CxQL Query Customization

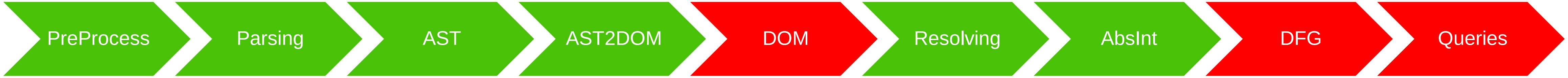
Happy.Yang@checkmarx.com

Agenda

- CxSAST Data Flow Analysis Process
  - Source Code Analysis Process By Engine
  - CxQL Query Analysis
- False Positive and False Negative
  - How False Positive Happens and How to Fix False Positive
  - How False Negative Happens and How to Fix False Negative
- A Case with Query Customization to Fix False Positive
  - False Positives Case Study – Reduce Reflected_XSS_All_Clients False Positives
  - False Positive Analysis and Fix False Positive by Customizing Query Find_XSS_Sanitize
- A Case with Query Customization to Fix False Negative
  - False Negative Case Study – Finding Missing SQL Injection Data Flows
  - False Negative Analysis and Fix False Negative by Customizing Query Find_Interactive_Inputs

CxSAST Data Flow Analysis Process

Source Code Analysis Process By Engine



DOM

Document Object Model

DFG

Data Flow Graph

CxQL

Queries are created using CxAudit

- All source code elements (class, method, variable, expression) will be represented by Nodes of DOM tree
- DFG is build by (InfluencingOn and InfluecingBy) relation between Nodes
- DOM and DFG stays in Memory
- CxQL Query is used in final stage to find all vulnerable data flows

CxQL Query Analysis

Exploitable: Found Input, Found Output, No Sanitizers

Not Exploitable: Found Input, Found Output, Found Sanitizers

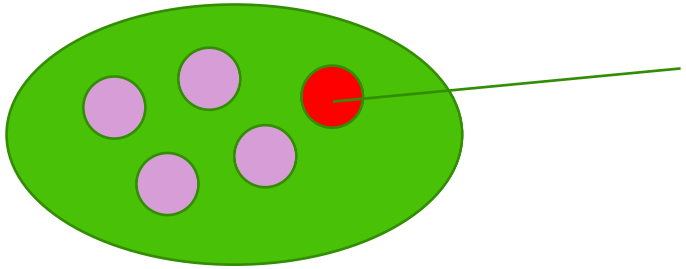
No Data Flows: No Input, or No Output, Or Neither

```
SQL_Injection
CxList db = general.Find_SQL_DB_In();
CxList inputs = general.Find_Interactive_Inputs();
CxList sanitized = general.Find_SQL_Sanitize();
result = inputs.InfluencingOnAndNotSanitized(db, sanitized);
```

False Positive and False Negative

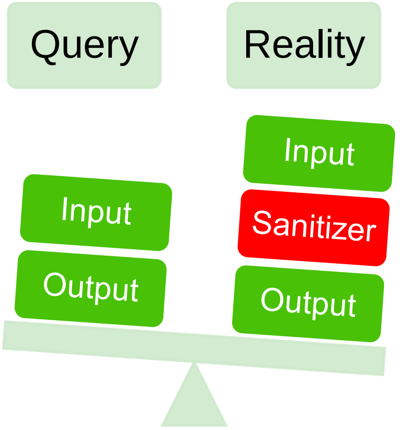
How False Positive Happens and How to Fix False Positive

- False Positive



A Scan result(mostly data flow) is confirmed as not exploitable, so it is called False Positive

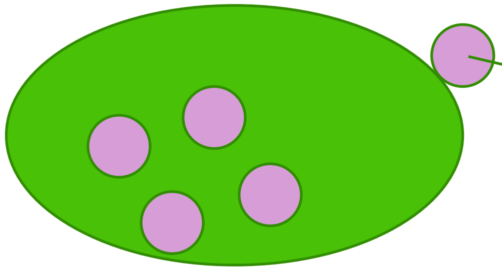
- Why False Positives Happen?
  - Unsupported(Customized) sanitizer
  - Source Code Missing, sanitizer not found



- How to Fix False Positive
  - Only a few happens, in a few projects, change the result state into Not Exploitable from Web Portal
  - Happens a lot, do query customization using CxAudit， add customized sanitizer into related Query

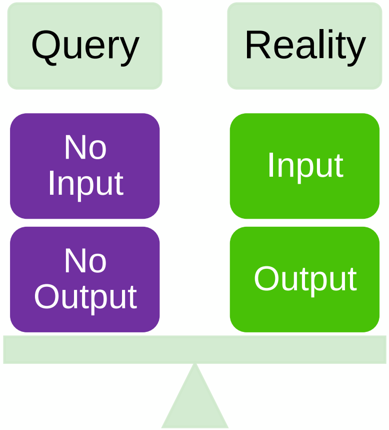
How False Negative Happens and How to Fix False Negative

- False Negative



A vulnerable data flow (confirmed exploitable) can be found by code review. But it's missing in scan results. So it is called False Negative.

- How False Negative Happens
  - Source code missing, can not find some element and data flow
  - Unsupported frameworks, programming languages
  - Input or output, or both can not be found by related Queries.



- How to Fix False Negative

  Is the input and output can be found by queries?

  No → Do query customization, add input and output into related queries

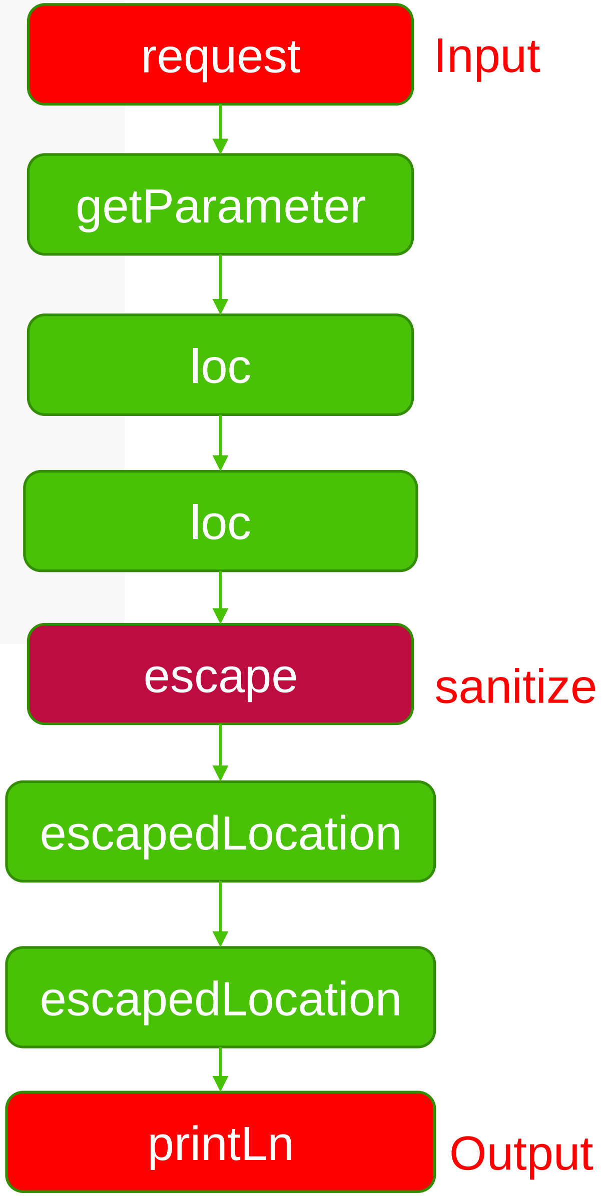
  Yes → check whether input is influencing on output, whether there is some data flow broken

CHECKMARX

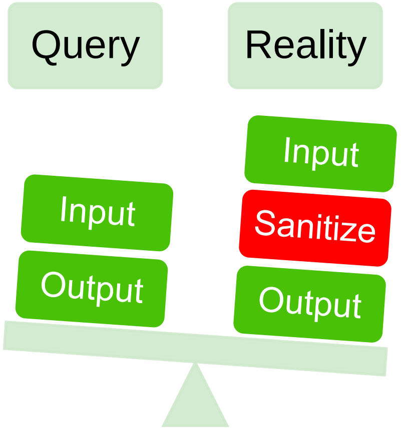
A Case with Query Customization to Fix False Positive

False Positives Case Study – Reduce Reflected_XSS_All_Clients False Positives

```
protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        String loc = request.getParameter("location");
        String escapedLocation =
HtmlEscapers.htmlEscaper().escape(loc);
        out.println("<h1> Location: " + escapedLocation +
"<h1>");
}
```



Query:
Input、
Output、
No Sanitizer,
Exploitable



Reality:
Input、
Output、
Sanitizer,
Not
Exploitable

False Positive Analysis and Fix False Positive by Customizing Query Find_XSS_Sanitize

- This data flow is confirmed Not Exploitable, this is False Positive
- Why not exploitable： the data flow is sanitized by escape method.
- It happens a lot.
- Why False Positive happened： customized sanitizer(escape method) can not be recognized by query.
- How to fix： customize query， put escape method into the query Find_XSS_Santize, the following query is auto-generated

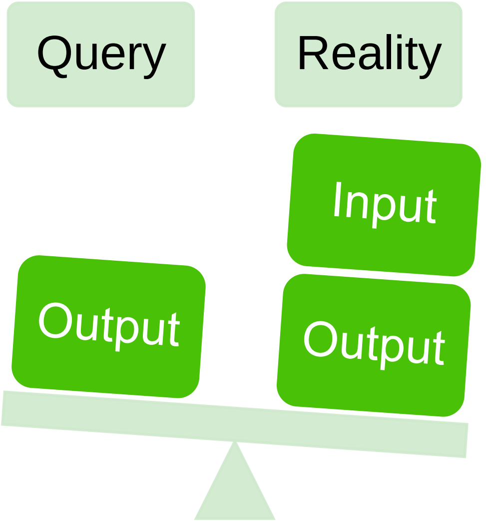
- result = base.Find_XSS_Sanitize();
- result.Add(All.FindByName("TestClass.doGet.HtmlEscapers.escape") +
- All.FindAllReferences(All.FindDefinition(All.FindByName("TestClass.doGet.HtmlEscapers.escape "))) +
- All.FindAllReferences(All.FindByName("TestClass.doGet.HtmlEscapers.escape "))+
- All.FindByMemberAccess(“HtmlEscapers.escape "));

A Case with Query Customization to Fix False Negative

False Negative Case Study – Finding Missing SQL Injection Data Flows

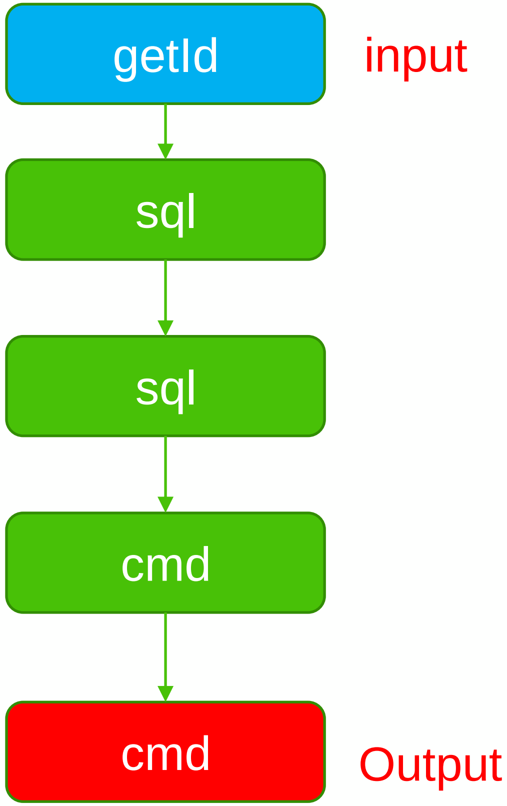
```
// Example 4: interactive input not detected
public void getUser(){
    string sql = "SELECT FROM users WHERE id='" + CurrentUser.getId() + “'”;
    MySqlCommand cmd = new MySqlCommand(sql, conn);
    conn.Open();
    SqlDataReader reader = cmd.ExecuteReader();
}
```

Query：
No Input，
Output，
No Sanitizer，
No data flow



Reality：
Input、
Output、
No Sanitizer，
Exploitable

A vulnerable data flow is found by code review



False Negative Analysis and Fix False Negative by Customizing Query Find_Interactive_Inputs

- A SQL Injection is found by code review
- The data flow is exploitable.
- Exploitable Reason： SQL query is built by string concatenation, the data is not sanitized
- Is able to find Output， Not Input
- Why False Negative Happened： getId which is an input can not be found by query
- How to Fix： add function getId into the query Find_Interactive_Inputs

- result = base.Find_Interactive_Inputs();
- result.Add(All.FindByName("CurrentUser.getId") +
- All.FindAllReferences(All.FindDefinition(All.FindByName("CurrentUser.getId"))) +
- All.FindAllReferences(All.FindByName("CurrentUser.getId"))+
- All.FindByMemberAccess("CurrentUser.getId"));

Thank you

happy.yang@checkmarx.com