



MathWorks
AUTOMOTIVE
CONFERENCE 2019

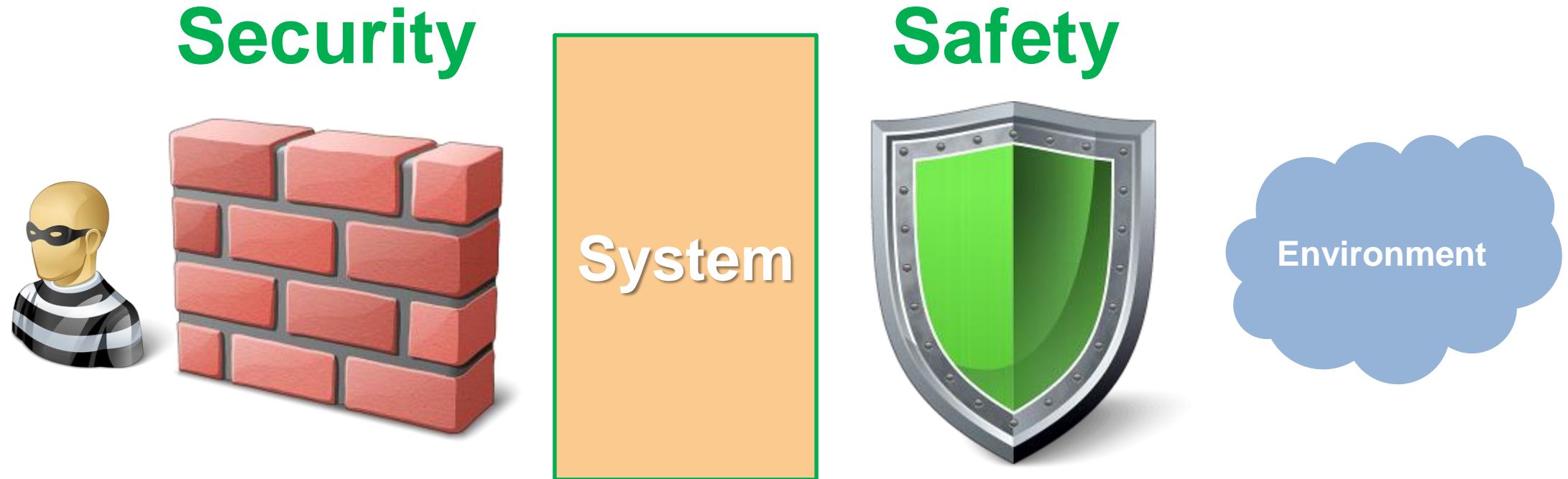
Secure Coding Guidelines를 위한
Polyspace 제품군

유용출

Agenda

- Safety vs Security
- Why Check Secure Coding Guidelines?
- How to Apply Secure Coding Guidelines?

Safety vs. Security



Note: Security issues may cause safety issues

5W1H - Secure Coding Guidelines

Who

You or Your colleagues
(Developers, QA, QE)

Where

at workplace

What

Most of your software running on target

When

Every day, week, month

5W1H - Secure Coding Guidelines

Why

1. Security issues may lead catastrophe
2. Required by your customers

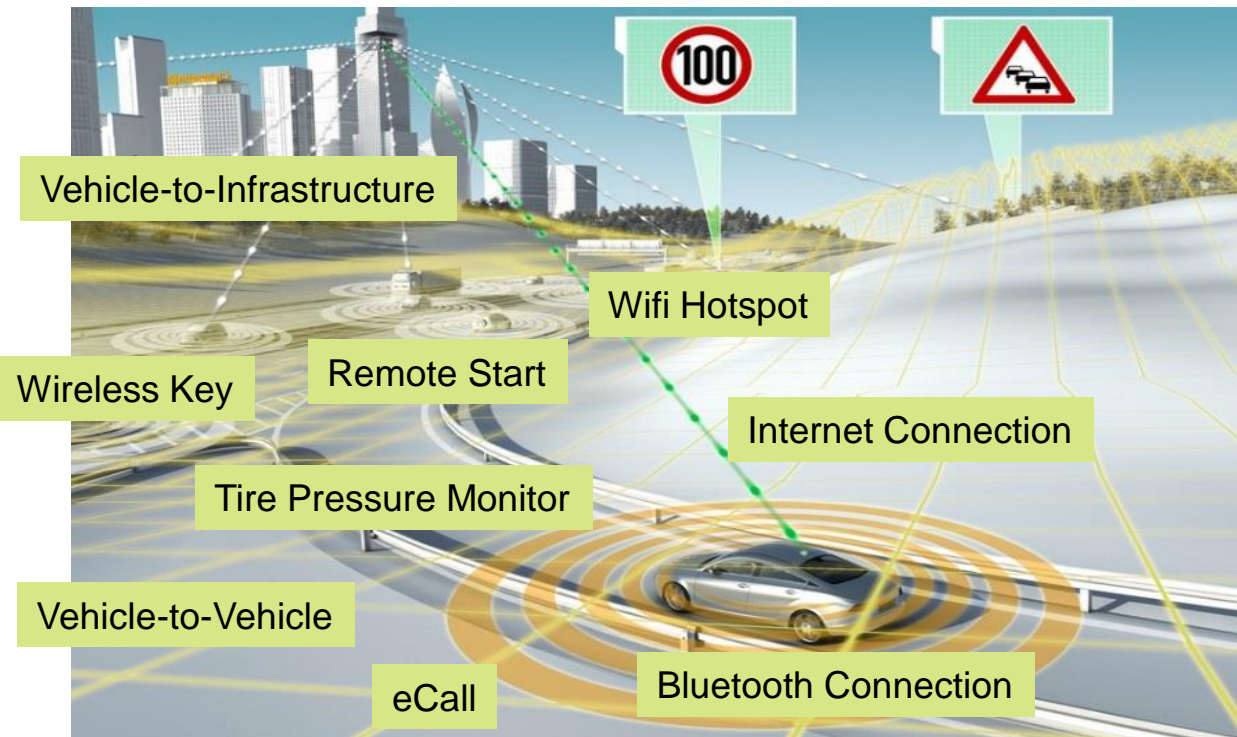
How

1. Use static analysis tool for security
2. Make analysis jobs automated

Why Check Secure Coding Guidelines?

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



<https://youtu.be/MK0SrxBC1xs>

Security in Consumers' Mind



57%

... of customers think
automakers and suppliers are
responsible for protecting data

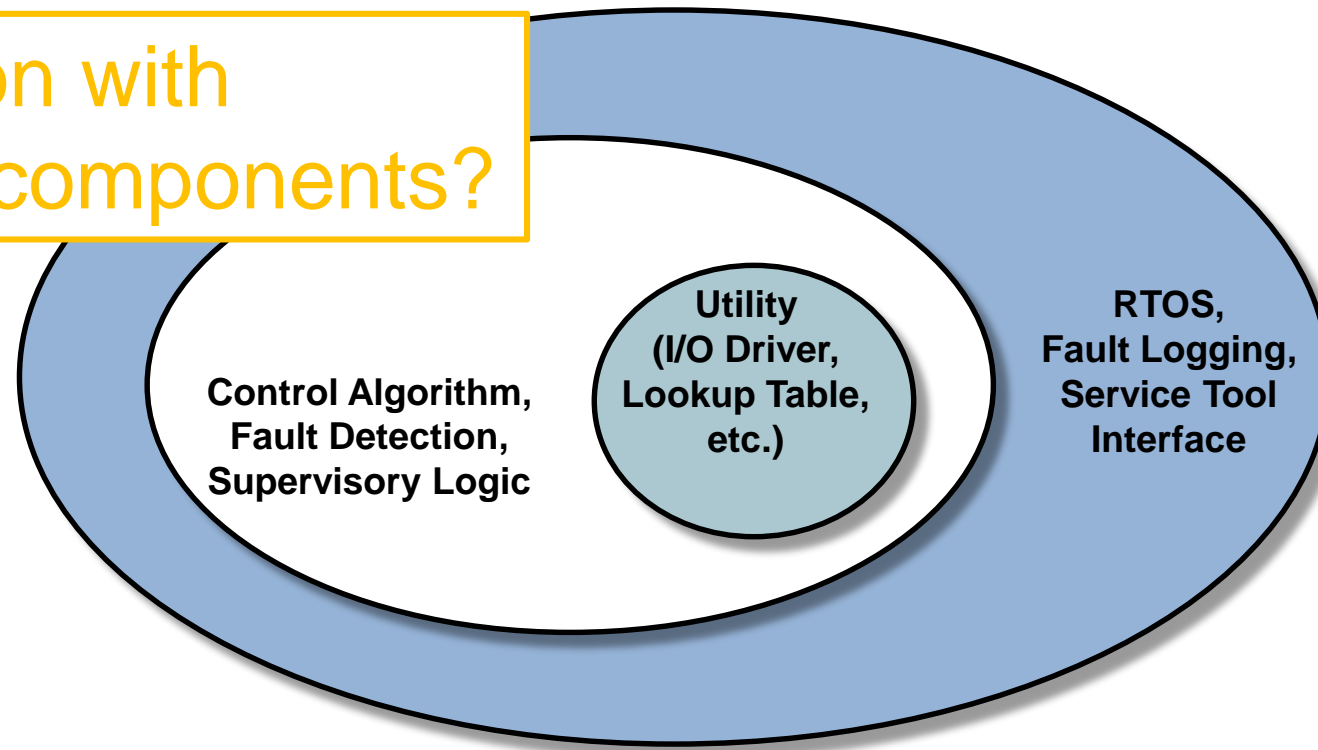


91%

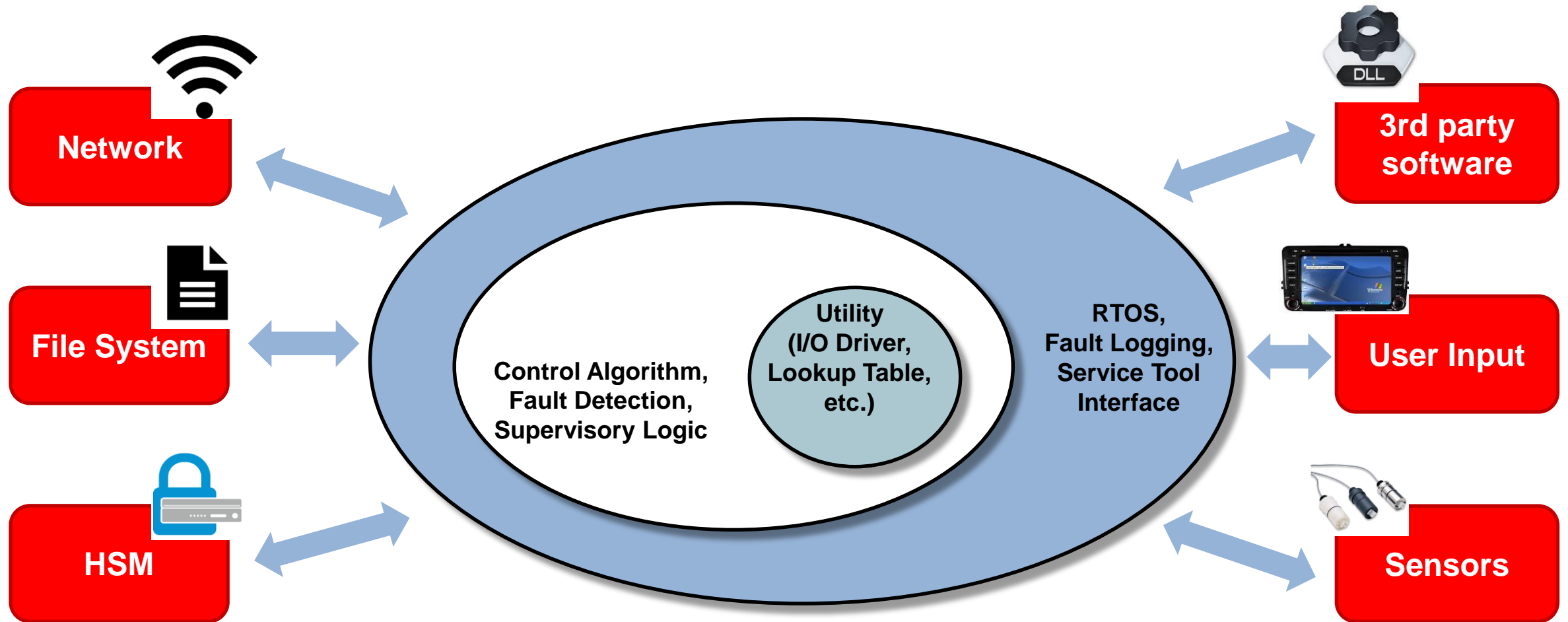
...of customers would never
buy or wary buying from
automakers were hacked

Typical Embedded Software Architecture

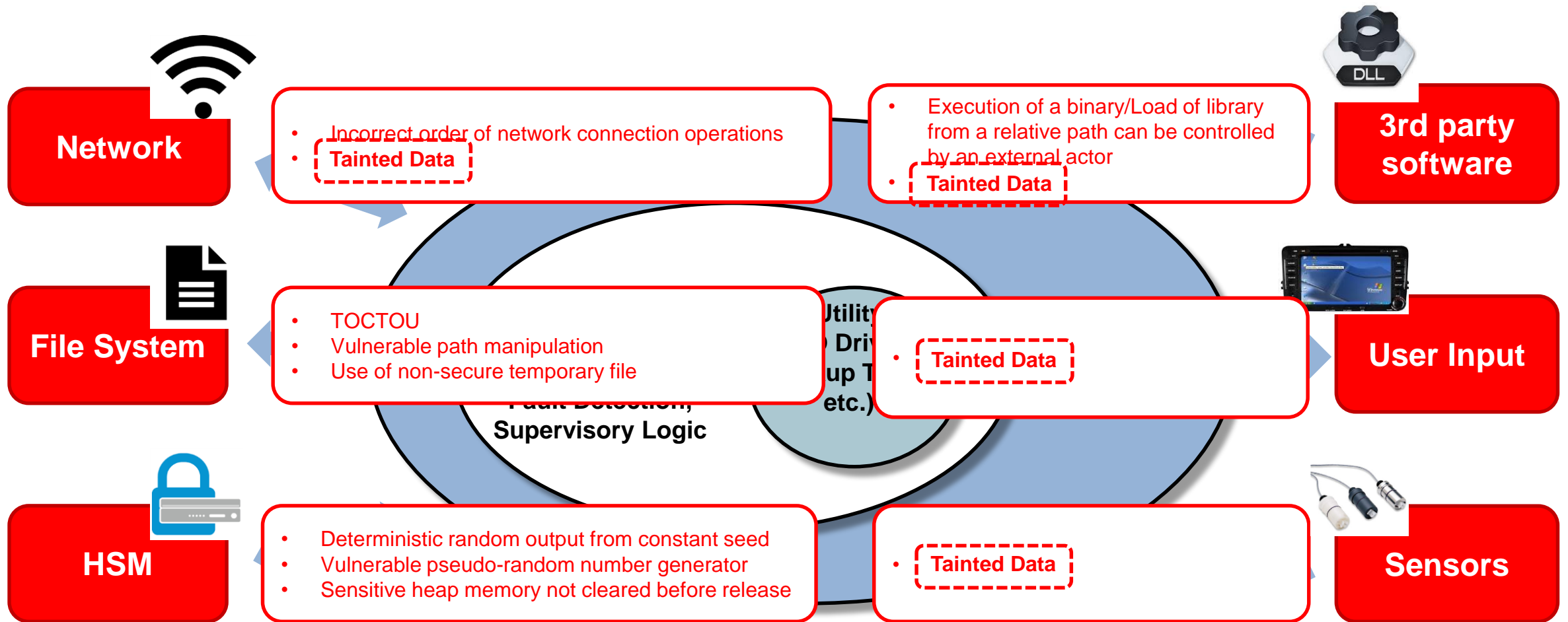
Interaction with
external components?



Embedded Software External Interactions



Embedded Software Security Concerns



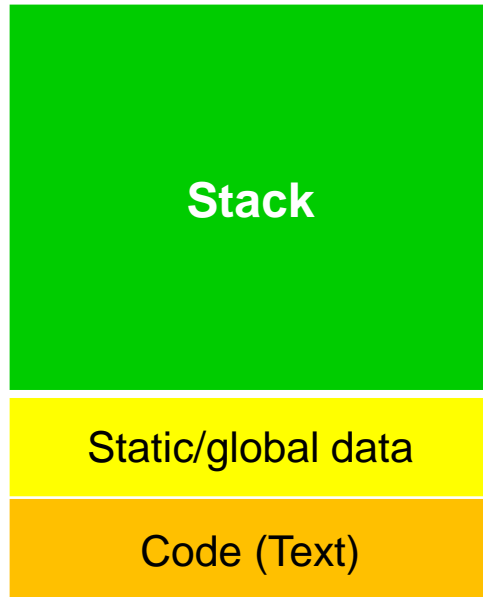
Let's look at an example - Tainted Data

```
1  #include <stdio.h>
2  #define ADCMAXSIZE 256
3
4  typedef signed int sint32;
5  typedef unsigned char uint8;
6
7  extern sint32 getLengthRxData (void);
8  extern sint32 readByte (void);
9
10 void receiveData(void)
11 {
12     sint32 i, length;
13     sint32 ADCdata[ADCMAXSIZE];
14
15     length = getLengthRxData();
16
17     for (i = 0; i < length; i++)
18     {
19         ADCdata[i] = readByte();
20     }
21 }
```

Let's look at an example - Tainted Data

- One of the most exploited

Application memory



```

1  #include <stdio.h>
2  #define ADCMAXSIZE 256
3
4  typedef signed int sint32;
5  typedef unsigned char uint8;
6
7  extern sint32 getLengthRxData (void);
8  extern sint32 readByte (void);
9
10 void receiveData(void)
11 {
12     sint32 i, length;
13     sint32 ADCdata[ADCMAXSIZE];
14
15     length = getLengthRxData();
16
17     for (i = 0; i < length; i++)
18     {
19         ADCdata[i] = readByte();
20     }
21 }

```

buffer overflow!

Stack is corrupted after array overflow.

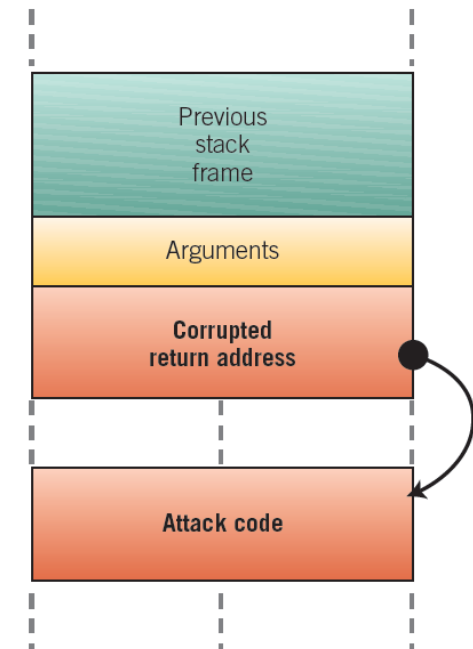


Figure 2

Polyspace helps you find those vulnerabilities



```

Source
StackAttack.c x
1 #include <stdio.h>
2 #define ADCMAXSIZE 256
3 typedef signed int sint32;
4 typedef unsigned char uint8;
5
6 extern sint32 getLengthRxData (void);
7 extern sint32 readByte (void);
8
9 void receiveData(void)
10 {
11     sint32 i, length;
12     sint32 ADCdata[ADCMAXSIZE];
13
14     length = getLengthRxData();
15
16     for (i = 0; i < length;
17         {
18         ADCdata[i] = readByte();
19     }
20 }
    
```

Select one or more results to review:

- Loop bounded with tainted value** (Impact: Medium)
- SEI CERT C MSC21-C (Recommendation)
- SEI CERT C INT04-C (Recommendation)
- ISO/IEC TS 17961 taintformatio

Loop bounded with tainted value (Impact: Medium)

Loop is controlled by a value from an unsecure source.
Loop may be infinite.

	Event	File	Scope	Line
1	Return of external getLengthRxData	StackAttack.c	receiveData()	14
2	Assignment to local variable 'length'	StackAttack.c	receiveData()	14
3	<input type="radio"/> Loop bounded with tainted value	StackAttack.c	receiveData()	16

Select one or more results to review:

- Loop bounded with tainted value** (Impact: Medium)
- SEI CERT C MSC21-C (Recommendation)
- SEI CERT C INT04-C (Recommendation)
- ISO/IEC TS 17961 taintformatio

SEI CERT C INT04-C (Recommendation)

Enforce limits on integer values originating from tainted sources
Loop is controlled by a value from an unsecure source.
Loop may be infinite.

	Event	File	Scope	Line
1	Return of external getLengthRxData	StackAttack.c	receiveData()	14
2	Assignment to local variable 'length'	StackAttack.c	receiveData()	14
3	<input type="checkbox"/> INT04-C Enforce limits on integer	StackAttack.c	receiveData()	16

```

1 #include <stdio.h>
2 #define ADCMAXSIZE 256
3 typedef signed int sint32;
4 typedef unsigned char uint8;
5
6 extern sint32 getLengthRxData (void);
7 extern sint32 readByte (void);
8
9 void receiveData(void)
10 {
11     sint32 i, length;
12     sint32 ADCdata[ADCMAXSIZE];
13
14     length = getLengthRxData();
15
16     for (i = 0; i < length; ++i)
17     {
18         ADCdata[i] = readByte();
19     }
20 }
    
```

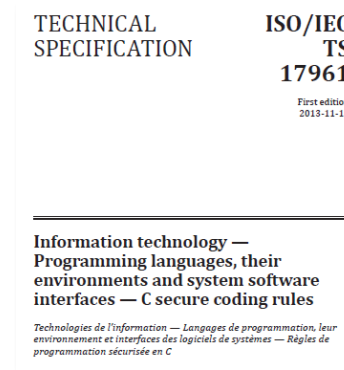
How to Apply Secure Coding Guidelines?

Most Frequently Heard Secure Coding or Security Standards

- **CERT C/C++**



- **ISO/IEC 17961**



- **MISRA-C:2012 Amendment 1, Addendum 2/3**



- **CWE (Common weakness enumeration)**

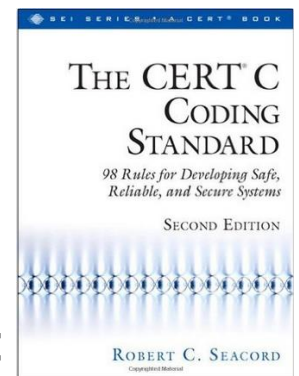


Embedded Safety and Security Coding Standards Overview

Coding Standard	C Standard	Security Standard	Safety Standard	International Standard
MISRA C:2004	C89	No	Yes	No
MISRA C:2012	C90/99	Yes*	Yes	No
CERT C99	C99	Yes	No	No
CERT C11	C11	Yes	Yes	No
ISO/IEC TS 17961	C11	Yes	No	Yes
CWE	None/all	Yes	No	No

* Additional security guidelines for MISRA-C:2012 Amendment 1

Source: Table is based on the book:



How does *Polyspace* help you with embedded software security?

- Detecting security vulnerabilities and underlying defects early
- Provides Exhaustive Documentation and recommendation for security fix
- Proving absence of certain critical vulnerabilities
- Complying with industry standards such as CERT C/C++ and ISO 17961

Polyspace Tools

Bug Finder

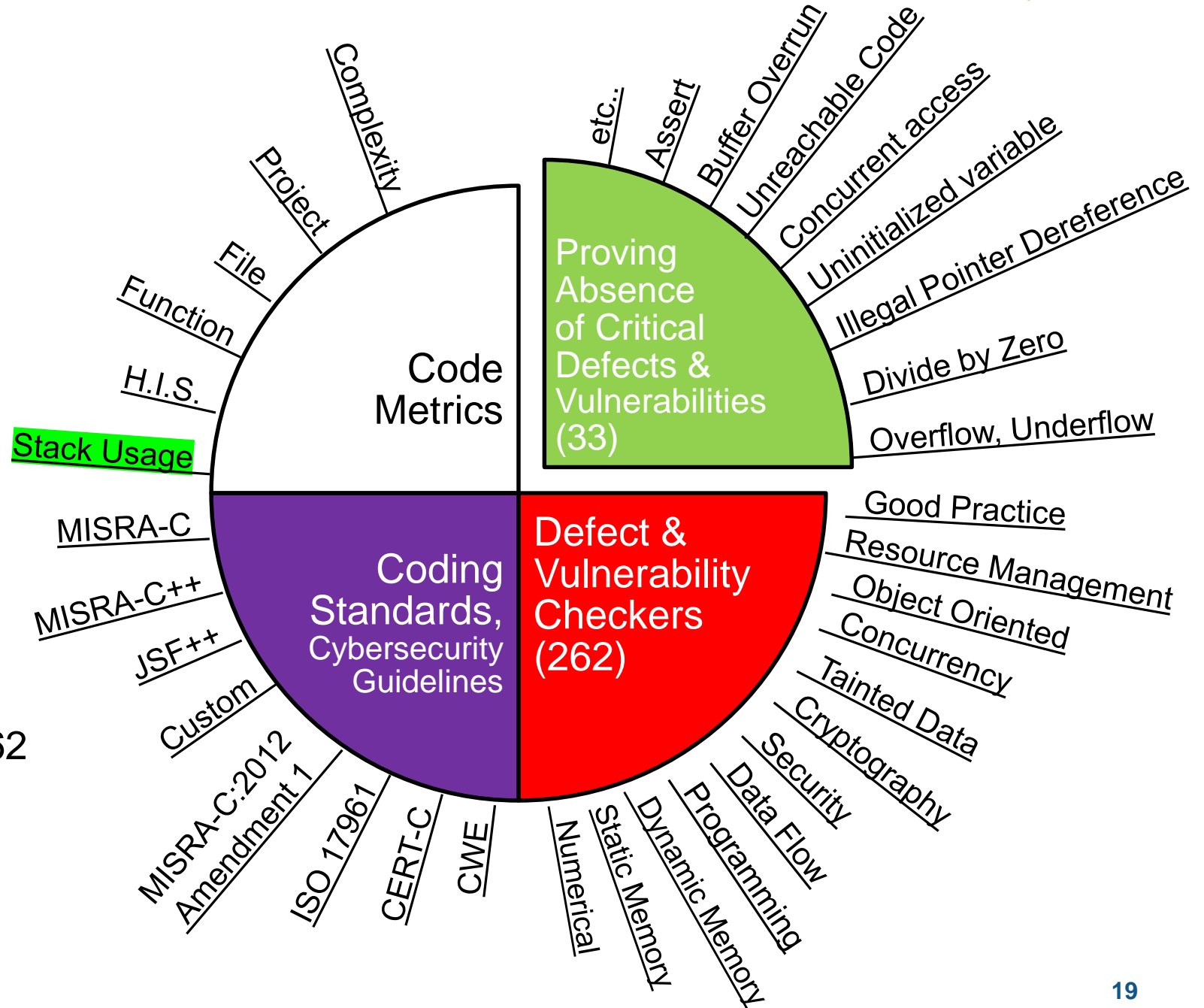


- Produce code metrics
- Check coding standards
- Find defects and vulnerabilities

Code Prover



- Proves code Safe and Secure
- 33 most critical run-time checks
- Supports DO-178 and ISO 26262



Easy to Configure and Review

- Provides explicit options to check Security Guidelines like CERT-C and ISO 17961
- Enable/Disable each rules easily
- Checks Security, Safety guidelines and Defects in One tool

Defect	159
Cryptography	28
Data flow	15
Dynamic memory	9
Numerical	20
Security	68
Tainted data	19
MISRA C:2012	1883
Custom Rule	642
SEI CERT C	1978
Application Programming Interfaces (API)	3
Arrays (ARR)	17
Characters and Strings (STR)	19
Concurrency (CON)	23
Declarations and Initialization (DCL)	384
Environment (ENV)	7
Error Handling (ERR)	11

Loop bounded with tainted value (Impact: Medium)
 SEI CERT C MSC21-C (Recommendation)
 SEI CERT C INTO4-C (Recommendation)
 ISO/IEC TS 17961 taintedformatio

ISO/IEC TS 17961 taintedformatio ?
 Using a tainted value to write to an object using a formatted input function (FLP).
 Loop is controlled by a value from an unsecure source.
 Loop may be infinite.

Event
1 Return of external getLengthRxData
2 Assignment to local variable 'length'
3 <input checked="" type="checkbox"/> taintedformatio Using a tainted value to write to an object using

<input type="checkbox"/> Check MISRA AC AGC	OBL-rules	View
<input type="checkbox"/> Check MISRA C:2012	mandatory-required	View
<input type="checkbox"/> Check MISRA C++:2008	required-rules	View
<input type="checkbox"/> Check JSF AV C++	shall-rules	View
<input checked="" type="checkbox"/> Check SEI CERT-C	all	View
<input checked="" type="checkbox"/> Check SEI CERT-C++	all	View

Select file

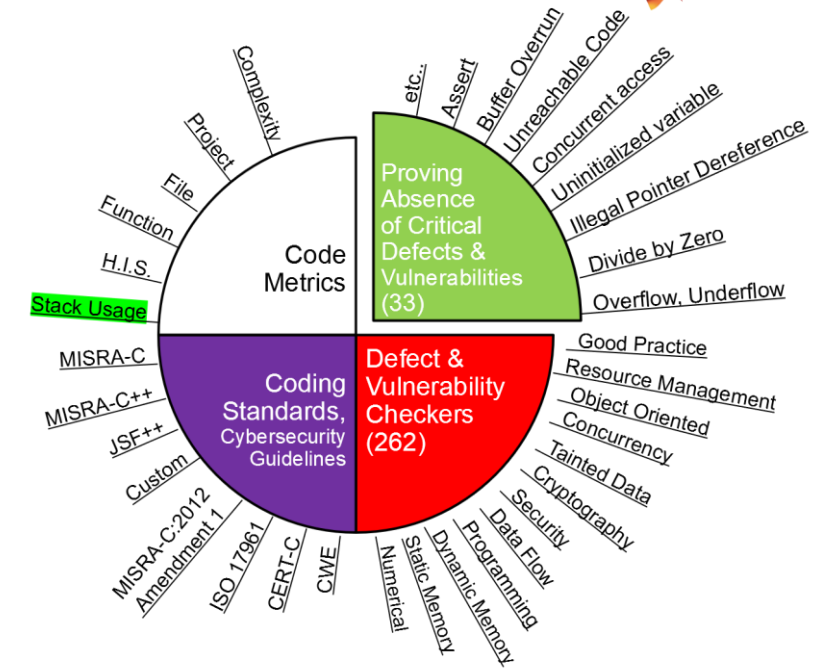
- MISRA C:2004 (131/131)
- MISRA AC AGC (129/129)
- MISRA C:2012 (170/170)
- MISRA C++:2008 (202/202)
- ISO/IEC TS 17961 taintedformatio (157/157)
- SEI CERT C MSC21-C (207/207)
- SEI CERT C INTO4-C (PRE)
- ISO/IEC TS 17961 taintedformatio (EXP)
- ISO/IEC TS 17961 taintedformatio (DCL)
- ISO/IEC TS 17961 taintedformatio (STR)
- ISO/IEC TS 17961 taintedformatio (MEM)
- ISO/IEC TS 17961 taintedformatio (ENV)
- ISO/IEC TS 17961 taintedformatio (ERR)
- Application Programming Interfaces (API)
- Concurrency (CON)
- Miscellaneous (MSC)
- POSIX (POS)
- Microsoft Windows (WIN)

Select rules in category: All

Status	Category	Name
<input checked="" type="checkbox"/>	rule	EXP45-
<input checked="" type="checkbox"/>	rule	EXP46-
<input checked="" type="checkbox"/>	rule	EXP47-
<input checked="" type="checkbox"/>	Integers (INT)	
<input checked="" type="checkbox"/>	Floating Point (FLP)	
<input checked="" type="checkbox"/>	recomme...	FLP00-
<input type="checkbox"/>	recomme...	FLP01-
<input checked="" type="checkbox"/>	recomme...	FLP02-
<input checked="" type="checkbox"/>	recomme...	FLP03-
<input type="checkbox"/>	recomme...	FLP04-
<input type="checkbox"/>	recomme...	FLP05-
<input checked="" type="checkbox"/>	recomme...	FLP06-
<input type="checkbox"/>	recomme...	FLP07-
<input checked="" type="checkbox"/>	rule	FLP30-
<input checked="" type="checkbox"/>	rule	FLP32-
<input checked="" type="checkbox"/>	rule	FLP34-
<input checked="" type="checkbox"/>	rule	FLP36-
<input checked="" type="checkbox"/>	rule	FLP37-
<input checked="" type="checkbox"/>	Arrays (ARR)	
<input checked="" type="checkbox"/>	Characters and Strings (STR)	

Important to close all the windows!

- False negatives - missed vulnerabilities
- All malicious attackers want is one loop hole
- Testing is not exhaustive, almost all static analysis tools are not exhaustive
- Polyspace Code Prover – proving **absence of specific vulnerabilities**
- For critical defects such as buffer overflows, illegal pointer dereferencing ..



Let's look at an example - Tainted Data

```
1  #include <stdio.h>
2  #define ADCMAXSIZE 256
3
4  typedef signed int sint32;
5  typedef unsigned char uint8;
6
7  extern sint32 getLengthRxData (void);
8  extern sint32 readByte (void);
9
10 void receiveData(void)
11 {
12     sint32 i, length;
13     sint32 ADCdata[ADCMAXSIZE];
14
15     length = getLengthRxData();
16
17     for (i = 0; i < length; i++)
18     {
19         ADCdata[i] = readByte();
20     }
21 }
```

Polyspace helps you find those vulnerabilities



Polyspace
Bug Finder

```

Source
StackAttack.c x
1  #include <stdio.h>
2  #define ADCMAXSIZE 256
3  typedef signed int sint32;
4  typedef unsigned char uint8;
5
6  extern sint32 getLengthRxData (void);
7  extern sint32 readByte (void);
8
9  void receiveData(void)
10 {
11     sint32 i, length;
12     sint32 ADCdata[ADCMAXSIZE];
13
14     length = getLengthRxData();
15
16     for (i = 0; i < length; i++)
17     {
18         ADCdata[i] = readByte();
19     }
20 }

```



Polyspace
Code Prover

```

9  void receiveData(void)
10 {
11     sint32 i, length;
12     sint32 ADCdata[ADCMAXSIZE];
13
14     length = getLengthRxData();
15
16     for (i = 0; i < length; i++)
17     {
18         ADCdata[i] = readByte();
19     }
20 }

```

? Out of bounds array index ?

Warning: array index may be outside bounds : [0..255]
array size: 256
array index value: [0 .. 256]

Polyspace helps you find those vulnerabilities



Polyspace Bug Finder

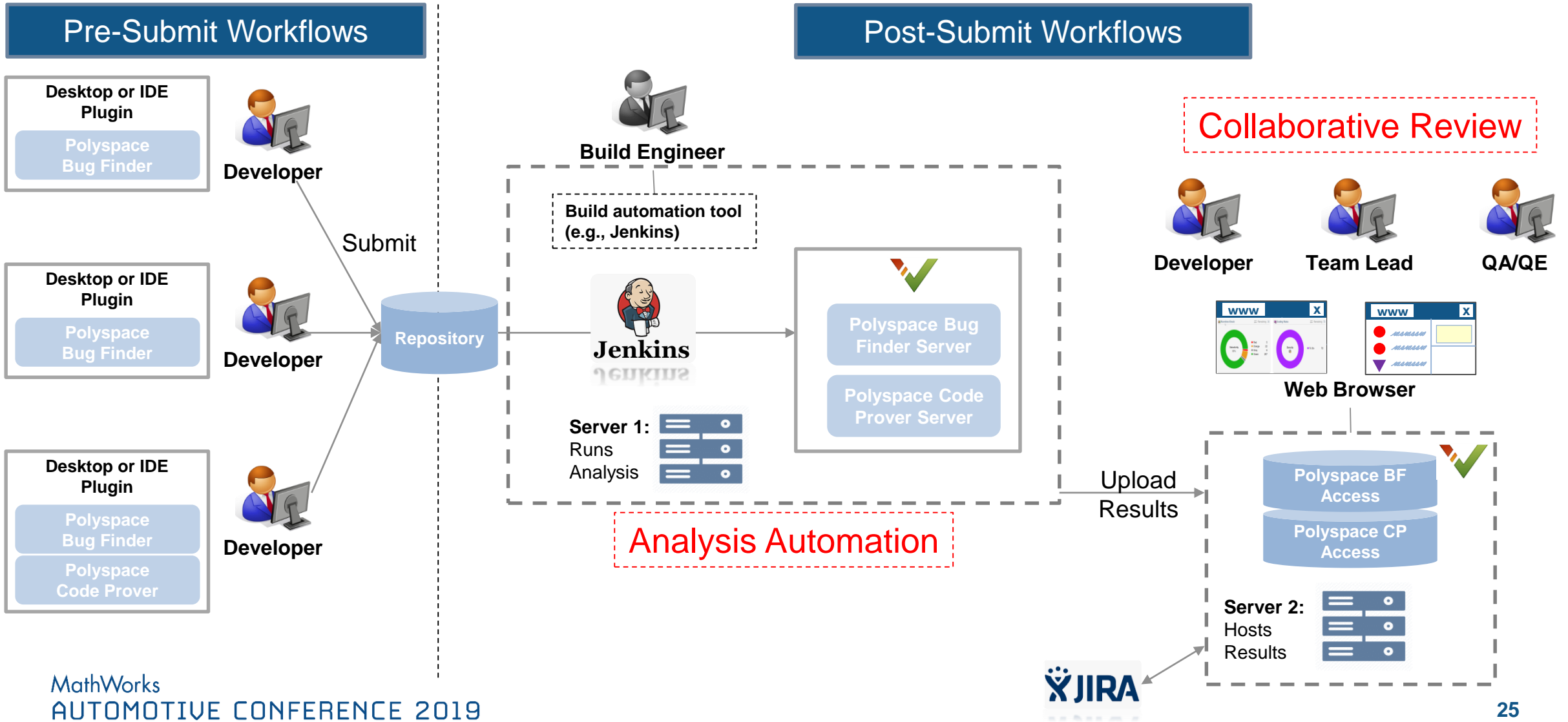


Polyspace Code Prover

The screenshot displays the Polyspace R2019b interface for a project named 'StackCorruption'. The 'Project Browser' on the left shows a tree structure with 'StackCorruption' expanded to show 'Module_1', which contains 'src', 'Configuration', and 'Result' folders. Under 'Result', 'BF_Result [Completed]' and 'CP_Result [Completed]' are visible. The 'Configuration' pane on the right shows settings for 'StackCorruption', including 'Target & Compiler' (C), 'Bug Finder Analysis', and 'Code Prover Verification'. The 'Output Summary' pane at the bottom shows the analysis process completed with a total elapsed time of 00:00:06. The summary table lists several messages:

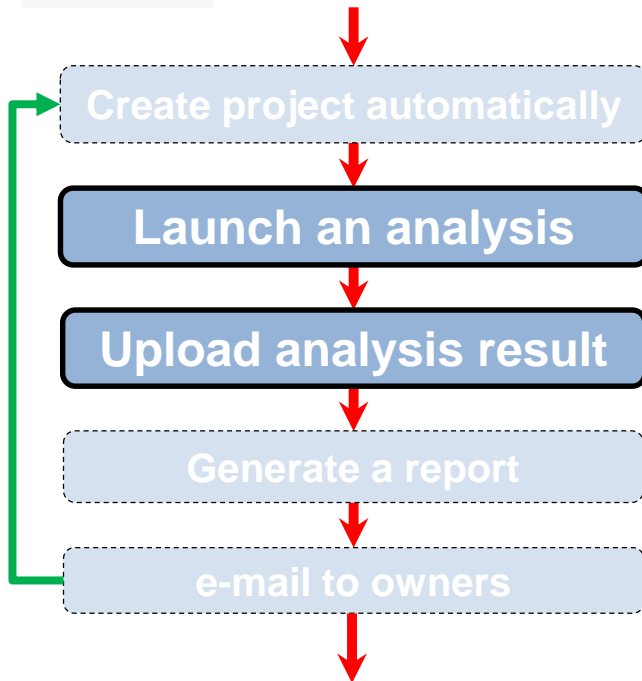
Type	Message	File	Line	Col
Information	C analysis starts at Sat Oct 26 15:55:23 2019			
Information	-float-rounding-mode is ignored in Polyspace Bug ...			
Information	-O2 is ignored in Polyspace Bug Finder.			
Information	-to is ignored in Polyspace Bug Finder.			

Recommended Workflow



Automate Polyspace Analysis by Jenkins plug-in

- Set env variables for Polyspace Access and Web Metrics
- Ease analysis automation and configuration for a standard CI workflow



The screenshot shows the Jenkins dashboard interface. The main content area displays a table of build jobs:

S	W	Name ↓	최근 성공	최근 실패	최근 소요 시간
		putty-src	2 hr 12 min -#3	—	2 hr 4 min

Below the table, there are sections for '빌드 대기 목록' (Build Queue) and '빌드 실행 상태' (Build Execution Status). The '빌드 대기 목록' section shows '빌드 대기 항목이 없습니다.' (No build items in queue). The '빌드 실행 상태' section shows '1 대기 중' (1 in progress) and '2 대기 중' (2 in progress).

Polyspace Access - Review Results in Collaborative Environment

DASHBOARD
? garyryu

Project Overview
Defects
Code Metrics
Custom Rules
MISRA C:2012

Layout
Open in Desktop
Review

PROJECT EXPLORER

- ProjectsWaitingForDeletion
- public
 - Bug_Finder_Example (Bug Finder)
 - Bug_Finder_Example (Code Prover)

Project Overview
Defects
Code Metrics
MISRA C:2012

Summary

Open Issues		Mandatory	Required	Advisory
Done	0%	0%	100%	
Open	1182	19	1163	0

0% Done 0/1182 violations

75 rules violated (130 rules enabled)

Trends

Open violations over time

Upload Date

Details

View by Category | View by File

Name	Total	To Do	In Progress	Done
1 A standard C environment	2	2	-	-
2 Unused code	1	1	-	-

PROJECT DETAILS

Project

Name Bug_Finder_Example (Bug Finder)

Language C

Tools Bug Finder

Coding Standards Custom Rules, MISRA C:2012

Number of Runs 3

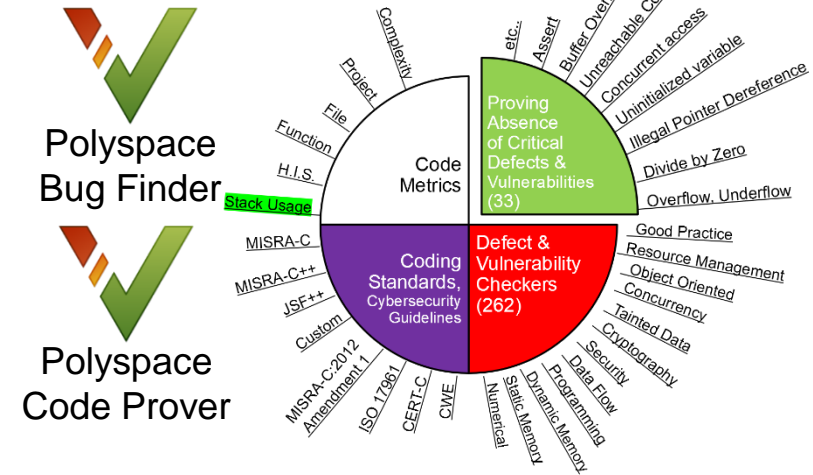
Current Run (ID 18)

Upload Date 9/6/19, 6:14 PM

Job 1.0

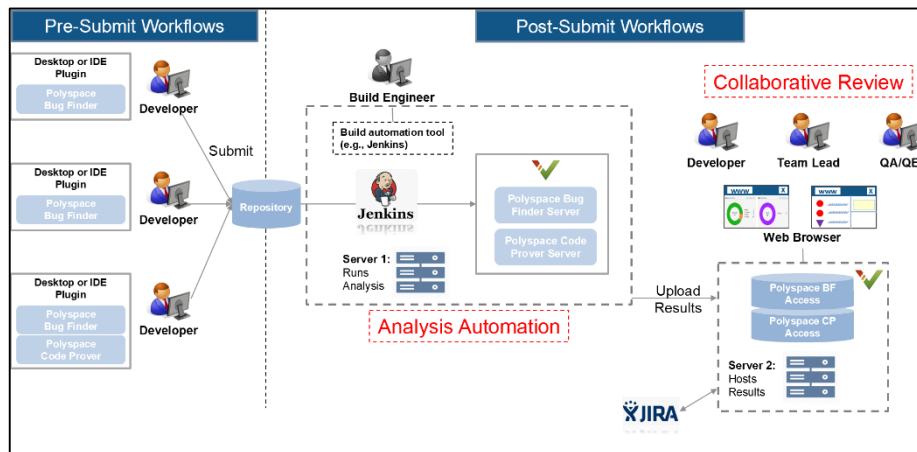
SUPPORT REPORT

Key takeaways



1. Achieve the Goals for Security and Safety by One Tool, Polyspace

2. Improve your workflow by Analysis Automation & Collaborative Review Environment





Thank you!