The Cool Future of Source Code Analysis

USING THE WISDOM OF THE CROWD TO ENHANCE APPLICATION SECURITY

By Moshe Lerner
VP Corporate Strategy at Checkmarx
ABOUT MYSELF

Moshe Lerner
VP Corporate Strategy at Checkmarx

- Led many Israeli software companies
- A technology, vision and business match-maker

moshel@checkmarx.com
APPLICATION SECURITY ANOMALY

Ratio   Developers VS QA experts
2 : 1

Ratio   Developers VS Appsec experts
150 : 1

can we truly secure apps?
NEW PARADIGMS

Agile Development

Continuous Deployment

Beta = GA

Hybrid Apps

CAN WE CREATE A TRUE SECURE SDLC?
ISSUES AT HAND

The biggest challenge of current source code analysis solutions is size and agility!

How to deliver:

1. Usable results
2. Automatically
3. Out-of-the-box
4. Accurate

for extra large code bases with thousands+ of results
AGENDA

1. How to automatically detect issues that the user does not even know how to describe.
   - “Extracting knowledge” from a large code base
     (wisdom of the crowd)

2. How to automatically correlate ???
   - Suggest remediation actions for a fraction of the time for extra large code bases
ZERO DAYS? ZERO CONFIGURATION?

• What happens if you do not even know what question to ask?
• What if you do not have the resources to configure the system?
• We want a “guru” that asks the questions for us.
  – Configures the system for us.
  – Finds the vulnerabilities for us.
  – Guides us how to fix.

Hold on for a few more slides ...
SOURCE CODE ANALYSIS

HISTORY
FIRST GENERATION CODE ANALYSIS

• The system came out of the box with the relevant security knowledge wired into the system.

• Little-to-no adaptation capabilities.
FIRST GENERATION CODE ANALYSIS

```c
void main()
{
    int j = 0;
    int i = 0;

    while (i < 10){
        if (i == 3){
            j=j*2;
        }
        j = j + i;
        i = i + 1;
    }

    printf("%d\n", j);
    printf("%d", i);
}
```
NEW GENERATION CODE ANALYSIS

- The system came out of the box with the relevant security knowledge
- Ability to customize existing security knowledge
- Ability to add your own business logic
- EASY!! Virtual Compiler. No need to compile your code.
- EASY!! Incremental scan.
- Detection ranges from SQL Injection to Backdoors
void main()
{
    int j = 0;
    int i = 0;

    while (i < 10){
        if (i == 3){
            j=j*2;
        }
        j = j + i;
        i = i + 1;
    }

    printf("%d", j);
    printf("%d", i);
}
SOQL/SOQL Injection:

```java
CxList db = Find_DB();
CxList inputs = Find_Interactive_Inputs();
CxList sanitized = Sanitize();

result = db.InfluencedByAndNotSanitized(inputs, sanitized);
```

SOQL Injection:

- **A** = Input
- **B** = Input
- **C** = `escapeSingleQuotes(A)`
- **DB** = `(C + B)`
CxList $\textbf{Input} = \text{All.FindByName(“input”)}$;
CxList $\textbf{DB} = \text{All.FindByName(“execute”)}$;
CxList $\textbf{Fix} = \text{All.FindByName(“fix”)}$;

Return $\text{DB.InfluencedByAndNotSanitized(\text{input}, \text{fix})}$;
void main()
{
    int j = 0;
    int i = 0;

    while (i < 10){
        if (i == 3){
            j=j*2;
        }
        j = j + i;
        i = i + 1;
    }
    printf("%dn", j);
    printf("%dn", i)
}
The Cool Future of Source Code Analysis

SCKD
Source Code Knowledge Discovery

“Using Wisdom of the crowd (Big Data) to identify security vulnerabilities via code irregularities”
ZERO DAYS? ZERO CONFIGURATION?

• What happens if you do not even know what question to ask?
• What if you do not have the resources to configure the system?
• We want a “Guru” that asks the questions for us.
• Configures the system for us.
• Finds the vulnerabilities for us.
• Guides us.
THERE IS SUCH A GURU

YOU
YOU
YOU
and... YOU!

All of you – Wisdom of the crowd
Most of the developers write good, standard, quality code, most of the time
CROWD

We can set a baseline based on code statistics and find deviations thereof
“Knowledge discovery describes the process of automatically searching large volumes of data for patterns that can be considered knowledge about the data. It is often described as deriving knowledge from the input data. Knowledge discovery developed out of the Data mining domain, and is closely related to it both in terms of methodology and terminology.”
TECHNIQUE

• Building reference data
• Finding common sequences
• Finding violations
S = input();
If (isValid(S))
{
    ...
    response.write(S);
    ...
}

A = input();
If (isValid(A))
{
    ...
    response.write(A);
    ...
}

K = input();
If (isValid(K))
{
    ...
    response.write(K);
    ...
}

M = input();
If (isValid(M))
{
    ...
    response.write(M);
    ...
}

C = input();
If (isValid(C))
{
    ...
    response.write(C);
    ...
}

BUILDING REFERENCE DATA
* = input();
If (isValid(*))
{
    ...
    response.write(*);
    ...
}

v = input();
X

response.write(v);
BACKDOOR
if my name is Moshe, login

If (isAuthenticated(user) || user.name == "Moshe") {
  .......
}
EXAMPLE: LEVERAGING CLOUD OF APPS?

Find similarities between different applications in order to set an intra-corporate standard.

With Zero-Definition!

It’s enough that some apps were fixed. They’ll allow us to find the apps that are not yet fixed.

VAT = 1.05

...  

VAT = 1.08  

...  

VAT = 1.08  

...  

VAT = 1.08
WORKS WELL FOR:

• General:
  – We can find the hidden knowledge of the crowd, give it a name and find breaches of it.

• Security:
  – Make sure the user is authenticated at each page
  – Auto-recognize sanitization routines
  – Backdoors ("if (isValid(user) or user=="Moshe")...")
  – Business logic ("if (qty > 0) {charge (qty*amnt)}")

• Quality
  – Always release a specific resource
  – Best coding practices (auto recognize conventions)
  – Initialize a variable
• Wisdom of the crowd
• Works better for larger enterprises and code bases
GRAPH VISUALIZATION

Optimize call for action

“Using smart graph methods to identify Vulnerability junctions and best fix locations ”
Findings thousands accurate results, does not make us happy ...

Webgoat, for example, has ~220 XXS+SQL Injection

Assuming 30 minutes to fix each one + 30 minutes to validate will take 220 hours - ~ 1 month of work

We’ll narrow this down to 16 places

~1/14 of the time

So we have some time to play golf ;)
CURRENT SITUATION

Each result has a data flow, presented independently from other findings.
SINGLE DATA FLOW PATH - XSS

String s = Request.QueryString["param1"];  
...  
Response.Write(s);  

Request.QueryString["param1"];  

s  

Response.Write(s);  

CHECKMARX
ONE IS EASY ... AND 14?!
Many Single-Path – XSS – a lot of work
BUT...

What do they have in common?
Combined paths
CAN WE:

• Point, click and check without even READING the source code?

• “What if I fix here? Or here?”
What-If I fix here?
Here it is more effective
And here?
Automatic “What-if” => Best Fix Location
Compare the two:

Vs.
BENEFITS

• Gives you the correlation between findings of the same type (e.g. SQLi) and different types.

• You are not dealing with individual findings – but with a complete system

• Use your time better
FIX LOCATIONS

• At the point of a click we narrow down 220 places into 16.
• The more results, the more effective this solution is
RECAP

The biggest challenge of current source code analysis solutions is size!

How to deliver:

1. Actionable results
2. Automatically
3. Out-of-the-box
4. Accurate

for extra large code bases with thousands+ of results
QUESTIONS?
Thank you

Moshe Lerner
Moshel@checkmarx.com