declares the inventory of components used to build a software artifact such as a software application.

Why SBOM: Understanding the code that makes up our products provides all parties with a blueprint for:

- Cybersecurity (vulnerabilities, transitive dependencies) risk management
- Legal (copyright, license, plagiarism) risk management
- Automation of risk identification processes
SBOM

```json
{
  "bomFormat": "CycloneDX", "specVersion": "1.5",
  "serialNumber": "urn:uuid:ff872069-01a6-458d-95f6-43830efc9407", "version": 1,
  "metadata": {
    "timestamp": "2023-08-30",
    "component": {
      "bom-ref": "pkg:conan/insecure_lib@1.0",
      "type": "application",
      "name": "insecure_lib",
      "version": "1.0"
    },
    "licenses": [{"license": {"name": "Proprietary License CARIAD SE"}}],
    "authors": [{"name": "Joe Doe","email": "joe.doe@cariad.technology.de"}]
  },
  "components": [{
    "bom-ref": "pkg:conan/libcurl@7.64.1", "type": "library",
    "licenses": [{"license": {"id": "MIT"}}],
    "name": "libcurl",
    "version": "7.64.1",
    "purl": "pkg:conan/libcurl@7.64.1",
    "cpe": "cpe:2.3:a:haxx:libcurl:7.64.1:::*:::*:::*:::*:*" }
}
```
VDR

What is VDR

**NIST SP 800-161**: Cybersecurity Supply Chain Risk Management SP 800-161/ RA-5 2022 / page 144 defines VDR as:

*Enterprises, where applicable and appropriate, may consider providing customers with a Vulnerability Disclosure Report (VDR) to demonstrate proper and complete vulnerability assessments for components listed in SBOMs. The VDR should include the analysis and findings describing the impact (or lack of impact) that the reported vulnerability has on a component or product. The VDR should also contain information on plans to address the CVE. Enterprises should consider publishing the VDR within a secure portal available to customers and signing the VDR with a trusted, verifiable, private key that includes a timestamp indicating the date and time of the VDR signature and associated VDR.*
VDR Properties

A VDR shall contain the following properties:

- All vulnerabilities affecting and non-affecting a product or its' any transitive dependencies.
- Analysis describing the impact (or lack thereof) that a reported vulnerability has on a product or dependency.
- Signing the VDR with a trusted, verifiable, private key that includes a timestamp indicating the date and time of the VDR signature to ensure Interiority and Confidentiality as needed.
- Ensure all NIST NVD vulnerabilities identified in flexible order per product or selective component in latest live status.
- Serve as critical vulnerability exposure qualifier with an SBOM at any product or component release as the final proof that each component was evaluated for vulnerabilities before production.
- Online, living document always updated by the software producer and consumer at any time.
- Can be flexible model.
- Supported by SPDX V2.3 and CycloneDX V14 SBOM standards and above.
VDR example: full disclosure

"vulnerabilities": [
{ "id": "CVE-2023-38545",
  "source": { "name": "NVD", "url": "https://nvd.nist.gov/vuln/detail/CVE-2023-38545" },
  "ratings": [{
    "source": { "name": "NVD" },
    "score": 9.8,
    "severity": "critical",
    "method": "CVSSv3",
    "vector": "CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H"
  }],
  "description": "This flaw makes curl overflow a heap based buffer in the SOCKS5 proxy...",
  "recommendations": "Upgrade curl to version 8.4.0",
  "advisories": [ "url": "https://curl.se/docs/CVE-2023-38545.html" ],
  "affects": [ { "version": "vers:generic/\geq7.69.0\lt8.4.0",
    "ref": "pkg:conan/libcurl" } ]
}]}
According to the CISA VEX WG, a Vulnerability Exploitability eXchange (VEX) is:

*a form of a security advisory that indicates whether a product or products are affected by known vulnerability or vulnerabilities.*

VEX Properties

- A VEX allows a supplier or other party to *deterministically* assert the status of specific vulnerabilities in a product
- Supported by CycloneDX-14+, CSAF-2.0, OpenVEX specifications
- VEX initial Minimum Data Elements is under extension by CISA with several workgroups *(When to Issue VEX, Minimum Requirements for VEX* and more under drafts)*
VEX example

"vulnerabilities": [
  {
    "id": "CVE-2023-38545",
    "source": {
      "name": "NVD",
      "url": "https://nvd.nist.gov/vuln/detail/CVE-2023-38545"
    },
    "ratings": [
      {
        "source": {
          "name": "NVD"
        },
        "score": 9.8,
        "severity": "critical",
        "method": "CVSSv3",
        "vector": "CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H"
      }
    ],
    "description": "This flaw makes curl overflow a heap based buffer in the SOCKS5 proxy...",
    "analysis": {
      "state": "not_affected",
      "response": ["will_not_fix"],
      "justification": "code_not_reachable",
      "detail": "SOCKS5 proxy is not used"
    },
    "affects": [{
      "ref": "pkg:conan/libcurl@7.64.1"
    }]
  }
]
Crypto Bill of Materials (CBOM)
Cryptography can decay, e.g.:
- DES, MD5, SHA-1
- RSA (in case of quantum computing)
- Certificates and keys can get invalid

=> Cryptographic inventory and asset management needed
Finer Grained SBOMs

class CIDS(ConanFile):
    name = "Cariad-SWC"
    version = "24.1.0"
    homepage = "cariad.technology"
    author = "andreas weichslgartner (andreas.weichslgartner@cariad.technology)"
    default_options = { "botan:enable_modules":'uuid,auto_rng,sha2_32,system_rng' }

    def requirements(self):
        self.requires("botan/2.19.4")
Finer Grained SBOMs

{"bomFormat": "CycloneDX",
"components": [{
"bom-ref": "pkg:conan/botan@2.19.4",
"type": "library",
"name": "botan",
"version": "2.19.4",
"purl": "pkg:conan/botan@2.19.4",
"data": {"type": "configuration",
"name": "botan:config",
"properties": [{"name": "enable_modules",
"value": "uuid,auto_rng,sha2_32,system_rng"}]}]}

Crypto Bill of Materials (CBOM)

- Extends SBOM format with crypto assets
- Proposed by IBM IBM/CBOM
- Will be part of the next CycloneDX standard (v 1.6)

```
"components": [{
  "bom-ref": "pkg:conan/libcrypto@24.3.4",
  "type": "library",
  "name": "libcrypto",
  "version": "24.3.4",
  "purl": "pkg:conan/libcrypto@24.3.4",
  "components": [{
    "type": "cryptographic-asset",
    "oid": "2.16.840.1.101.3.4.2.1",
    "bom-ref": "oid:2.16.840.1.101.3.4.2.1",
    "name": "SHA-256",
    "cryptoProperties": {
      "assetType": "algorithm",
      "primitive": "hash"
    }
  }]
}
```
CBOM Tooling

- **Static Analysis:**
  - Function calls/parameters
  - Offered API
- **Dynamic Analysis:**
  - Runtime config
  - Negotiated ciphers

```csharp
// CodeQL broken crypto
string getAnInsecureAlgorithmName() {
    result = [
        "DES", "RC2", "RC4", "RC5", "ARCFOUR", // ARCFOUR is a variant of RC4
        "3DES", "DES3" // also appears separated, e.g. "TRIPLE-DES", which will be matched as "DES".
    ]
}

string getInsecureAlgorithmRegex() {
    result = // algorithms usually appear in names surrounded by characters that are not
    // alphabetical characters in the same case or numerical digits. This
    // handles the upper case:
    "(^|.*[^A-Z0-9])([^A-Z0-9]+strictconcat(getAnInsecureAlgorithmName(), "]") + "([^A-Z0-9]+\$" + ""] +
    // for lowercase, we want to be careful to avoid being confused by
    // camelCase, hence we require two preceding uppercase letters to be
    // sure of a case switch (or a preceding non-alphabetic, non-numeric
    // character).
    "([^A-Z][2].[^a-zA-Z0-9]+") +
    strictconcat(getAnInsecureAlgorithmName().toLowerCase(), "]") + "([^a-zA-Z0-9]+\$"
}
```

**CodeQL @Github**
Crypto Bill of Materials (CBOM)

- Generate CBOM of libcrypto with all offered crypto algorithms
- Use program analysis tool to search for function calls to libcrypto (e.g., regex, custom static analysis)
- Reference to CBOM to libcrypto

Component SBOM

Generate SBOM for every component
Automotive Vulnerability Management at Scale

“the challenges of scaling..”
Automotive Vulnerability Management at Scale

**ECUs:**
- Gateway ECU
- Door controller(s)
- Immobilizer
- ...

**Functions:**
- Remote wake-up/climate control
- 3rd party applications
- ...

**Services:**
- TLS
- Secure Onboard Communication
- Key management
- ...

ECUs:
- Gateway ECU
- Door controller(s)
- Immobilizer
- ...

Functions:
- Remote wake-up/climate control
- 3rd party applications
- ...

Services:
- TLS
- Secure Onboard Communication
- Key management
- ...
Automotive Vulnerability Management at Scale

Integrators (sometimes internals, or often contracted)

ECUs

Functions

Services

Supplier, sub-suppliers, sub-sub ..
Automotive Vulnerability Management at Scale
Automotive Vulnerability Management at Scale

- Tackling complex supplier matrix
- Tracing of vulnerability to the source
- Prioritizing the vulnerability reports
- Ensuring every patch is thoroughly tested
- Delivering the patch in a timely manner – updates are hard as we have multiple variants!
Automotive Vulnerability Management at Scale

- Average age of German vehicles [KBA (2023)]:
  - Cars (10.1 years)
  - Trucks (8.5 years)
  - Tractor unit (30 years)
- Average car lifetime until scrappage [Held et. al. (2021)]:
  - 14.8 years (Germany)
  - 8 years (Luxemburg)
  - 35.1 years (Poland)
Automotive Vulnerability Management at Scale

What does the regulations say?

UN Regulation No. 155 - Cyber security and cyber security management system:

- Also called UNECE R155
- UNCEC – United Nations Economic Commission for Europe:
  - Promotes pan-European economic integration

*UNECE R155 - 7.2.2.2* The vehicle manufacturer shall demonstrate that the processes used within their Cyber Security Management System ensure security is adequately considered, including risks and mitigations listed in Annex 5. This shall include:
  
  (c) The processes used for the assessment, categorization and treatment of the risks identified
  
  (g) The processes used to monitor for, detect and respond to cyber-attacks, cyber threats and vulnerabilities on vehicle types and the processes used to assess whether the cyber security measures implemented are still effective in the light of new cyber threats and vulnerabilities that have been identified.

*UNECE R155 - 7.2.2.3* The vehicle manufacturer shall demonstrate that the processes used within their Cyber Security Management System will ensure that, based on categorization referred to in paragraph 7.2.2.2 (c) and 7.2.2.2 (g), cyber threats and vulnerabilities which require a response from the vehicle manufacturer shall be mitigated within a reasonable timeframe.
Automotive Vulnerability Management at Scale

What does the regulations say?

China GB Standards:

- New (vehicle) cyber security standards being introduced in China
- **204-05 GB Whole Vehicle Cyber Security** - Analogous to UNECE R155
- As of March 2024, still in **draft**, car in-production from end of Q4-2027/early Q1-2028 will be affected (forecast)

7.3 General requirements for external connections:

7.3.1.2 External connection systems, such as systems with remote control function on the vehicle side and authorized third-party applications, shall be free of security vulnerabilities of high risk level or above that were announced by any authoritative vulnerability platform of the automotive industry 6 months ago and have not been handled yet.

- Note 1: Authoritative vulnerability platforms of the automotive industry include the NVDB-CAVD and other vulnerability platforms recognized by competent government authorities
- Note 2: Handling includes vulnerability elimination, development of mitigation measures, etc.
Automotive Vulnerability Management at Scale

Facing the challenges..

- Minor releases
- Major project maturity milestone
- OTA updates

- Start of new car project
- Start of production
- End of life

- Major project maturity milestone
-minor releases
- OTA updates
Automotive Vulnerability Management at Scale

Facing the challenges...

Some more hard challenges:

- What if the suppliers, sub-suppliers goes out of business after SOP?
- Financial model for approx. 10 years of support!
  - More suppliers == more maintenance contracts == expensive cars!
Automotive Vulnerability Management at Scale

*Facing the challenges..*

**Tackling complex supplier matrix:**

- **Solution for the VW Group:**
  - Shift from supplier dependent development to more in-house development!

- Basic security requirements for the suppliers: [VW KGAS](https://www.volkswagen-group.com/de/konzepte-kompetenzen/vw-kgas) (German: "Konzern Grundanforderungen Software")

- Group-wide cross-sectional specifications that define Volkswagen AG's minimum requirements for vehicle-related software installed in vehicles.

- Ensure suppliers deliver vulnerability free software, and any known vulnerabilities must be justified with risk assessments.
Automotive Vulnerability Management at Scale

*Facing the challenges...*

**Tracing of vulnerabilities to the source:**
- Vulnerabilities can arise anywhere: in hardware, services, functions

**Assume:**

**Deriving facts:**
- TLS service is affected
- List of ECUs which have affected openssl
- List of functions using TLS services

**Challenges:**
- Source of openssl package - ECU, function, or function/application?
- Applications can package their own openssl lib (e.g. 3rd party apps)
- Who is the owner of this risk?
Automotive Vulnerability Management at Scale

*Facing the challenges..*

**Tracing of vulnerabilities to the source:**

- Vulnerabilities can arise anywhere: in hardware, services, functions
- We setup multi-directional dependency tracking
- SBOMs!!
  - [TR-03183: Cyber Resilience Requirements for Manufacturers and Products by BSI](https://www.bsi-trustedshopservice.de) (German: Bundesamt für Sicherheit in der Informationstechnik)*
  - UNECE R155 based security roles are defined. Every function, ECU, and service has a dedicated security owner

*Not automotive relevant, but we follow!*

---

4 **SBOM formats**

An SBOM MUST be in a format that meets one of the following specifications in one of the specified versions:

- CycloneDX, version 1.4 or higher
- Software Package Data eXchange (SPDX), version 2.2 or higher

---

2024-03-26 | VulnCon | An Automotive Perspective | Weichslgartner/Ghosh/Prasanna
PUBLIC | CSD class 9.1
Automotive Vulnerability Management at Scale

*Facing the challenges..*

**Prioritizing vulnerability reports:**

- On average, companies were only able to remediate about 15.5% of their open vulnerabilities in a month. Source: Prioritization to Prediction, vol. 8, 2022 by Kenna Security and Cyentia

- CVSS & EPSS score is not always sufficient for prioritizing

![Diagram](image)

- **SRA = Security Risk Assessment**
- **ASIL = Automotive Safety Integrity Level** ([What is ASIL (Automotive Safety Integrity Level)? – Overview | Synopsys Automotive](https://www.synopsys.com/automotive/technical-resources/resource-center/what-is-asil-automotive-safety-integrity-level.html))
- **CAL (Cybersecurity Assurance Level) from ISO/SAE 21434:2021 is being standardized**
Automotive Vulnerability Management at Scale

*Facing the challenges..*

**Testing and delivering patches:**

- 10 years of support time – mobile devices (iOS & Android) support range from approx. 3-8 years
- Linux LTS kernel support ~ 2 years [Linux Kernel | endoflife.date](https://www.kernel.org/doc/Documentation/Kernel-Release-Milestones.txt)
- Maintaining HiLs, SiLs, vehicles, testing tools & equipments for ~10 years is a huge task!
- Patches need to be tested for every car variant before OTA updates

**Solutions:**

- Knowledge transfer when suppliers are involved
- Investing in test management – dedicated teams are setup
- Penetration tests before customer delivery
- Using open-source tools for testing (re-produce any test reports ~10 years!)
Lessons Learned

- Automotive vulnerabilities are analogous to any generic software vulnerabilities
- Memory (un)safety is a threat
- Use machine-processable standardized artefacts (SBOMs, CBOMs, VEX, VDR) in JSON
- Build SBOMs from package managers
- Ensure all missing software identifiers are registered/captured like CPE/PURL/SWID consistently
- Prefer public vulnerability reporting databases like NVD over any proprietary vendor lock-in databases.
- Automation is key
- Supplier & vulnerability management at scale is a big challenge – learn and evolve!
Questions?