

## “Red” project

- Goal:  
*develop a language for specifying software verification procedures*
- Application area: software security, ...

## “Red” project

- See also (for eg):  
*[MOPS: an Infrastructure for Examining Security Properties of Software, Chen, Wagner]*

## “Red” project

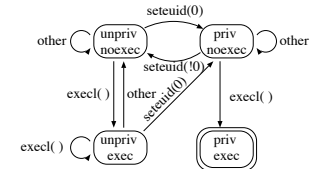
- See also (for eg):  
*[MOPS: an Infrastructure for Examining Security Properties of Software, Chen, Wagner]*

## “Red” project

```
int main(int argc, char *argv[])
{
  // start with root privilege
  do_something_with_privilege();
  m0:
  drop_privilege();
  m1:
  m2:
  m3:
  execl("/bin/sh", "/bin/sh", NULL); // risky syscall
}
```

```
void drop_privilege()
{
```

```
  struct passwd *passwd;
  d0:
  d1:
  d2:
  d3:
  d4:
  if ((passwd = getpwuid(getuid())) == NULL)
    return; // but forget to drop privilege!
  fprintf(log, "drop priv for %s", passwd->pw_name);
  seteuid(getuid()); // drop privilege
}
```



(a) An FSA describing Property 3

## “Red” project

---

```
int
main(int argc, char **argv)
{
    char buf[100];
    int x;

    if(argc != 2)
        exit(1);

    x = 1;

    snprintf(buf, sizeof buf, argv[1]);
    buf[sizeof buf - 1] = 0;
    printf("buffer (%d): %s\n", strlen(buf), buf);
    printf("x is %d/%#x (@ %p)\n", x, x, &x);
    return 0;
}
```

## “Red” project

---

- Problem:
  - *define a language for specifying properties (temporal, control and data flow...)*
  - *implement an algorithm for generating a static analyzer for that property*
  - *analyze C code!*

## “Red” project

---

- Approach:
  - *start with a trivial language and a simple analyzer*
  - *become more ambitious with time*

## “Green” project

---

- Goal:
  - *develop a tool for (semi)automatic migration of legacy C code to Java*
- Application area: software maintenance

## “Green” project

---

- Problems:
  - *Java JNI allows C code to interact with Java in unsafe ways*
  - *Java compilers can not optimize JNI*
  - *Static analysis defaults to “top” when encounter a JNI call OR unsound*
  - *JNI code is not portable (or at least requires inspection)*

*Sun Research has started a project on translation of C code to bytecode.*

## “Green” project

---

- See also

*[Safe Heterogeneous Applications: Curing the Java Native Interface, Tan, Appel, Chakradhar, Raghunathan, Ravi, Wang, Princeton TR]*

## “Green” project

---

- A step-wise approach (think in term of use-case):
  - *systems calls (and calls to libraries that have no access to Java objects) are always “OK”*
  - *procedures that could be written in Java (e.g. translation requires no work)*
  - *procedures that do not access Java objects*
  - *procedures that access Java objects*
  - *procedures that manipulate C data structures*

## “Green” project

---

- Challenge:
  - *analyze C code!*
  - *exit strategy: even partial translation is a big step forward.*
  - *Use the GNU Java (gcj) system for validation*

---

```

JNIEXPORT jobject JNICALL
Java_java_io_ObjectInputStream_currentClassLoader( JNIEnv * env,
                                                    jclass clazz,
                                                    jobject loader )
{
    jmethodID id = (*env)->GetMethodID( env,
                                        (*env)->GetObjectClass( env, loader ),
                                        "currentClassLoader",
                                        "()Ljava/lang/ClassLoader;" );

    if( id == NULL )
        return;

    (*env)->CallObjectMethod( env, loader, id );
}

```

---

```

static jobject _javanet_create_inetaddress(JNIEnv *env, int netaddr) {
    char buf[16];
    jclass ia_cls;
    jmethodID mid;
    jstring ip_str;
    jobject ia;
    /* Build a string IP address */
    sprintf(buf, "%d.%d.%d.%d", ((netaddr & 0xFF000000) >> 24),
          ((netaddr & 0x00FF0000) >> 16), ((netaddr & 0x0000FF00) >> 8), (netaddr & 0x000000FF));
    DBG("Created ip addr string\n");
    /* Get an InetAddress object for this IP */
    ia_cls = (*env)->FindClass(env, "java/net/InetAddress");
    if (!ia_cls){
        _javanet_throw_exception(env, IO_EXCEPTION, "Can't load InetAddress class");
        return(0);
    }
    DBG("Found InetAddress class\n");
    mid = (*env)->GetStaticMethodID(env, ia_cls, "getByName",
                                   "(Ljava/lang/String;)Ljava/net/InetAddress;");
    if (!mid){ _javanet_throw_exception(env, IO_EXCEPTION, "Internal Error"); return(0); }
    DBG("Found getByName method\n");
    ip_str = (*env)->NewStringUTF(env, buf);
    if (!ip_str){ _javanet_throw_exception(env, IO_EXCEPTION, "Internal Error"); return(0); }
    ia = (*env)->CallStaticObjectMethod(env, ia_cls, mid, ip_str);
    if (!ia) { _javanet_throw_exception(env, IO_EXCEPTION, "Internal Error"); return(0); }
    DBG("Called getByName method\n");
    return(ia);
}

```

## Evaluation

---

- You should produce
  - Software
  - ~10 page methodology paper that documents your experience in development, what process you used, what worked what did not...
  - 10 page (ACM format, latex) paper on the problem & solution
    - *Grades will be based on all three*