Challenges Analyzing SATE VI Classic Track

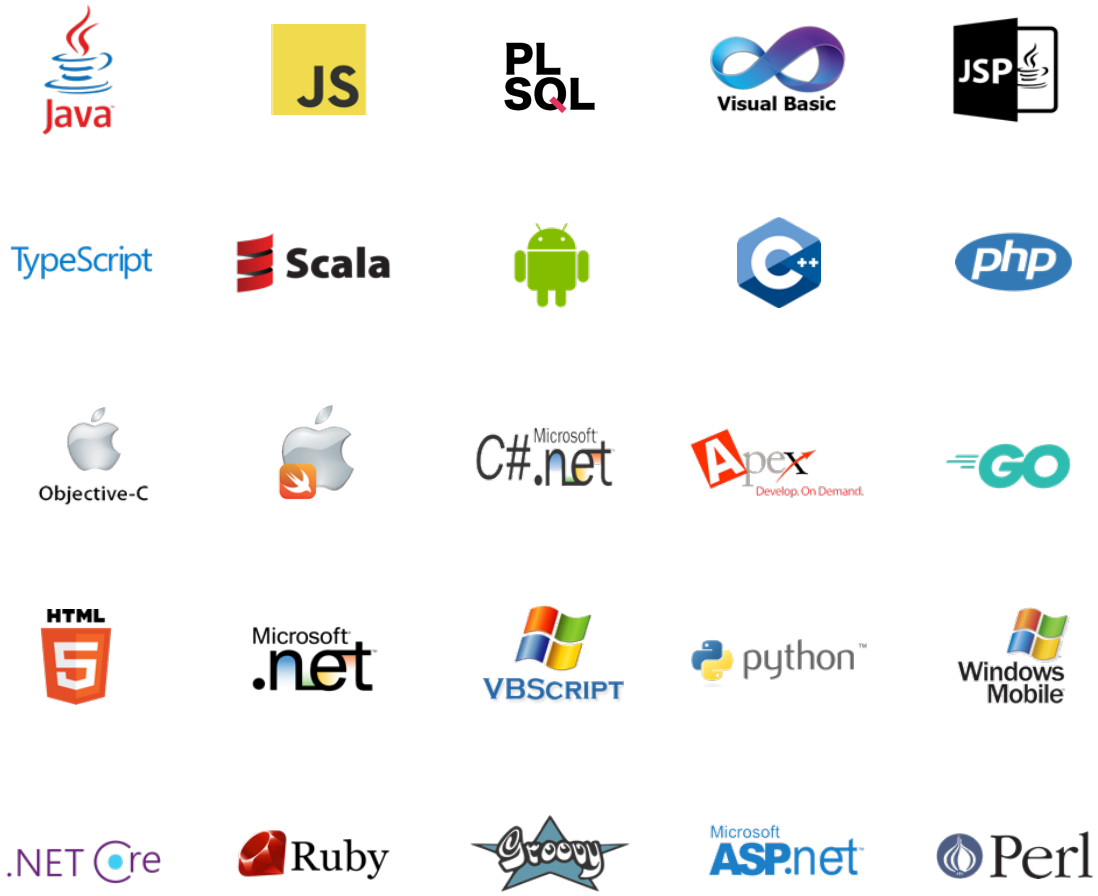
September 2019

Agenda

- About Checkmarx
- CxSAST Results Overview
- “Challenging” Cases
- Feedback for the team

About Checkmarx

- Founded 2006 in Tel Aviv, Israel
- 600+ employees, 1,400+ customers in 70+ countries
- Supports **25 coding and scripting languages** and their frameworks
- Coverage for the **latest development technologies**
- **Zero configuration** to scan any language



“Behind the Scenes” Look at CxSAST

- Common Parser
  - Not compiler based; permissive
  - No build required!
  - C and C++ are challenging
- Object Model, AST, Data Flow Graphs
  - Security-focused “Tainted Data Flow” analysis
- User accessible, modifiable, flexible “queries”
  - Snippets on C# code executed by the analysis engine
  - Easy to create new or modify existing checks
- Pros and Cons

SATE VI Result Overview

- Dspace
  - OOB: 8 missed findings
  - Partial matches: sources or sinks do not align exactly
  - Fixed version: a few FPs due to missed JSP validation

- Sakai
  - OOB: 13 missed findings
  - Mostly partial matches: sources or sinks do not align exactly
  - Fixed: prepared statements identified correctly

Challenging Cases: Sink Mismatch

Sink listed in the XML result file:

```
<location><cwe>89</cwe><path>/sakai-11.2-buggy/lessonbuilder/tool/src/java/org/sakaiproject/lessonbuildertool/tool/beans/SimplePageBean.java</path><line>4307</line><length>4</length><comment>SINK: Write a 'query' without prepared statements causing a potential SQL injection</comment></location>
```

```
4306        String query = "";
4307        query = "INSERT INTO FORUM_SATE (ID, TITLE, BODY) VALUES('" + uuid + "', '" + title + "', 'body')";
4308        statement = connection.createStatement();
4309        int i = statement.executeUpdate(query);
4310        connection.commit();
```

Challenging Cases: Preprocessor Directives

```
/** This file contains code that is specific to Windows.*/
#include "sqliteInt.h"
#if SQLITE_OS_WIN               /* This file is used for Windows only */
…


/* Figure out the effective temporary directory.  First, check if one
 ** has been explicitly set by the application; otherwise, use the one
 ** configured by the operating system.
 */
nDir = nMax - (nPre + 15);
assert( nDir>0 );
if( sqlite3_temp_directory ){
  int nDirLen = sqlite3Strlen30(sqlite3_temp_directory);
  if( nDirLen>0 ){
    ...
    sqlite3_snprintf(nMax, zBuf, "%s", sqlite3_temp_directory);
  }
}
```

What's next for us?

- Better C/C++ support
  - Build-time parameters tracking
  - Control flow tracking

- Easier configuration of custom JSP validation

Feedback for the SATE Team

- Project selection
  - CGC, Sakai: large, time consuming
  - Sqlite and Dspace: optimal size
- Result Processing
  - Allow tool customization
  - DTD has hard-coded paths
  - Automated result matching?
- Better visibility
  - Juliet's foster sibling?
  - Much better benchmark than OWASP Benchmark Project
- Other languages to consider?

Final Thoughts

- Easy to work with
- *Useful* feedback
- Our feedback is considered .. and sometimes even accepted
- Thank you for helping *us* make a better product for our customer!

/ Thank you

Igor Matlin
Phone: +1 (336) 462-8546
Email: igor.matlin@checkmarx.com
www.checkmarx.com