Capabilities by Example (in C#)

Assumptions

1. ***No networking:*** SocketPermission, WebPermission, etc. all denied
2. ***No file system:*** FileIOPermission, etc. all denied
3. ***No static class fields:*** public static object SomeObject; is forbidden

Two Conspirators and a Secret

```
public class Mole {
  DirtySecret secret;
  public DirtySecret Payoff() {
    return secret;
  }
  public void LeakSecret() {
    //attempt to gain access to a
    //Tabloid and leak a secret
    Tabloid paper = ...? //unfinished
    paper.Publish( secret );
  }
}
```

```
public class Tabloid {
  TextWriter headline;
  public void Publish(DirtySecret s) {
    headline.Write( s );
  }
  public void DigForInfo() {
    //attempt to gain access to a Mole
    //to obtain a secret
    Mole mole = ...? //unfinished
    Publish( mole.Payoff() );
  }
}
```

```
public class DirtySecret {
  //holds my dirty little secret...
}
```

Let the Conspiracy Begin

```
//'enquirer' is a capability to an instance of Tabloid
public void AttemptPressLeak(Tabloid enquirer)
{
  //'secret' is a capability to a juicy tidbit that would devastate my fame
  DirtySecret secret = new DirtySecret( "I play with Barbies" );

  //'mole' is a capability to an instance of a Mole that wants to leak my secret
  Mole mole = new Mole(secret);

  //Tabloid and Mole concurrently try to access each other to leak my secret
  new Thread( enquirer.DigForInfo ).Start();
  new Thread( mole.LeakSecret ).Start();
}
```

The Challenge

How can the Mole leak our dirty secret to Tabloid?

Remember: no file system, no networking, no static fields, etc.

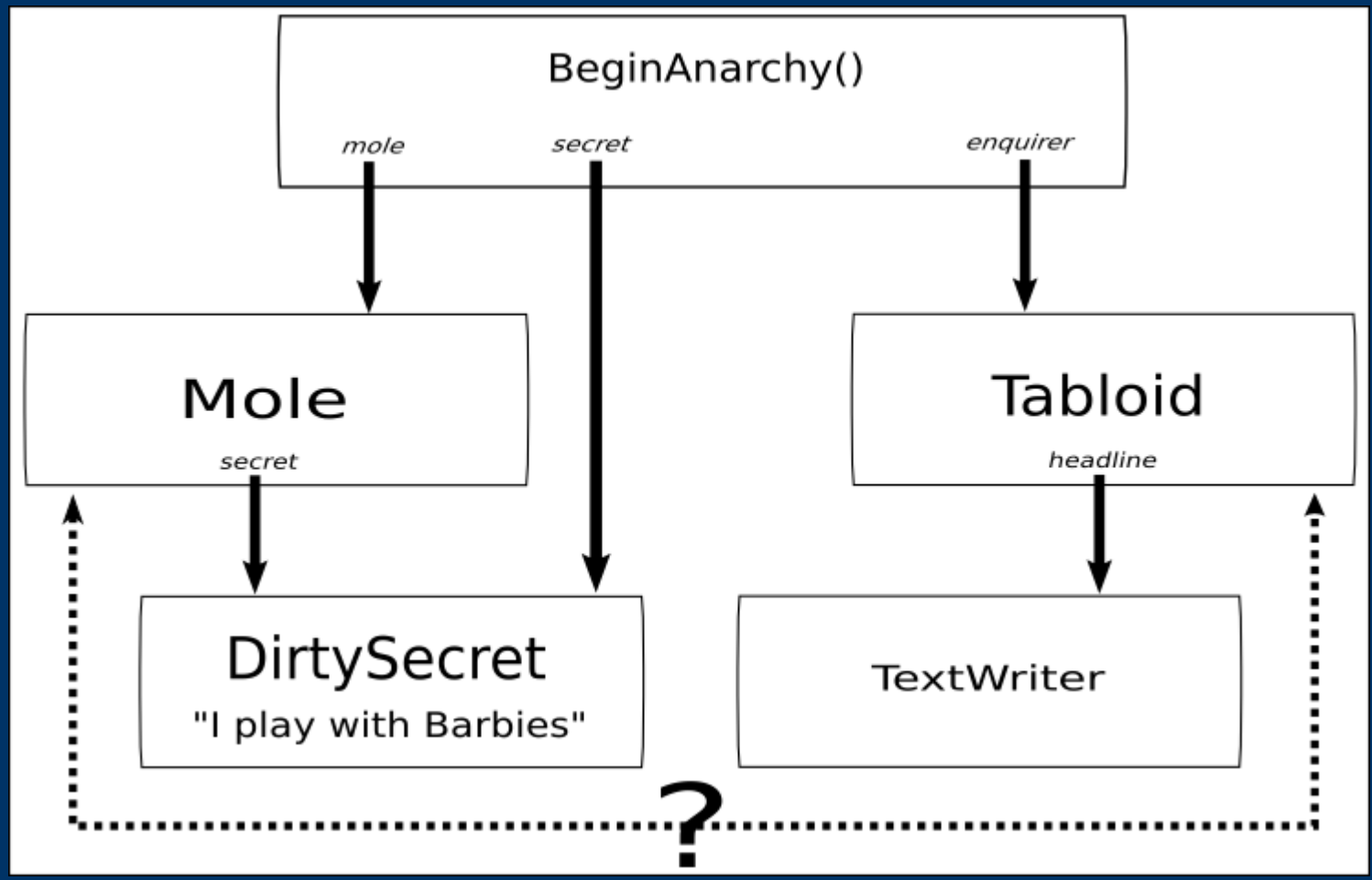
Answer

It can't.

Why Not?

Because there are no shared references between Mole and Tabloid via which Mole could leak our dirty secret.

Reference Graph



Reference Graph Defines Access Restrictions

- Mole can't just "create" a reference to Tabloid to leak our secret.
- Tabloid can't somehow manufacture a reference to our secret or to Mole out of thin air.

So...?

Our secret is safe, even in the presence of a Mole!

So what is a capability?

A capability is an object reference (in C#).

```
//'secret' is a capability to our dirty secret
DirtySecret secret = new DirtySecret( "I play with Barbies" );
```

What is capability-based security?

An architecture that uses only capabilities for access control.

In other words, only objects and object references.

How do we grant permissions?

Simple parameter passing and assignment!

```
//copy capability 'mole' to 'mole2'
object mole2 = mole;

//give the Enquirer the same capability to the secret that the mole has
enquirer.Secret = mole.Payoff();

//call 'someFunction' and pass in a capability/reference to the Mole
someFunction(mole);
```

In other words, programming with capabilities is just normal OO programming!

Capabilities Encapsulate Permissions

A reference carries an implicit permission to cause side-effects to that object via its methods, the permission to pass it around, etc.

If you hold a reference in C#, there is nothing stopping you from invoking methods via that reference *.

* except the intrusive and useless Code-Access Security model

Permissions and Functionality

If a function needs the ability to do something, such as write to a file, then it also needs the permission to do it!

Since permission and functionality are inextricably linked, why not just bundle them together?

Denying Permission

If you don't want to grant permission to do X, then just don't provide a reference to an object that can do X!

Gaining Authority

One can gain authorities in ***only*** two ways with capabilities:

1. Communication
2. Construction

Gaining Authority-Communication

Communication: being given an object reference is granting a capability to the object

```
enquirer.Secret = secret; //assignment is capability copy
someFunction(secret); //pass in capability to function
```

Gaining Authority-Construction

Creating an object yields a capability to the new object, and can provide the new object with its own capabilities.

```
//construct a new Tabloid, granting it a capability to a TextWriter, and
//assign the capability to the new Tabloid to the 'enquirer' variable
Tabloid enquirer = new Tabloid( headlineWriter );
```

Capability-secure programming

All authority is encapsulated in references/capabilities, and all authority can only be communicated by references/capabilities.

Is C# capability-secure?

Since C# object references are capabilities, does that mean C# is capability-secure?

C# is NOT capability-secure

NO.

(but a fully functional subset is)

Why not capability secure?

The example specifically excluded:

1. file system
2. networking
3. static class fields

Those APIs and features are sources of "ambient authority".

Ambient Authority

"Ambient Authority" is a source of authority that does not derive from capabilities.

In other words, ambient authority permits any code to manufacture a capability and increase its authority without being ***explicitly given*** the capability to do so.

Example of Ambient Authority

```
Stream stream = new FileStream( @"C:\AUTOEXEC.BAT" );
```

I just turned a capability to a string into a capability to a Windows startup script!

Wow!

Ambient Authority Bypasses Capability Security

With ambient authority, Tabloid and Mole could have opened a socket or a file, or simply declared a static class field, and leaked my secret that way.

You've already learned this lesson

Ambient authority is the security equivalent of global variables in ordinary programming.

In other words, use sparingly ***if ever***.

What does that mean exactly?

It means that Tabloid and Mole can't just open sockets or files, and neither can anyone else.

They can only obtain a FileStream or a Socket, if they are explicitly given a FileStream or a Socket reference.

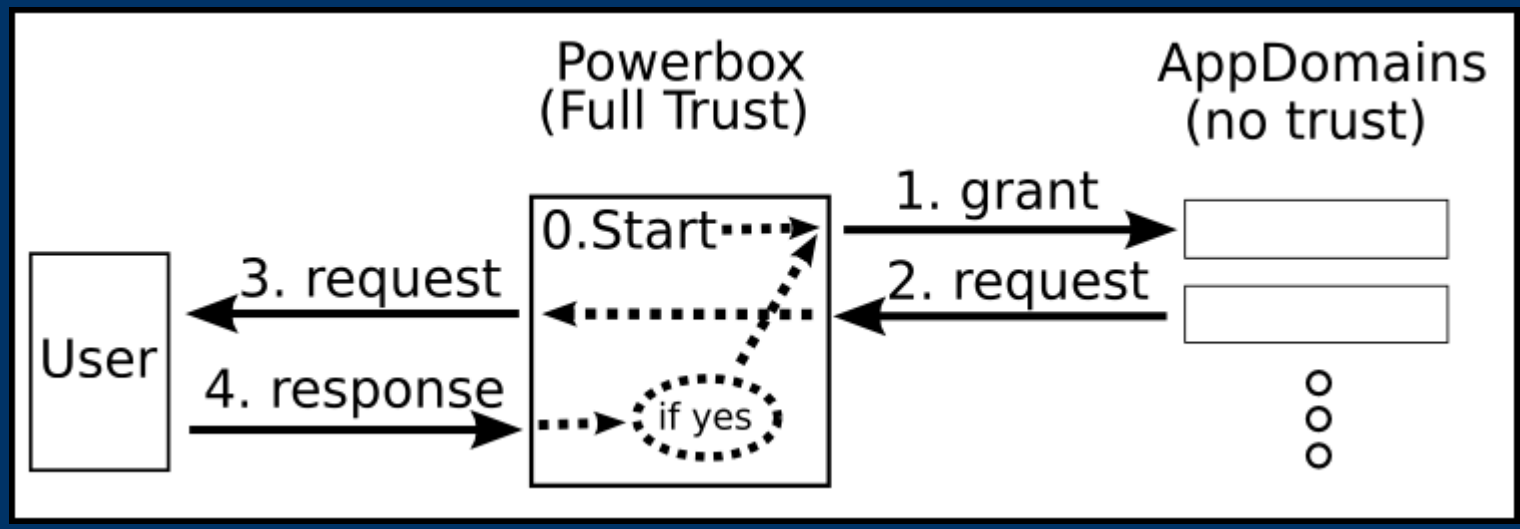
Chicken and the Egg?

So if no one can create a Socket or a FileStream, how are these objects ever created?

The Powerbox Pattern

The Powerbox is a small, trusted and audited code module that has FileIOPermission, WebPermission, etc. and which initializes the rest of the application with the appropriate references/capabilities.

Simplified Powerbox



User Requests are Unspoofable

The user trusts the Powerbox, so the Powerbox ensures that all communication with the user cannot be spoofed *.

* See the “DarpaBrowser” report, and “How Emily Tamed the Caml” for anti-spoofing via the Powerbox

CapDesk and the DarpaBrowser

CapDesk is a capability-secure desktop running on the Java VM.

The DarpaBrowser is a malicious web browser launched from CapDesk, that cannot harm the machine *.

* an intensive external security review confirmed this

Unmatched Security Guarantees

CapDesk and the DarpaBrowser both provide security guarantees that no other platform or technology can match, including unspoofability.

They provide this safety with the same easy interface as Explorer, KDE, Gnome, etc.

All thanks to their capability-secure design.

Capability Security is here!

We already use capabilities! All we have to do is get rid of any “features” that bypass them, such as ambient authority.

"Don't add security, remove insecurity."
~ Mark S. Miller

Small Steps to Capability Security

How to make the .NET VM into a "secure VM" *.

* idea is currently in research stage

Step 1

Trusted "Powerbox" runs in "Full Trust" mode with all permissions enabled.

Step 2

Powerbox launches new AppDomains for potentially malicious code with ALL permissions denied.

Step 3

Provide the new AppDomain only the specific capabilities (MarshalByRefObjects) that you wish to grant it.

Step ... wait, that's it!

There is now no way for code in any malicious AppDomain to gain any authority beyond that which it can access via the MarshalByRefObjects it was given.

Restriction #3 Relaxed

3. no static class fields

Each AppDomain initializes its own copy of static fields, so we can somewhat relax our initial restriction on static class fields when using this technique.

Leaky Libraries

We need to isolate potentially malicious code in separate AppDomains only because of classes that "leak authority".

These are classes that provide ambient authority.

Capability-Secure Classes

If class libraries are designed only with capabilities, and the language followed capability discipline*, then AppDomains would be completely unnecessary!

* like C# without static class fields

Capability-Secure Classes? It's Just Good OO Design

```
//good OO design, and also secure
public void EvilLogger(TextWriter output, string msg)
{
  //can't open any other files, sockets, etc.
  output.WriteLine( “Crap, at best I can only ignore your 'msg' parameter.” );
}

//bad, leaky, “ambient authority” design; bad OO design too
public void EvilLogger(string file, string msg)
{
  using (TextWriter o = new TextWriter(File.Open(@”C:\AUTOEXEC.BAT”)))
  {
    //Hope you like your new paperweight.
    o.WriteLine( “Evil will always win because good is dumb!” );
  }
}
```

Summary

- No viral Code Access Security.
- No more surprising, random and non-sensical SecurityExceptions.
- No need for partial trust modes.
- The VM is vastly simplified by eliminating stack inspection.
- Increased performance, as stack walking is very CPU-intensive and prevents important JIT optimizations.

Conclusion

Golden Rule: If you have a reference to it, you can use it, no muss, no fuss.

Capability security is objects all the way!

P.S.

I don't really play with Barbies... no, really!

References

.NET web-calculus implementation in C#
http://sourceforge.net/projects/web-calculus

Original Java web-calculus implementation and design docs
http://waterken.sourceforge.net/

E capability-secure language and papers
http://www.erights.org/

DarpaBrowser report
http://www.combex.com/papers/darpa-report/index.html

How Emily Tamed the Caml (capability-secure subset of OCaml)
http://www.hpl.hp.com/techreports/2006/HPL-2006-116.pdf