2021 REPORT

CYBELLUM

The State of Automotive Software Security

Contents

01 Introduction

Connected cars have become a reality. This is an important milestone towards autonomous vehicles that will transform transportation and mobility services as we know them today. But with such advances come challenges.

Cyber threats in the automotive ecosystem can impact vehicle operation and the safety and security of road users. Without effective cybersecurity measures, modern vehicles, with their broad attack surface, are prone to malicious activities that could cause physical harm.

Industry-wide recognition of these threats is evident in the recent adoption of the UNECE WP.29 R155 regulation and the draft proposal for the ISO/SAE 21434 standard. Both put the cybersecurity onus on the OEM. This includes the need to manage risks associated with suppliers, service providers, and other organizations.

Managing cyber risk is, however, not a simple feat. The issues that underpin day-to-day security management are overwhelming and product security teams are under constant pressure to support the digital transformation that vehicles and their manufacturers undergo, despite limited resources and scarce cybersecurity talent.

This report sheds light on the key security risks associated with automotive software, based on analysis of over 100 automotive software components throughout 2020-2021.

The new regulations drive a fundamental change in automotive cybersecurity:

01

A "shift left" approach that addresses security early in the design and development process

02

The need for cybersecurity management throughout the vehicle lifespan, including post-development

03

Extending cybersecurity responsibility across the automotive supply chain

02 Key Findings

Automotive Vulnerabilities are Not "Regular" Vulnerabilities

While the past year saw the discovery of more vulnerabilities than ever before, only a fraction of these applied to automotive software. This has to do with the complex and diverse technological ecosystem of the automotive industry, which makes detection a challenge equivalent to searching for a needle in a haystack.

Automotive Software is Not Immune to Supply Chain Vulnerabilities

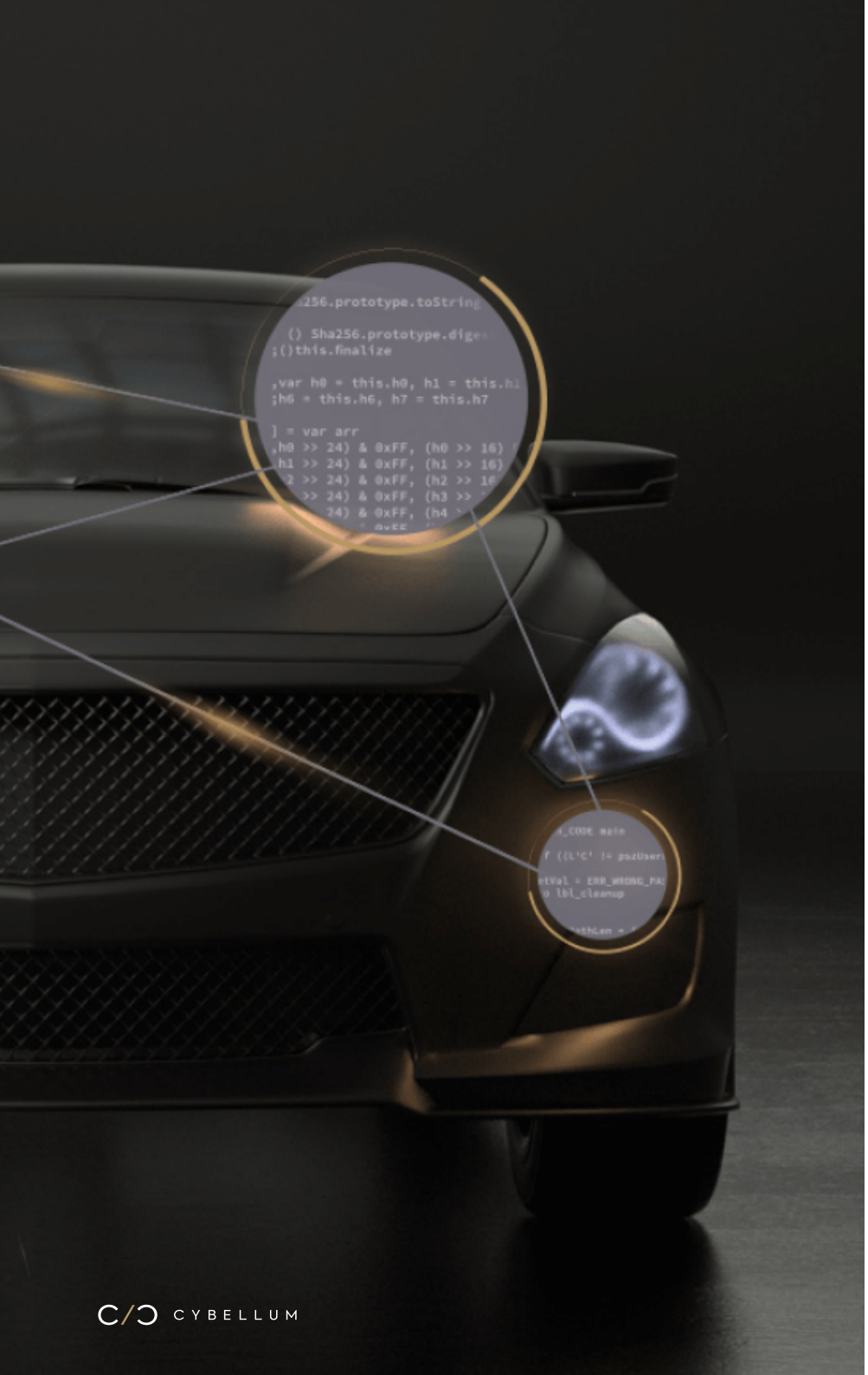
This may not come as a surprise given the complex automotive supply chain, but our research shows that fairly old vulnerabilities such as **BlueBorne** (discovered 4 years ago) as well as newer infamous vulnerabilities, such as **Ripple20**, impact automotive software.

Aging Software Poses Operational Risk

Manufacturers currently use a lot of outdated software and 90% of the ECUs we've analyzed contained software that is more than 5 years out-of-date, despite the availability of newer, more secure versions. Software that is not properly maintained poses a serious operational cybersecurity risk.

Memory Corruption Weaknesses Loom Over Vehicle Software

Most of the coding weaknesses affecting automotive software are of various memory mismanagement types. Buffer overflows, Null Dereferences and Use-After-Free are low-level weaknesses that have been known for years, yet it seems they remain even more relevant now than before.



03

Methodology

This report provides an in-depth snapshot of the current state of automotive software security. The Cybellum Research Lab has provided all information and data in this report without explicit reference.

At its core, the report is based on over 100 automotive software components that have been analyzed throughout 2020-2021.

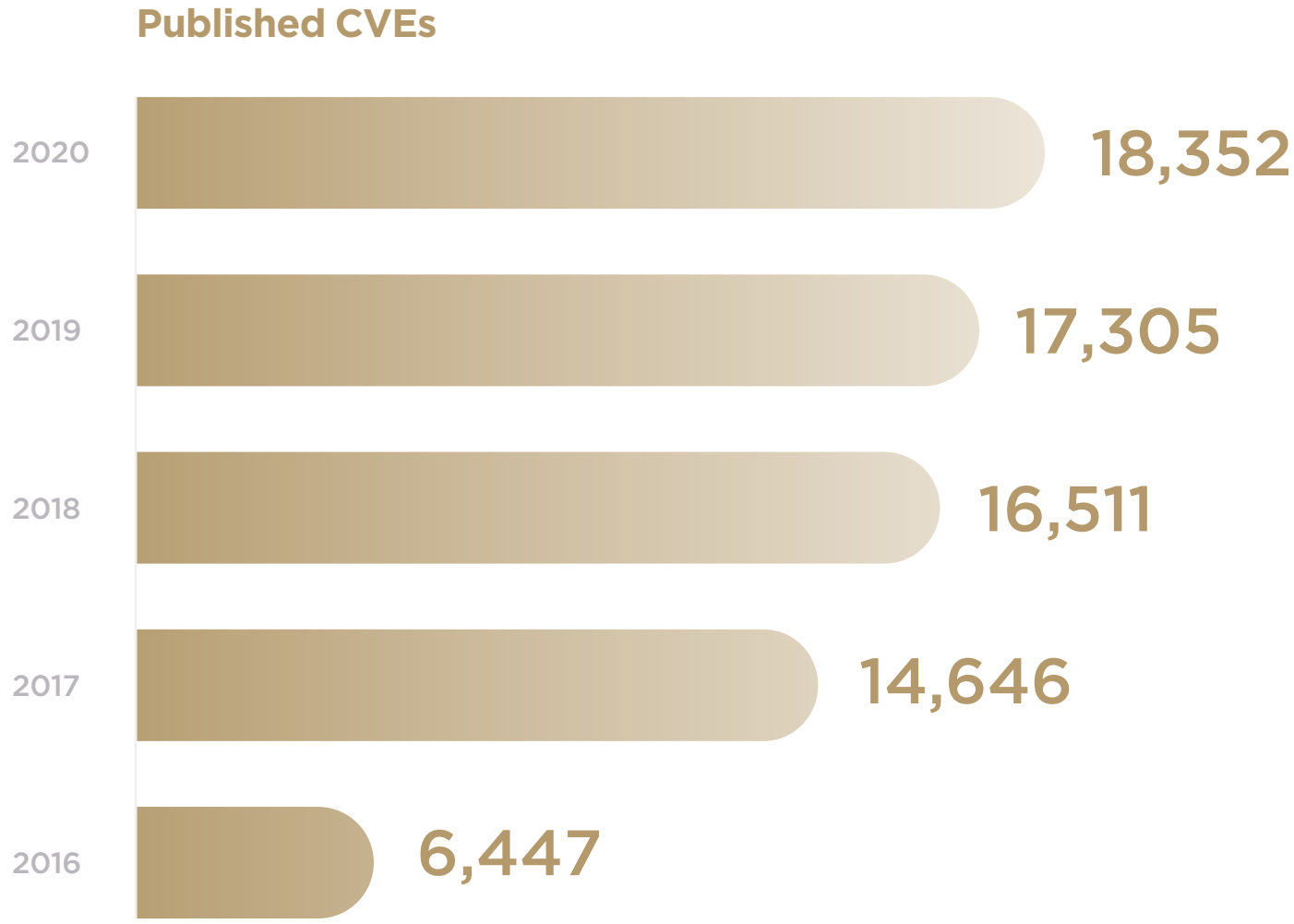
Finally, some information is based on various open/public sources – explicit reference is made where appropriate.

04 Automotive Vulnerabilities

The Needle in the Haystack Challenge

The continued increase in reported vulnerabilities is worrying: growth in vulnerabilities complicates prioritization, slows remediation processes and may lead to malicious activities in the future. Ultimately, it gets in the way of advancing the cyber resilience of automotive products and organizations.

However, these statistics do not tell the story as it relates to the automotive industry.



Source **NIST NVD**

The reality is that when analyzing vulnerabilities in the context of the software found in vehicle ECUs – its composition, characteristics and behavior – we find that most vulnerabilities are irrelevant, largely due to the different technological ecosystem in the automotive industry.

This is apparent in the following table listing **the 10 most searched CVEs on Google** - a simple proxy to the most popular CVEs in the research period – compared to **the most prevalent CVEs detected by Cybellum** in automotive software.

The lists are quite different. Where popular CVEs on Google cover a myriad of enterprise/IT products they are irrelevant in the context of embedded systems found in vehicles.

Most Searched CVEs *Google*			Most Popular Critical Automotive CVEs *Cybellum*		
CVE	**CVSS Score**	**Affected Software**	**CVE**	**CVSS Score**	**Affected Software**
2019-0708	9.8	**Remote Desktop Services**	2020-11656	9.8	**SQLite**
2017-11882	7.8	**Microsoft Office**	2019-19646	9.8	**SQLite**
2017-0199	7.8	**Microsoft Office**	2019-8457	9.8	**SQLite**
2018-11776	8.1	**Apache Struts**	2019-5482	9.8	**cURL**
2017-5638	10	**Jakarta Multipart parser**	2018-16842	9.1	**cURL**
2019-5544	9.8	**OpenSLP (ESXi / Horizon DaaS)**	2018-14618	9.8	**cURL**
2017-0143	8.1	**SMBv1 server (Microsoft Windows)**	2017-12652	9.8	**libpng.**
2020-0549	5.5	**Intel(R) Processors**	2017-10989	9.8	**SQLite**
2020-2555	9.8	**Oracle Coherence**	2018-1000301	9.1	**cURL**
2018-7600	9.8	**Drupal**	2018-1000120	9.8	**cURL**

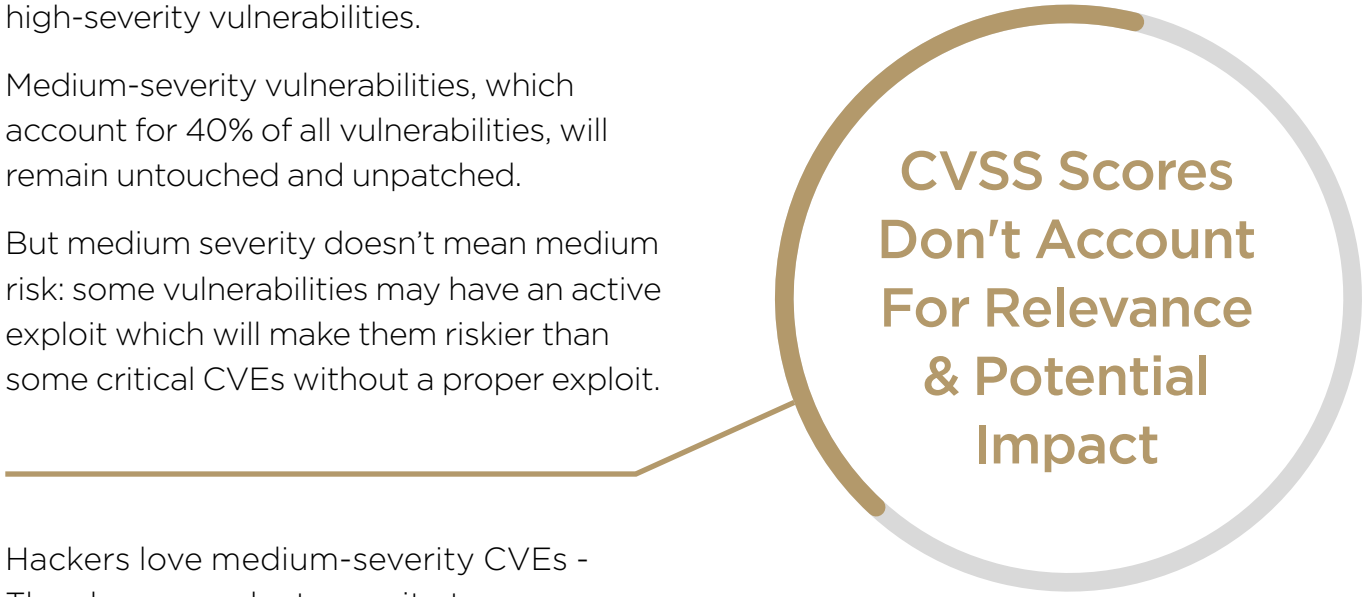
Severity vs. Relevance

The severity of a vulnerability, denoted in the above table by the CVSS score, is an important aspect of understanding the risks a vulnerability poses in a vehicle. It does not, however, show the full picture.

Because of resource constraints, automotive organizations will typically focus on critical vulnerabilities, and if possible, take a stab at high-severity vulnerabilities.

Medium-severity vulnerabilities, which account for 40% of all vulnerabilities, will remain untouched and unpatched.

But medium severity doesn't mean medium risk: some vulnerabilities may have an active exploit which will make them riskier than some critical CVEs without a proper exploit.



Hackers love medium-severity CVEs - They know product security teams are distracted by masses of critical and high-severity vulnerabilities and that they are ripe for attack.

These vulnerabilities act as a “foot in the door” for an attack chain, allowing lateral movement within the vehicle.

As previously indicated, there are several of these in automotive software.

By understanding the context in which automotive software operates and focusing on relevant CVEs that actually affect vehicle components, many vulnerabilities can be eliminated, making time to handle lower severity CVEs that have previously been neglected.

Work done by the Cybellum Research Lab shows that on average, over 80% of detected CVEs in vehicle components can be eliminated due to irrelevance.

Context awareness improves overall product security and resilience, while optimizing the Return-On-Investment (ROI) of your vulnerability management program.

Infamous Supply Chain Vulnerabilities

What's missing from the list above? Infamous vulnerabilities that have won multiple headlines in the press, have not made it to our list of top 10 automotive vulnerabilities.

But it does not mean they are not out there The top three positions on our **"Infamous Vulnerabilities"** list (BlueBorne, DirtyCow and Drown) were discovered 4-5 years ago, and they are still impacting automotive software. Solid cybersecurity practice should have mitigated these a long time ago.

Others, such as Ripple20, Amnesia:33 and Urgent/11 are fairly recent supply chain vulnerabilities lurking in popular communication stacks from multiple vendors.

Top Infamous Vulnerabilities in Automotive Software

Rank	Vulnerability Alias	Disclosure Year
01	BlueBorne	2017
02	DirtyCow	2016
03	Drown	2016
04	SweynTooth	2020
05	Ripple20	2020
06	Amnesia:33	2020
07	Urgent/11	2019
08	Ghost	2015

KEY TAKEAWAYS

Most published CVEs do not affect automotive software (but there are still plenty of those around) – detecting those relevant to your vehicle or component is like searching for a needle in a haystack

Vulnerability management programs can benefit dramatically by incorporating context awareness to focus remediation efforts at the most relevant risks. In fact, most of the security issues can be detected and remediated during the R&D process (pre-SOP)

Legacy practices rooted in manual detection and response don't provide the insight and lack the automation needed to scale remediation

Supply chain risks have a real impact on automotive software; vulnerabilities from a few years ago should and could be mitigated through solid security practice

05

Aging Software Risks

All software ages, be it of open-source nature, commercial or of our own making. As it ages, it loses support which means no more feature improvements, stability updates and security fixes.

Developers are often concerned that using a newer software version will require modifying other code, causing a knock-on effect that could slow down development. As long as the code continues to function as it's supposed to, most development teams will ignore it and eventually it will be forgotten.

The 5 Oldest Packages Found In Automotive Software

Package name	Version date	Latest release date	Out of date	Exploits	Critical CVEs
libxml2-1.1	2 JUN 1999	30 OCT 2019	**20 Years**	Exploit 1	2016-1834 2016-4448 2016-4658 2017-7376
glibc-2.2	9 NOV 2000	1 FEB 2021	**20 Years**	Exploit 2, Exploit 3, Exploit 4*, Exploit 5*, Exploit 6*	2015-0235
ncurses-5.0	29 OCT 1999	12 FEB 2021	**19 Years**		
glib-2.0	13 APR 2002	8 APR 2021	**19 Years**		
mesa-6.3	20 JUL 2005	7 APR 2021	**16 Years**		

**Exploits available on DarkNet*

But using outdated software is an operational risk, as at some point fixing or patching may become too difficult or impossible, putting at risk the proper functioning of a component, and perhaps the entire vehicle.

Cybellum findings validate the above – in the researched firmware, we detected software packages that had not been updated in 20 years and had critical CVEs as well as exploits.

These are quite extreme examples, but the research shows that over 90% of analyzed ECUs contained software packages that were more than 5 years out-of-date, exposing the components to a higher risk of vulnerabilities and exploits.

KEY TAKEAWAYS

Software packages in analyzed ECUs

Aging software is common in the automotive industry - policies should be created to address the risks of out-dated code

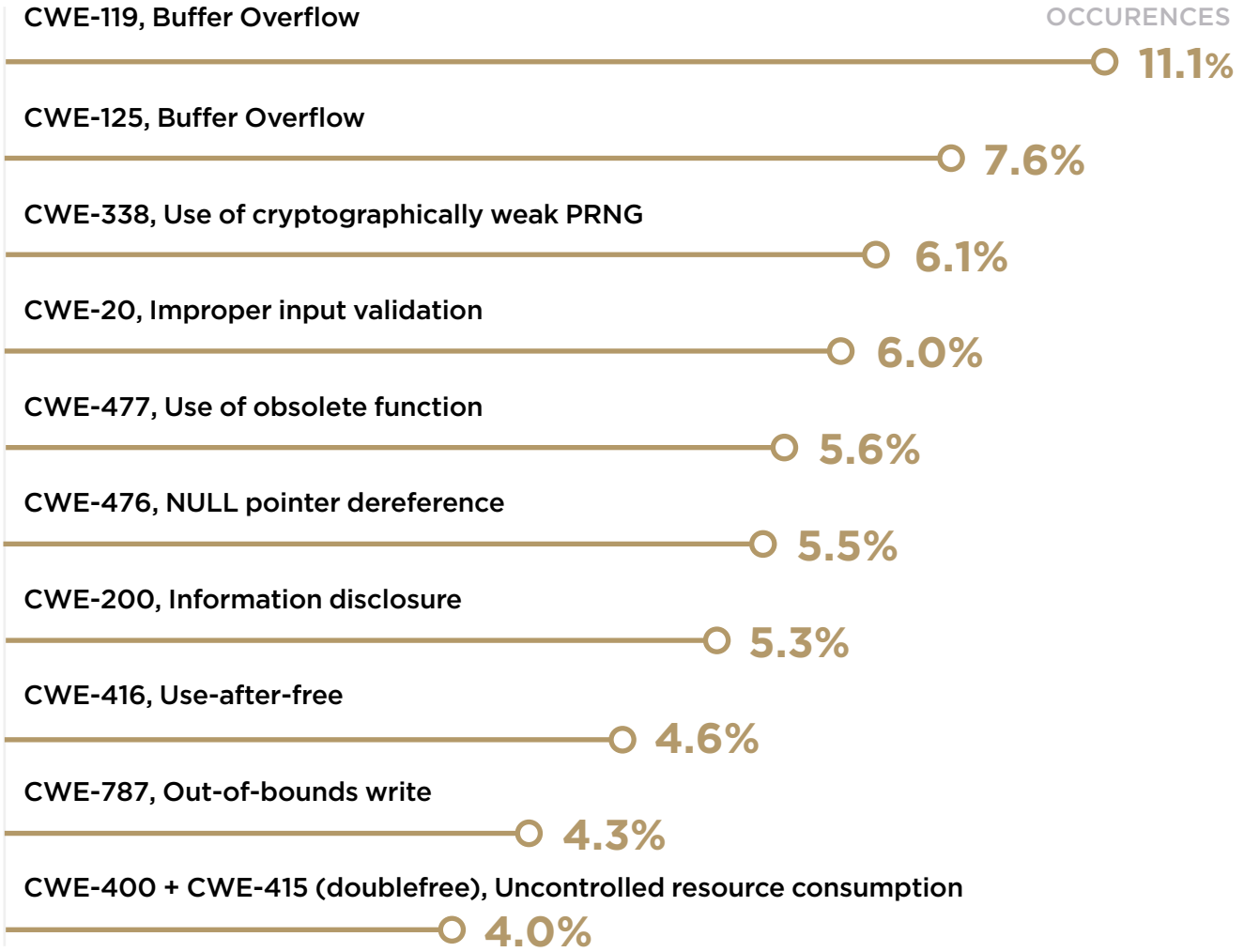
Mitigation of vulnerabilities through software updates seems like a reasonable solution, but in reality, it faces many challenges - product security teams should consider alternative methods such as patching, configurations and hardening mechanisms

06 Memory Corruption Looms Over Automotive Software

Our research shows that most of the coding weaknesses discovered in automotive software, be those underlying CWEs of public CVEs or coding weaknesses in proprietary code, are forms of memory corruption - buffer overflows, null dereferences, use-after-free, out-of-bound-writes and more.

These are all low-level weaknesses that have affected computer systems for years so one would expect high coder awareness, resulting in code which is almost completely secure in this regard. Unfortunately This is not the case.

Top CWEs Found in Automotive Software



Unlike most of the CWEs in the table, Improper Input Validation (CWE-20, ranked number four) is not a memory corruption vulnerability and is highly exploitable hence worth special attention.

Input validation is a technique for checking potentially dangerous inputs to ensure that they are safe for processing within the code, or when communicating with other components.

When the software does not validate it properly, an attacker can craft the input in a certain form so that parts of the application receive unintended input that will result in altered control flow, arbitrary control of a resource, or arbitrary code execution.

The exploit process of this coding weakness is relatively simple and requires lesser reverse engineering skills and experience, so even a novice attacker can execute a severe attack relatively easily.

What is Memory Corruption?

Memory corruption can be described as a vulnerability that may occur in a computer system when its memory is altered without an explicit assignment. The contents of a memory location are modified due to programming errors which enable attackers to execute an arbitrary code. Examples of memory corruptions include:

BUFFER OVERFLOW occurs when the volume of data exceeds the storage capacity of the memory buffer. Attackers exploit it by overwriting the memory of an application, changing the execution path of the program, triggering a response that damages files or exposes private information.

USE-AFTER FREE refers to a memory corruption flaw that occurs when an application tries to use memory no longer assigned to it. In cyber attack scenarios this can allow an attacker to gain remote code execution capabilities.

DOUBLE FREE errors occur when you release or deallocate memory blocks more than once with the same memory address as an argument. It may result in a write-what-where condition, allowing an attacker to execute arbitrary code.

KEY TAKEAWAYS

Most CWEs in the automotive ECUs relate to memory corruption

Explore automated solutions that will assist in identifying coding weaknesses

Product security teams should take note of this and employ continuous testing and validation practices throughout the SDLC

Education of engineering teams and security awareness programs should be created (or refreshed) to promote secure coding best practices

07 Conclusion

You need look no further than the pages of this report to see that, while software is literally driving today's vehicles, automotive cyber risks are growing, specifically in the areas of security, code quality and maintenance.

R&D organizations and product security teams need to establish ongoing processes and supportive policies to proactively manage automotive cyber risk.

This involves gaining clear visibility and understanding of the make and characteristics of their software asset inventory, reliable and timely vulnerability data and guidance on how to handle affected code.

This is not an easy feat, but the following checklist will help you get there:

- ☑ Create a comprehensive software asset inventory with accurate S-BOM and enhanced contextual data for your vulnerability management program
- ☑ Infuse your decision making with context awareness to help prioritize efforts on the most pressing issues, automate triaging activities and more
- ☑ Use smart automation (where appropriate) to operate effectively at scale
- ☑ Take control of your supply chain - vulnerabilities that should have been dealt with years ago are still present, while new ones are being introduced
- ☑ Re-evaluate your vulnerability mitigation strategy - consider patching, configurations and hardening mechanisms in parallel to software updates
- ☑ Establish policies to address the potential risks associated with using out-dated code
- ☑ Employ continuous testing and validation throughout the SDLC to mitigate known vulnerabilities and coding errors
- ☑ Drive awareness to memory corruption risks and promote secure coding practices through educational programs for your engineering team

ABOUT CYBELLUM

Cybellum empowers automotive OEMs and their suppliers to identify and remediate security risks at scale, throughout the vehicle lifespan. Our Cyber Digital Twins™ platform analyzes embedded software components without needing access to their source code, exposing known vulnerabilities, zero days and compliance violations.

Manufacturers can then take immediate actions to eliminate any cyber risk in the development and production process, before any harm is done, while continuously monitoring for emerging threats impacting vehicles on the road.

For more information, please visit
www.cybellum.com

or connect with us on