

Stack-based Access Control and Secure Information Flow

Anindya Banerjee

Joint work with David A. Naumann, Stevens Instt. of Technology

Secure Information Flow

Goal: Modular, static checking of security policies, e.g., confidentiality, for extensible software: no information flow from secret/classified input channels to public output channels (including covert channels).

Formalize for systems programmed in Java-like languages.

- ◆ System with **H**igh and **L**ow inputs, $L \leq H$.
H \equiv secret/private/classified
- ◆ **L** users permitted to see **L** outputs.

(Security Policy: Confidentiality \equiv “PROTECT SECRETS”)

Noninterference

- ◆ Noninterference (NI) [Goguen-Meseguer '82]

“No matter how **H** inputs change, **L** outputs remain same”.

≡ No information flow from **H** to **L**.

Example: Aliasing

```
class LPatient extends Object {//basic patient record
  String name;
  String getName() {return self.name;}
  unit setName(String n) {self.name := n;} }
```

```
class XPatient extends LPatient {
  String hiv; //SECRET
  String getHIV() {return self.hiv;}
  unit setHIV(String s) {self.hiv := s;} }
```

Example: Aliasing

```
LPatient lp := readFile();  
String LBuf := lp.getName(); String HBuf := lp.getName();
```

LBuf ~ lp.name ~ HBuf

```
XPatient xp := new XPatient(); xp.setName(LBuf);
```

LBuf ~ lp.name ~ HBuf ~ xp.name

```
String HBuf := readFromTrustedChannel(); xp.setHIV(HBuf);
```

HBuf ~ xp.hiv; LBuf ~ lp.name ~ xp.name

LBuf := HBuf; lp.setName(xp.getHIV())

lp.name ~ xp.hiv

Annotated Types Prevent Direct Data Flows

```
class LPatient extends Object {  
  (String, L) name;  
  (String, L) getName() {return self.name;}  
  (unit, L) setName((String, L) n) {self.name := n;} }  
}
```

```
class XPatient extends LPatient {  
  (String, H) hiv; //SECRET  
  (String, H) getHIV() {return self.hiv;}  
  (unit, L) setHIV((String, H) s) {self.hiv := s;} }  
}
```

Example: Aliasing revisited

```
(LPatient, L) lp := readFile();
```

```
(String, L) LBuf := lp.getName();
```

```
(String, H) HBuf := lp.getName();
```

```
(XPatient, L) xp := new XPatient(); xp.setName(LBuf:L);
```

```
(String, H) HBuf := readTrusted...; xp.setHIV(HBuf:H);
```

Example: Aliasing revisited

```
(LPatient, L) lp := readFile();
```

```
(String, L) LBuf := lp.getName();
```

```
(String, H) HBuf := lp.getName();
```

```
(XPatient, L) xp := new XPatient(); xp.setName(LBuf:L);
```

```
(String, H) HBuf := readTrusted...; xp.setHIV(HBuf:H);
```

```
LBuf:(String,L) := HBuf:(String,H)
```

```
lp.setName( xp.getHIV():(String,H) )
```

Example: Implicit Control Flow (Conditional)

```
class XPatient extends LPatient { //hiv, getHIV, setHIV }

String leakStatus() {
    var String s;
    if (self.hiv) {s := 'YES';} else {s := 'NO'};
    return s;
}
```

Example: Implicit Control Flow (Conditional)

```
class XPatient extends LPatient { //hiv, getHIV, setHIV }
```

```
String leakStatus() {  
    var String s; //level of s???  
    if (self.hiv) {s := 'YES';} else {s := 'NO'};  
    return s;  
}
```

```
xp := new XPatient();    xp.setName(xp.leakStatus())
```

Example: Implicit Control Flow (Conditional)

```
class XPatient extends LPatient { //hiv, ...
  (String, H) leakStatus() {
    var (String, H) s;
    if (self.hiv) {s := 'YES';} else {s := 'NO'};
    return s;}
xp := new XPatient();    xp.setName(xp.leakStatus():H)
```

Example: Implicit Control Flow (Conditional)

```
class XPatient extends LPatient { //hiv, ...
  (String, H) leakStatus() {
    var (String, H) s;
    if (self.hiv) {s := 'YES';} else {s := 'NO';}
    return s;}
xp := new XPatient();    xp.setName(xp.leakStatus():H)
```

If guard is **H**, only **H**-variables and **H**-fields may be modified.

“No write down”

Example: Implicit Control Flow (Dynamic Dispatch)

```
class XPatient extends LPatient { //hiv, ...
```

```
class YN extends Object {(bool, L)val() {return true;}}  
class Y extends YN {(bool, L)val() {return true;}}  
class N extends YN {(bool, L)val() {return false;}}
```

Example: Implicit Control Flow (Dynamic Dispatch)

```
class XPatient extends LPatient { //hiv, ...
  (YN, H) leak() {
    var (YN, H) o;
    if (self.hiv) {o := new Y();} else {o := new N();}
    return o;}}

class YN extends Object {(bool, L)val() {return true;}}
class Y extends YN {(bool, L)val() {return true;}}
class N extends YN {(bool, L)val() {return false;}}
```

Example: Implicit Control Flow (Dynamic Dispatch)

```
class XPatient extends LPatient { //hiv, ...
  (YN, H) leak() {
    var (YN, H) o;
    if (self.hiv) {o := new Y();} else {o := new N();}
    return o;}}
xp.leak() : (YN, H);

class YN extends Object {(bool, L)val() {return true;}}
class Y extends YN {(bool, L)val() {return true;}}
class N extends YN {(bool, L)val() {return false;}}
```

Example: Implicit Control Flow (Dynamic Dispatch)

```
class XPatient extends LPatient { //hiv, ...
  (YN, H) leak() {
    var (YN, H) o;
    if (self.hiv) {o := new Y();} else {o := new N();}
    return o;}}
xp.leak() : (YN, H);    xp.leak().val() : (bool, ???)

class YN extends Object {(bool, L)val() {return true;}}
class Y extends YN {(bool, L)val() {return true;}}
class N extends YN {(bool, L)val() {return false;}}
```

Example: Implicit Control Flow (Dynamic Dispatch)

```
class XPatient extends LPatient { //hiv, ...  
  (YN, H) leak(){...}  
}
```

- ◆ `xp.leak()` : (YN, H)
- ◆ `xp.leak().val()` : (bool, H)

If level of receiver **H**, level of returned result from method call **H**.

Leaks via aliasing in field update

Assume XPatient has L field bloodGroup.

```
(string, L) test((bool, H) g) {
  (XPatient, L) xp1 := new XPatient;
  (XPatient, L) xp2 := new XPatient;
  (XPatient, H) hp;
  (String, L) bg := xp1.bloodGroup;
  //initial value of xp1's bloodGroup
  if g then hp := xp1 else hp := xp2; // aliasing
  hp.bloodGroup := "Z";
  if bg = xp1.bloodGroup then
    result := "no" else result := "yes"; // g is leaked}
```