Type-Based Capability for Java

Xi Wu, Yi Lu, Ian J. Hayes and Larissa A. Meinicke

The University of Queensland
Oracle Labs, Australia

Sydney November 2017

Under the ARC Linkage Project with Oracle Labs, Australia

Outline

- An Overview of Capabilities for Java
- Motivation
- Ongoing Work
- Summary and Future Direction

Java Security Issues

- doPrivileged blocks
  - wide privileges granted to the Java Class Library
  - code can run with fewer restrictions

Java Security Issues

- doPrivileged blocks
  - wide privileges granted to the Java Class Library
  - code can run with fewer restrictions
- Subclassing privileged classes
  - unprivileged subclasses of privileged classes
  - overriding existing methods with rogue code

Java Security Issues

- doPrivileged blocks
  - wide privileges granted to the Java Class Library
  - code can run with fewer restrictions
- Subclassing privileged classes
  - unprivileged subclasses of privileged classes
  - overriding existing methods with rogue code
- Privileged access escape
  - access to a privileged object escapes to unauthorized domains

Java Security Issues

- doPrivileged blocks
  - wide privileges granted to the Java Class Library
  - code can run with fewer restrictions
- Subclassing privileged classes
  - unprivileged subclasses of privileged classes
  - overriding existing methods with rogue code
- Privileged access escape
  - access to a privileged object escapes to unauthorized domains
- Caller-sensitive methods
  - depend on the privileges of the class loader of the caller

How to provide a more secure access to resources for Java, with the aim of preventing security flaws

Philosophy behind Capabilities in Java

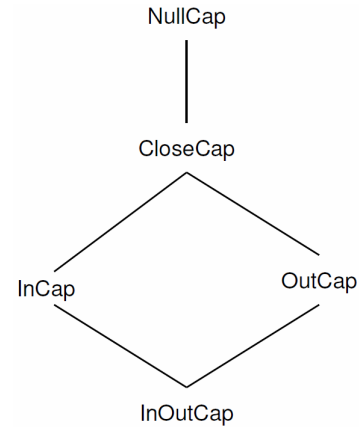
- All access to resources given by explicit "capabilities"
- An object with a restricted interface
  - a set of operations that can be invoked
  - encapsulate what one can do with a resource
- Permission checking done when a capability created
  - access the resource via methods of the capability
  - no further permission checking is required

Reference

- Ian J. Hayes, Xi Wu and Larissa A. Meinicke.: Capabilities for Java: Secure Access to Resources. In: Proc. 15th Asian Symposium on Programming Languages and Systems, APLAS 2017. pp 67-84.

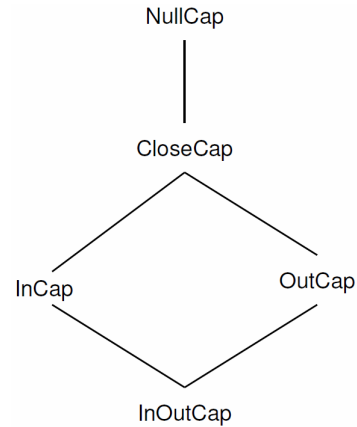
Running Example: File Input and Output Streams

```
capability CloseCap { // Implicitly extends NullCap
    void close();
}
capability InCap extends CloseCap {
    int read();
    int read(byte[] b);
    int read(byte[] b, int off, int len);
    long skip(long n);
}
capability OutCap extends CloseCap {
    void write(int b);
    void write(byte[] b);
    void write(byte[] b, int off, int len);
    void flush();
}
capability InOutCap extends InCap, OutCap {
    // Only the methods from InCap and OutCap.
    // Note that InCap and OutCap share method close.
}
```



Running Example: File Input and Output Streams

```
capability CloseCap { // Implicitly extends NullCap
    void close();
}
capability InCap extends CloseCap {
    int read();
    int read(byte[] b);
    int read(byte[] b, int off, int len);
    long skip(long n);
}
capability OutCap extends CloseCap {
    void write(int b);
    void write(byte[] b);
    void write(byte[] b, int off, int len);
    void flush();
}
capability InOutCap extends InCap, OutCap {
    // Only the methods from InCap and OutCap.
    // Note that InCap and OutCap share method close.
}
```



Inherit from the empty capability **NullCap**

- methods inherited from class object are disallowed unless explicitly included with restrictions

Generating Capabilities

Generating Capabilities

- Capability Manager

Generating Capabilities

- Capability Manager

```
capability FileAccessCap {
    InCap requestInCap(String name)
        throws FileNotFoundException, SecurityException;
    OutCap requestOutCap(String name)
        throws FileNotFoundException, SecurityException;
    InOutCap requestInOutCap(String name)
        throws FileNotFoundException, SecurityException;
}
```

Generating Capabilities

- Capability Manager

```
capability FileAccessCap {
    InCap requestInCap(String name)
        throws FileNotFoundException, SecurityException;
    OutCap requestOutCap(String name)
        throws FileNotFoundException, SecurityException;
    InOutCap requestInOutCap(String name)
        throws FileNotFoundException, SecurityException;
}
```

```
class RandomAccessFileManager implements FileAccessCap {
    RandomAccessFileManager() { super(); }
    /* Factory method for InCap */
    public InCap requestInCap(String name)
        throws SecurityException, FileNotFoundException {
        SecurityManager sm = System.getSecurityManager();
        if( sm != null ) {
          sm.checkPermission(new FilePermission(name, "read"));
        }
        return capability (InCap) new RandomAccessFile(name);
    }
```

Capabilities Escape to Untrusted Code

```
capability FileAccessCap {
   InCap requestInCap (String name)
      throws FileNotFound Exception, SecurityException;
}

class RandomAccessFileManager implements FileAccessCap {
  RandomAccessFileManager() { }
  public InCap requestInCap (String name)
       throws SecurityException, FileNotFoundException {
  SecurityManager sm = System.getSecurityManager();
  If (sm != null) {
     sm.checkPermission (new FilePermission (name, “read”));
   }
  return capability (InCap) new RandomAccessFile (name);
}
```

Capabilities Escape to Untrusted Code

```
public class A {
  public static void main (String[] args)
        throws Exception {
    FileAccessCap f = ...;
    B b = ...;
    InCap in = f.requestInCap (fileName);
    b.use(in);
  }
}

public class B {
  ... ...
  public void use (InCap in) { ... ... }
}
```

```
capability FileAccessCap {
   InCap requestInCap (String name)
      throws FileNotFound Exception, SecurityException;
}

class RandomAccessFileManager implements FileAccessCap {
  RandomAccessFileManager() { }
  public InCap requestInCap (String name)
       throws SecurityException, FileNotFoundException {
  SecurityManager sm = System.getSecurityManager();
  If (sm != null) {
     sm.checkPermission (new FilePermission (name, “read”));
   }
  return capability (InCap) new RandomAccessFile (name);
}
```

Capabilities Escape to Untrusted Code

```
public class A {
  public static void main (String[] args)
        throws Exception {
    FileAccessCap f = ...;
    B b = ...;
    InCap in = f.requestInCap (fileName);
    b.use(in);
  }
}

public class B {
  ... ...
  public void use (InCap in) { ... ... }
}
```

Class	Permission
RandomAccess FileManager	All Permission
A	Permission(“read”)
B	Φ

```
capability FileAccessCap {
   InCap requestInCap (String name)
      throws FileNotFound Exception, SecurityException;
}

class RandomAccessFileManager implements FileAccessCap {
  RandomAccessFileManager() { }
  public InCap requestInCap (String name)
       throws SecurityException, FileNotFoundException {
  SecurityManager sm = System.getSecurityManager();
  If (sm != null) {
     sm.checkPermission (new FilePermission (name, “read”));
   }
  return capability (InCap) new RandomAccessFile (name);
}
```

Capabilities Escape to Untrusted Code

```
public class A {
  public static void main (String[] args)
        throws Exception {
    FileAccessCap f = ...;
    B b = ...;
    InCap in = f.requestInCap (fileName);
    b.use(in);
  }
}
```

```
public class B {
  ... ...
  public void use (InCap in) { ... ... }
}
```

Class	Permission
RandomAccess FileManager	All Permission
A	Permission(“read”)
B	Φ

```
capability FileAccessCap {
   InCap requestInCap (String name)
      throws FileNotFound Exception, SecurityException;
}
```

```
class RandomAccessFileManager implements FileAccessCap {
  RandomAccessFileManager() { }
  public InCap requestInCap (String name)
       throws SecurityException, FileNotFoundException {
  SecurityManager sm = System.getSecurityManager();
  If (sm != null) {
     sm.checkPermission (new FilePermission (name, “read”));
   }
  return capability (InCap) new RandomAccessFile (name);
}
```

Type-based Capability

Type-based Capability

- Attempt to solve
  - capabilities obtained by trusted code may be received by untrusted code

Type-based Capability

- Attempt to solve
  - capabilities obtained by trusted code may be received by untrusted code
- Avoid dynamic permission check
  - regarding capabilities as types
  - proper use of capabilities by type checking

Type-based Capability

- Attempt to solve
  - capabilities obtained by trusted code may be received by untrusted code
- Avoid dynamic permission check
  - regarding capabilities as types
  - proper use of capabilities by type checking
- Capabilities as permissions
  - grant to code by user-defined policy files
  - restrict capabilities to only authorised code

Security Goal

Security Goal

$$\text{access}(\text{code}, \text{cap}) \Rightarrow \text{grant}(\text{code}, \text{cap})$$

- access(code, cap): code uses the capability cap
- grant(code, cap): code is granted the capability cap by user

Security Goal

$$\text{access}(\text{code}, \text{cap}) \Rightarrow \text{grant}(\text{code}, \text{cap})$$

- access(code, cap): code uses the capability cap
- grant(code, cap): code is granted the capability cap by user
  - Transitivity:

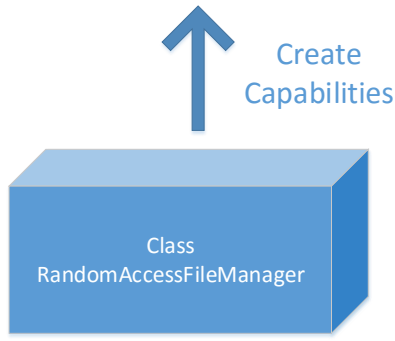
    $\text{grant(code, cap2)} \wedge \text{cap1} <: \text{cap2} \Rightarrow \text{grant(code, cap1)}$
  - cap1 $<:$ cap2: is satisfied if cap2 is more privileged than cap1
  - relation $<:$ is opposite of the standard Java subset relation
  - e.g., InCap $<:$ InOutCap and InCap $\not<:$ OutCap

Revisited Capabilities Escape

```
public class A {
 public static void main (String[] args)
       throws Exception {
   FileAccessCap f = ...;
   B b = ...;
   InCap in = f.requestInCap (fileName);
   b.use(in);
  }
}

public class B {
 ... ...
 public void use (InCap in) { ... ... }
}
```

```
capability FileAccessCap {
   InCap requestInCap (String name)
      throws FileNotFound Exception, SecurityException;
}
```



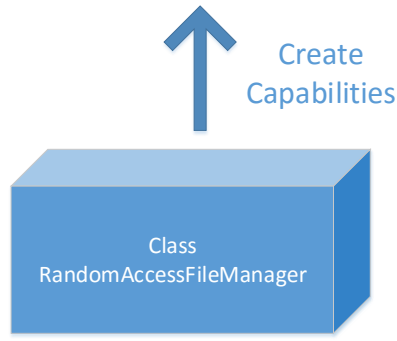
Revisited Capabilities Escape

```
public class A {
 public static void main (String[] args)
       throws Exception {
   FileAccessCap f = ...;
   B b = ...;
   InCap in = f.requestInCap (fileName);
   b.use(in);
  }
}
```

```
public class B {
 ... ...
 public void use (InCap in) { ... ... }
}
```

Class	Capability
A	FileAccessCap InCap
B	NullCap

```
capability FileAccessCap {
   InCap requestInCap (String name)
      throws FileNotFound Exception, SecurityException;
}
```

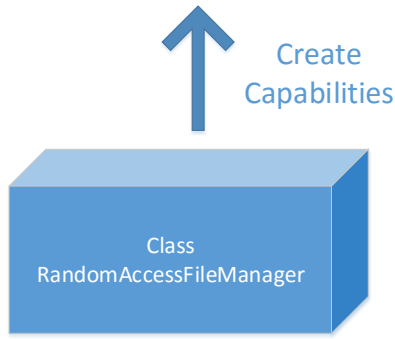


Revisited Capabilities Escape

```
@grant{FileAccessCap, InCap}
public class A {
 public static void main (String[] args)
       throws Exception {
   FileAccessCap f = ...;
   B b = ...;
   InCap in = f.requestInCap (fileName);
   b.use(in);
 }
}

@grant{NullCap}
public class B {
 ... ...
 public void use (InCap in) { ... ... }
}
```

```
capability FileAccessCap {
   InCap requestInCap (String name)
      throws FileNotFound Exception, SecurityException;
}
```

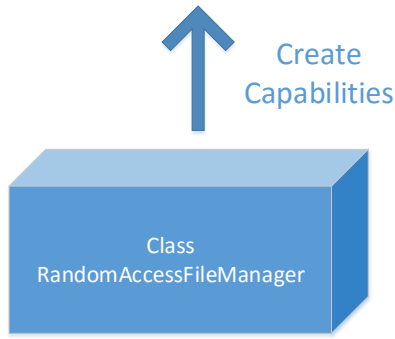


Revisited Capabilities Escape

```
@grant{FileAccessCap, InCap}
public class A {
 public static void main (String[] args)
       throws Exception {
   FileAccessCap f = ...;
   B b = ...;
   InCap in = f.requestInCap (fileName);
   b.use(in);
 }
}

@grant{NullCap}
public class B {
 ... ...
 public void use (InCap in) { ... ... }
}
```

```
capability FileAccessCap {
   InCap requestInCap (String name)
      throws FileNotFound Exception, SecurityException;
}
```



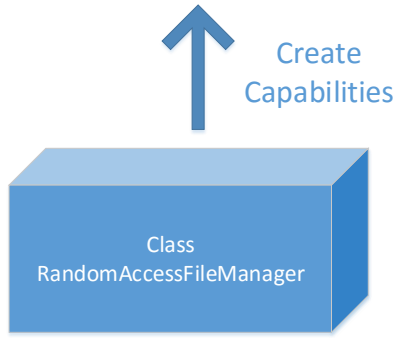
$$access(B,InCap) \Rightarrow grant(B,InCap)$$

Revisited Capabilities Escape

```
@grant{FileAccessCap, InCap}
public class A {
 public static void main (String[] args)
       throws Exception {
   FileAccessCap f = ...;
   B b = ...;
   InCap in = f.requestInCap (fileName);
   b.use(in);
 }
}

@grant{NullCap}
public class B {
 ... ...
 public void use (InCap in) { ... ... }
}
```

```
capability FileAccessCap {
   InCap requestInCap (String name)
      throws FileNotFound Exception, SecurityException;
}
```



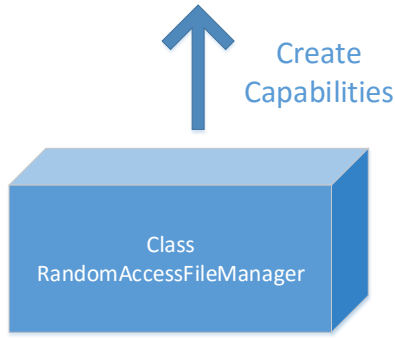
$access(B,InCap) \Rightarrow grant(B,InCap)$ ✗

Revisited Capabilities Escape

```
@grant{FileAccessCap, InCap}
public class A {
 public static void main (String[] args)
       throws Exception {
   FileAccessCap f = ...;
   B b = ...;
   InCap in = f.requestInCap (fileName);
   b.use(in);
 }
}

@grant{InOutCap}
public class B {
 ... ...
 public void use (InCap in) { ... ... }
}
```

```
capability FileAccessCap {
   InCap requestInCap (String name)
      throws FileNotFound Exception, SecurityException;
}
```



$access(B,InCap) \Rightarrow grant(B,InCap)$

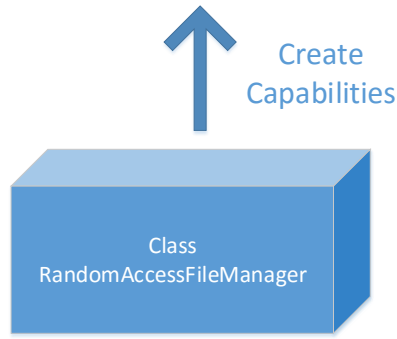
$grant(B,InOutCap) \wedge InCap <: InOutCap \Rightarrow grant(B,InCap)$

Revisited Capabilities Escape

```
@grant{FileAccessCap, InCap}
public class A {
 public static void main (String[] args)
       throws Exception {
   FileAccessCap f = ...;
   B b = ...;
   InCap in = f.requestInCap (fileName);
   b.use(in);
 }
}

@grant{InOutCap}
public class B {
 ... ...
 public void use (InCap in) { ... ... }
}
```

```
capability FileAccessCap {
   InCap requestInCap (String name)
      throws FileNotFound Exception, SecurityException;
}
```



$access(B,InCap) \Rightarrow grant(B,InCap)$ ✔

$grant(B,InOutCap) \wedge InCap <: InOutCap \Rightarrow grant(B,InCap)$

Summary and Future Directions

Summary and Future Directions

- Summary
  - prevent capabilities from escaping to unauthorised code
  - security goal can be enforced statically by type system

Summary and Future Directions

- Summary
  - prevent capabilities from escaping to unauthorised code
  - security goal can be enforced statically by type system
- Future Direction
  - Capabilities as Module Dependency
    - applies capabilities on describing dependency in module system

Summary and Future Directions

- Summary
  - prevent capabilities from escaping to unauthorised code
  - security goal can be enforced statically by type system

- Future Direction
  - Capabilities as Module Dependency
    - applies capabilities on describing dependency in module system
  - Properties from Object-Capability and Design Patterns
    - describes object-capability properties and design patterns

Summary and Future Directions

- Summary
  - prevent capabilities from escaping to unauthorised code
  - security goal can be enforced statically by type system

- Future Direction
  - Capabilities as Module Dependency
    - applies capabilities on describing dependency in module system
  - Properties from Object-Capability and Design Patterns
    - describes object-capability properties and design patterns
  - Parameterization
    - specifies the specific file names that the code with capabilities can access

capability FileAccessCap (filename) {.....}

sm.checkPermission (new FilePermission (filename, “read”));

Thanks.

Questions?