Some clarification about the previous projects:
Design of Taint-droid based project. Professor recommends that to not to place any policy in java VM in that project rather what we can do is we can modify taint droid so that instead of logging the event whenever there is any tainted data, we can throw exception. The app has to handle the exception.

Once we get that thing, the way we get the data out on the network, is to design and implement a class that does not have any public accessible structure. The classifier object will be the object of that class. It is like location provider, the app has to use location manager to get one. The classifier class will have more than one method, which takes an argument as an object. And then makes a duplicate of the arguments on the copy. The application that has classifier object will be send over to the network.

The classifier class, one of which allows you to call at run time. So that would be basic idea how to go about it.
There are two parts of the project, one is the taint droid (one half of the project) and second is user interface (other half of the project). So the other part is to built a trusted user interface, that when get clicked, they grant the app to perform some action and also tell it to perform the action. The way of doing this is that the user interface widget, the app will ask the OS to construct one and give a call back. When the widget calls back the app, it will pass the classifier object. So there is extra call back parameter. If we go deep in this it tells to enforce specific data paths for the classifier object. Like, it gives a classifier object that allows declassifying a type of data. So we get GPS data and send it over to the network but we cannot do anymore with it. The classifier class involves dealing with some of the native code. We can have a classifier class in which we can tell it what type of classifier data it is allowed to declassify and when it is called it will check whether it the data being passed has that bit ON in it. If the declassifying object is constructed to only declassify GPS data, when the app calls it declassify method, it will first check all the data that is being passed is right or not. If it is right, then it will say to copy or add sink permission bit. As this copy is allowed to go to the sink.

Lecture Begins:

Last time we were talking about linear capabilities. We started with hardware capabilities. Then we talked about OS capability.
- Hardware capability (FAT pointers)
- OS capability (File descriptor)
Here capability will be objects. The thing that got us to capability was native client. When we were running untrusted code inside the same address space as trusted code. We wanted to restrict the memory access of the untrusted code. We decided to make calls from untrusted to trusted code. The code still requires copy. So let's see how we can avoid hardware copy.

There are other properties:

1. Unforgeability
2. Possession = permission
3. Possible/communicable
4. Attenuable

We want to construct a weaker capability based on the capabilities we have. As we want to pass out capability to another system that we do not fully trust, rather than giving full access to capability, we give a restricted access to the capability that we want to work on and then it does it work and we get the result. For example, if we have a function that sum up integers and we have read/write pointer to array of integers and now pass to summation function. We do not want the summation function to write to the integers. So we have to restrict the capability. This is an example of attenuation.

What we need to do if we want to use object as capabilities?
- We will use a language Joe-e (java).

A major motivation of capability is

1. **Principle of least privilege**: Each piece of code gets the least amount of power it needs to do its job.

Least privilege is a relative term. If we go back to the example, which adds a bunch of integers, the least privilege is the ability to read those integers. If the hardware does not support the read only pointer, the least privilege means that the hardware can only give the access to write pointers. So can make more and more restrictions to achieve a given task. Someway these privileges are defined relative to how expressive your system can express privileges.

So when we are facing the issue of capability, then one of our goals will be to define objects that need least about to privilege to get their job.

We discussed confused deputy problem in earlier lectures. What kind of authority gave rise to confused deputy problem?

**That is called ambient authority.** The victim get confused because the OS would look at all the possible sources of authority and if any one grant the ability to do something then it will give up. So we need to eliminate ambient authority.
Eliminating ambient authority from languages:

Design a language where we can construct a program, were different modules in the program will have restricted access from another module. For example, we have Module1 and Module2.

![Diagram showing access restrictions between Module1 and Module2]

What we need to do is that, take a small piece of code from each module and give least amount of privileges to them.

**Address space:** If modules can read/write any memory in the program that would be a problem. We need to protect the address space. We are doing in java so that would be taking care of this. It turns out in java we need something stronger than memory safety. What we actually need? It is called type safety. Java memory safety depends on type safety. If we memory have access to two different reference to same memory address that have different types then we can break the memory safety. Lets assume java has type safety.

**Public field:** Now public field can be dangerous. But public field is a public field of what?
It is a field of an object. It is a static method for example, print. We have to restrict this method so that it cannot print.

**Static public field:** These are closest to ambient public authority. This module does not need to be granted access to the system, they just got it. The problem with static public field is that they are shared and public so anyone can get to them.
These are kind of like ambient authority.

What about abstract methods?
That is not a problem. It is a definition of an interface. An attacker does not learn anything from the definition of interface. We cannot use definition of the interface to make changes to the system. Like someone has to implement it. So it is not a problem. We do not hide the definition of the APIs, we just control the access to those APIs.

We can give an example of,
- system.out
- File constructor: file (struct filename)

They both can be a problem. File construction can be a problem as they use file resources to access system resources. This file constructor does not allow a piece of code to access a file that the user with JVM. It is restricted by the user. If this constructor exists in the language then any module, which is running as the part of the program, and that program has user id, then that program can access similar to what can be access to the user. So we want to eliminate this constructor (global variables).

So, If I want to give a module restricted access to a file system then i can have a file like open:

class File{

    ....
    ....
    ....
    public File openAt(name){...}
}

What openAt method does?
It will restrict to very few allowed names. Let's suppose this is not static and this is the only method that returns a file that a untrusted module can see.

What an untrusted module need to have in order to get a file?
It needs a file.

How would I prevent it (module) to have access to the file system?
If we do not pass the module a file object, and this is the only way to restrict the access of the module. By making a slight change in the file API, it will eliminate the constructor. So if we do not give it initial file object then our purpose will get solved.

We can make the class static.
It would not be usable. We want it to be usable. So we want globally accessible file objects. So what we will have is:
If we construct a new file:

File root = newFile("/temp");
Untrusted.Object.doWork(root);

Now we can pass untrusted object beneath some directory to store some temporary files. We have to make sure it does not make files above its directory. So inside the core trusted part of the file system, we construct a file object that only allows to
access files underneath ‘temp’. The root is passed to the doWork() function. This name has to be checked no “..” and no “/”.

How can u get access to the object in java?
- It can be passed as an arguments
- Call constructor
- Reflection
- Static globals
- Return to you
- Exception (It can be thrown to us)

A goal that would make it easy to understand the hardware that we give to module is to be able to trace down all the objects. If we are reading the code, then we can easily see the call to constructors, classes, etc.

There is some problem with static globals because if we leave static global out there and we have to make sure that it will not get abused, and then we have to analyze the whole code. We can’t just analyze the static global itself. That’s why we eliminate those.

Next thing: Now we will look, how to use a capability to get another capability, so that’s what power constructor talks about. There is another method,

class File{

    ....
    ....
    ....

    public File openAt(name){...}

    Public File getParent();

}

This is a method in java.
We have to eliminate this method. If we have to easily reason about a set of files that a module can access then here we know that only there is one file that we have access which is ’/temp’, so it has only one object which is pass to it.

But this method getParent() destroys that concept. So we have to eliminate this method.

**Question rose in the class: - What if we have the access to the constructor of the subclass but not that of parent class?**

In that case we have to go through every API and all the constructors that allow you to
construct an object and gives you access to system resources. So we have to eliminate those constructors.

Only way to construct a file is to have a file.

If we have to make a system that uses a file API then we have to change the signature for main. The main will be given access to root directory and string args. We can delegate the file system access to sub system.

```
Main(File root,string args){ ... }
....
File tmp=root.openAt('tmp');

class File{

-
-
-
  public File openAt(name){...}

  File root=newFile("/temp");

  File tmp=root.openAt('tmp');
  Untrusted.Object.doWork(tmp);
  Untrusted.Object.doWork(root);
}

That is secure delegation.
```

---

**Reflection API violates access keywords**

```
Class File{

  String name;
}

Suppose this file class has string name, and then we pass this file object to an evil code. The evil code has reflection API, will reflect and change the name to ‘/’. We have to eliminate it. We can allow some reflection, but we cannot keep it.

- Allows program to read/modify private fields.
- Modify Reflection API so called cannot access private/protected fields and methods.

The reason we do this: we want to allow objects to enforce their own constraints but if we allow other class to access internal field of an object then we do not want that. We have to weaken this (-Reflection). Finally, we get:
- It can be passed as an arguments
- Call constructor
- Reflection *(weaken)*
- Static global *(eliminate this)*
- Return to you
- Exception

Q: If I give you a code and ask you to find out all different objects, that code can access. What would you do?
Read the code, look at what kind of objects are passed. If we pass a file, then we get methods of the file. We get method, these methods contain file, then we get methods from this file. Then the process goes on and on. We have to do a graph traversal. If we eliminate the static globals, do we even need to look at the body of the code? Suppose we have untrusted code and we need to know that are the different code that this untrusted piece of code can access. We have to see the constructors that are called. What a class can access, is the stuff that we have passed to it.

So capability oriented programming supports easy code review. Just look at the stuff passed to an untrusted module.

Some times global variables are not so bad. Suppose we have global variables that never changes. In java there is a notion of directory separator. It is a static field. It could be final. So it was like:

```java
Public Static final separator = "/";
```

Is there any danger in having this public piece of data: we cannot use this in modifying any piece of data, it is just a constant. It is ok to have global piece of constant. So we have to change the static globals to only static final globals.

Suppose we have a final:

```java
Public static final StringBuffer foo= new StringBuffer();
```

If something is final then we cannot change the reference. But we can change the object of reference where it points. The variable foo can change. So that becomes dangerous. So if there is one program, which is reading the data and another part of the program, which is messing up with the data, then we get the new value. So this get the shared state between evil modules and non-evil modules. We need a deeper concept of mutable than final. Immutable objects are deeply immutable. It means they are immutable all the way down.

So we actually need a notion of only static final immutable globals.
We cannot change the reference cannot change as it is final. Immutable objects are deeply immutable. We cannot change the field of the objects and the field of the reference will never change. Immutable stuff as long as it is secret every one will get access to it. Lets talk about attenuation:

**Attenuating capability:**

If we get a capability for file system. And we want to pass the capability to another module but we do not want to give full capability, we just want to give limited capability. How would we do that?

We can write wrappers and this wrapper should have same interface. We can wrap objects to create attenuation capabilities.

For example, we have to write/read a file handle to a file and we want to give somebody read access of the file. So we write code like this,

```java
class ROFile extends File {

    ROFile (File f)
    {
        Myf=f;
    }

    public OutputStream getOutputStream() {
        throw BadCall();
    }

}
```

If the file is declared final then we have some methods to make it read only. One method would be to declare a new class read only files. From now the untrusted code has to be written for ready only file API, which would be easy to avoid.

How we enforce the untrusted code to use this API? We are the vendor of a large database and we support plugin written in this language. We say this API is to be used and say it is necessary plugin. The goal here is to give the least privilege to do what it has to do its job.

Also the case study will be done for the last class.

Class concluded.