DLint and JITProf

DLint: Dynamically Checking JS Coding Practice

[ISSTA'15] DLint: Dynamically Checking Bad Coding Practices in JavaScript

Liang Gong, Michael Pradel, Manu Sridharan, Koushik Sen

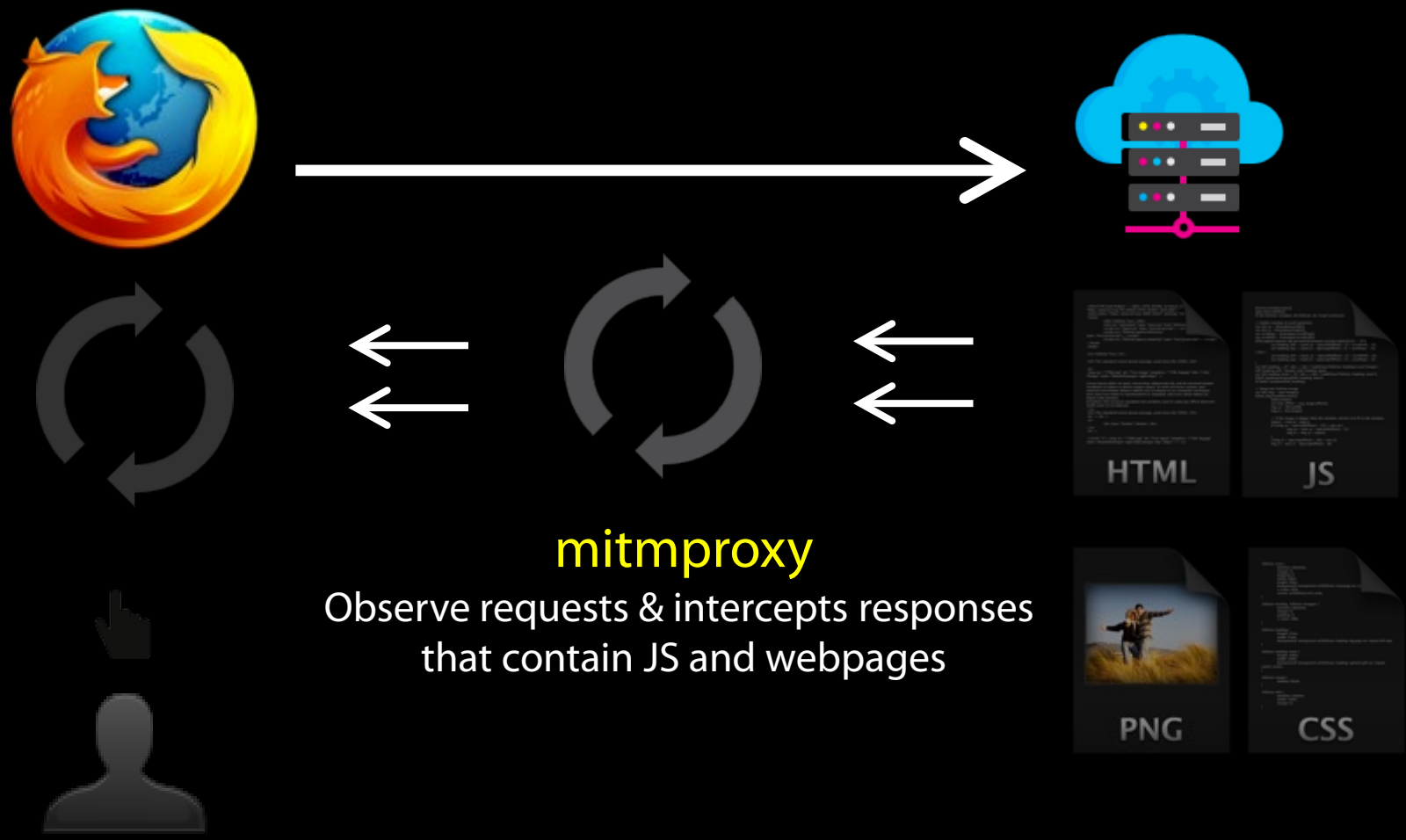
JITProf: Find JS code that prohibit JIT-optimization

[FSE'15] JITProf: Pinpointing JIT-unfriendly JavaScript code

Liang Gong, Michael Pradel, Koushik Sen

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

DLint and JITProf for Web Pages



mitmproxy

Observe requests & intercepts responses that contain JS and webpages

DLint and JITProf

DLint: Dynamically Checking JS Coding Practice

[ISSTA'15] DLint: Dynamically Checking Bad Coding Practices in JavaScript
Liang Gong, Michael Pradel, Manu Sridharan, Koushik Sen

JITProf: Find JS code that prohibit JIT-optimization

[FSE'15] JITProf: Pinpointing JIT-unfriendly JavaScript code
Liang Gong, Michael Pradel, Koushik Sen

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

What are coding practices?

- Good coding practices
  - Informal rules
  - Improve code quality

- Better quality means:
  - Fewer correctness issues
  - Better performance
  - Better usability
  - Better maintainability
  - Fewer security loopholes
  - Fewer surprises
  - …

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Rule: avoid using *for..in* over arrays

```
var sum = 0, value;
var array = [11, 22, 33];
for (value in array) {
    sum += value;
}
> sum ?
```


Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Rule: avoid using *for..in* over arrays

```
var sum = 0, value;
var array = [11, 22, 33];
for (value in array) {
    sum += value;
}
> sum ?
```

array index
(not array value)

```
11 + 22 + 33 => 66
```

array index : string

```
0 + 1 + 2 => 3
```

```
0+"0"+"1"+"2" => "0012"
```


Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Rule: avoid using *for..in* over arrays

```
var sum = 0, value;
var array = [11, 22, 33];
for (value in array) {
    sum += value;
}
> sum ?
```

array index
(not array value)

```
11 + 22 + 33 => 66
```

array index : string

```
0 + 1 + 2 => 3
0+"0"+"1"+"2" => "0012"
```

- Cross-browser issues
- Result depends on the Array prototype object

```
> "0012indexOftoString..."
```


Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Rule: avoid using *for..in* over arrays

```
var sum = 0, value;
var array = [11, 22, 33];
for (value in array) {
    sum += value;
}
> sum ?
```

```
for (i=0; i < array.length; i++) {
    sum += array[i];
}
```

```
function addup(element, index, array) {
  sum += element;
}
array.forEach(addup);
```


Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Rule: avoid using *for..in* over arrays

```
var sum = 0, value;
var array = [11, 22, 33];
for (value in array) {
    sum += value;
}
> sum ?
```

```
for (i=0; i < array.length; i++) {
    sum += array[i];
}
```

```
function addup(element, index, array) {
  sum += element;
}
array.forEach(addup);
```


Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Coding Practices and Lint Tools

- Existing Lint-like checkers
  - Inspect source code
  - Detect common mistakes
- Limitations:
  - Approximates behavior
  - Unknown aliases
  - Lint tools favor precision over soundness
- Difficulty: Precise static program analysis

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

DLint

- Dynamic Linter checking code quality rules for JS
- Open-source, robust, and extensible framework
- Formalized and implemented 28 rules
  - Counterparts of static rules
  - Additional rules
- Empirical study
  - It is better to use DLint and static linter together

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Detect *for..in* over arrays with Jalangi

```
var sum = 0, value;
var array = [11, 22, 33];
for (value in array) {
    sum += value;
}
> sum ?
```

```
for (i=0; i < array.length; i++) {
    sum += array[i];
}
```

```
function addup(element, index, array) {
  sum += element;
}
array.forEach(addup);
```


Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Detect *for..in* over arrays with Jalangi

```
for (value in obj) {
    sum += value;
}
```


Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Detect *for..in* over arrays with Jalangi

```
for (value in obj) {
    sum += value;
}
```

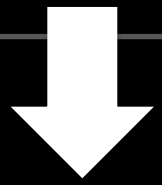
Have a warning when *obj* in *for-in* is an array.

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Detect *for..in* over arrays with Jalangi

```
for (value in obj) {
    sum += value;
}
```

Have a warning when *obj* in *for-in* is an array.



instrumentation

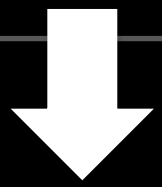
Jalangi Instrumented Code

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Detect *for..in* over arrays with Jalangi

```
for (value in obj) {
    sum += value;
}
```

Have a warning when *obj* in *for-in* is an array.



instrumentation

Jalangi Instrumented Code

```
function forinObject(iid, val) {



}
```


Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Detect *for..in* over arrays with Jalangi

```
for (value in obj) {
    sum += value;
}
```

Have a warning when *obj* in *for-in* is an array.

instrumentation

Jalangi Instrumented Code

```
function forinObject(iid, val) {


}
```


Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Detect *for..in* over arrays with Jalangi

```
for (value in obj) {
    sum += value;
}
```

Have a warning when *obj* in *for-in* is an array.

instrumentation

Jalangi Instrumented Code

```
function forinObject(iid, val) {
    if (isArray(val)) {
        // report warning!
    }
}
```


Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Detect *for..in* over arrays with Jalangi

```
for (value in obj) {
    sum += value;
}
```

Have a warning when *obj* in *for-in* is an array.

instrumentation

Jalangi Instrumented Code

```
function forinObject(iid, val) {
    if (isArray(val)) {
        // report warning!
    }
}
```


Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Detect *for..in* over arrays with Jalangi

```
for (value in obj) {
    sum += value;
}
```

Have a warning when *obj* in *for-in* is an array.

instrumentation

Jalangi Instrumented Code

```
function forinObject(iid, val) {
    if (isArray(val)) {
        J$.iidToLocation(iid);
    }
}
```


Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Detect *for..in* over arrays with Jalangi

```
for (value in obj) {
    sum += value;
}
```

Have a warning when *obj* in *for-in* is an array.

instrumentation

Jalangi Instrumented Code

```
function forinObject(iid, val) {
    if (isArray(val)) {
        // report warning!
        J$.iidToLocation(iid);
    }
}
```

file.js:<start line>:<start col>:<end line>:<end col>

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Detect *for..in* over arrays with Jalangi

```
for (value in obj) {
    sum += value;
}
```

Have a warning when *obj* in *for-in* is an array.

instrumentation

Jalangi Instrumented Code

```
function forinObject(iid, val) {
    if (isArray(val)) {
        // report warning!
        J$.iidToLocation(iid);
    }
}
```

file.js:<start line>:<start col>:<end line>:<end col>

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Checkers

CheckNaN.js
ConcatUndefinedToString.js
NonObjectPrototype.js
SetFieldToPrimitive.js
OverFlowUnderFlow.js
StyleMisuse.js
ToStringGivesNonString.js
UndefinedOffset.js
NoEffectOperation.js
AddEnumerablePropertyToObject.js
ConstructWrappedPrimitive.js
InconsistentNewCallPrefix.js
UncountableSpaceInRegexp.js
FloatNumberEqualityComparison.js

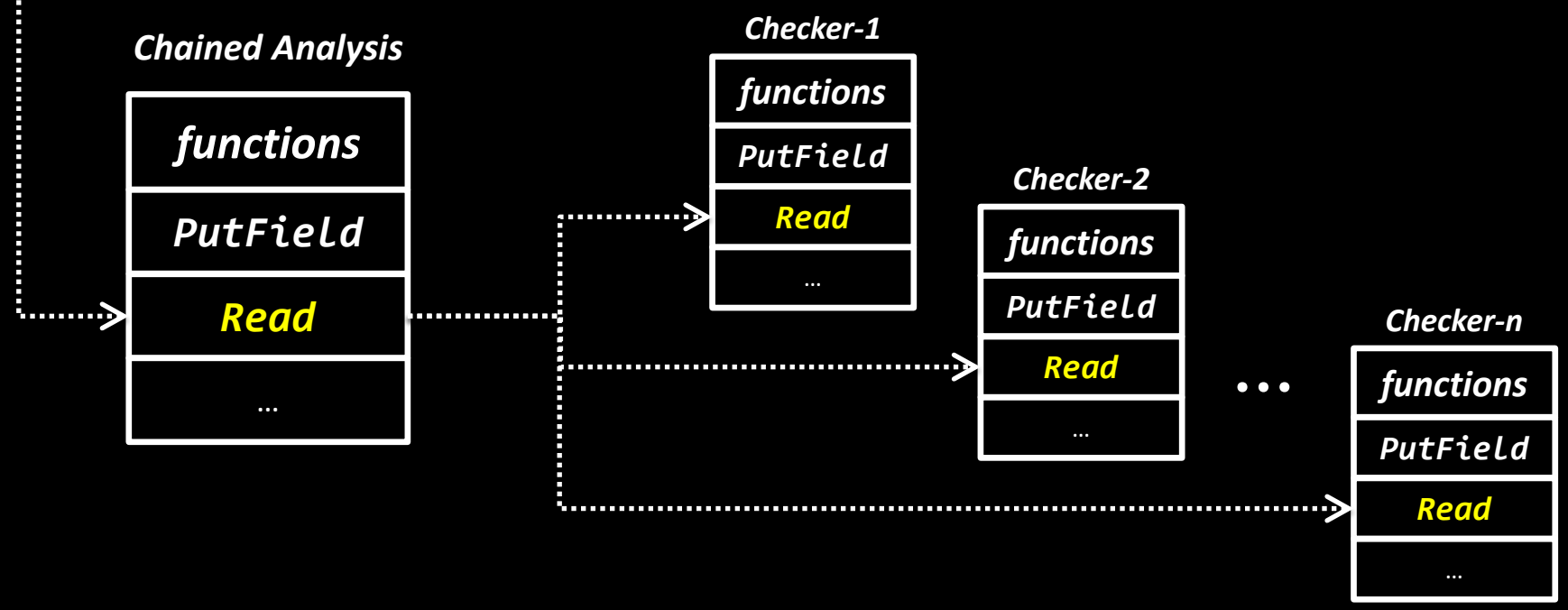
FunctionToString.js
ShadowProtoProperty.js
ForInArray.js
NonNumericArrayProperty.js
OverwrittenPrototype.js
GlobalThis.js
CompareFunctionWithPrimitives.js
InconsistentConstructor.js
FunctionCalledWithMoreArguments.js
IllegalUseOfArgumentsVariable.js
DoubleEvaluation.js
EmptyClassInRegexp.js
UseArrObjConstrWithoutArg.js
MissRadixArgInParseNum.js

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Chained Analysis

```
a.f = b.g
```

```
PutField(Read("a", a), "f", GetField(Read("b", b), "g"))
```



Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Other Resources

Jalangi (v2) Github

https://github.com/Samsung/jalangi2

DLint + JITProf Github based on Jalangi (v2)

https://github.com/ksen007/jalangi2analyses

JITProf Visualization Github based on Jalangi (v2)

https://github.com/JacksonGL/jitprof-visualization

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

DLint and JITProf

DLint: Dynamically Checking JS Coding Practice

[ISSTA'15] DLint: Dynamically Checking Bad Coding Practices in JavaScript
Liang Gong, Michael Pradel, Manu Sridharan, Koushik Sen

JITProf: Find JS code that prohibit JIT-optimization

[FSE'15] JITProf: Pinpointing JIT-unfriendly JavaScript code
Liang Gong, Michael Pradel, Koushik Sen

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Motivation of JITProf

Dynamic language features:

Simplifies coding

- Write less, do more
  → more productive
- Code is less verbose
  → easier to understand

Motivation of JITProf

Dynamic language features:

Simplifies coding

- Write less, do more
  → more productive
- Code is less verbose
  → easier to understand

Slow execution

- Too many runtime checks
- Object property lookup -> hash table lookup

...

Pinpointing JIT-unfriendly JavaScript Code

- Code snippet from Google Octane Benchmark:

```
SplayTree.prototype.insert = function(key, value) {
  ...
  var node = new SplayTree.Node(key, value);
  if (key > this.root_.key) {
    node.left = this.root_;
    node.right = this.root_.right;
    ...
  } else {
    node.right = this.root_;
    node.left = this.root_.left;
    ...
  }
  this.root_ = node;
};
```


Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Pinpointing JIT-unfriendly JavaScript Code

- Code snippet from Google Octane Benchmark:

```
SplayTree.prototype.insert = function(key, value) {
  ...
  var node = new SplayTree.Node(key, value);
  if (key > this.root_.key) {
    node.left = this.root_;
    node.right = this.root_.right;
    ...
  } else {
    node.right = this.root_;
    node.left = this.root_.left;
    ...
  }
  this.root_ = node;
};
```

Cause of poor performance:

- ***node*** has two layouts: offset of ***left*** in ***node*** can be ***0*** or ***1***
- JIT cannot replace ***node.left*** with ***node[0]*** or ***node[1]***

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Pinpointing JIT-unfriendly JavaScript Code

- Code snippet from Google Octane Benchmark:

```
SplayTree.prototype.insert = function(key, value) {
  ...
  var node = new SplayTree.Node(key, value);
  if (key > this.root_.key) {
    node.left = this.root_;
    node.right = this.root_.right;
    ...
  } else {
    node.right = this.root_;
    node.left = this.root_.left;
    ...
  }
  this.root_ = node;
};
```

Performance boost:

15%

6.7%

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Pinpointing JIT-unfriendly JavaScript Code

- Code snippet from Google Octane Benchmark:

```
SplayTree.prototype.insert = function(key, value) {
  ...
  var node = new SplayTreeNode(key, value);
  if (key > this.root_.key) {
    node.left = this.root_;
    node.right = this.root_.right;
    ...
  } else {
    node.right = this.root_;
    node.left = this.root_.left;
    ...
  }
  this.root_ = node;
};
```

JITProf Simulates the Hidden Classes
based on the information provided by Jalangi

Performance boost:

15%

6.7%

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Back to the Motivating Example

```
function Thing(flag) {
    if (!flag) {
        this.b = 4;
        this.a = 3;
    } else {
        this.a = 2;
        this.b = 1;
    }
}

for(var i = 0; i<1000000;i++) {
    var o = new Thing(i%2);
    result += o.a + o.b;
}
```

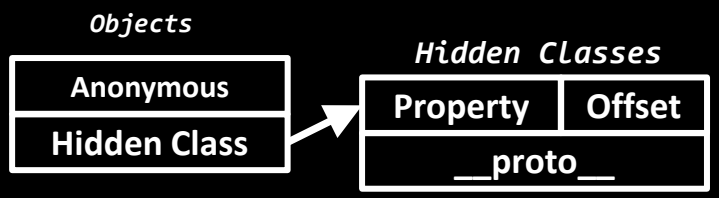
- Each object has a meta information associated with it
- The meta information keeps track of its object layout and its transition history.

Back to the Motivating Example

```
function Thing(flag) {
    if (!flag) {
        this.b = 4;
        this.a = 3;
    } else {
        this.a = 2;
        this.b = 1;
    }
}

for(var i = 0; i<1000000;i++) {
    var o = new Thing(i%2);
    result += o.a + o.b;
}
```

Back to the Motivating Example

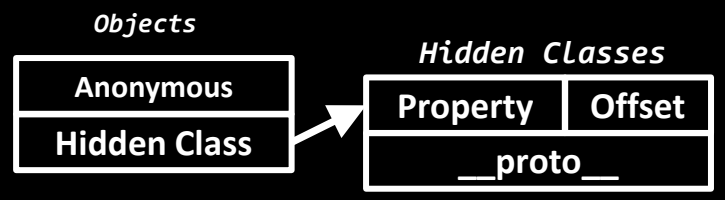
```
function Thing(flag) {
    if (!flag) {
        this.b = 4;
        this.a = 3;
    } else {
        this.a = 2;
        this.b = 1;
    }
}

for(var i = 0; i<1000000;i++) {
    var o = new Thing(i%2);
    result += o.a + o.b;
}
```



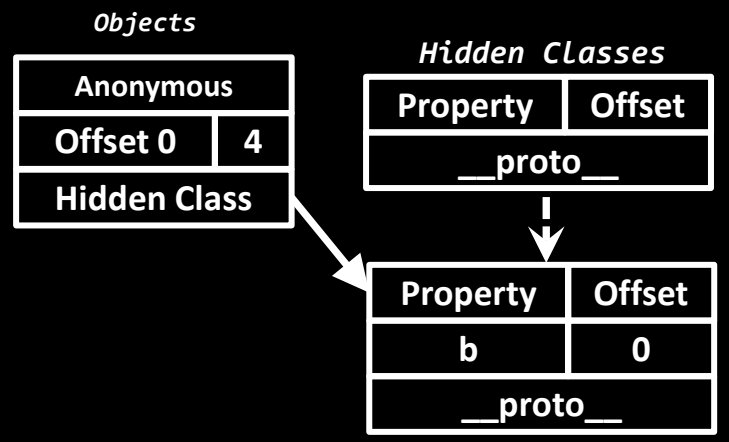
Hidden class simulation before the statement

Back to the Motivating Example

```
function Thing(flag) {
    if (!flag) {
        this.b = 4;
        this.a = 3;
    } else {
        this.a = 2;
        this.b = 1;
    }
}

for(var i = 0; i<1000000;i++) {
    var o = new Thing(i%2);
    result += o.a + o.b;
}
```



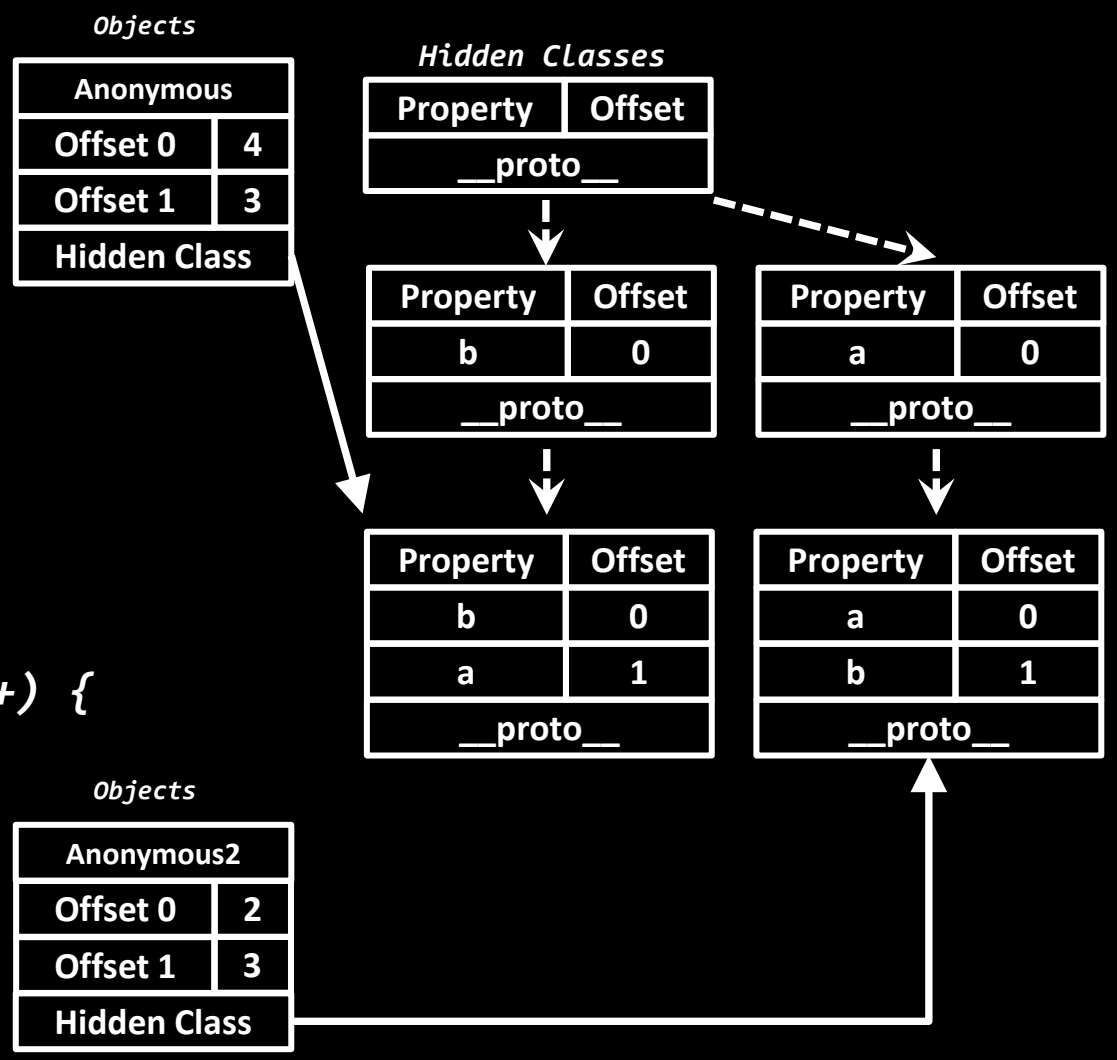
Hidden class simulation before the statement



Hidden class simulation after the statement

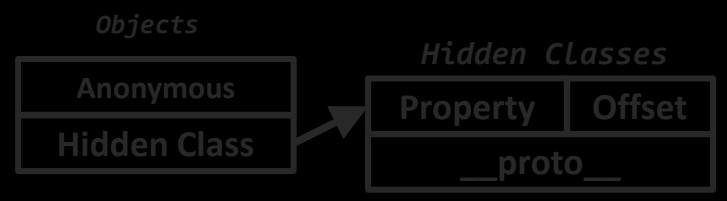
Back to the Motivating Example

```
function Thing(flag) {
    if (!flag) {
        this.b = 4;
        this.a = 3;
    } else {
        this.a = 2;
        this.b = 1;
    }
}

for(var i = 0; i<1000000;i++) {
    var o = new Thing(i%2);
    result += o.a + o.b;
}
```

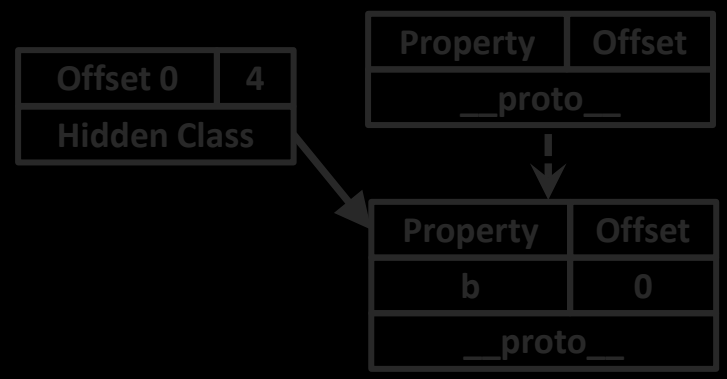


Back to the Motivating Example

```
function Thing(flag) {
    if (!flag) {
        this.b = 4;
        this.a = 3;
    } else {
        this.a = 2;
        this.b = 1;
    }
}

for(var i = 0; i<1000000;i++) {
    var o = new Thing(i%2);
    result += o.a + o.b;
}
```



Hidden class simulation before the statement



Hidden class simulation after the statement

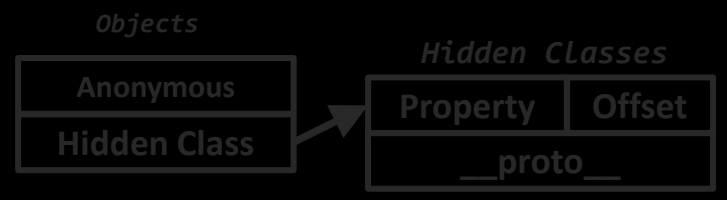
Back to the Motivating Example

```
function Thing(flag) {
    if (!flag) {
        this.b = 4;
        this.a = 3;
    } else {
        this.a = 2;
        this.b = 1;
    }
}

for(var i = 0; i<1000000;i++) {
    var o = new Thing(i%2);
    result += o.a + o.b;
}
```

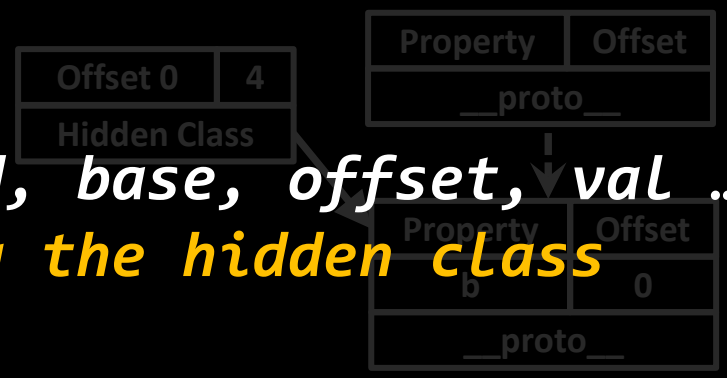
Jalangi

invoke

```
function putFieldPre (iid, base, offset, val … ) {
    // logic for updating the hidden class

}
```



Hidden class simulation before the statement



Hidden class simulation after the statement

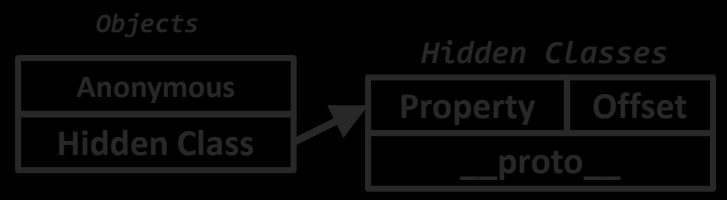
Back to the Motivating Example

```
function Thing(flag) {
    if (!flag) {
        this.b = 4;
        this.a = 3;
    } else {
        this.a = 2;
        this.b = 1;
    }
}

for(var i = 0; i<1000000;i++) {
    var o = new Thing(i%2);
    result += o.a + o.b;
}
```

Jalangi

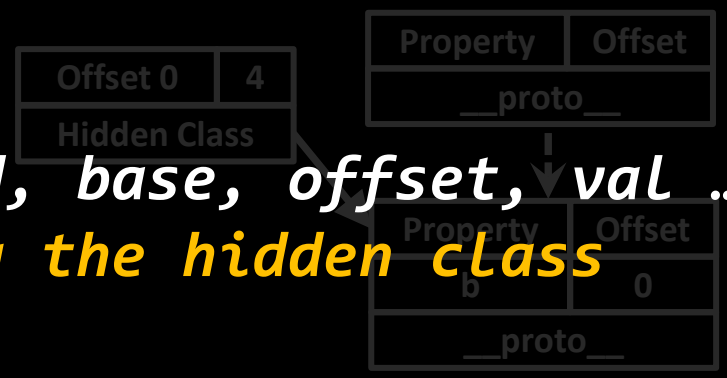
invoke

```
function putFieldPre (iid, base, offset, val … ) {
    // logic for updating the hidden class

}
```



Hidden class simulation before the statement

```
this.b = 4;
```



Hidden class simulation after the statement

Back to the Motivating Example

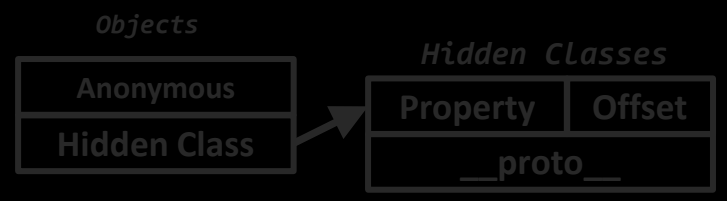
```
function Thing(flag) {
    if (!flag) {
        this.b = 4;
        this.a = 3;
    } else {
        this.a = 2;
        this.b = 1;
    }
}

for(var i = 0; i<1000000;i++) {
    var o = new Thing(i%2);
    result += o.a + o.b;
}
```

Jalangi

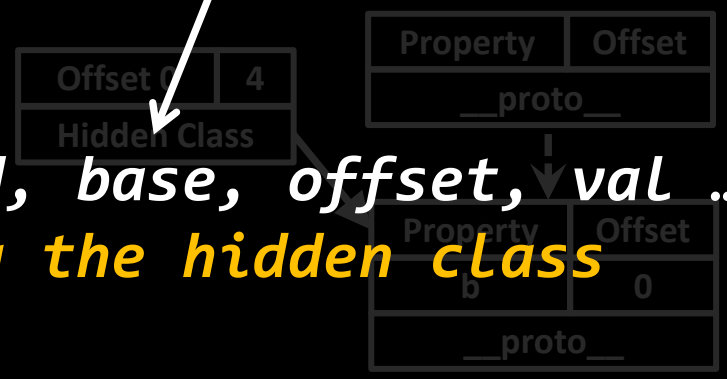
invoke

```
this.b = 4;
```

```
function putFieldPre (iid, base, offset, val … ) {
    // logic for updating the hidden class

}
```



Hidden class simulation before the statement



Hidden class simulation after the statement

Back to the Motivating Example

```
function Thing(flag) {
    if (!flag) {
        this.b = 4;
        this.a = 3;
    } else {
        this.a = 2;
        this.b = 1;
    }
}

for(var i = 0; i<1000000;i++) {
    var o = new Thing(i%2);
    result += o.a + o.b;
}
```

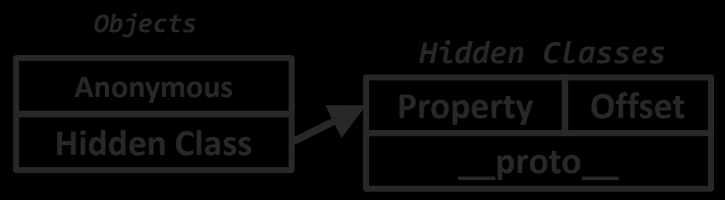
Jalangi

invoke

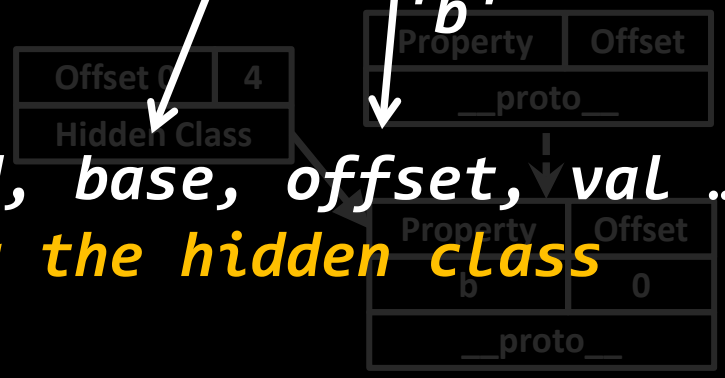
```
this.b = 4;
```

'b'

```
function putFieldPre (iid, base, offset, val … ) {
    // logic for updating the hidden class

}
```



Hidden class simulation before the statement



Hidden class simulation after the statement

Back to the Motivating Example

```
function Thing(flag) {
    if (!flag) {
        this.b = 4;
        this.a = 3;
    } else {
        this.a = 2;
        this.b = 1;
    }
}

for(var i = 0; i<1000000;i++) {
    var o = new Thing(i%2);
    result += o.a + o.b;
}
```

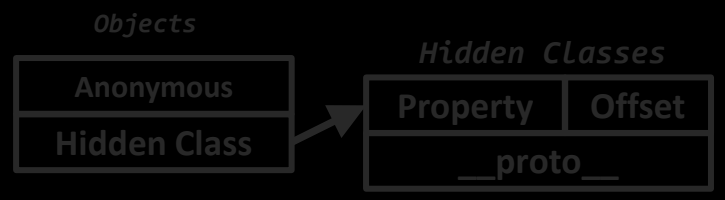
Jalangi

invoke

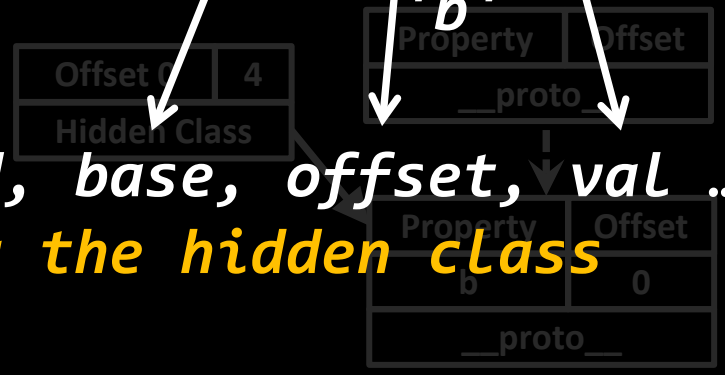
```
function putFieldPre (iid, base, offset, val … ) {
    // logic for updating the hidden class

}
```

this.b = 4;

'b'



Hidden class simulation before the statement



Hidden class simulation after the statement

Back to the Motivating Example

```
function Thing(flag) {
    if (!flag) {
        this.b = 4;
        this.a = 3;
    } else {
        this.a = 2;
        this.b = 1;
    }
}

for(var i = 0; i<1000000;i++) {
    var o = new Thing(i%2);
    result += o.a + o.b;
}
```

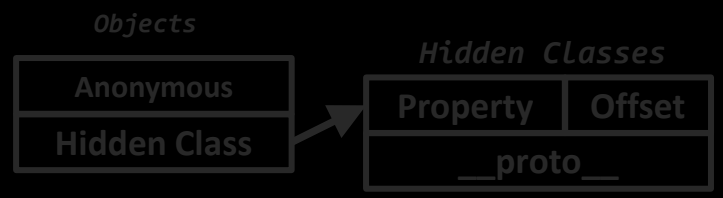
Jalangi

invoke

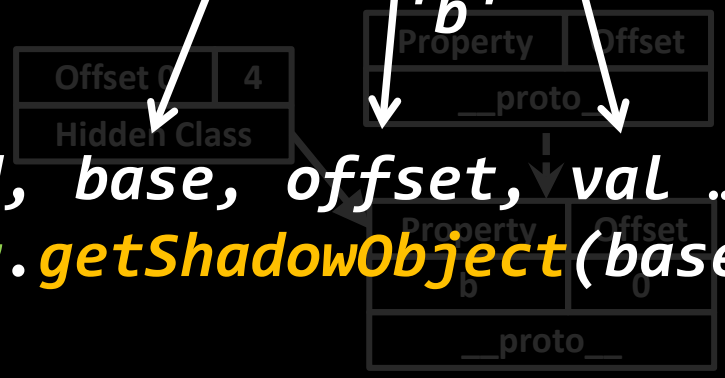
```
function putFieldPre (iid, base, offset, val … ) {
    var sobj = J$.smemory.getShadowObject(base);
    sobj.hiddenClass ...
}
```

this.b = 4;

'b'



Hidden class simulation before the statement



Hidden class simulation after the statement

Back to the Motivating Example

```
function Thing(flag) {
    if (!flag) {
        this.b = 4;
        this.a = 3;
    } else {
        this.a = 2;
        this.b = 1;
    }
}

for(var i = 0; i<1000000;i++) {
    var o = new Thing(i%2);
    result += o.a + o.b;
}

var o = {a: 1, b: 2};
```

Intercept *putField* to update the hidden class

Back to the Motivating Example

```
function Thing(flag) {
    if (!flag) {
        this.b = 4;
        this.a = 3;
    } else {
        this.a = 2;
        this.b = 1;
    }
}

for(var i = 0; i<1000000;i++) {
    var o = new Thing(i%2);
    result += o.a + o.b;
}

var o = {a: 1, b: 2};
```

Intercept *putField* to update the hidden class

Intercept *invokeFun* to record object creation location

Back to the Motivating Example

```
function Thing(flag) {
    if (!flag) {
        this.b = 4;
        this.a = 3;
    } else {
        this.a = 2;
        this.b = 1;
    }
}

for(var i = 0; i<1000000;i++) {
    var o = new Thing(i%2);
    result += o.a + o.b;
}

var o = {a: 1, b: 2};
```

Intercept *putField* to update the hidden class

Intercept *invokeFun* to record object creation location

Intercept *getField* to record inline cache misses

Back to the Motivating Example

```
function Thing(flag) {
    if (!flag) {
        this.b = 4;
        this.a = 3;
    } else {
        this.a = 2;
        this.b = 1;
    }
}

for(var i = 0; i<1000000;i++) {
    var o = new Thing(i%2);
    result += o.a + o.b;
}

var o = {a: 1, b: 2};
```

Intercept *putField* to update the hidden class

Intercept *invokeFun* to record object creation location

Intercept *getField* to record inline cache misses

Intercept *literal* to update hidden class + record object creation location

JIT-unfriendly Code Checked by JITProf

- Use inconsistent object layout
- Access undeclared property or array element
- Store non-numeric value in numeric arrays
- Use in-contiguous keys for arrays
- Not all properties are initialized in constructors
- … and more

Install DLint and JITProf with Jalangi2

https://github.com/ksen007/jalangi2analyses

```
npm install
```

mitmproxy (third-party framework)

```
pip install pyOpenSSL
pip install mitmproxy==0.11.3
```

Install the mitmproxy certificate manually (drag-and-drop)

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

mitmproxy (third-party framework)

- man-in-the-middle proxy
- Interactive, SSL-capable proxy for HTTP with a console interface.
- Intercept http communication between the client and the server for instrumentation.

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Install mitmproxy

- `pip install pyOpenSSL`
- `pip install mitmproxy==0.11.3`

```
~/git/public/mitmproxy (Python)
   GET https://github.com/
       ←200 text/html 5.52kB
   GET https://a248.e.akamai.net/assets.github.com/stylesheets/bundles/github2-24f59e3ded11f2a
       1c7ef9ee730882bd8d550cfb8.css
       ←200 text/css 28.27kB
   GET https://a248.e.akamai.net/assets.github.com/images/modules/header/logov7@4x-hover.png?1
       324325424
       ←200 image/png 6.01kB
   GET https://a248.e.akamai.net/assets.github.com/javascripts/bundles/jquery-b2ca07cb3c906cec
       cfd58811b430b8bc25245926.js
       ←200 application/x-javascript 32.59kB
   ↺ GET https://a248.e.akamai.net/assets.github.com/stylesheets/bundles/github-cb564c47c51a14
         af1ae265d7ebab59c4e78b92cb.css
         ←200 text/css 37.09kB
   GET https://a248.e.akamai.net/assets.github.com/images/modules/home/logos/facebook.png?1324
       526958
       ←200 image/png 5.55kB
>> GET https://github.com/twitter

[7]    [i:.*]                                                    ?:help [*:8080]
```


Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Install mitmproxy

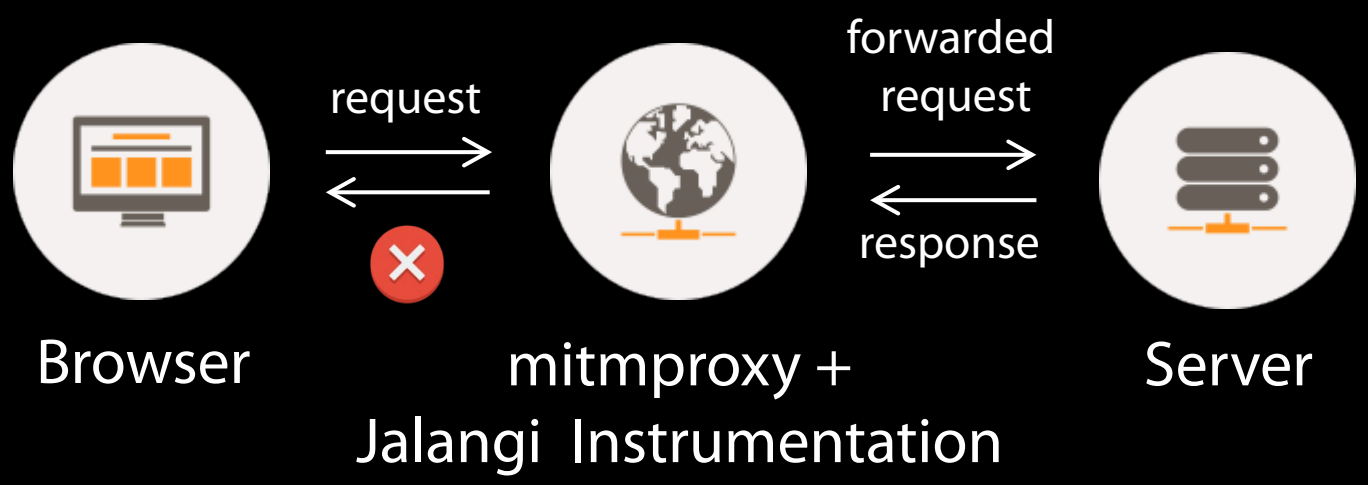
- `pip install pyOpenSSL`
- `pip install mitmproxy==0.11.3`

```
~/git/public/mitmproxy (Python)
    GET https://github.com/
        ←200 text/html 5.52kB
    GET https://a248.e.akamai.net/assets.github.com/stylesheets/bundles/github2-24f59e3ded11f2a
        1c7ef9ee730882bd8d550cfb8.css
        ←200 text/css 28.27kB
    GET https://a248.e.akamai.net/assets.github.com/images/modules/header/logov7@4x-hover.png?1
        324325424
        ←200 image/png 6.01kB
    GET https://a248.e.akamai.net/assets.github.com/javascripts/bundles/jquery-b2ca07cb3c906cec
        cfd58811b430b8bc25245926.js
        ←200 application/x-javascript 32.59kB
    ↺ GET https://a248.e.akamai.net/assets.github.com/stylesheets/bundles/github-cb564c47c51a14
          af1ae265d7ebab59c4e78b92cb.css
          ←200 text/css 37.09kB
    GET https://a248.e.akamai.net/assets.github.com/images/modules/home/logos/facebook.png?1324
        526958
        ←200 image/png 5.55kB
>> GET https://github.com/twitter

[7]    [i:.*]                                                    ?:help [*:8080]
```


Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

The HTTPS Problem

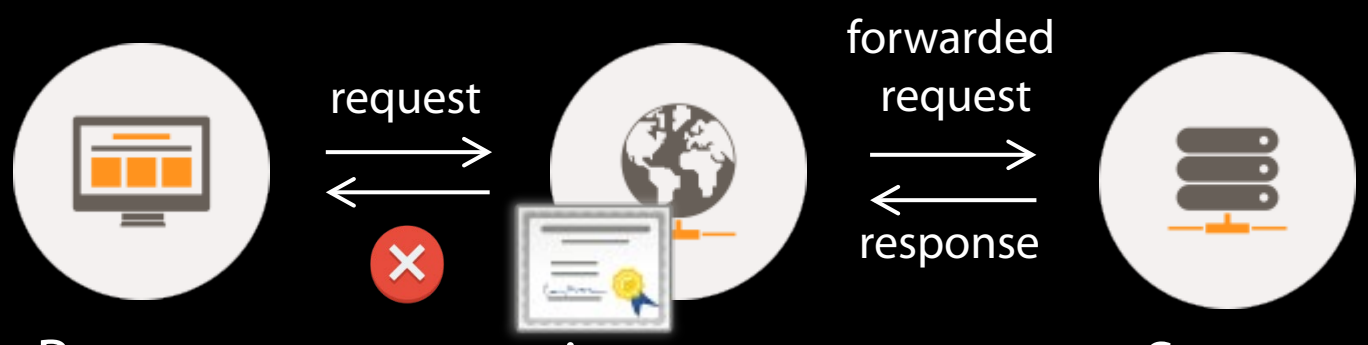
- Man-in-the-middle Proxy
- SSL and HTTPS is designed against MITM
- HTTPS Handle shake error due to uncertified modification via instrumentation



Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

The HTTPS Problem

- Man-in-the-middle Proxy
- SSL and HTTPS is designed against MITM
- HTTPS Handle shake error due to uncertified modification via instrumentation



Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

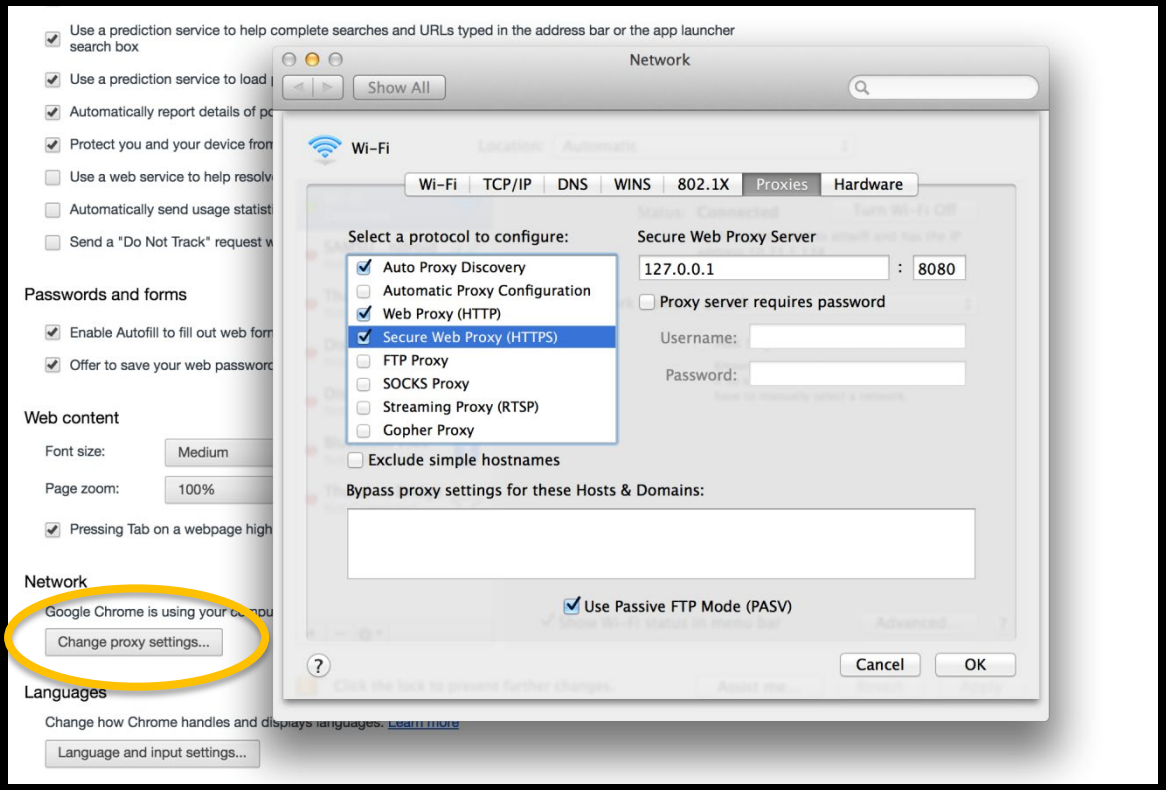
The HTTPS Problem

- Man-in-the-middle Proxy
- SSL and HTTPS is designed against MITM
- HTTPS Handle shake error due to uncertified modification via instrumentation

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Install the CA System of mitmproxy

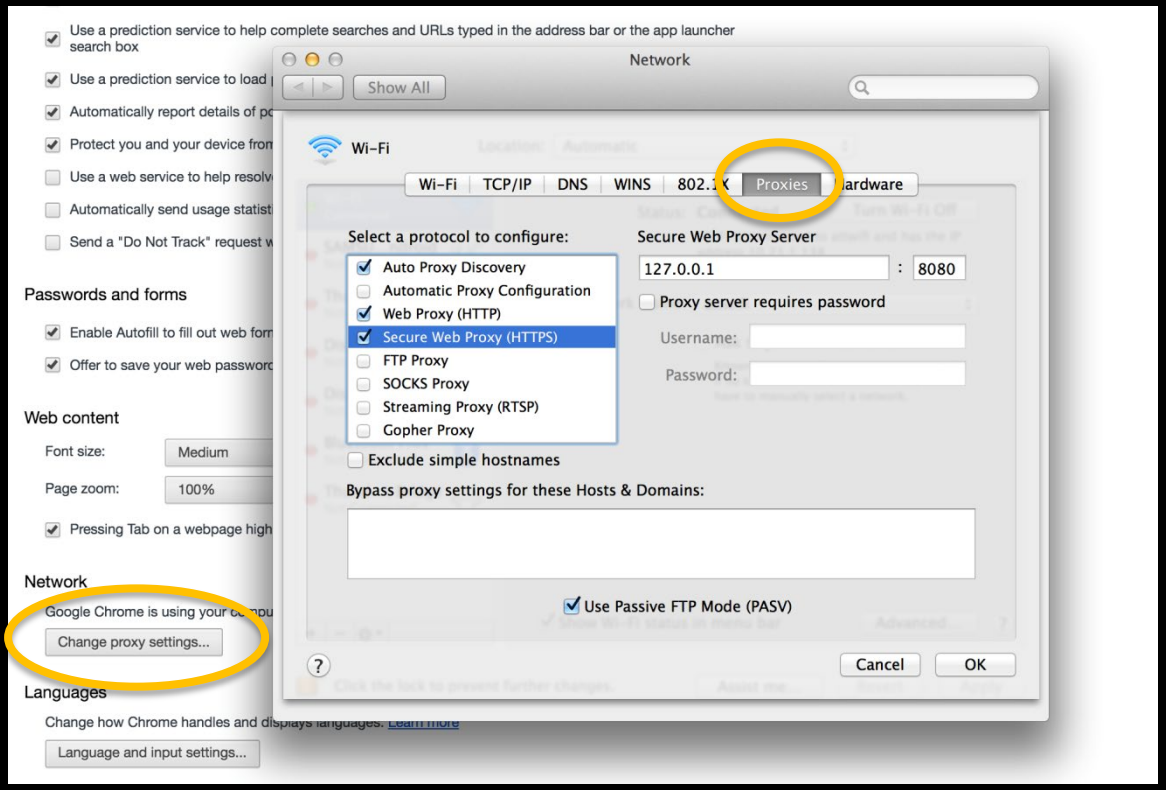
- `pip install mitmproxy==0.11.3`
- Then run `mitmproxy` in the terminal
- In browser, configure HTTP and HTTPS proxy
  - Server: 127.0.0.1 Port: 8080



Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Install the CA System of mitmproxy

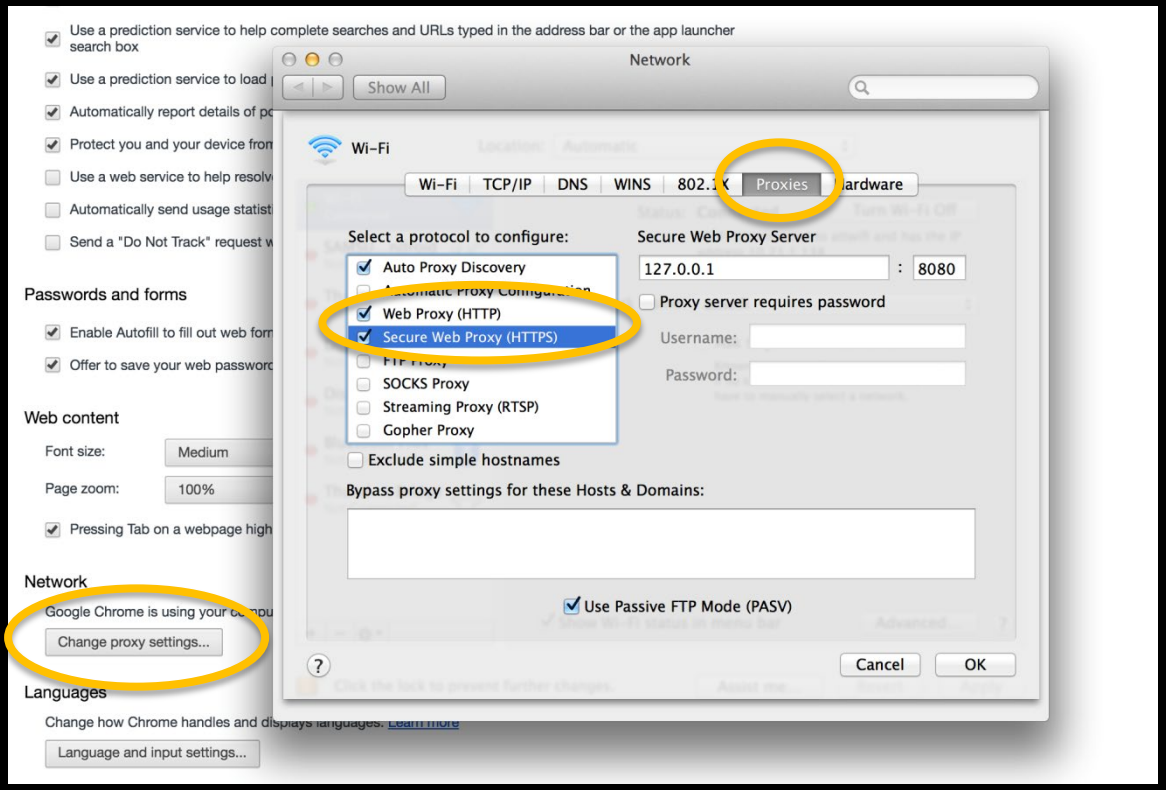
- `pip install mitmproxy==0.11.3`
- Then run `mitmproxy` in the terminal
- In browser, configure HTTP and HTTPS proxy
  - Server: 127.0.0.1 Port: 8080



Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Install the CA System of mitmproxy

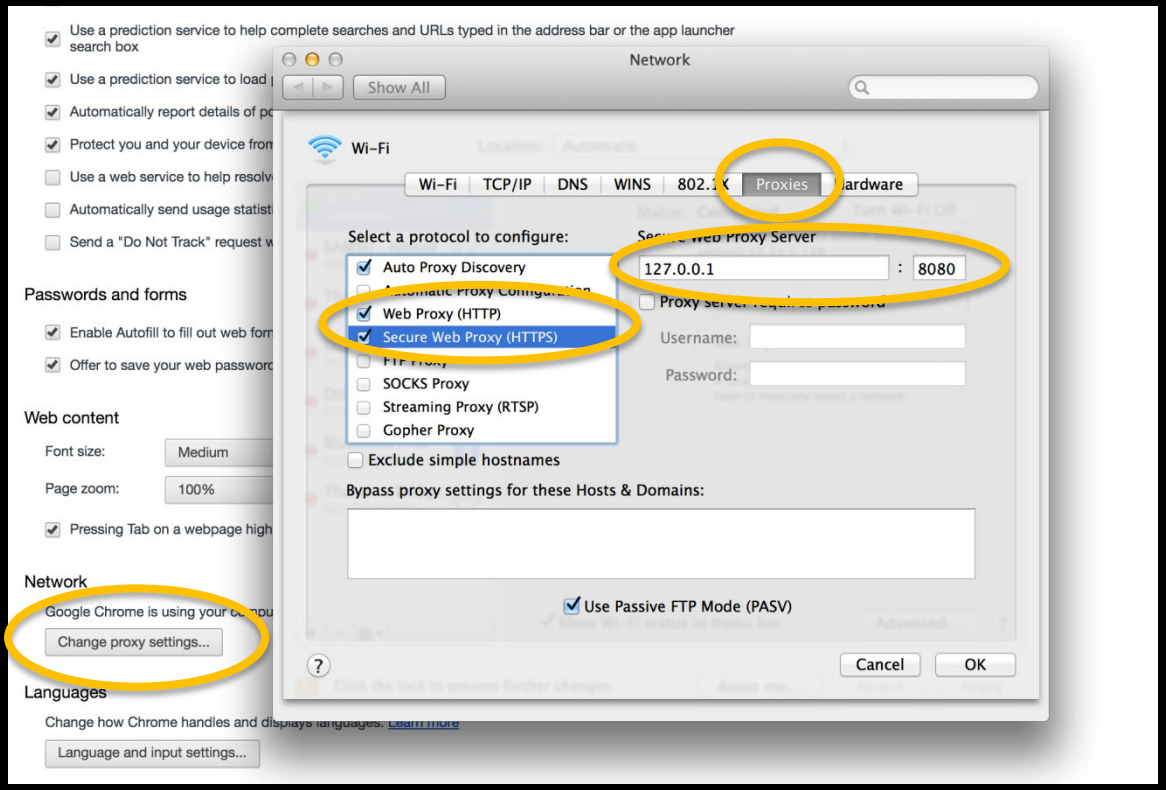
- `pip install mitmproxy==0.11.3`
- Then run `mitmproxy` in the terminal
- In browser, configure HTTP and HTTPS proxy
  - Server: 127.0.0.1 Port: 8080



Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

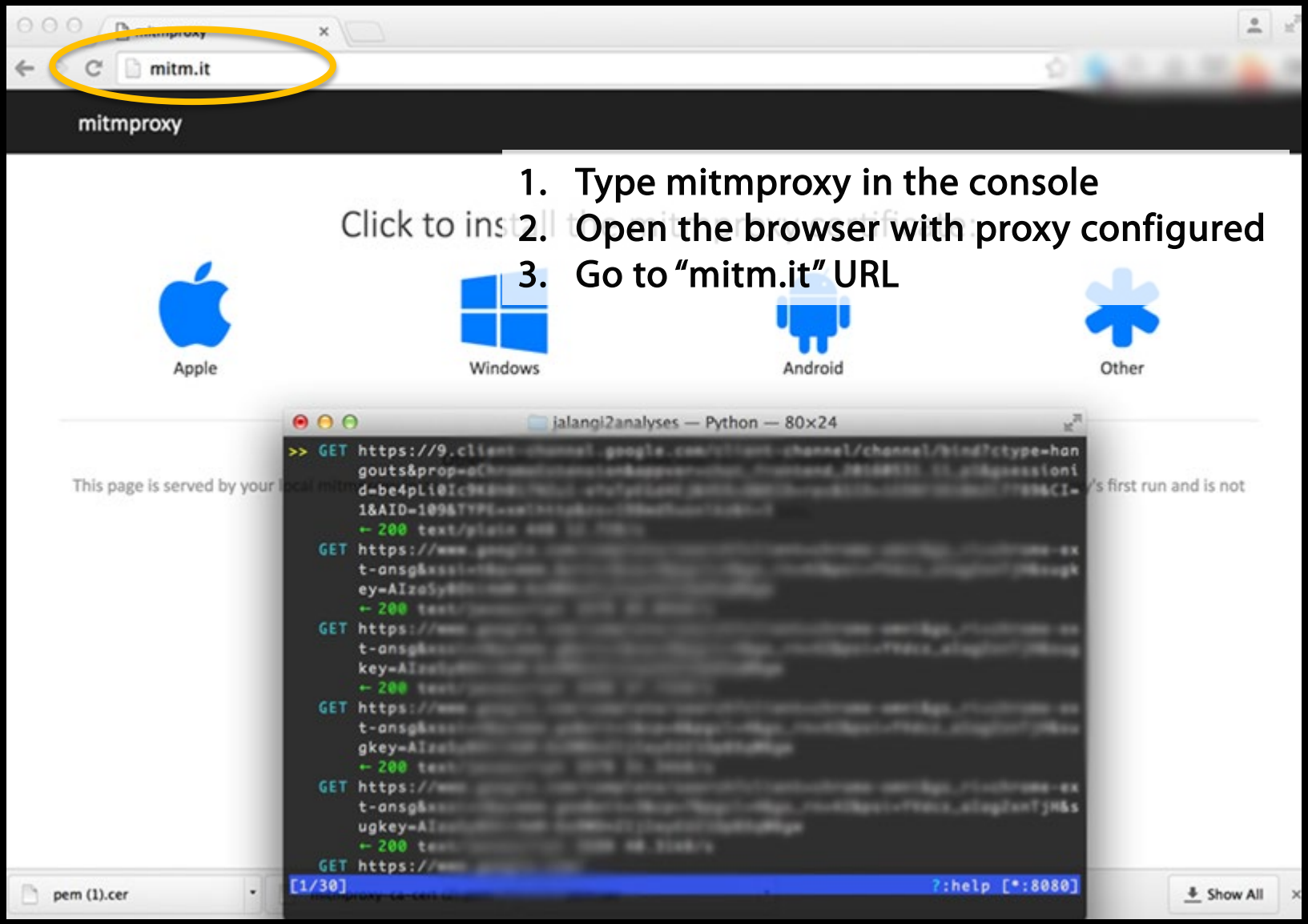
Install the CA System of mitmproxy

- `pip install mitmproxy==0.11.3`
- Then run `mitmproxy` in the terminal
- In browser, configure HTTP and HTTPS proxy
  - Server: 127.0.0.1 Port: 8080



Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

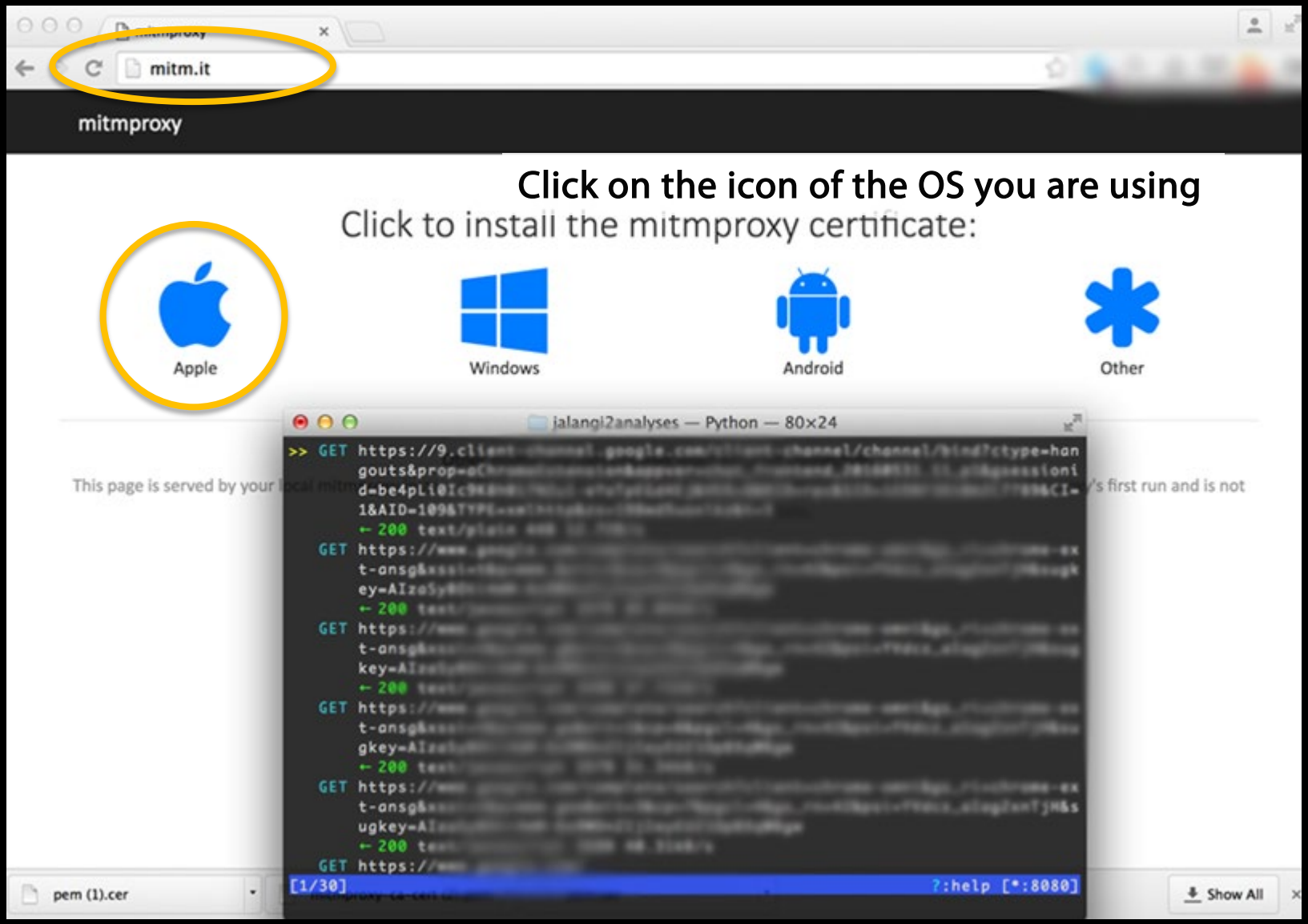
Install the CA System of mitmproxy



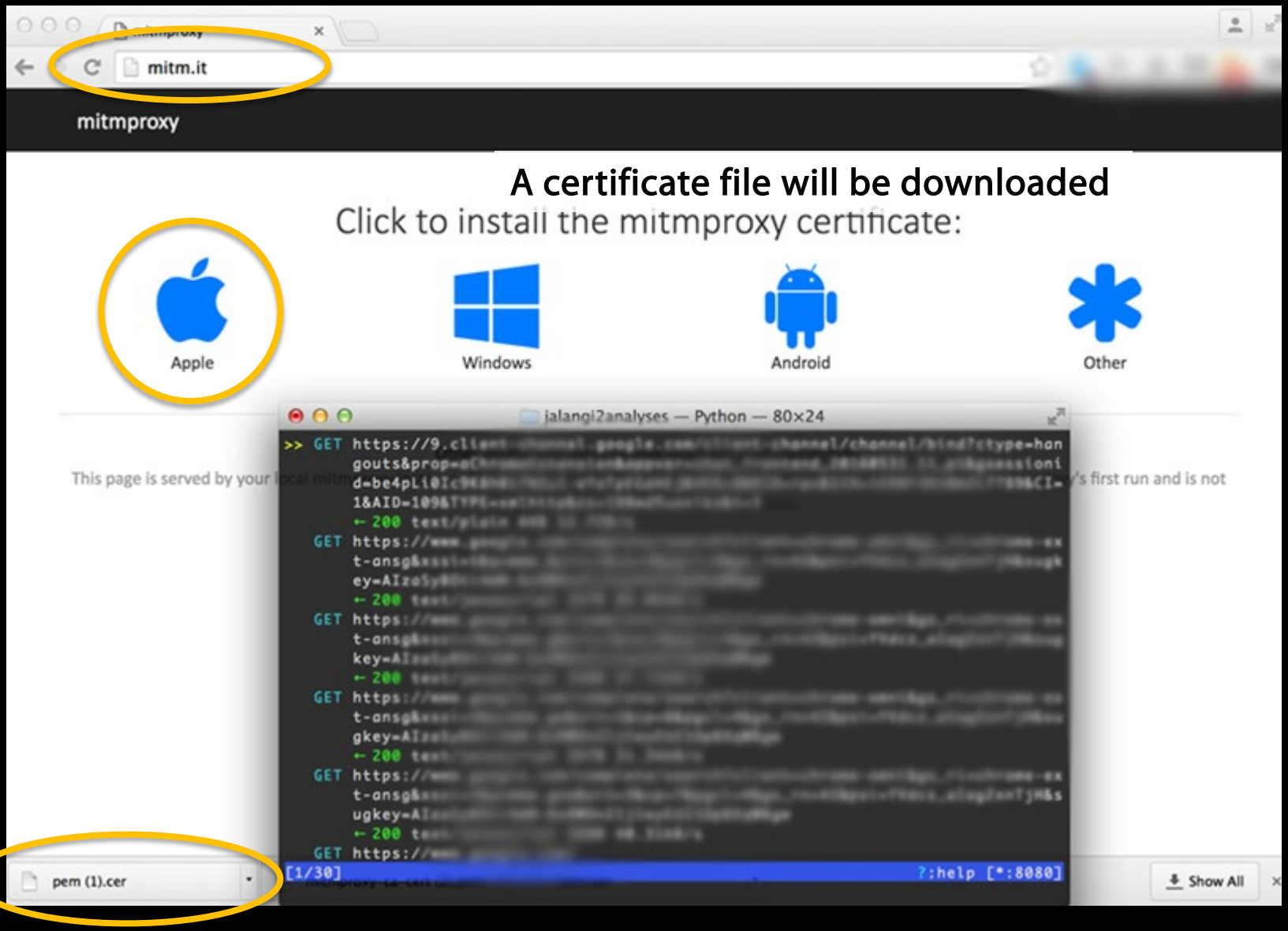
1. Type mitmproxy in the console
2. Open the browser with proxy configured
3. Go to “mitm.it” URL

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

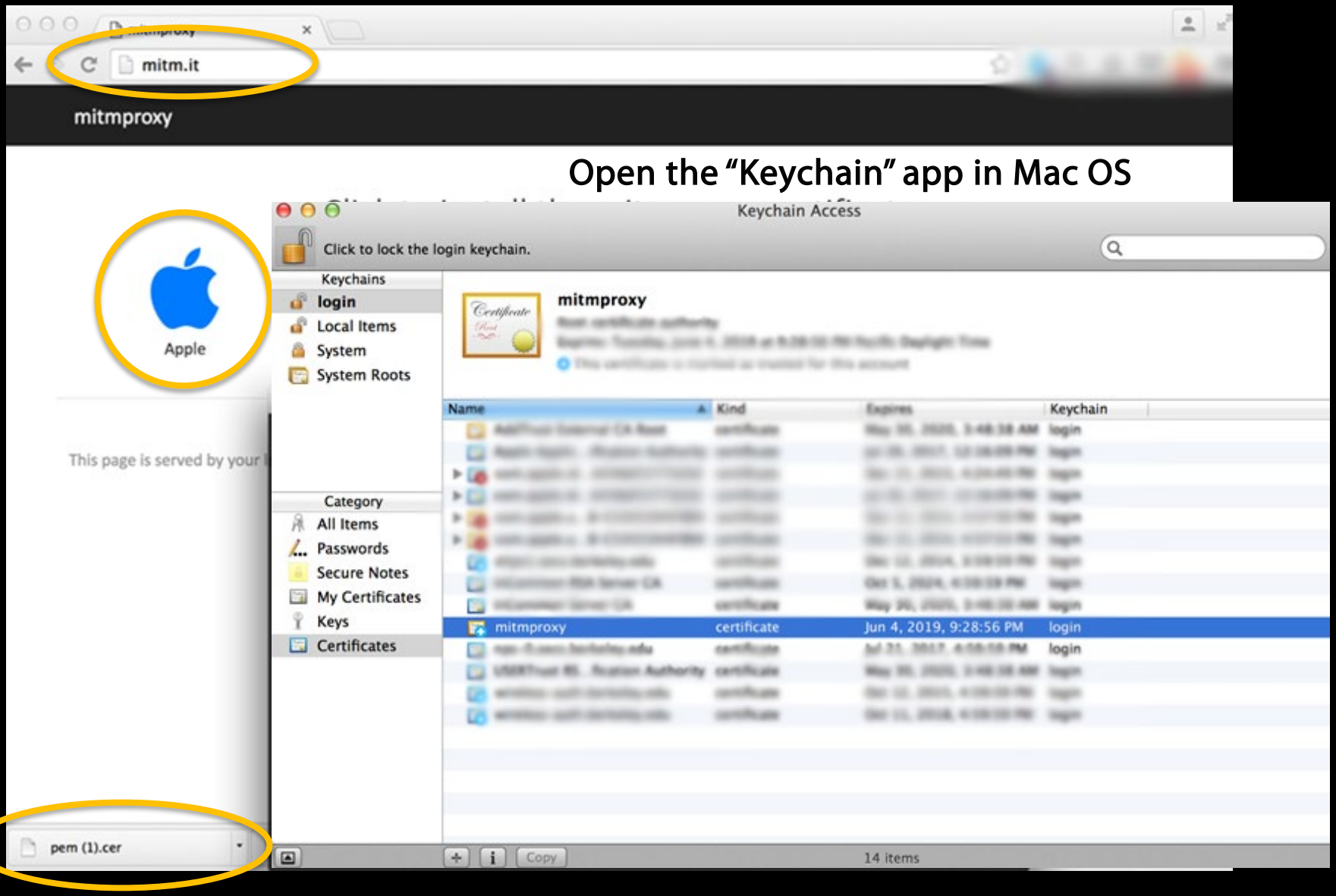
Install the CA System of mitmproxy



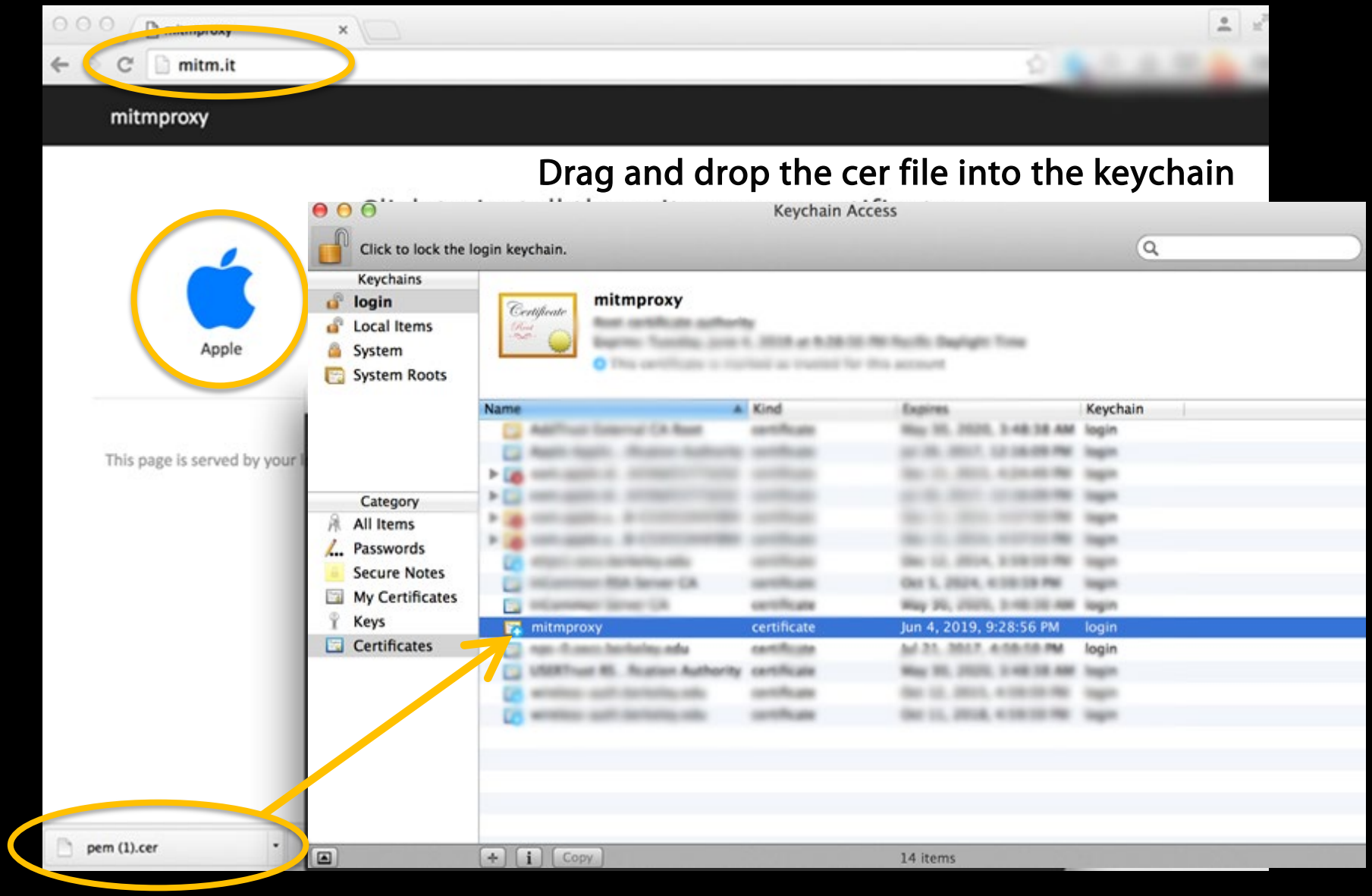
Install the CA System of mitmproxy



Install the CA System of mitmproxy



Install the CA System of mitmproxy



Other Resources

Jalangi (v2) Github

https://github.com/Samsung/jalangi2

DLint + JITProf Github based on Jalangi (v2)

https://github.com/ksen007/jalangi2analyses

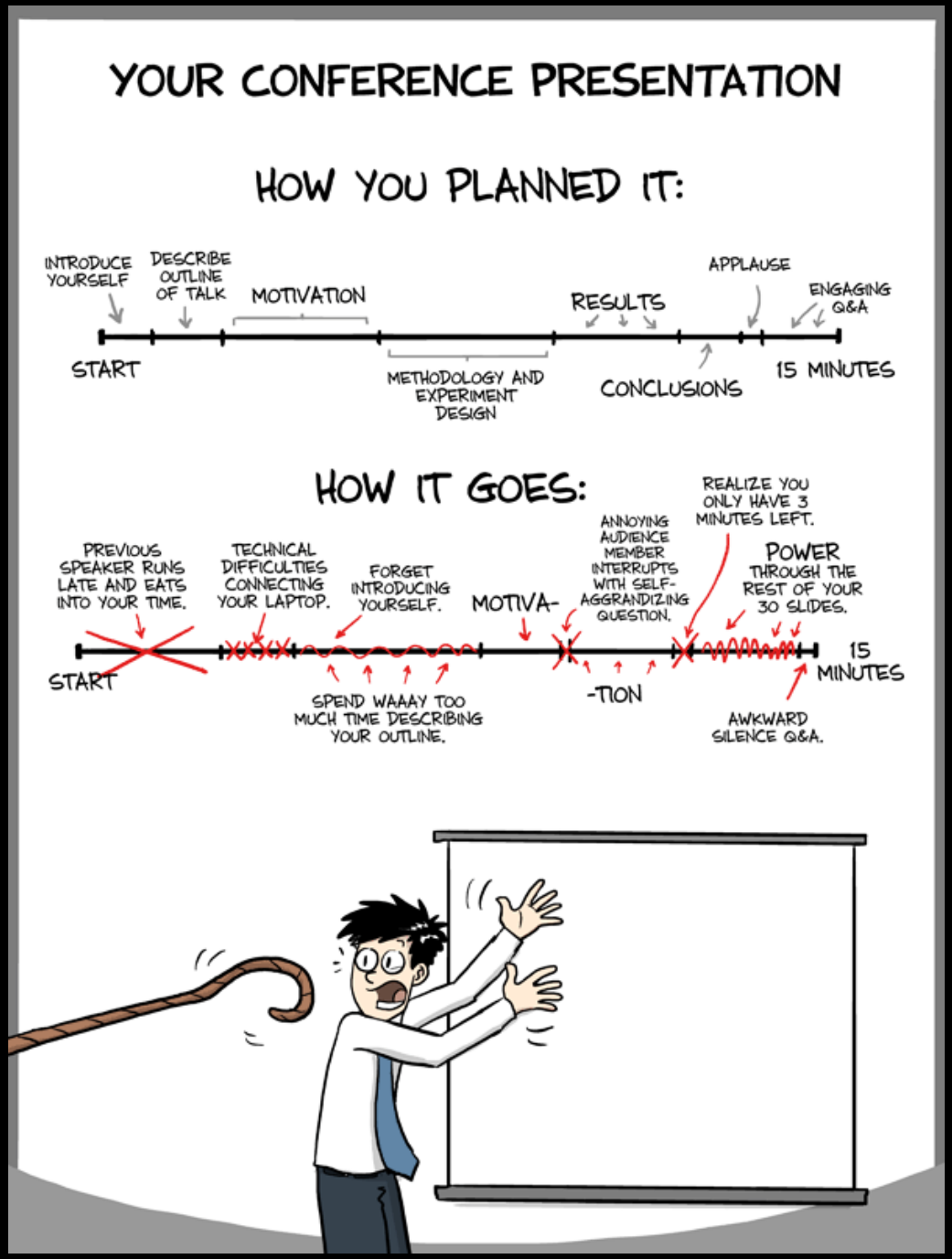
JITProf Visualization Github based on Jalangi (v2)

https://github.com/JacksonGL/jitprof-visualization

Questions ?

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.


Thank You
감사합니다
Dank Je
Blagodaram
Ngiyabonga
Dziekuje
Juspaxar
நன்றி
Ua Tsaug Rau Koj
Děkuji
Suksama
Rahmat
Nuwun
Matur
Misaotra
Danke
Danksche
спасибо
köszönöm
Kiitos
Vinaka
Maake
Kam Sah Hammida
Mauruuru
Asante
Shukria
Dhanyavadagalu
Manana
Dankon
Biyan
Chokrane
Arigato
Gracias
Grazas
Nirringrazzjak
Hvala
Welalin
Di Ou Mèsi
Merci
Salamat
Go Raibh Maith Agat
ขอขอบคุณคุณ
Najis Tuke
Eskerrik Asko
Djiere Dieuf
Kop Khun Khap
Kia Ora
cảm ơn bạn
Paldies
Gratias Tibi
Obrigado
Tingki
Mochchakkeram
Grazie
Tack
Taiku
Terima Kasih
Diolch i Chi
Matondo
ありがとう
谢谢
спасибо


YOUR CONFERENCE PRESENTATION
HOW YOU PLANNED IT:
INTRODUCE YOURSELF
DESCRIBE OUTLINE OF TALK
MOTIVATION
RESULTS
APPLAUSE
ENGAGING Q&A
START
METHODOLOGY AND EXPERIMENT DESIGN
CONCLUSIONS
15 MINUTES
HOW IT GOES:
PREVIOUS SPEAKER RUNS LATE AND EATS INTO YOUR TIME.
TECHNICAL DIFFICULTIES CONNECTING YOUR LAPTOP.
FORGET INTRODUCING YOURSELF.
MOTIVA-
ANNOYING AUDIENCE MEMBER INTERRUPTS WITH SELF-AGGRANDIZING QUESTION.
REALIZE YOU ONLY HAVE 3 MINUTES LEFT.
POWER THROUGH THE REST OF YOUR 30 SLIDES.
START
SPEND WAAAY TOO MUCH TIME DESCRIBING YOUR OUTLINE.
-TION
AWKWARD SILENCE Q&A.
15 MINUTES

Rule #5: Use Contiguous Keys for Array

```
var array = [];
for (var i=10000;i>=0;i--){
    array[i] = i;
}
```

Rule #5: Use Contiguous Keys for Array

```
var array = [];
for (var i=10000;i>=0;i--){
    array[i] = i;
}
```

```
array[10000] = 10000;
array[9999] = 9999;
   ...
```

- non-contiguous array
- To save memory, JIT-engine decides to represent the array with slow data structures like hash table.

Rule #5: Use Contiguous Keys for Array

```
var array = [];
for (var i=10000;i>=0;i--){
    array[i] = i;
}
```

```
for (var i=0;i<=10000;i++){
    array[i] = i;
}
```

10X+ speedup!

Rule #5: Use Contiguous Keys for Array

```
var array = [];
for (var i=10000;i>=0;i--){
loc1: array[i] = i;
}
```

- **Intercept *putField* operation of arrays**
- **Rank locations by number assignments to non-contiguous arrays**

(*)means smaller is better	group	average	improve rate
sunspider-chrome-sha1 (*)	original	1884.7588	26.3%
	refactored	1299.0706	
octane-firefox-Splay	original	11331.59	3.5%
	refactored	12198.65	
Sunspider-String-Tagcloud (*)	original	9178.76	11.7%
	refactored	9457.53	
octane-firefox-DeltaBlue	original	28473.53	1.4%
	refactored	31154.06	
octane-chrome-Box2D	original	24569.47	7.5%
	refactored	24915.00	
octane-chrome-RayTrace	original	43595.94	12.9%
	refactored	48140.35	

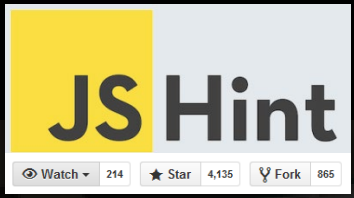
higher → better

(*)means smaller is better	group	average	improve rate
octane-chrome-Splay	**original**	10278.59	15.1%
	refactored	11885.71	
octane-chrome-SplayLatency	**original**	20910.24	3.8%
	refactored	21994.82	
sunspider-chrome-3d-Cube (*)	**original**	597.047059	1.1%
	refactored	593.744118	
sunspider-firefox-sha1 (*)	**original**	680.476471	3.3%
	refactored	669.932353	
sunspider-firefox-Xparb (*)	**original**	364.6824	19.7%
	refactored	357.2235	
sunspider-chrome-md5 (*)	**original**	774.3500	24.6%
	refactored	665.8382	
sunspider-chrome-format-tofte (*)	**original**	212.2029	3.4%
	refactored	200.9000	

higher → better

DLint: Dynamically Checking Bad Coding Practices in JavaScript

Liang Gong, Michael Pradel, Manu Sridharan, Koushik Sen [ISSTA'15]



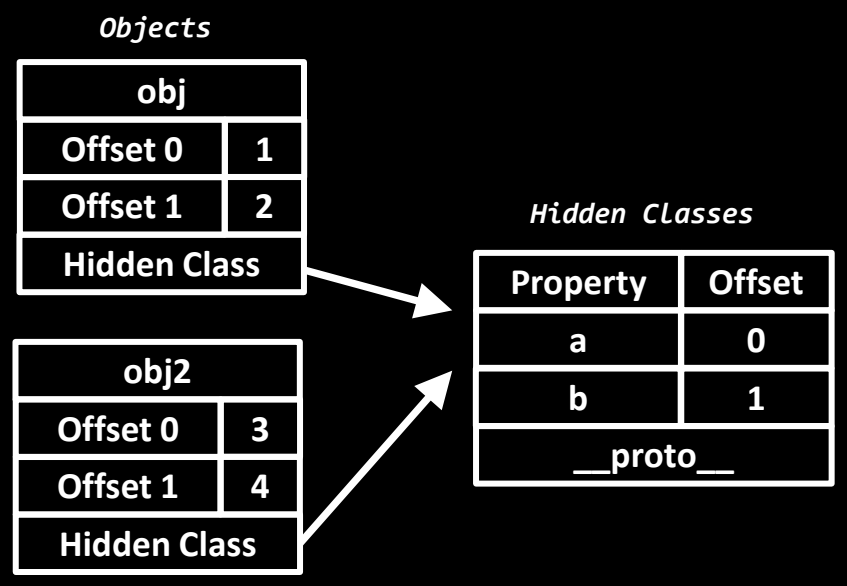
- Designed and Implemented in 10 days
- Not all decisions were well-thought
- Problematic language features
  - Error prone
  - Poor performance
  - Prone to security vulnerabilities
- Problematic features are still around
  - Backward compatibility

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Hidden Class

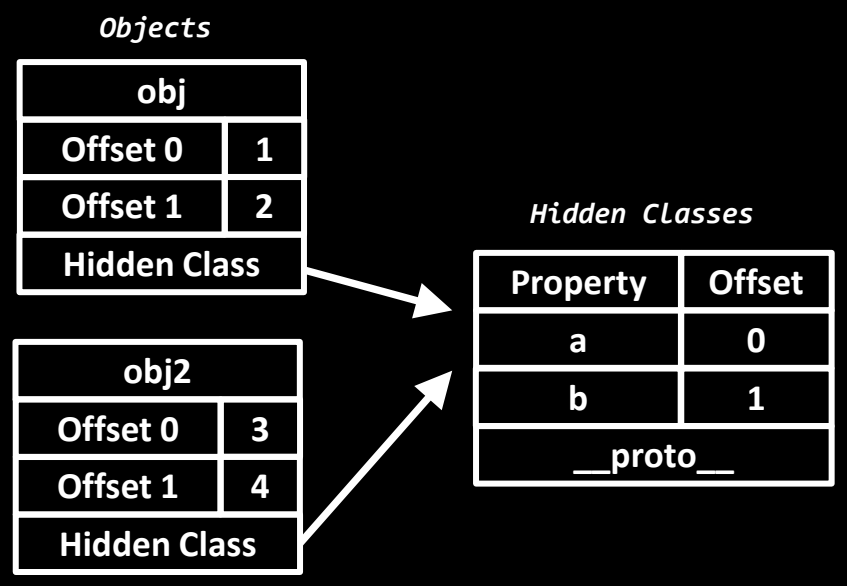
```
var obj = {a:1,b:2}
var obj2 = {a:3,b:4}
```

Map *in V8*
Shape *in SpiderMonkey*
Structure *in JavaScriptCore*



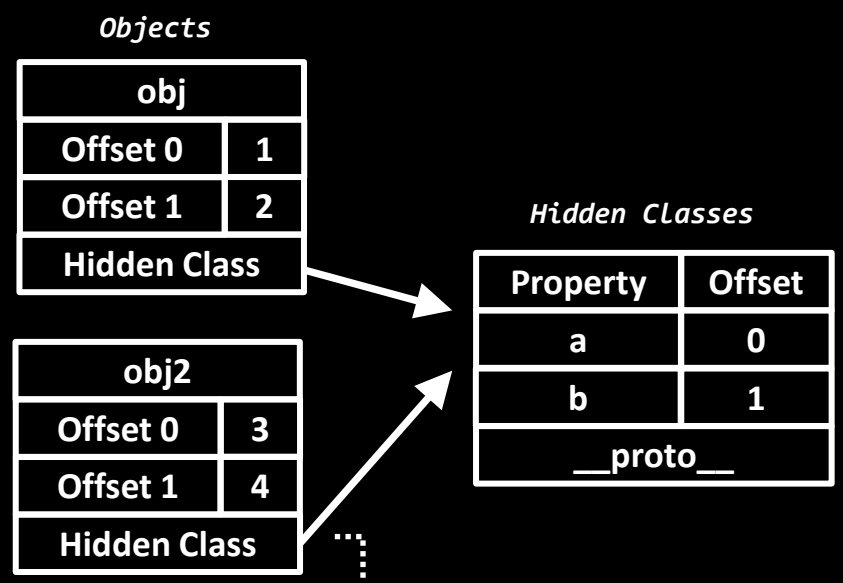
Hidden Class

```
var obj = {a:1,b:2}
var obj2 = {a:3,b:4}

function getA(o){
    return o.a;
}
getA(obj);
```



Hidden Class + Inline Caching

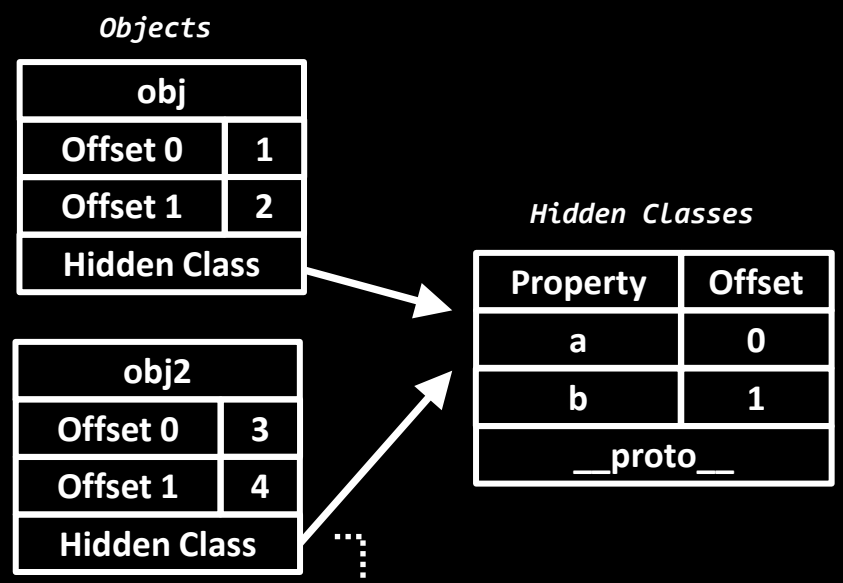
```
var obj = {a:1,b:2}
var obj2 = {a:3,b:4}

function getA(o){
    return o.a;
}
getA(obj);
```




```
function getA(o){
    if(o is an object &&
       o.hiddenclass == cached_hiddenclass)
        return o[cached_a_offset];
    else{
        // jump to V8 runtime
    }
}
```

Hidden Class + Inline Caching

```
var obj = {a:1,b:2}
var obj2 = {a:3,b:4}

function getA(o){
    return o.a;
}
getA(obj);
```

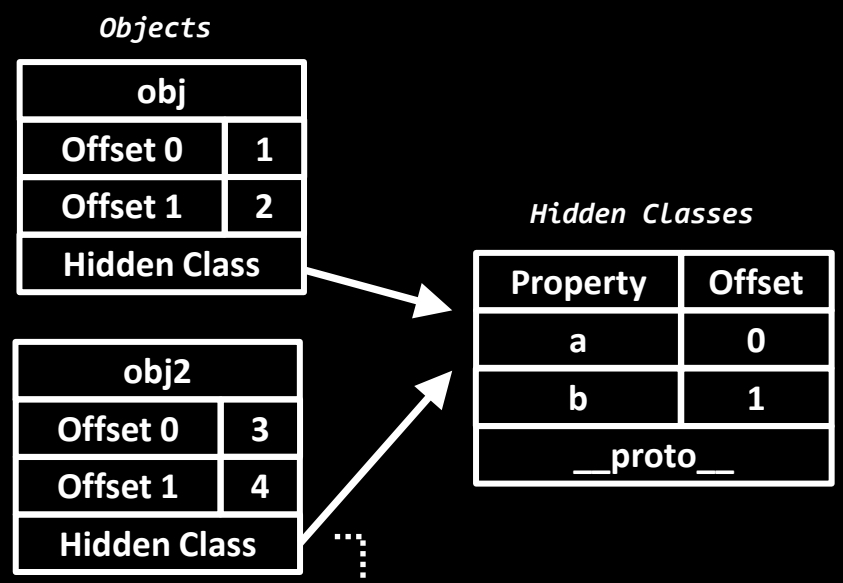



```
function getA(o){
    if(o is an object &&
       o.hiddenclass == cached_hiddenclass)
        return o[cached_a_offset];
    else{
        // jump to V8 runtime
    }
}
```

An inline cache hit

Hidden Class + Inline Caching

```
var obj = {a:1,b:2}
var obj2 = {a:3,b:4}

function getA(o){
    return o.a;
}
getA(obj);
```




```
function getA(o){
    if(o is an object &&
       o.hiddenclass == cached_hiddenclass)
        return o[cached_a_offset];
    else{
        // jump to V8 runtime
    }
}
```

An inline cache hit

An inline cache miss

Hidden Class + Inline Caching

- ***A monomorphic inline cache hit requires 3-10 instructions,***
- ***while an inline cache miss requires 1000 ~ 4000 instructions*** [1]

```
function getA(o){
    if(o is an object &&
       o.hiddenclass == cached_hiddenclass)
        return o[cached_a_offset];
    else{
        // jump to V8 runtime
    }
}
```

An inline cache hit

An inline cache miss

[1] ***Wonsun Ahn et al. PLDI '14***