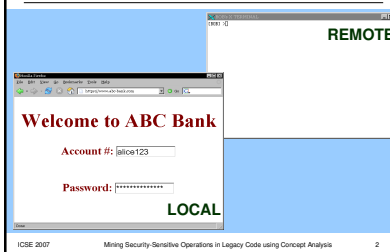


Mining Security-Sensitive Operations in Legacy Code

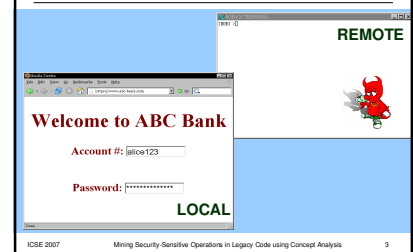
Vinod Ganapathy University of Wisconsin-Madison
Trent Jaeger Pennsylvania State University
David King Pennsylvania State University
Somesh Jha University of Wisconsin-Madison

29th ICSE, Minneapolis, Minnesota
May 25th, 2007

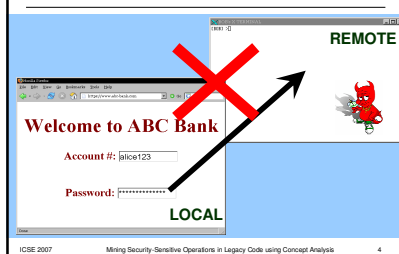
The X Window system



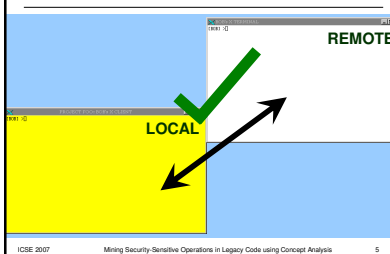
Malicious remote X client



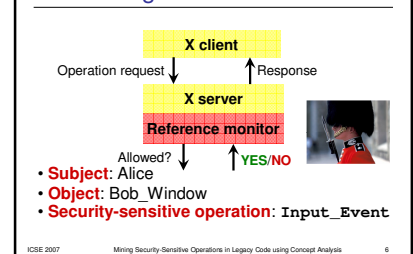
Undesirable information flow



Desirable information flow



Retrofitting the X server



Retrofitting for authorization

- Mandatory access control for Linux
 - Linux Security Modules [Wright et al., '02]
 - SELinux [Lescovoco and Smalley, '01]
- Painstaking, manual procedure**
 - Trusted X, Compartmented-mode workstation, X11/SELinux [Epstein et al., '90][Berger et al., '90][Kilpatrick et al., '03]
 - Java Virtual Machine/SELinux [Fletcher, '06]
 - IBM Websphere/SELinux [Hocking et al., '06]

Contributions

Static analyses to retrofit legacy code for authorization policy enforcement

- Mining security-sensitive operations via concept analysis
- Application to real-world servers
 - ext2 file system
 - X11 server
 - PennMUSH

Outline

- Motivation
- Challenges
- Solution
- Case studies
- Conclusion

Retrofitting lifecycle

1. Identify security-sensitive operations
2. Locate where they are performed in code
3. Instrument these locations

Security-sensitive operations

Source Code

Policy checks

Input_Event

Create
Destroy
Copy
Paste
Map

Can the client receive this Input_Event?

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 10

Problems

Time-consuming

- X11/SELinux ~ 2 years [Kilpatrick et al., '03]
- Linux Security Modules ~ 2 years [Wright et al., '02]

Error-prone

- Violation of complete mediation
- Time-of-check to Time-of-use bugs [Zhang et al., '02][Jaeger et al., '04]

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 11

Our contributions

Fingerprints

Mining

Matching

Focus of this work [IEEE S&P 2006]

Security-sensitive operations

Source Code

Policy checks

Input_Event

Create
Destroy
Copy
Paste
Map

Can the client receive this Input_Event?

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 12

Outline

- Motivation
- Challenges
- Solution
 - Fingerprints
 - Matching fingerprints
 - Mining fingerprints
- Case studies
- Conclusion

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 13

What are fingerprints?

Code-level signatures of security-sensitive operations

- Resource accesses that are unique to a security-sensitive operation
- Denote key steps needed to perform the security-sensitive operation on a resource

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 14

Examples of fingerprints

- Input_Event :-
 Cmp xEvent->type == KeyPress

Security-sensitive operations

Source Code

Input_Event

Create
Destroy
Copy
Paste
Map

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 15

Examples of fingerprints

- Input_Event :-
 Cmp xEvent->type == KeyPress
- Input_Event :-
 Cmp xEvent->type == MouseMove
- Enumerate :-
 Read Window->firstChild &
 Read Window->nextSib &
 Cmp Window # 0

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 16

Fingerprint matching

```

Enumerate :- Read Window->firstChild &
             Read Window->nextSib &
             Cmp Window # 0

MapSubWindows(Window *pParent, Client *pClient) {
    Window *pWin;
    // Run through linked list of child windows
    pWin = pParent->firstChild;
    for (; pWin != 0; pWin=pWin->nextSib) {
        // Code that maps each child window
        ...
    }
}

```

Performs Enumerate

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 17

Placing authorization checks

- X server function MapSubWindows

```


MapSubWindows(Window *pParent, Client *pClient) {
    Window *pWin;
    // Run through linked list of child windows
    if CHECK(pClient, pParent, Enumerate) == ALLOWED {
        pWin = pParent->firstChild;
        for (; pWin != 0; pWin=pWin->nextSib) {
            // Code that maps each child window
            ...
        }
    } else { HANDLE_FAILURE }
}

```

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 18

Fingerprint mining

Security-sensitive operations
Input_Event
Create
Destroy
Copy
Paste
Map

Source Code


Resources
• Window
• xEvent

• **Cmp** xEvent->type == KeyPress
• **Read** Window->firstChild &
Read Window->nextSib &
Cmp Window != 0

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 19

Straw-man solution I

Each resource access is a fingerprint

- Finest level of granularity
- Cmp** xEvent->type == KeyPress
- Read** Window->firstChild
- Read** Window->nextSib
- Cmp** Window != 0

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 20

Problem with this solution

Difficult to write and maintain policies at this level of granularity

- Cmp** xEvent->type == KeyPress
- Read** Window->firstChild
- Read** Window->nextSib
- Cmp** Window != 0

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 21

Straw-man solution II

Each API call is a fingerprint

- Coarsest level of granularity
- Call** MapSubWindows
- Call** MapWindow
- Write policies allowing/disallowing the use of an API call

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 22

Problem with this solution

Does not reflect actual resource accesses performed by API call

- Call** MapSubWindows
 - Enumerates child windows and maps them to the screen
- Call** MapWindows
 - Maps a window onto the screen

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 23

Our solution

Cluster resource accesses that always happen together

- Each API entry point implicitly defines a set of resource accesses
- Cluster resource accesses based upon the API entry points that perform them

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 24

Clustering resource accesses

	MapSub Windows	Map Window	Keyboard Input
Set xEvent->type To MapNotify	✓	✓	
Set Window->mapped To True	✓	✓	
Read Window->firstChild	✓		
Read Window->nextSib	✓		
Cmp Window != 0	✓		
Cmp xEvent->type == KeyPress			✓

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 25

Concept analysis

Instances	MapSub Windows	Map Window	Keyboard Input
Set xEvent->type To MapNotify	✓	✓	
Set Window->mapped To True	✓	✓	
Read Window->firstChild	✓		
Read Window->nextSib	✓		
Cmp Window != 0	✓		
Cmp xEvent->type == KeyPress			✓

Clustering using concept analysis

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 26

Concept analysis

	A	B	C
1 Set xEvent->type To MapNotify	✓	✓	
2 Set Window->mapped To True	✓	✓	
3 Read Window->firstChild	✓		
4 Read Window->nextSib	✓		
5 Cmp Window != 0	✓		
6 Cmp xEvent->type == KeyPress			✓

```

graph TD
    A["{A,B}, {1,2}"] --- B["{A,B,C}, Φ"]
    A --- C["{C}, {6}"]
    A --- D["Φ, {1,2,3,4,5,6}"]
    B --- D
    C --- D
  
```

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 27

Mining fingerprints

	A	B	C
	Window	Map	Keyboard
	Window	Window	App

Fingerprint 1: 1. ReadWindow->DrawableRec->width
2. ReadWindow->DrawableRec->height

Fingerprint 2: 3. ReadWindow->DrawableRec->width
4. ReadWindow->DrawableRec->height

Fingerprint 3: 5. ReadWindow->DrawableRec->width
6. ReadWindow->DrawableRec->height

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 28

Outline

- Motivation
- Challenges
- Solution
- Case studies
- Conclusion

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 29

Results on case studies

Software	LOC	Fingerprints	Avg. Size
ext2	4,476	18	3.7
X Server/dix	30,096	115	3.7
PennMUSH	94,014	38	1.4

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 30

Results on case studies

Benchmark	Manually identified Security-sensitive ops	Fingerprints
ext2	11	18
X Server/dix	22	115

Able to find **at least one fingerprint** for each security-sensitive operation

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 31

Results on case studies

Benchmark	Manually identified Security-sensitive ops	Fingerprints
ext2	11	18
X Server/dix	22	115

Identified automatically in a **few minutes**

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 32

Results on case studies

Benchmark	Manually identified Security-sensitive ops	Fingerprints
ext2	11	18
X Server/dix	22	115

- Associated **59** candidate fingerprints with security-sensitive operations
- Remaining are likely security-sensitive too
Read Window->DrawableRec->width & Read Window->DrawableRec->height

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 33

Summary

Static approach to retrofit legacy code for authorization policy enforcement

- Fingerprints
- Fingerprint mining with concept analysis
- Results:
 - Mined fingerprints for security-sensitive operations in ext2, X server and PennMUSH

ICSE 2007 Mining Security-Sensitive Operations in Legacy Code using Concept Analysis 34

Mining Security-Sensitive Operations in Legacy Code

Vinod Ganapathy David King
 vg@cs.wisc.edu dhking@cse.psu.edu

Trent Jaeger Somesh Jha
 tjaeger@cse.psu.edu jha@cs.wisc.edu