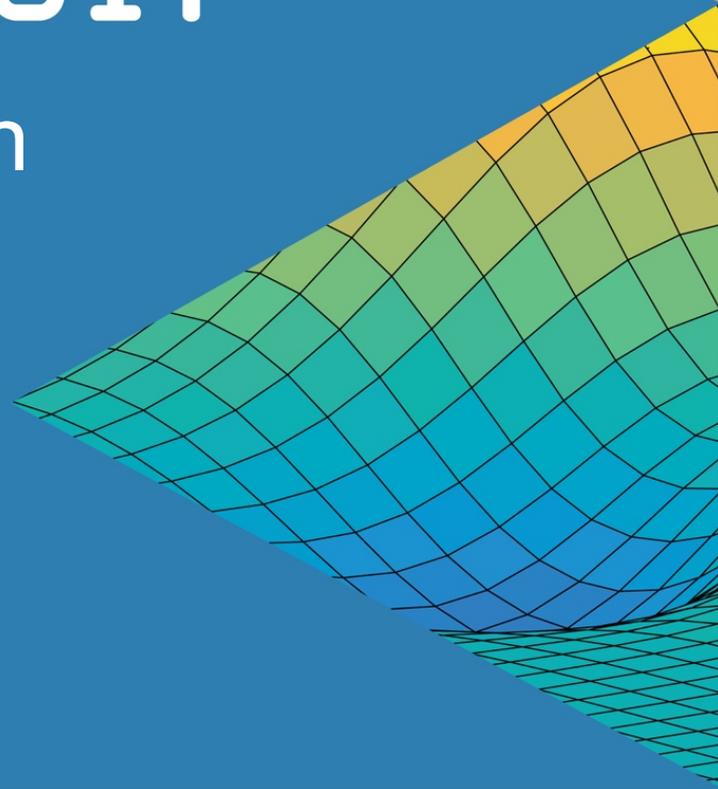


# MATLAB EXPO 2017

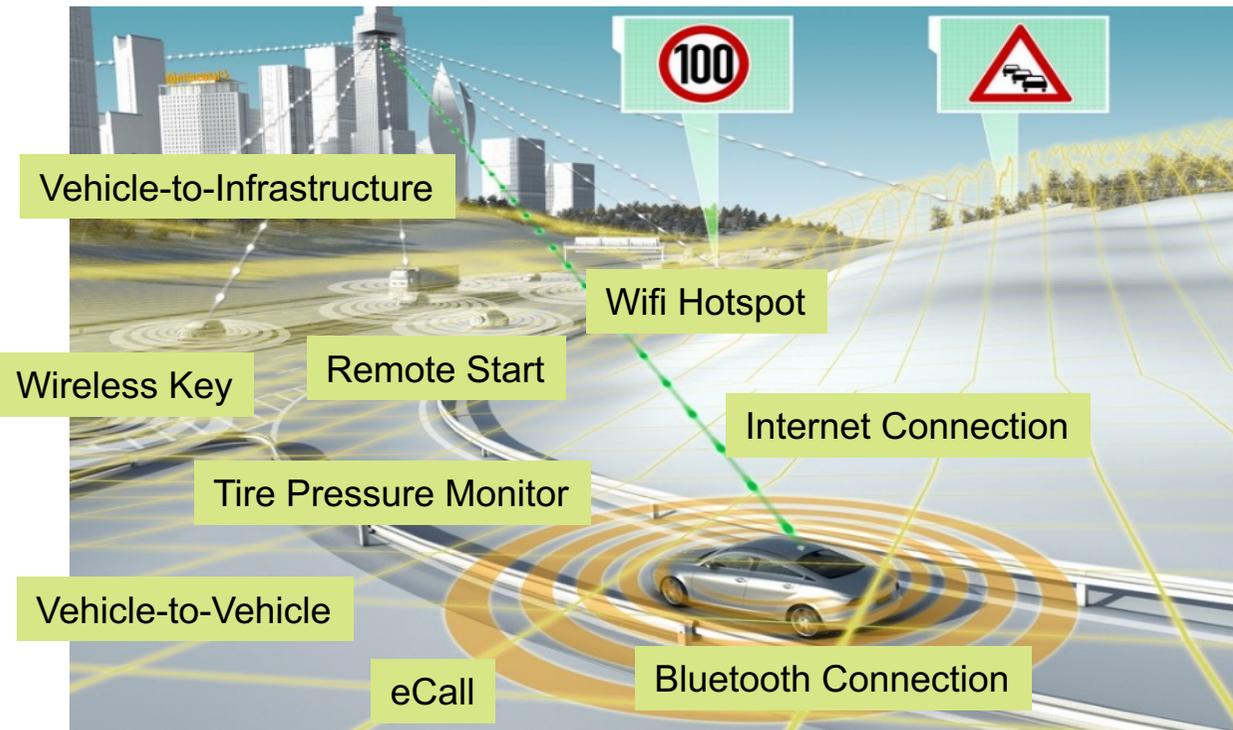
## Automatisches Erkennen von Sicherheitslücken mit Polyspace

Christian Guß



# Cybersecurity – Emerging Topic in the Auto Industry

## On-Board & V2X Communication



- Growing communication of on-board systems, sensors and external sites
- Car becomes another node of IoT
- Security of automotive embedded systems increasingly important (possible cyber attacks)

## FCA recalls 1.4 Million cars after Jeep hack

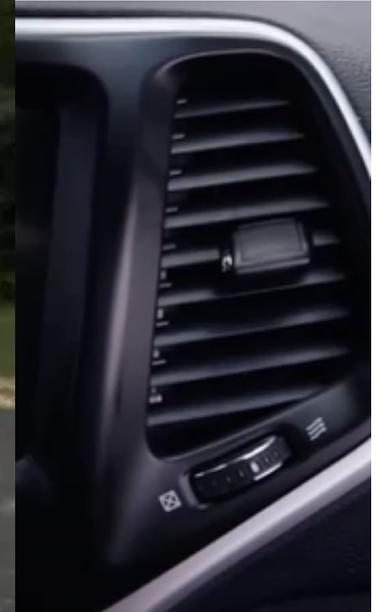


# Embedded Software Security New Challenge



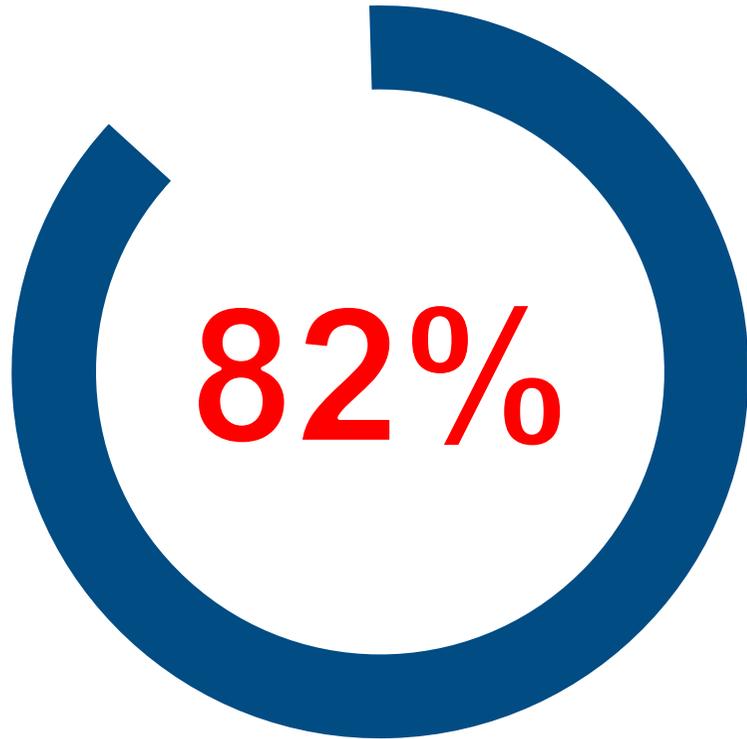
Miller (left)  
drove it

ON THE

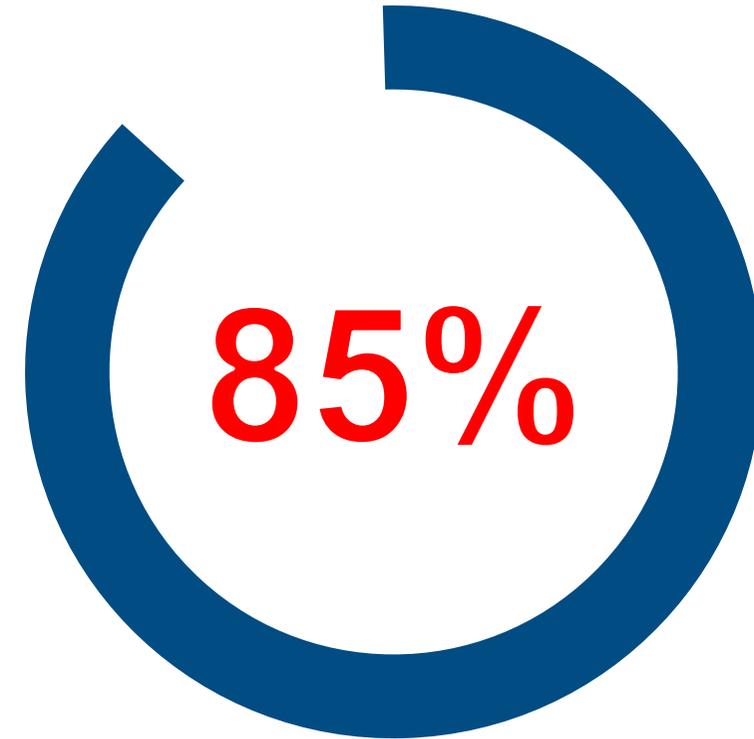




## Security is on consumers' mind

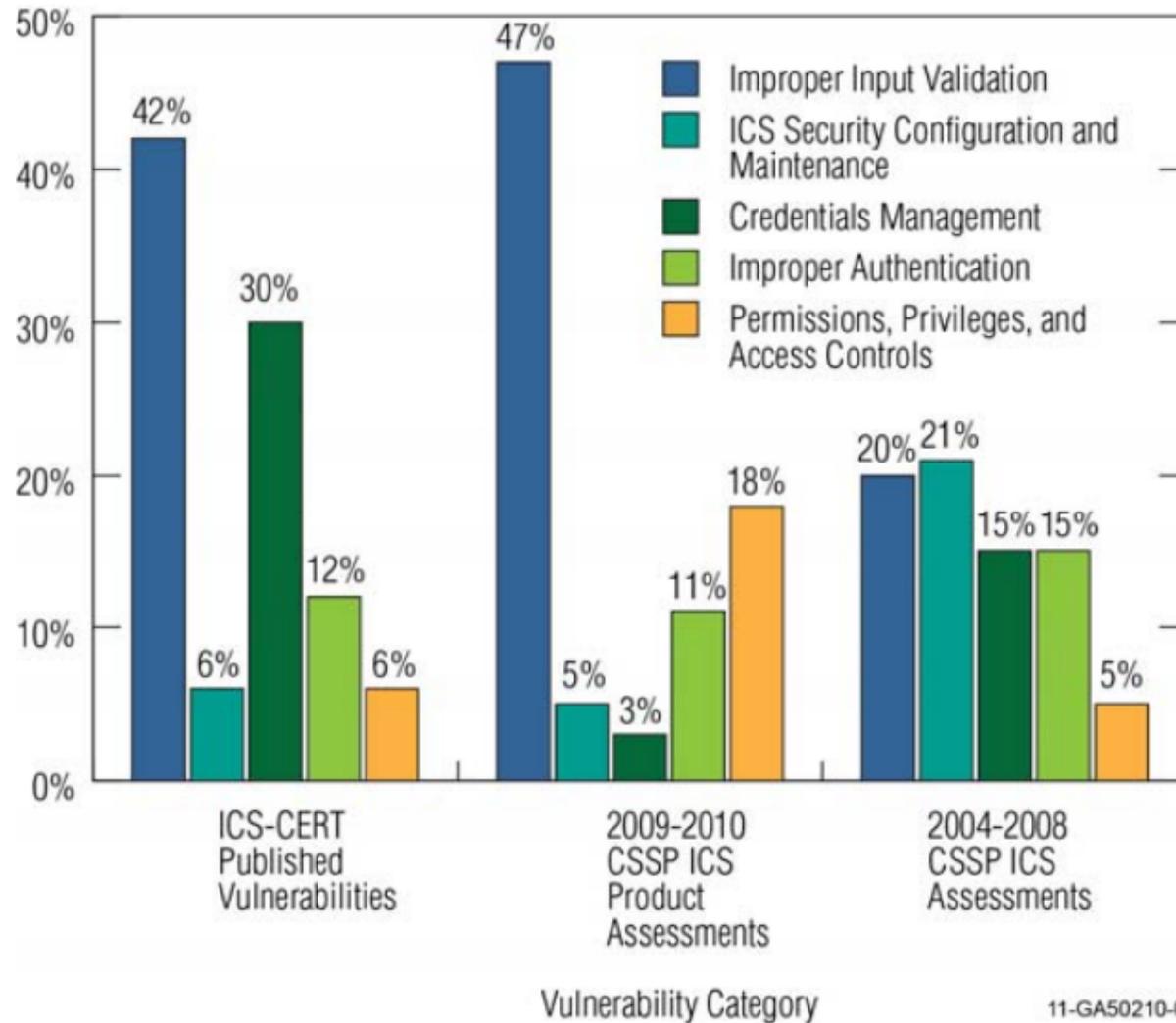


...of customers would never buy from automaker if they had been hacked



...of automakers admit their organization have been breached in the past 2 years

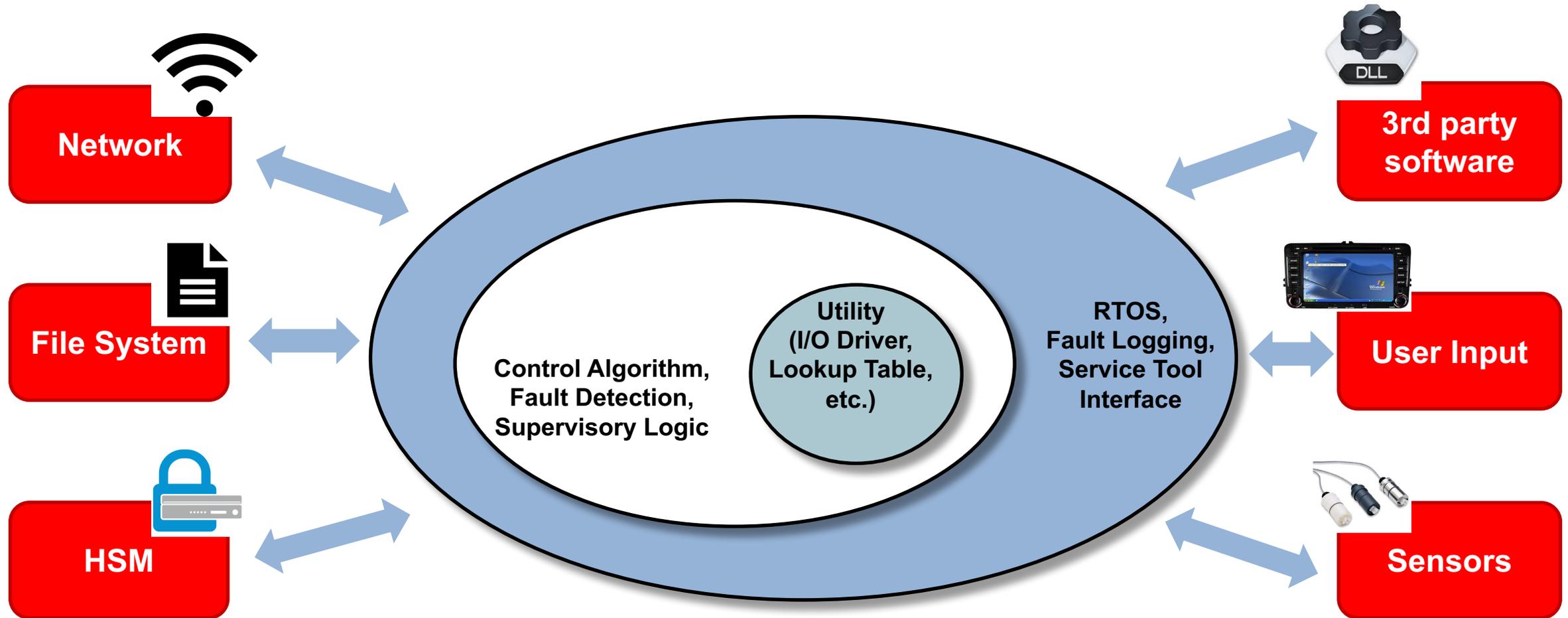
# Cybersecurity – Emerging Topic in the Internet of Things



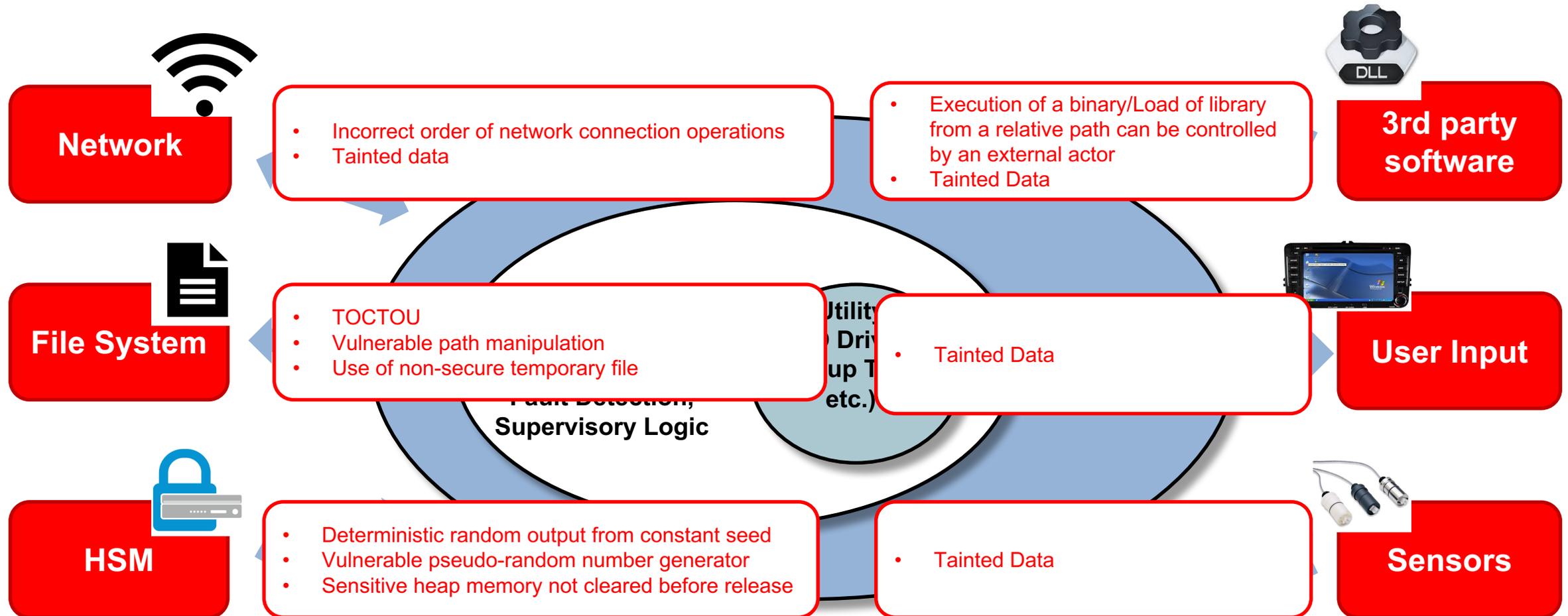
11-GA50210-01

Source:  
 „Industrial Control Systems (ICS)  
 Cyber Emergency Response Team“ (ICS-CERT)

# Embedded Software External Interactions



# Embedded Software Security Concerns



# Polyspace helps you to....

# Identify and prove absence of critical defects

## Enforce coding rules

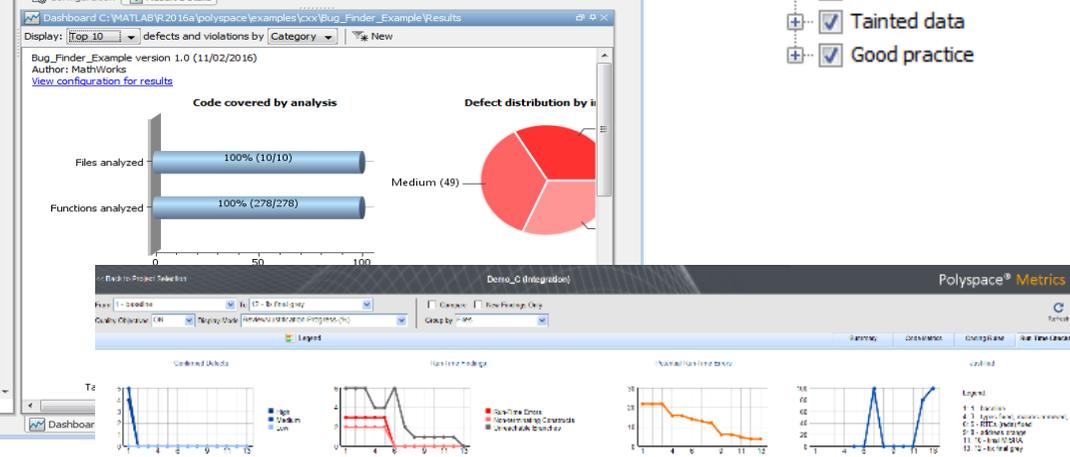
- +1 A standard C environment 2
- +2 Compilation and build 1
- +2 Unused code 13
- +4 Code design 12
- +5 Identifiers 93
- +7 Literals and constants 17
- +8 Declarations and definitions 503
- +15 Control flow 31
- +16 Switch statements 1
- +17 Functions 159
- +18 Pointers and arrays 6
- +21 Standard libraries 289
- +22 Resources 16

The screenshot shows the Polyspace Bug Finder interface. On the left, a tree view lists various defect categories such as Programming, Concurrency, and Numerical. The main window displays a detailed view of a defect: 'Data race (Impact: High)'. The description states: 'Certain operations on variable 'bad\_glob' can interfere with each other. To avoid interference, operations on 'bad\_glob' must be in the same thread.' Below this, an 'Event' list shows:
 

- 1 Take the address of variable 'i'
- 2 Formal argument number 1 (ptr\_) of call to f
- 3 ! Invalid free of pointer

 A large green checkmark is overlaid on the interface, indicating that the defect has been resolved or proven absent.

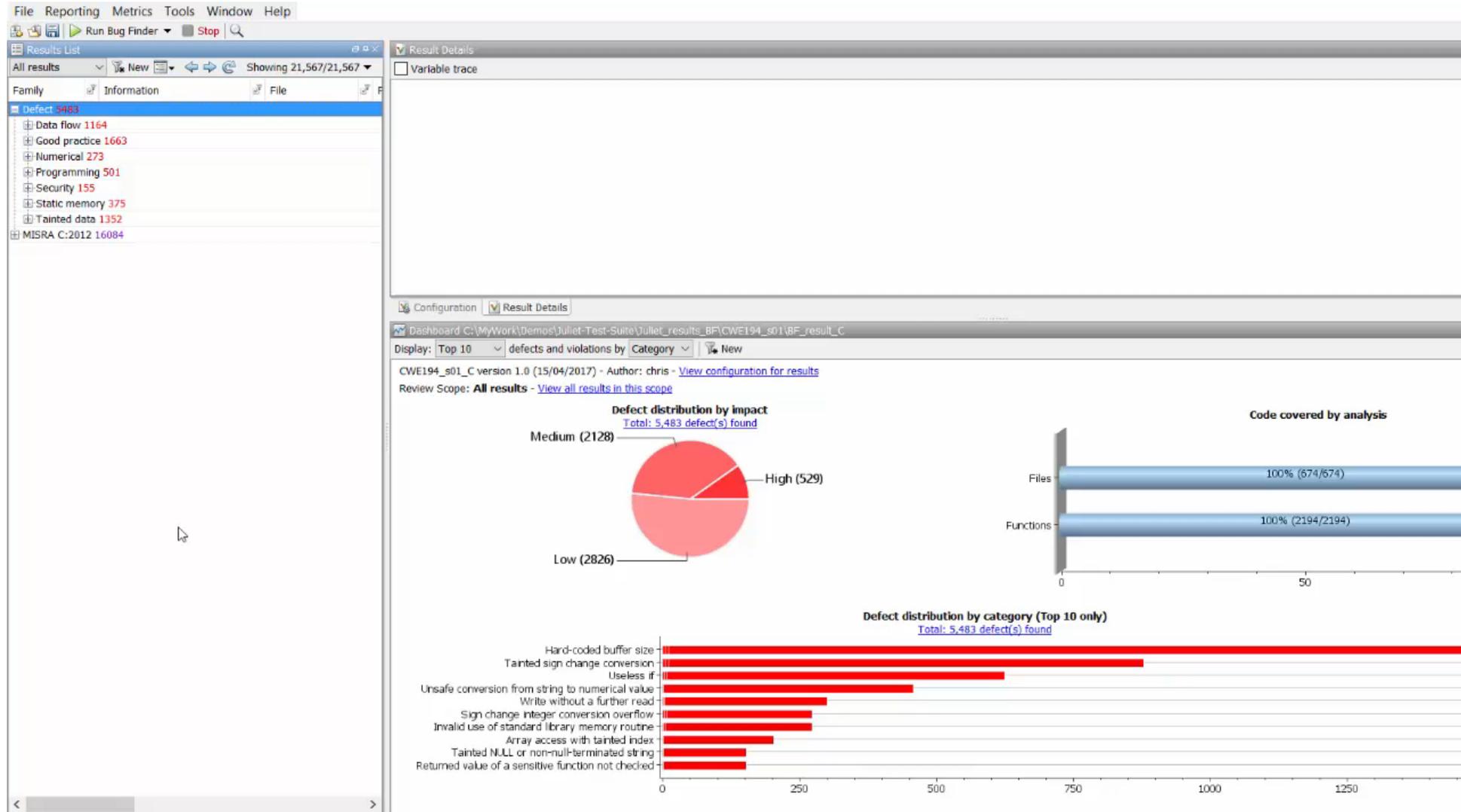
- Defects
- Numerical
- Static memory
- Dynamic memory
- Data flow
- Resource management
- Programming
- Concurrency
- Security
- Tainted data
- Good practice



## Produce and monitor quality metrics

Validator	Controlled Defects	Run-Time Defects	Open Code	Synthetic Software (Pre-Run Checks)	Unresolvable Priorities (Pre-Run Checks)	Open Software (Always Checks)	Unresolvable Priorities (Always Checks)	Unresolvable Priorities (Always Checks)	Software Quality Collector	Quality Status	Level	Items Progress
0 - Full test suite	100.0%	212	0.0%	0.0%	0.0%	100.0%	4	2	PASS	100.0%	100.0%	100.0%
1 - Compiler default	99.7%	311	0.0%	0.0%	1	100.0%	4	2	FAIL	100.0%	100.0%	100.0%
2 - Final C/C++	99.4%	316	0.0%	0.0%	1	100.0%	4	2	FAIL	100.0%	100.0%	100.0%
3 - Binary C/C++	99.0%	306	0.0%	0.0%	1	100.0%	4	2	FAIL	100.0%	100.0%	100.0%
4 - Address range	99.0%	316	0.0%	0.0%	1	100.0%	4	2	FAIL	100.0%	100.0%	100.0%
5 - Address range (64)	99.0%	205	0.0%	0.0%	1	100.0%	12	2	FAIL	100.0%	100.0%	100.0%
6 - Address range (16)	98.5%	291	0.0%	0.0%	2	100.0%	13	1	FAIL	100.0%	100.0%	100.0%
7 - RTTI (disabled)	99.3%	218	0.0%	0.0%	0	100.0%	14	1	FAIL	100.0%	100.0%	100.0%
8 - RTTI (enabled)	99.4%	376	0.0%	0.0%	4	100.0%	16	1	FAIL	100.0%	100.0%	100.0%
9 - System (new)	99.0%	376	0.0%	0.0%	2	100.0%	16	1	FAIL	100.0%	100.0%	100.0%
10 - System (old)	99.0%	351	0.0%	0.0%	1	100.0%	13	1	FAIL	100.0%	100.0%	100.0%
11 - System (new)	99.0%	291	0.0%	0.0%	2	100.0%	13	2	FAIL	100.0%	100.0%	100.0%
12 - System (old)	98.8%	304	0.0%	0.0%	4	100.0%	13	2	FAIL	100.0%	100.0%	100.0%

# What is “Tainted Data”?



# Cybersecurity – Industry Activities & Standards

## **SAE – Vehicle Cybersecurity Systems Engineering Committee**

- SAE J3061 - Cybersecurity Guidebook for Cyber-Physical Vehicle Systems
- SAE J3101 - Requirements for Hardware-Protected Security for Ground Vehicle Applications (WIP)
- SAE “Cybersecurity Assurance Testing Task Force” (TEVEES18A1)

## **Coding standards & practices that we observe at automotive customers**

- CERT C
- ISO/IEC TS 17961 – C Secure Coding Rules
- CWE – Common Weakness Enumeration
- MISRA-C:2012 Amendment 1

## Polyspace helps you to....

- ✓ Enforce **new coding rules**  
(all of them required or mandatory)
  
- ✓ **1 new directive** [4.14]
  
- ✓ **13 new rules** on
  - ✓ Expressions [12.5]
  - ✓ Standard libraries [21.13, 21.14, 21.15, 21.16, 21.17, 21.18, 21.19, 21.20]
  - ✓ Resources [22.7, 22.8, 22.9, 22.10]
  
- ✓ Changes to existing rules [21.8]

### MISRA C:2012 Amendment 1

Additional security  
guidelines for MISRA C:2012

April 2016

# Example of new MISRA directive 4.14

▼ **MISRA C:2012 D4.14** (Required) ?

The validity of values received from external sources shall be checked.  
 Division (/) operation operands are from an unsecure source. Check for:

- Denominator of zero.
- Numerator of minimum value and denominator of -1.

	Event	File	Scope	Line
1	Formal parameter is a tainted value	tainteddata.c	bug_taintedintdivision()	108
2	▼ MISRA C:2012 D4.14	tainteddata.c	bug_taintedintdivision()	109

Configuration Result Details

Source

tainteddata.c x

```

104
105 /*=====
106  * USING TAINTED DATA AS NUMERATOR IN DIVISION
107  *=====*/
108 int bug_taintedintdivision(int usernum, int userden) {
109     int r = usernum/userden;          /* Defect: Parameter is not checked before being used as numerator */
110     print_int(r);
111     return r;
112 }
  
```

# Summary

## Cybersecurity ...

- ✓ enable new future markets
- ✓ entire vehicle lifecycle process
- ✓ by design

