ORACLE®

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

On Dynamic Information-Flow Analysis for Object-Oriented Programs

Yi Lu
Oracle Labs Australia

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Introduction

- Modern object-oriented languages, such as Java and C#, are designed for extensible systems and internet applications
  - Untrusted code may run in the same process as trusted code

- Stack-based access control is used to manage the security requirements of program code
  - Cannot prevent information-flow security vulnerabilities such as confidentiality and integrity violations

ORACLE

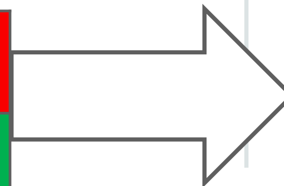
Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Tainted Information Used in Sensitive Operation

```
public class A {
   public static void main(String[] args) throws Exception {
      L l = ...; B b = ...;
      String name = b.getName();
      l.createResource(name);
      ...
   }
}

public class B {
   public String getName() {
      return "password";
   }
   ...
}
```

```
public class L {
   private Resource resource;

   private native Resource create(String name);

   public void createResource(String name) {
      AccessController.checkPermission(
         new ResourcePermission(name, "create"));
      resource = create(name);
   }
}
```

B.getName	ϕ
A.main	ResourcePermission("*", "create")



AC.checkPermission	AllPermission
L.createResource	AllPermission
A.main	ResourcePermission("*", "create")

ORACLE

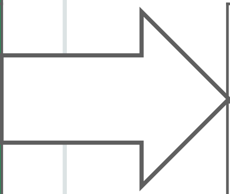
Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Leaked Sensitive Information

```
public class A {
   public static void main(String[] args) throws Exception {
      L l = ...; B b = ...;
      ...
      Resource r = l.getResource();
      b.useResource(r);
   }
}

public class B {
   ...
   public void useResource(Resource res) { ... }
}
```

```
public class L {
   private Resource resource;

   ...

   public Resource getResource() {
      AccessController.checkPermission(
         new ResourcePermission("*", "get"));
      return resource;
   }
}
```

AC.checkPermission	AllPermission
L.getResource	AllPermission
A.main	ResourcePermission("*", "create") ResourcePermission("*", "get")



B.useResource	ϕ
A.main	ResourcePermission("*", "create") ResourcePermission("*", "get")

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Our Research on Information-flow Security for Java

- A security model extends existing access control to track secure information flow
  - Static analysis with proved whole-program security guarantee
  - A points-to analysis based prototype can detect known vulnerabilities
  - False positives due to over-approximation of data structures, call graphs and dynamic features

- Dynamic program analysis (**this talk**)
  - Track information propagation during execution
  - Add runtime checks to reject insecure executions
  - More precise than static analysis, but with limited coverage

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Core Language

$$
\begin{array}{lll}
C & ::= & \text{class } c\ [\text{extends } c]\ \{\overline{f};\ \overline{M}\} \\
M & ::= & m(x)\ \{s\} \\
s & ::= & x{=}\text{new } c \mid x{=}e \mid x.f{=}x \mid x.m(x) \mid \text{if } x \text{ then } s \text{ else } s \mid s;\ s \\
e & ::= & x \mid x.f
\end{array}
$$

- Variables and fields are labelled with a security label
  - e.g. label(x), label(f)

- Standard type safety is assumed

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Dynamic Semantics

- Stack maps local variables to **labelled values** and heap maps locations and fields to **labelled values**
  - e.g. $S(x) = v\ \varphi$, $H(l)(f) = v\ \varphi$
- Programs are evaluated in the execution state $E ::= S\ H$
- Expression evaluation in big-step: $e\ E \Downarrow v\ \varphi$
- Statement evaluation for side effects: $s\ \varphi\ E_1 \Downarrow E_2$

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Generic Information-Flow Analysis

- Security labels φ can be levels, principals, roles, permissions, etc.
  - Information-flow relation $\varphi_1 \rhd \varphi_2$
  - Joined labels $\varphi_1 \sqcup \varphi_2$

- Simple information-flow rules:

$$\text{[VAR]}\quad \frac{S(x) = v\ \varphi}{x\ S\ H\ \Downarrow\ v\ \varphi \sqcup label(x)}$$

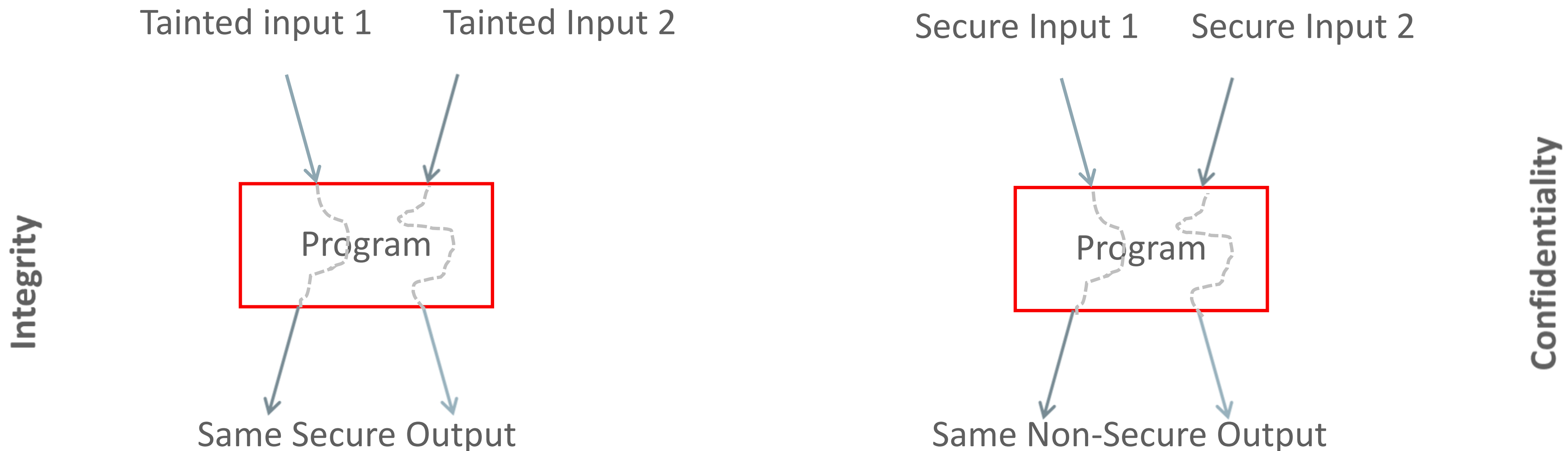
$$\text{[ASSIGN]}\quad \frac{e\ S\ H\ \Downarrow\ v\ \varphi_0 \qquad \varphi \sqcup \varphi_0 \rhd label(x)}{x{=}e\ \varphi\ S\ H\ \Downarrow\ S[x \mapsto (v\ \varphi \sqcup \varphi_0)]\ H}$$

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Noninterference

- Attacker/system should not be able to distinguish two executions from their outputs with a given security label, if they only vary in their inputs with security labels that are not allowed to access it



Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Noninterference

- Attacker/system should not be able to distinguish two executions from their outputs with a given security label, if they only vary in their inputs with security labels that are not allowed to access it

$$\left.\begin{array}{c} E_1 \overset{\varphi}{\approx} E_2 \\ E_1 \overset{\triangleright\varphi}{\approx} E_2 \\ \vdash E_1 \\ \vdash E_2 \\ s\ \varphi_0\ E_1 \Downarrow E_3 \\ s\ \varphi_0\ E_2 \Downarrow E_4 \end{array}\right\} \Longrightarrow E_3 \overset{\varphi}{\approx} E_4$$

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Implicit Flows via Control Structures

```
x = new c;  // location o

if (h) {
    x = a;
}

l = x;
```

- Assume information with label(h) must not flow to label(l)
- Security labels are propagated along with information flows
  - S(x)=o H
  - Sufficient for static analysis

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Implicit Flows via Control Structures

```
x = new c;  // location o

if (h) {
   x = a;
}

l = x;
```

- Assume information with label(h) must not flow to label(l)
- Security labels arepropagated along with information flows
  - S(x)=o H
  - Not sufficient for dynamic analysis
  - Attacker may observe from l that h is false

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

State Convergence of Branches

$$\frac{x\ E \Downarrow l\ \varphi_0 \qquad s_1\ \varphi \sqcup \varphi_0\ E \Downarrow E_1 \qquad s_2\ \varphi \sqcup \varphi_0\ E \Downarrow E_2}{(\textsf{if}\ x\ \textsf{then}\ s_1\ \textsf{else}\ s_2)\ \varphi\ E \Downarrow E_1 \uplus E_2}\ [\text{TRUE}]$$

$$\frac{\left.\begin{array}{r} S_1(x) = v\ \varphi_1 \\ S_2(x) = _\ \varphi_2 \end{array}\right\} \Longrightarrow S(x) = v\ \varphi_1 \sqcup \varphi_2 \qquad \left.\begin{array}{r} H_1(l)(f) = v\ \varphi_1 \\ H_2(l)(f) = _\ \varphi_2 \end{array}\right\} \Longrightarrow H(l)(f) = v\ \varphi_1 \sqcup \varphi_2}{S_1\ H_1 \uplus S_2\ H_2 = S\ H}$$

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Implicit Flows via Dynamic Dispatch

```
x = new c;  // location o

h.m(x);
l = x.f;

class c1 extend c {
    m(x) { x.f = a; }
}

class c2 extend c {
    m(x) { // not update x.f
    }
}
```

- Assume information with label(h) must not flow to label(l)
- Dynamic dispatch is an implicit switch on the type of target object
  - Attacker may observe from l that h is type c1

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

State Convergence of Possible Target Methods

$$\frac{\begin{array}{c} x\ S\ H\ \Downarrow\ l^o\ \varphi_0 \qquad y\ S\ H\ \Downarrow\ v\ \varphi_1 \qquad S_0 = \{\textsf{this} \mapsto (l^o\ \varphi_0), z \mapsto (v\ \varphi_1)\} \\ method(type(o), m) = m(z)\{s\} \qquad s\ \varphi \sqcup \varphi_0\ S_0\ H\ \Downarrow\ _\ H_0 \\ c_{1..n} = \{c \mid c \preceq type(x)\} \qquad \forall i \in 1..n \cdot \left\{ \begin{array}{c} method(c_i, m) = m(z)\{s_i\} \\ s_i\ \varphi \sqcup \varphi_0\ S_0\ H\ \Downarrow\ _\ H_i \end{array} \right. \end{array}}{x.m(y)\ \varphi\ S\ H\ \Downarrow\ S\ H_0 \uplus H_1 \uplus .. \uplus H_n}\ [\text{CALL}]$$

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Implicit Flows on Heap Objects

```
x = new c;  // location o
b = x.f;

if (h) {
    x = a;
}

l = (x.f==b);
```

- Assume information with label(h) must not flow to label(l)
- Field selection can be affected indirectly by variable update
  - Attacker may observe from l that h is false

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Field Selections Depend on Target Object

$$\text{[LOAD]}\quad \frac{S(x) = l\ \varphi_0 \qquad H(l)(f) = v\ \varphi}{x.f\ S\ H\ \Downarrow\ v\ \varphi \sqcup \varphi_0 \sqcup label(f)}$$

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Implicit Flows on Heap Objects

```
x = new c;  // location o
y = x;
b = y.f;

if (h) {
    x.f = a;
}

l = (y.f==b);
```

- Assume information with label(h) must not flow to label(l)
- Aliases must be protected
  - Attacker may observe from l that h is false

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Field Updates Depend on Target Object

$$\frac{S(x) = l\ \varphi_0 \qquad y\ S\ H\ \Downarrow\ v\ \varphi_1 \qquad \varphi \sqcup \varphi_1 \rhd label(f)}{x.f{=}y\ \varphi\ S\ H\ \Downarrow\ S\ H[l \mapsto H(l)[f \mapsto (v\ \varphi \sqcup \varphi_0 \sqcup \varphi_1)]]}\ [\text{STORE}]$$

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Related Work

- Pure dynamic analysis
  - Pistoia, Banerjee and Naumann, S&P'07
  - Askarov and Sabelfeld, CSF'09
  - Austin and Flanagan, PLAS'09
- Hybrid analysis
  - Le Guernic, Banerjee, Jensen, and Schmidt, ASIAN'06
  - Shroff, Smith, and Thober, CSF'07

- *Some works do not consider objects; restricted dynamic analysis may be more conservative than static analysis in cases; none of the works has been used in practice*

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Conclusion

- Theoretically possible to obtain soundness result for dynamic analysis of information-flow security
- Practically infeasible to develop a purely dynamic analysis in the presence of shared objects and implicit information flows
  - Need exhaustive evaluation on all possible branches

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Q&A

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Integrated Cloud

Applications & Platform Services

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

ORACLE®