Today Discussion:

First discussion on the questions raised in office hours by students regarding the final project was done:

1) First one is with respect to design of the taint droid based project. **Recommendation is do not place any policy decision into the Dalvik virtual machine**, rather modify taint droid so that instead of logging the event whenever there is a tainted data written out, you can have it as exception, so this way it should be handled and data is not sent. There might be already existing exception handling in the Virtual machine infrastructure. So it does have event raising at run time.

2) Once you have this in place, then the way you can allow application to send data out to the network is that by implementing or designing the class that doesn’t have any publically accessible constructor. It can be classifier class, classifier object have no publically accessible constructor. If it does not have one, it has to get it from somewhere else and so in this way it is like Location Provider where an application cannot construct a location provider, there is no publically available constructor, rather it has to use location manager to get one. So the classifier class will have one or may be more than one method which takes an argument as an object or you might make it special like making it as a byte array an then what it does is, it duplicates it input argument but if it moves all the tainted bits into the input argument on the copy. So now
the application that has a classifier object can take any data which it wants to send over the data, it is an untainted copy. Now the policy decision that who gets which copy is not something we would not want to be done inside virtual machine.

3) Any policy decision is not to be placed inside VM. It is only doing enforcement. Not a good place to place policy decision.

**Capability**

Last time we were talking about capabilities and we will continue with that. We started with hardware capability and then we talked about OS capability.

- **Hardware Capability** – Fat pointers
- **OS capability** – File descriptors
- **Language based Capability** – today’s topic. It is the model called object capability model. In this case, object will be capability or capability will be an object.

Let’s back up a bit.

➔ Do we remember, kind of capability rode on this topic was Native client where we were trying running untrusted code inside the same address space as of the trusted code so as to confine the untrusted code?

➔ So it was restricted to not only access memory that was assigned to it, it was design in such way that we could call untrusted code form trusted code and vice versa. The call still requires copying the stuff.

➔ So we saw, that how we can avoid the data copy and that led to hardware capability. It is the capability where I can give a pointer to the data and hardware is going to ensure that the
recipient of that pointer or we can use that pointer to escape and get out to other places in memory then allows me to pass the data to untrusted code.

Certain Properties that capability possess:-

- Unforgeable
- Possession = permission
- Passable/communicable
- Attenuable: - it means to get weaker/make weaker. It would be nice to have one capability and build one weaker capability based on that because we might want to pass the capability to another subsystem that you don’t fully trust. So you give them the restricted access and let them do their work.
- For example we have got the function that sums up the bunch of integers and I have got a read/write pointer to an array of integers. So now I want to pass it to the summation function but I don’t want summation function to write to it. So we going to construct read only pointer for read/write pointer. So by this way we are restricting the access.

Let's think about what we need to do, if we want to use object as capabilities:

Here, we talk here about Joe-e (based on Java).

But before we discuss that we see the motivation behind it. A major motivation of capabilities is the principle of least privilege which means each piece of code gets the least amount of power it needs to do its job.

Least privilege is kind of relative.
Like if we go back to example of integer function summation, least privilege it needs is to read those integers. But if hardware doesn't support the legal pointers, well the least privilege it needs that hardware can allow you to do it will be to read/write pointers.

So these privileges are relative to how precise your system can express the privileges. So the hardware can allow read/write pointers than read only pointers.

One of the first things to do build up these kind of capability based system is we need to eliminate the ambient authority. This came from the concept of confused deputy that we discussed in the class before.

**Eliminating ambient authority from language.**

Design a language where we can construct a program and different modules in our program where it only has restricted access to other modules in program or our computer system in general. Let us see the figure to understand it:
How does ambient authority come in the programming language?

➜ **Address space** – if the module in the above figure can read/write any memory in the program that would be a problem. It will start playing with modules data structures. It might right the network buffers etc. **So protecting the address space is important** and if we are Building top of java, it is taken care of. *(Memory safety) In java we need more stronger than memory safety, that is type safety. Java memory safety depends on its type safety.* It means if you can get two different references to same address space having different types then you can break the memory safety with java. But we will assume that java has type safety.

➜ **Access specifier** – public fields and private. It could be dangerous. It is the **public field of an object/class**. Two different version of public field of object and class in java is known as static. So there are static method in java like main, println or print. So we would like to restrict the module in the above diagram so that it do not print. And it is a problem if print is the static method in system.out.

➜ **Why static public fields are problem?** - > They are shared and if they are public anybody can access them. They are kind of like ambient authority. They are there, anybody can use it anytime.

➜ **What about abstract methods?** - > These methods do not pose any problem. Abstract methods are simply viewed as like a definition of the universe. So, hacker doesn’t learn about them anything based on the interface. So we cannot use abstract interface for the sake of it, someone has to implement it with meaning. So it is not the problem.

➜ **Example for static public field:** -
- **System.out** (no global variables).
- Sometimes **constructors** can be a problem particularly when they are used to create objects for system resources.
- For example there is a **File Constructor**.
  
  ```java
  File(String filename);
  ```
- This constructor **do not allow** piece of code to access file that user is running in JVM, we are still restricted by user. User’s access rights for the matter of fact. If **this constructor exist in language**, any module that is running as the part of the program and that program has your user-id then that program can access any file that we as user can access.
- So we need to eliminate these constructor and global variables like system.out.

What we can do is, if I need to give a module restricted access to file system. We can do like:

```java
Class File {
    
    .
    
    .
    
    openAt(String filename) {....}
}
```

This method can restrict you to very specific allowed names but in particular this is not static. This is the only method that returns a file which an untrusted module will see.

**If it is not static as shown in the code below, what does the untrusted module need to have before it gets a file.**

**It needs a file to get a file. If it don't need a file, don't give it a file.**
What if I want the module to don’t have access to the file system at all?

- If I never pass the module the file object in the code below and if it is the only way to get the file object, then it cannot get any file handle.
- So by making a slight change in the API we can do this as shown below in the code snippet.

```java
Class File {
    .
    .
    .
    Name is checked to be no "/" or no '..',
    Public File openAt (name) {....} // not static
}
```

If we construct a new file we can do it with globally access file objects allow me to construct a file in the open constructor. What we can have it is like:

```java
File root = new File("/tmp"); // it allows you to access files under tmp
untrustedObject.doWork (root); // maybe it is a plugin to the code
```

Now it can use open at on root thing, to get the things globally. So the name changes to be as shown in the above figure.

How to get an object in Java:

1) Passed as an argument.
2) Call constructor.
3) Reflection (weaken this).
4) Only Static final Immutable global variable- why so? We eliminate static global variable. They are kind of problem as we don’t know who will be accessing them. To analyze them, we have to see rest of the code so that to see they are not misused in any way. That’s why we eliminate those.

5) Returned to you.

6) Exceptions (issues, we going to talk about later)

Next, we have to analyze how we can use one capability to another capability. There comes file constructor trick in picture.

Class File {
.
.
.

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

Public file getParent();
}

Another method in file object that we have to eliminate. This is a method in java class file. If we have to reasonable set the access right on a module that which files it can access, then our goal was that we can only access files under /tmp. That is the only file object that we pass it in the function. But the getParent method destroys this, so we have to eliminate the getParent() method.

Class File {
.
.
.

Public File openAt(name){....}
Question raised 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 essentially from nothing construct an object and gives you access to system resources. So we have to eliminate those constructors.

For example: Main is passed an array of strings, so if we wanted to build a system, will it be able to access any file at all? No.

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

Change signature of main: -

Main (File root, string[] args){
    ...
    ...
}

Now idea will be main program will be given handle access to real root directory, and then if plugin is loaded and provide a restricted access to its subdirectory then one can use open at , so what we actually do is seen below in the snippet of code: -

Class File {

    ...

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

    File root = newFile("/tmp");
File temp = root.openAt('temp'); // This is done
UntrustedObject.doWork(tmp);
Public file getParent();
}

it is secure delegation to the sub file system.

**Reflection API:**

Even if the field of an API is declared private, we can make use of reflection API to mess with it. It is terrible for security.

- It violates access keywords.
- Imagine that file class defined above had string name and we pass it to the evil code and reflected code reflects, reflects and change name to '/' and then call openAt.
- It allows program to read/modify private fields. We have to eliminate this. We are going to allow some reflection.
- We have to modify Reflection API so that caller cannot access private/protected fields and methods.
- Reason we need to do this is that we going to allow objects to enforce their own **constraints**. If we allow another class to mock with the contents or internal fields of the class then we won’t be able to maintain the consistency, so we have to weaken the reflection.

Let’s see the goal we are trying to achieve here and then we get back to the Exceptions.

Q. If the only way you can use reflection in a clean way or let’s just assume that static global variables have been eliminated then if you are given the piece of code and said to find out all the different type of objects that you can access what will you do?
-> Read the code is the first thing we do. We would look at what kind of objects got passed, then we will look at, if I pass file, then what I can get from the methods of the file and then we keep digging in further. So it will be like a graph traversal which we can start with the body of the code.

-> If I eliminate global static variables, do we even need to look at the body of the code? No.

**Capability oriented programming supports easy code reviews**

**Just look at the stuff passed to an untrusted module**

Example of a concept before we jump to exceptions: -

Sometimes global variables aren’t so bad. In java, for files there is a notion of the directory separator.

```java
    Public Static final separator = "/";
    Public static final StringBuffer foo = new StringBuffer();
```

String Buffer is immutable. **Final means we can’t change the reference but we can change/ mutate the reference to the object.**

This foo can change which is dangerous. It becomes a global shared state that is it is sharable 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.**

Let us talk about attenuation.

**Attenuating capability:** -If we have capability for network and if we want to give it to a module, how do we do that?
We can wrap objects to create attenuation capabilities. Wrapper can implement the same interface as the original one.

Suppose we have got read and write file handle to root directory and we have to give access to read a file. We might write code something like that:

```java
Class ROFile extends File {
    myf
    ROFile(File f);
    Myf = f;
}
Public OutputStream getOutputStream(){
    Throw BadCall(); // Exception call
}

.... Then normal execution of program
```

If file is declared final or method is final then we have method to make it read only. One method here would be to declare a new class read only files. Now the untrusted code has to be written for read only file API which would be nice to avoid it.

**How we enforce the untrusted code to use this API?**

Well, if we are the vendor of large database and we support plugin written in this language and we say this API to be used and we say it is a necessary plugin. Goal here is to give plugin the least privilege to do what it has to do its job.

Issue with exceptions will be covered in next class. Also the case study will be done for the last class.

**Class Concluded**