

# MATLAB EXPO

## 코딩 초기부터 통합 과정까지 Polyspace를 활용하는 방안

유용출 부장, 매스웍스코리아  
한지동 부장, 매스웍스코리아



# Do Polyspace Bug Finder & Code Prover!

## Bug Finder



→ High Quality, Secure, Compliant Code:

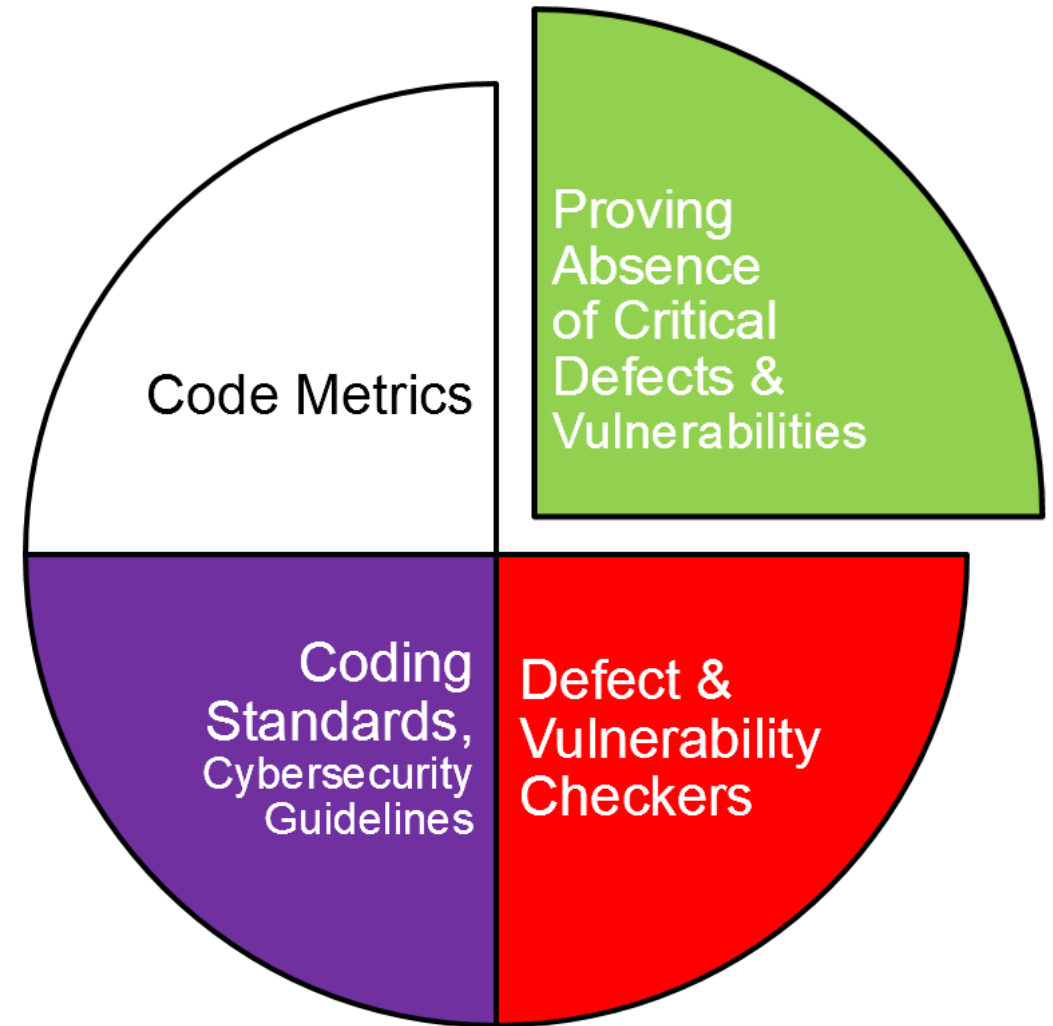
- Various defects or vulnerabilities
- Credits for functional safety & cybersecurity standards

## Code Prover



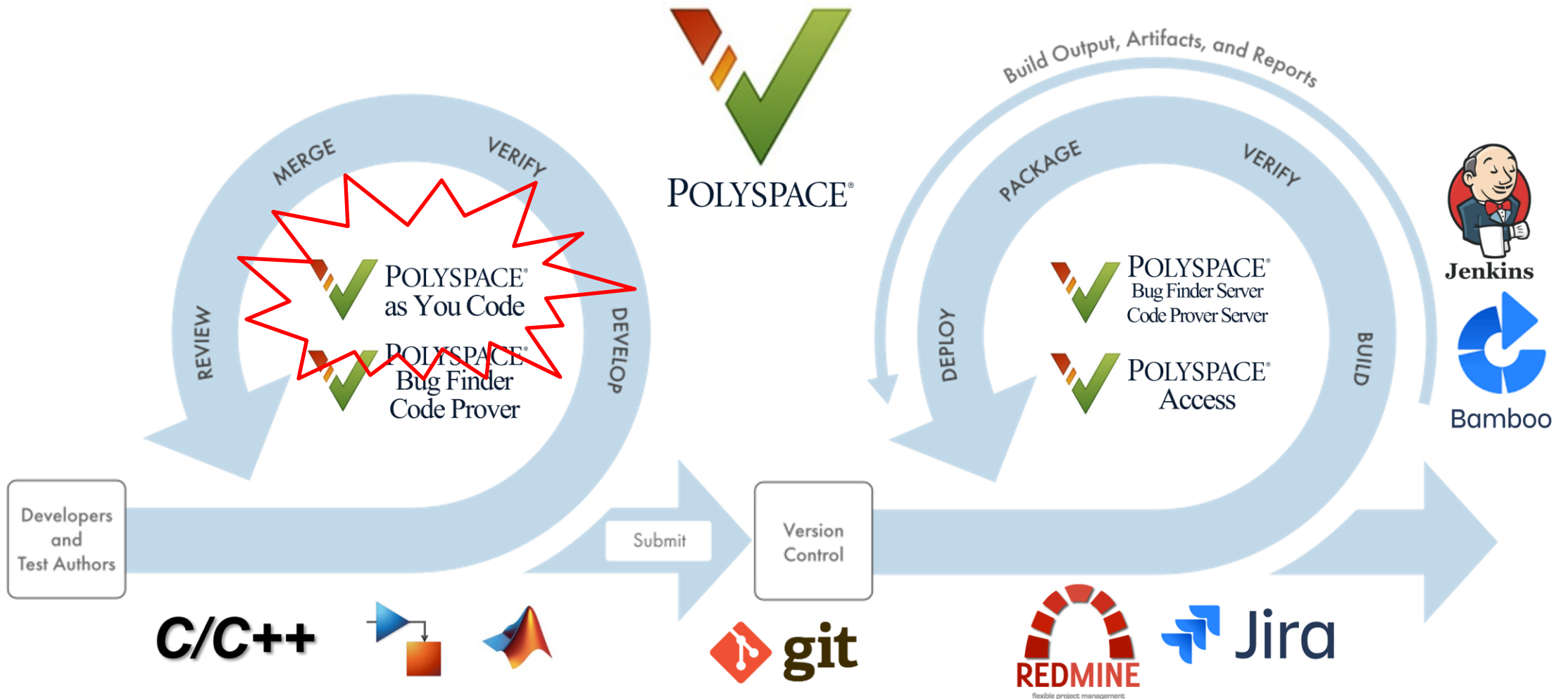
→ Fully Trusted Components:

- Proven free of critical runtime defects & vulnerabilities
- Additional credits for standards



# Do Polyspace as You Code!

*Review Guidelines Violations in your Visual Studio, Visual Studio Code, and Eclipse*



# Do Polyspace as You Code!

Review Guidelines Violations in your Visual Studio, Visual Studio Code, and Eclipse

```
4
5 #define BUFF_SIZE 128
6
7
8 int secure_print(char *str) {
9     char dst[BUFF_SIZE];
10    int r = 0;
11
12    if (sprintf(dst, "%s", str) == 1) {
13        r += 1;
14        dst[BUFF_SIZE-1] = '\0';
15    }
16 }
```

**Risk**

These functions can cause buffer overflow, which attackers can use to infiltrate your program.

**Fix**

The fix depends on the root cause of the defect. Often the result details show a sequence of events that led to the defect. You can implement the fix on any event in the sequence. If the result details do not show the event history, you can trace back using

**Result Details**

Variable trace

Result Review

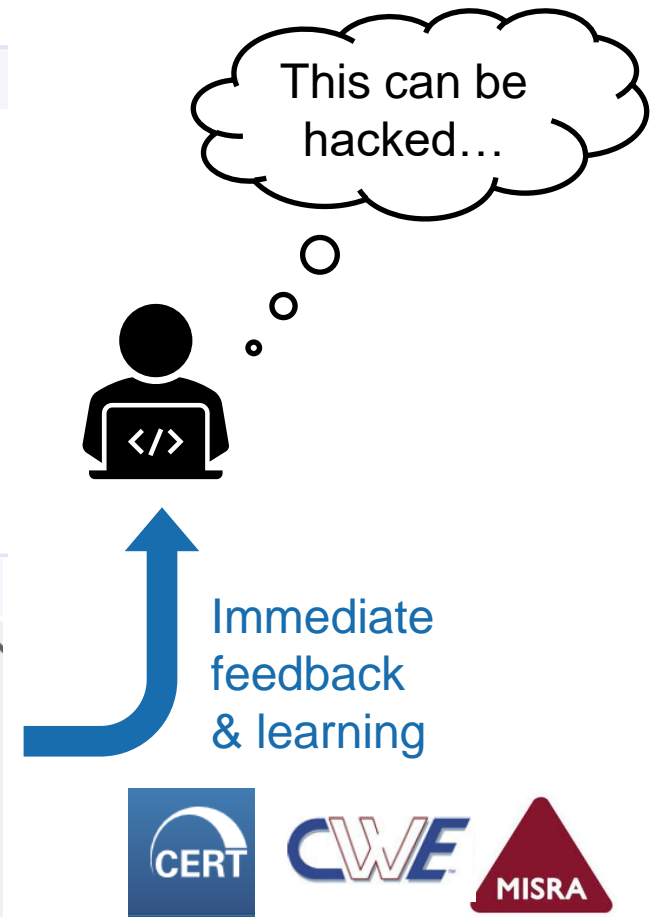
Status: Unreviewed Enter comment here...

Severity: Unset

**Use of dangerous standard function** (Impact: Low) ?

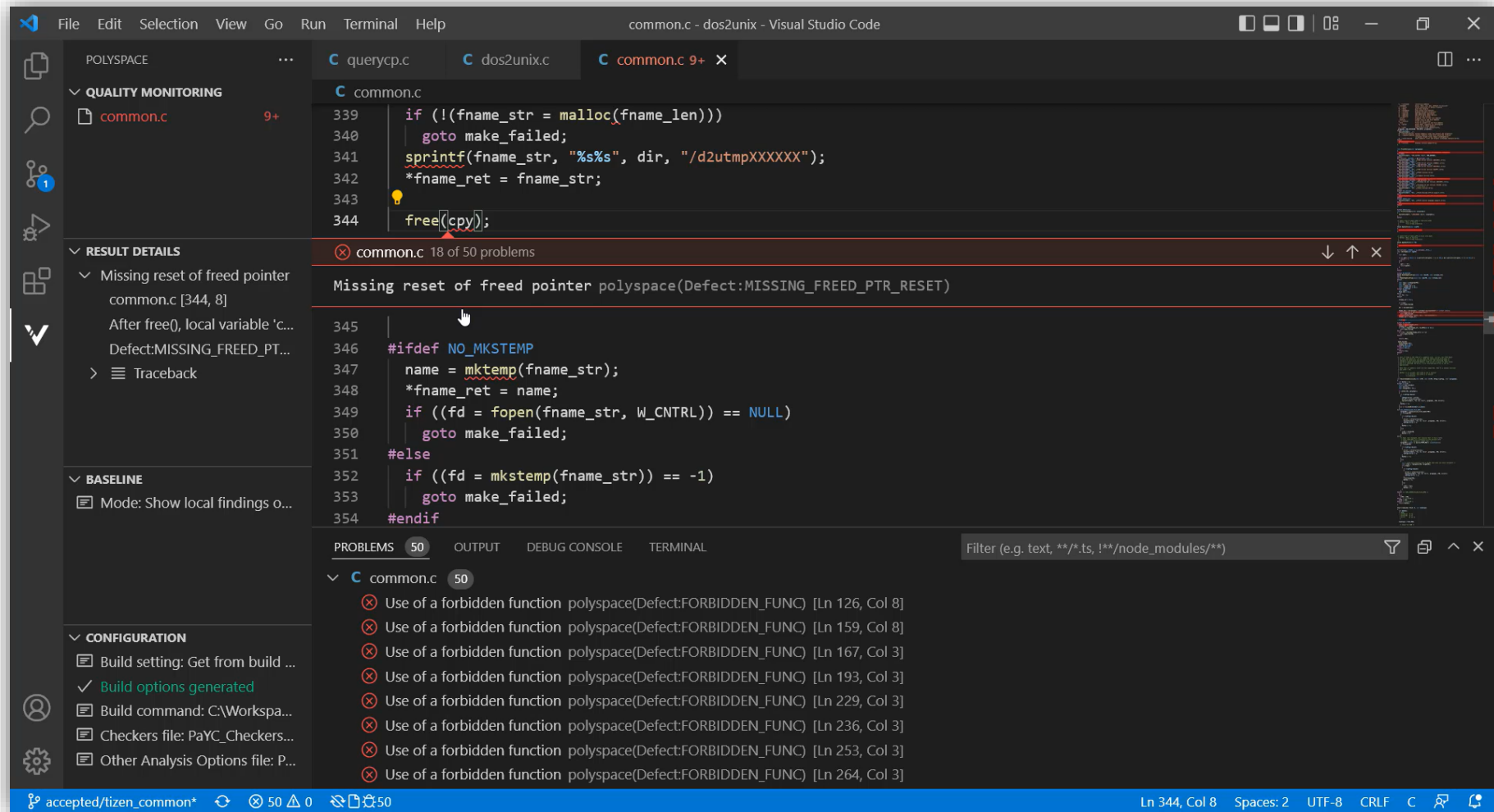
Using 'sprintf' can cause the destination buffer to overflow. The output length depends on unknown values that 'sprintf' cannot control.

Event	File	Scope	Line
1 Take the address of variable 'dst'	sectest.c	secure_print()	12
2 Use of dangerous standard function	sectest.c	File Scope	12

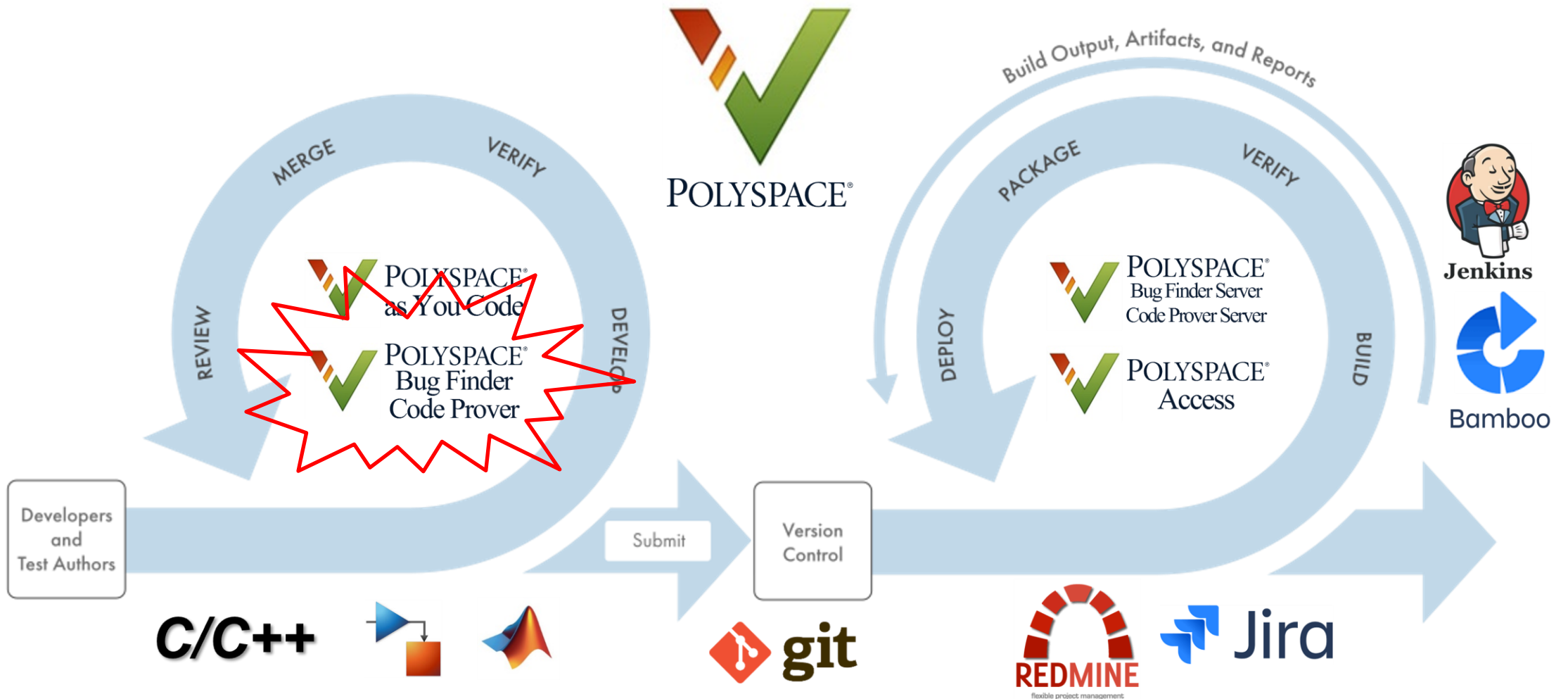


# Do Polyspace as You Code!

*Fast Analyze/Review Single Source File to Comply with HMC Secure Coding Guidelines*



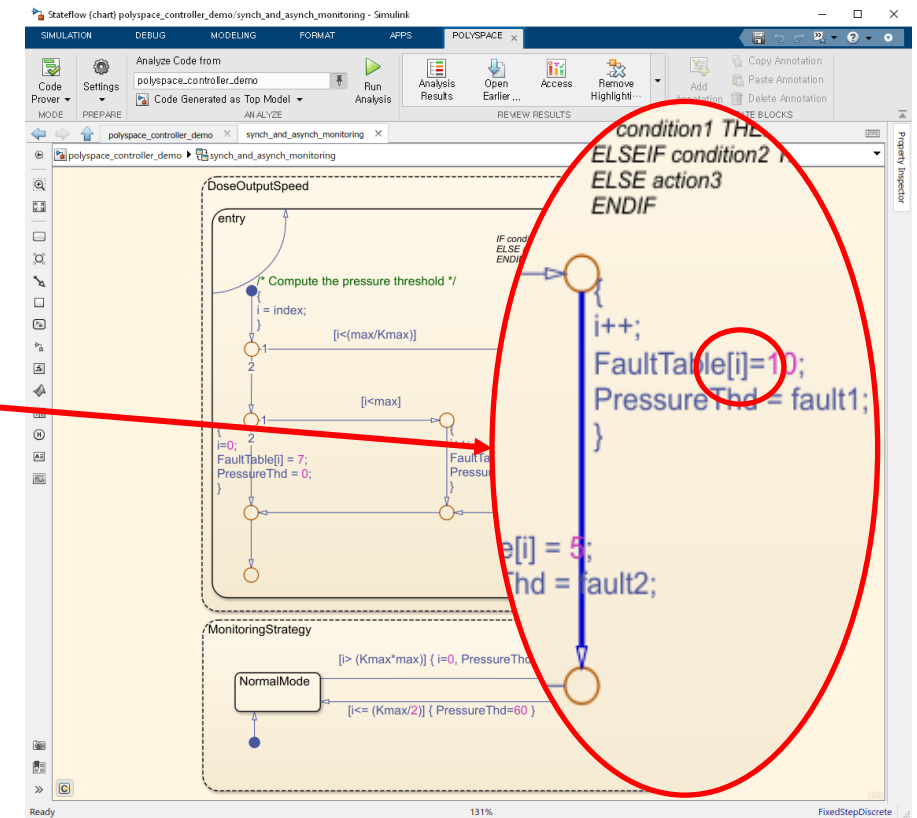
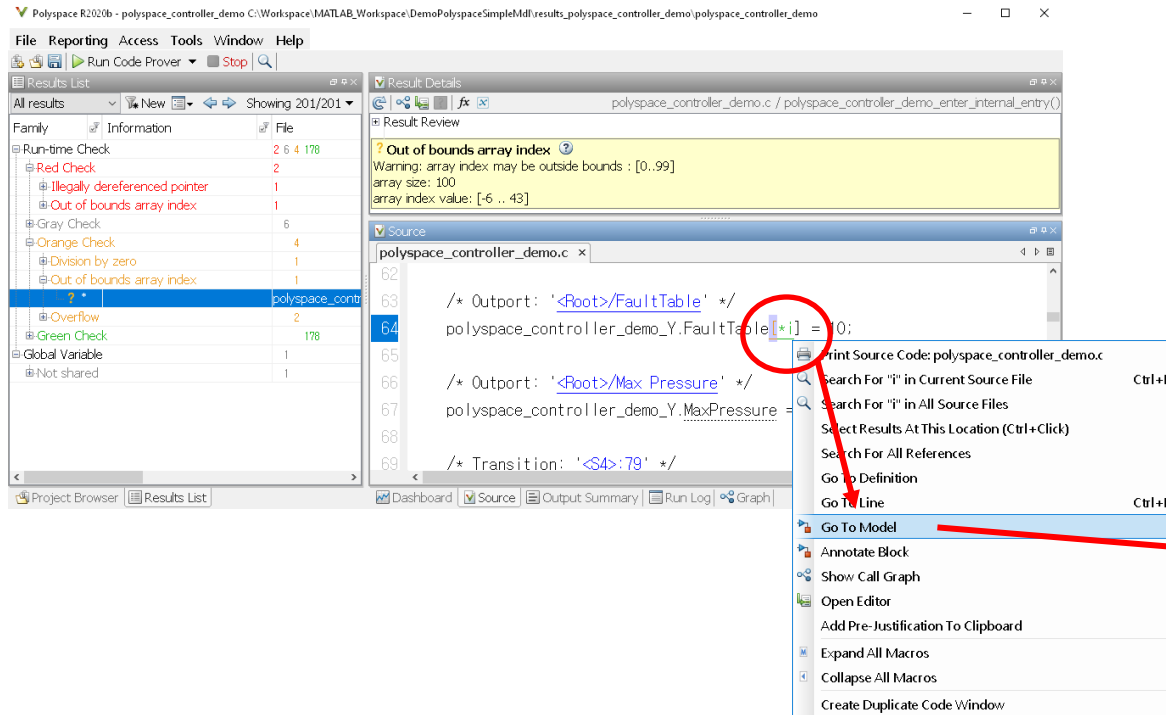
# Do Polyspace Bug Finder & Code Prover!





# Do Polyspace Bug Finder from Simulink!

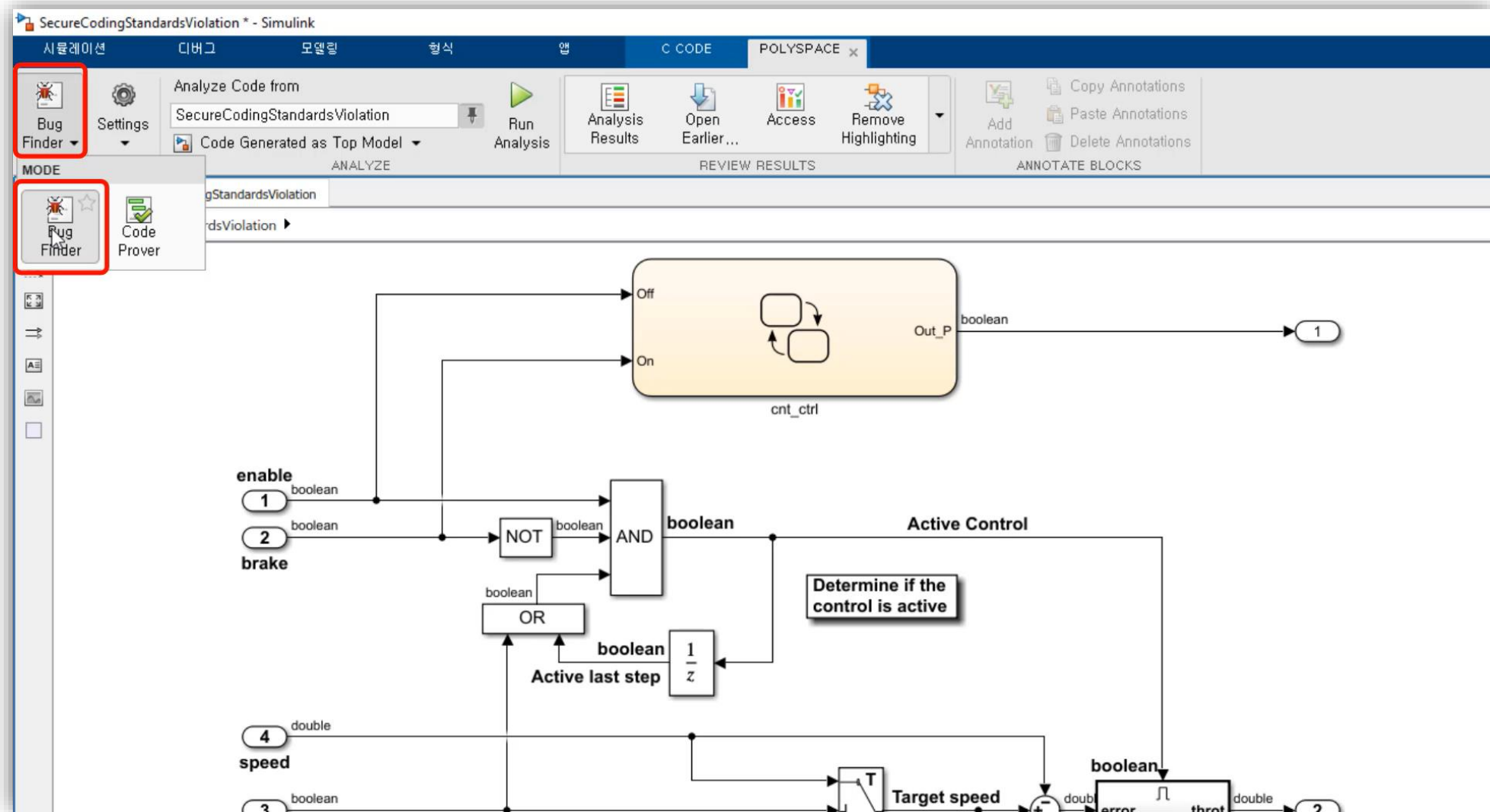
## Trace back from Polyspace to Simulink Model



- Seamless workflow for generated code review
  - Trace back from code to model at once
  - Easy configuration for static analysis with less effort
  - Simple annotation for Simulink blocks to justify issues

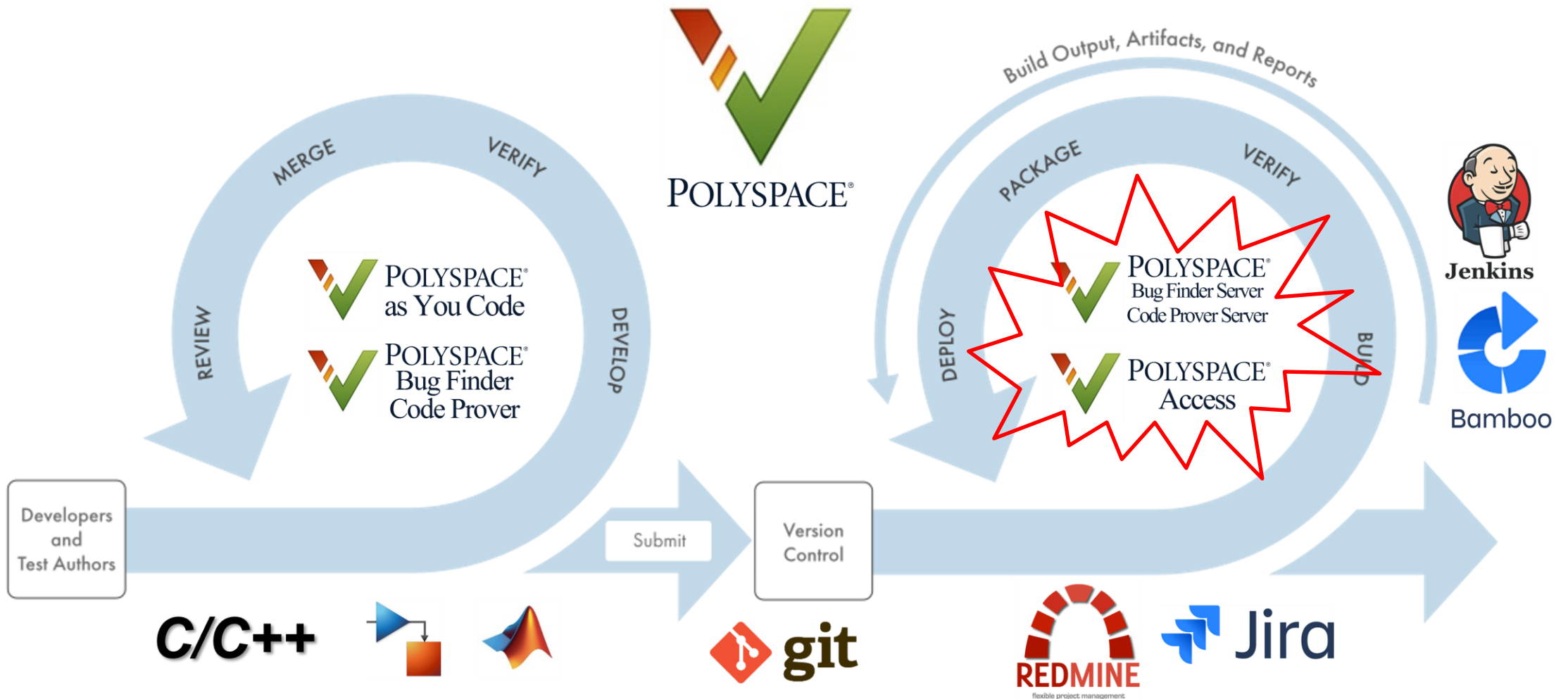
# Do Polyspace Bug Finder from Simulink!

*Use the Existing Configurations for HMC Secure Coding Guidelines*



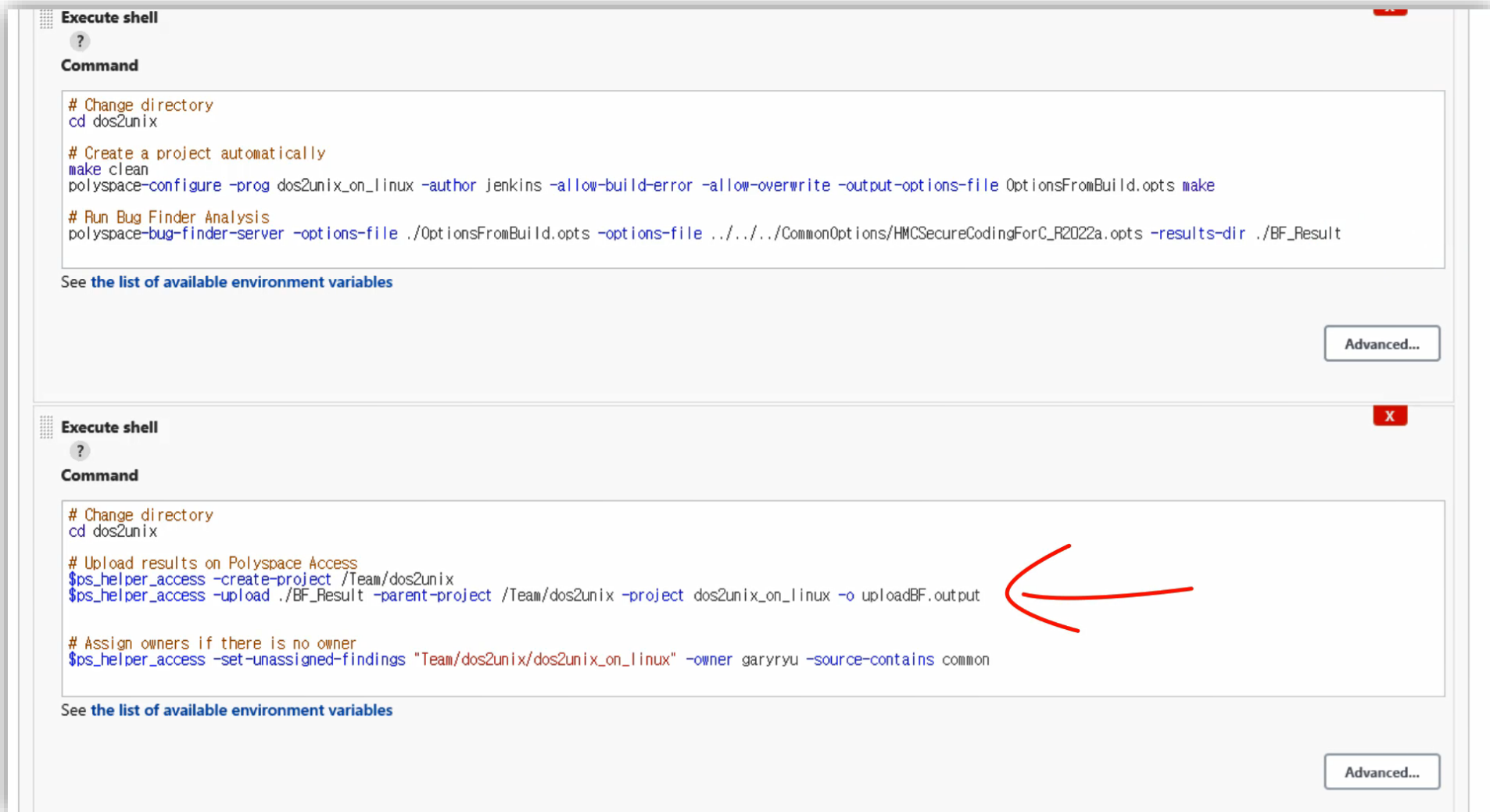


# Do Polyspace Server & Access in Continuous Integration!



# Do Polyspace Server & Access in Continuous Integration!

*Submitting source code, Automatic Analysis and Collaborative Review*



**Execute shell** ?

**Command**

```
# Change directory
cd dos2unix

# Create a project automatically
make clean
polyspace-configure -prog dos2unix_on_linux -author jenkins -allow-build-error -allow-overwrite -output-options-file OptionsFromBuild.opts make

# Run Bug Finder Analysis
polyspace-bug-finder-server -options-file ./OptionsFromBuild.opts -options-file ../../../../CommonOptions/HMCSecureCodingForC_R2022a.opts -results-dir ./BF_Result
```

See [the list of available environment variables](#)

Advanced...

---

**Execute shell** ?

**Command**

```
# Change directory
cd dos2unix

# Upload results on Polyspace Access
$ps_helper_access -create-project /Team/dos2unix
$ps_helper_access -upload ./BF_Result -parent-project /Team/dos2unix -project dos2unix_on_linux -o uploadBF.output

# Assign owners if there is no owner
$ps_helper_access -set-unassigned-findings "Team/dos2unix/dos2unix_on_linux" -owner garyryu -source-contains common
```

See [the list of available environment variables](#)

Advanced...

# Do Polyspace Server & Access in Continuous Integration!

## Submitting source code, Automatic Analysis and Collaborative Review

