

◊ Ivannikov ISP RAS Open Conference 2019

Deploying SVACE to Samsung

Youil Kim (Samsung Research)



Shape the Future with Innovation and Intelligence

Background

❖ SVACE is a static analysis tool for early detection of SW bugs.

- The name stands for “Security Vulnerabilities And Critical Errors.”

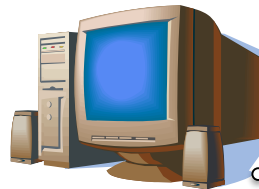
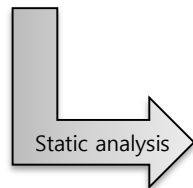
❖ Static analysis estimates a program’s runtime behavior without actual execution by analyzing the source code.

- It does not require runtime environments or test inputs.
- It is easily applicable to large-scale SW.
- We may consider it as an automated code review.

```
1: void f(int *ptr) {  
2:   if ( !ptr ) {  
3:     *ptr = 0;  
4:   }  
5: }
```



ptr can be NULL
@ line #3
→ Defect



ptr can be NULL
@ line #3
→ Defect

A Short Collaboration History

- ❖ **In 2009, we started a joint project on SVACE of ISP RAS.**
 - To introduce SVACE as an auxiliary tool in addition to commercial tools
- ❖ **In 2012, we decided to develop SVACE for the purpose of deploying it as the main static analyzer in our SW development process.**
 - We put a lot of time and effort in improving SVACE from 2013 to 2015.
 - ISP RAS colleagues also had very tough time.
- ❖ **At the end of 2015, we successfully deployed SVACE!**
 - We improved SVACE every year and released SVACE 3.0 in this year.
- ❖ **Our collaboration is to be continued in 2020.**

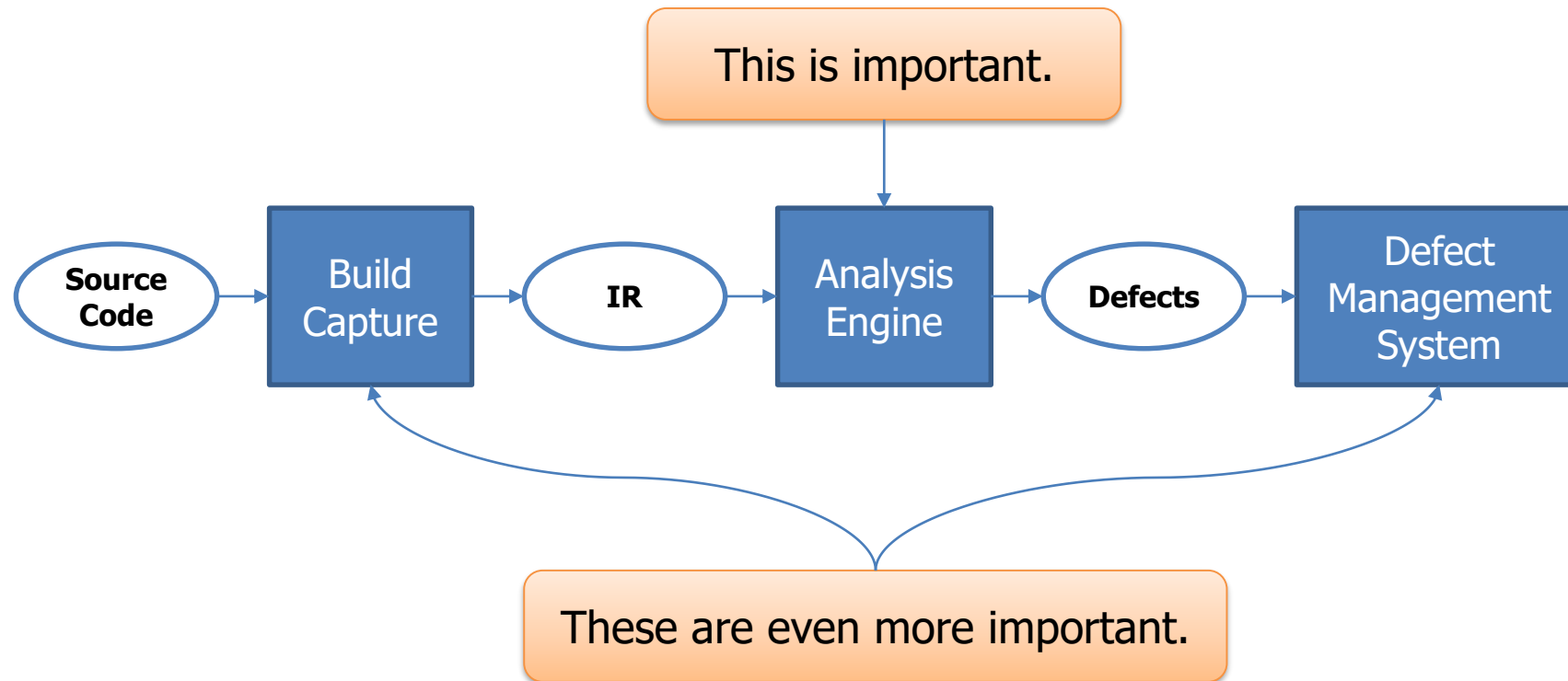
Our Current Position

- ❖ **SVACE has been deployed to most business divisions in the company.**
 - Samsung Research
 - Digital Appliances Business
 - Health & Medical Equipment Business
 - Mobile Communications Business
 - Network Business
 - Visual Display Business
- ❖ **Most of C, C++, C#, and Java source code is regularly checked with SVACE.**
 - Including Tizen TVs, Samsung mobile devices, FamilyHub, etc.
 - More than 10,000 users (developers)
 - More than 300 billion lines code has been checked (since 2015.)

From Research to the Field

There were several challenges.

- We knew that it would be never easy, but we underestimated the effort.



Challenges #1 – Build Capture

⬠ **What happened to us when we tried to analyze C/C++ code in the company**

- Build capturing is essential.
- Clang was not enough to support 90 different C/C++ compilers in the company.
- We had to support various systems including an old legacy OS.
- Conventional build capture implementation may not work with Git Build System (GBS) or Open Build Service (OBS).

⬠ **Some achievements**

- SVACE provides quite stable build capture feature.
- We extended Clang so that it can handle most of the company code, with a small workaround to ignore unknown constructs.
- We made a special RPM package to install SVACE into GBS/OBS build roots and enable SVACE's build capture.

Challenges #2 – Analysis Engines (1/2)

◈ IR level analysis was not enough.

- At the beginning, SVACE was a static analyzer for LLVM bitcode.
- Some coding errors could not be detected after we translate the code into LLVM bitcode.

◈ Path-sensitive analysis was not enough.

- Path-sensitive analysis was essential but was not enough to satisfy our quality goals.
- There is no one silver bullet to address many different false positive patterns.
- Some warnings are technically true positives, but developers don't want to see them.

◈ Some achievements

- SVACE has lightweight checkers on AST. (e.g., Clang Static Analyzer for C/C++ language)
- SVACE performs path-sensitive analysis using an SMT solver.
- In this year, we removed 47% of useless alarms that developers marked as 'Won't Fix' and 'False Alarm' in 52 C/C++ checkers.

Challenges #2 – Analysis Engines (2/2)

⬠ **Analysis results should be deterministic and stable.**

- We should not rely on hardware timer, random values, or any non-deterministic components.
- A build process is often not deterministic, due to timestamps, temporary files, and so on.
- Even if some part of a project has been changed, developers expect to get the same set of defects for the rest (unchanged part) of the project.

⬠ **The performance is always not enough.**

- Developers expect to get daily analysis results even for the largest codebase.
- Our codebase becomes larger and larger.
- Developers even expect to get analysis results within several minutes after code review integration.

⬠ **Some achievements**

- It takes 11 hours for SVACE to build and analyze Tizen 5.0 platform (17M LOC).
- SVACE produces the same results for each trial, when we build and analyze the Tizen platform 5 times.

Challenges #3 – Defect Management

It is crucial to have a good defect management system.

- Some of key features are supports for code navigation and grouping similar warnings.
- It requires considerable efforts to develop such system as it should be able to handle a lot of data.

Some achievements

- SVACE provides some necessary information (e.g., warning grouping factors, relations among code tokens) for defect management systems.
- We developed our own defect management system and improved it for several years.

The screenshot displays the SonarQube Defect Manager interface. The main window shows a table of warnings with columns for WQID, Classification, Checker, Severity, Status, and File Path. The table lists several warnings, including one with WQID 90867, which is highlighted. Below the table, a code editor shows the source code for the file path .task/sonarjs-test/Ghost/core/server/services/urldata.js. The code includes logic for handling image paths and URLs. On the right side, a 'Warning Information' panel provides details for the selected warning (WQID: 90867), including its classification (Undecided), checker (SONARJS.S2259), and a warning message: 'TypeError can be thrown as "data" might be null or undefined here.' The panel also includes fields for 'Warning Message', 'Trace Message', 'WQID URL', and 'Grouping Factors', along with buttons for 'Apply', 'Apply & Next', and 'Review History'.

One More Challenge – Competitiveness

◈ **We should keep our competitiveness against commercial tools.**

- By meeting company-specific requirements better
- By tight integration into our SW development infrastructure

◈ **Some achievements**

- In 2015, we provided SVACE for an old legacy RHEL 3 system, which a commercial tool stopped to support.
- In 2016, SVACE enabled the analysis of a new Android platform using Jack toolchain, 6 months earlier than a commercial tool.
- SVACE also has a number of checkers developed by the requests from our business divisions.

Conclusion – Beyond A Standalone Tool

- ❖ **We successfully deployed SVACE to the company through active collaboration with ISP RAS colleagues.**
- ❖ **We integrated SVACE into our SW development infrastructure.**
 - AnalysisHub provides SVACE as a service.
 - Code Review Bot provides SVACE results at code-review time.

Integration #2 – Code Review Integration

On top of AnalysisHub, we developed Code Review Bot.

- It generates automated code review comments including SVACE analysis results.

Tizen Gerrit

All My Projects People Documentation
Open Merged Abandoned

Change 211187 - **Blocked on Code-Review Label**
Change context manifest's type to c++ class

Related :
[tpk-backend] <https://review.tizen.org/gerrit/#/c/platform/core/applw/tpk-backend/+212844/>
[wgt-backend] <https://review.tizen.org/gerrit/#/c/platform/core/applw/wgt-backend/+212842/>

Change-Id: I9e79f17cb85fcbf4d798dc1db894f4cf3e9dcb6d
Signed-off-by: SA_Auto

Author: SA_Auto Jul 29, 2019 8:51 AM
Committer: SA_Auto Nov 25, 2019 5:58 PM
Commit: c684b39ea61219aee257a75156f8ff6e1f5c2b85 (gitweb)
Parent(s): 3a9fdce551eca71da4a5cc3d3022aa117e4a4a7e (gitweb)
Change-Id: I9e79f17cb85fcbf4d798dc1db894f4cf3e9dcb6d

History Expand All Hide tagged comments

SA_Auto
Patch Set 10: Code-Review-2
(3 comments)
SVACE Result

src/common/plugins/plugin_manager.cc
Line 107: [SVACE][WGID: 49051][Major] This statement in the source code might be unreachable during program execution.
src/common/plugins/types/metadata_plugin.cc
Line 121: [SVACE][WGID: 49049][Critical] Dynamic memory referenced by 'md->key' was allocated at metadata_plugin.cc:121 by calling function 'strdup' and lost at metadata_plugin.cc:177.
Line 128: [SVACE][WGID: 49050][Critical] Dynamic memory referenced by 'md->value' was allocated at metadata_plugin.cc:128 by calling function 'strdup' and lost at metadata_plugin.cc:177.

review-bot reviewed on 19 Jul [View changes](#)

review-bot left a comment Member + 😊 ...

[Potential Defect] (View it on [Defect Manager](#))
Analyze Information: SVACE_2.6 / java
Total defects in this repository : Critical 3, Major 35
New defects in this pull request : Critical 2, Major 7

► Click to view details

...om/samsung/retailremote/component/decorator/SpotPatrollingDecorator.java [Hide outdated](#)

Outdated

```
276 +  
277 +  
278 +  
279 + newPoint.setid(oldId);
```

review-bot on 19 Jul Member + 😊 ...

After having been compared to NULL value at SpotPatrollingDecorator.java:271, field 'newPoint' is dereferenced at SpotPatrollingDecorator.java:279.

Need help? [DEREF_AFTER_NULLEX](#) | Set this defect as [won't fix](#) or [false alarm](#)

GitHub Enterprise

Future Work

◈ **Structural analysis as another important use case of static analysis techniques**

- We developed SCRA, a static analysis tool for extracting a set of metrics and data/call relations, currently integrated into SVACE.
- We're developing code analysis services to support refactoring work.

◈ **Combining AI techniques with code analysis techniques**

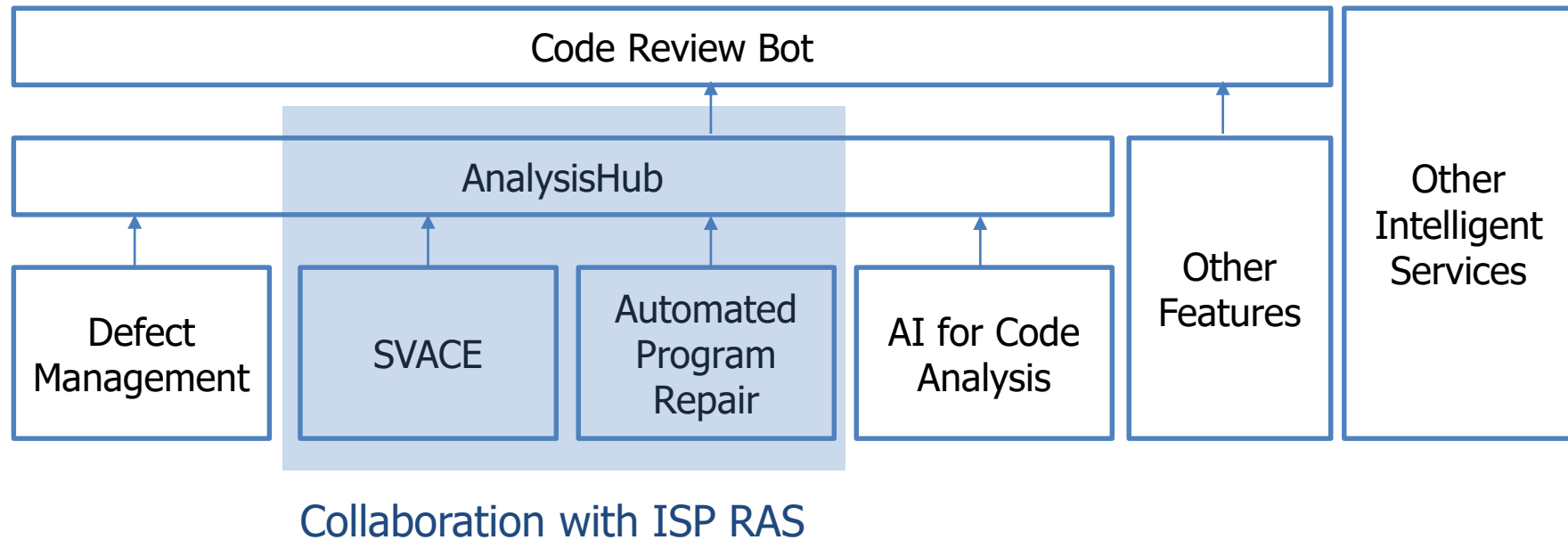
- Further reducing false positives
- Learning bug patterns from big code in repositories

◈ **Reducing code writing effort**

- We started work on automated code fix tools in 2018.
- SVACE can suggest how to fix bugs, for those detected by some C/C++ checkers.
- We're extending this feature for other checkers.

Future Work – Collaboration with ISP RAS

- ◆ Our collaboration is to be continued in the next years.
- ◆ We hope to extend our collaboration to other areas as well.



Thank you



Shape the Future with Innovation and Intelligence