An Approach to Software Vulnerability Analysis (SVA)

Kestrel Institute

Jim McDonald

March 29, 2001

Kestrel Institute

Outline

- Project goals
- Project strategy and flow
- Initial success story
- Current vision
- Description of the demo
- Project plans

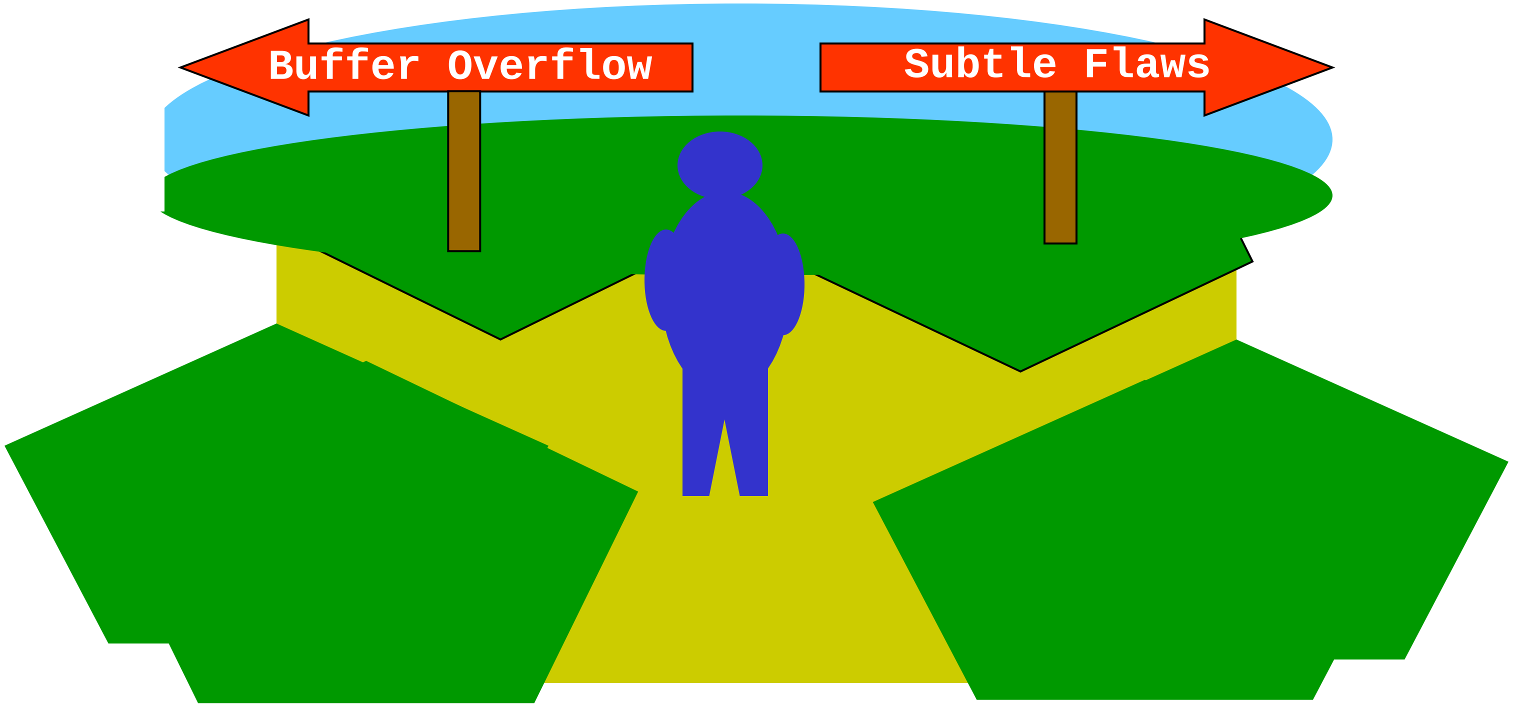
Kestrel Institute

SVA Project goals

- Build characterization of software vulnerability to support computation
  - Organized
  - Semantic rigor
  - Reusable
  - Extendable
- Build inference & analysis tools to detect vulnerabilities
  - Automation
  - Mixed initiative
- Demonstrate detection of real vulnerabilities

Kestrel Institute

SVA Project strategy

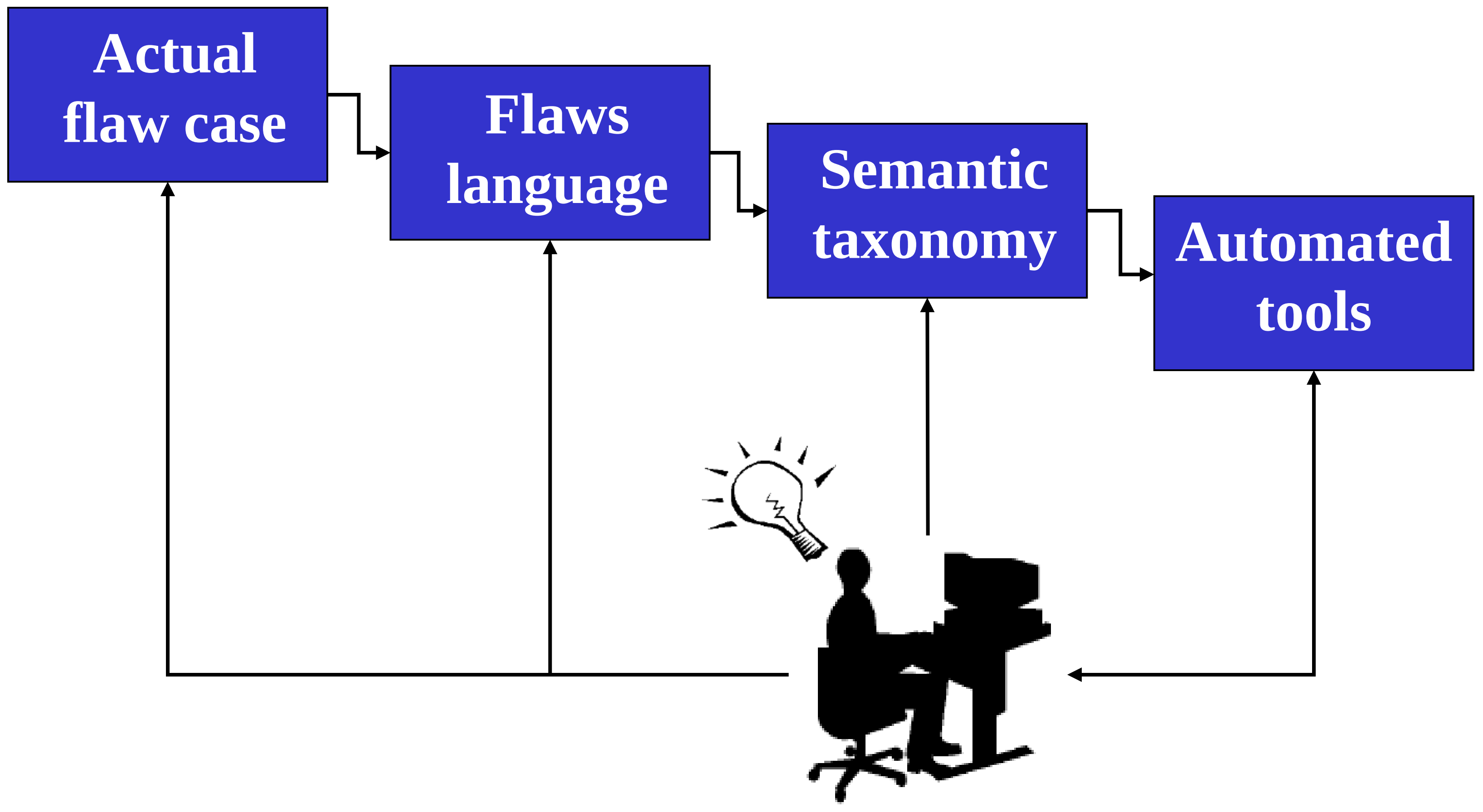


Subtle flaws

- Elude smart compiler – buffer overflow detection increasingly tractable
- Multiple element interactions – possibly great complexity
- Handle protocol implementations – optimization can cloud interactions
- Typically require human assessment & guided search to assess impact

Kestrel Institute

SVA Project flow



Kestrel
Institute

August 8, 2000: real flaws

[ed note: text taken from Dan Brumleve’s website]

2000.08.03, San Francisco

I've discovered a pair of new capabilities in Java, one residing in the Java core and the other in Netscape's Java distribution. The first (exploited in **BOServerSocket and BOSocket) allows Java to open a server which can be accessed by arbitrary clients.** The second (**BOURLConnection and BOURLInputStream) allows Java to access arbitrary URLs, including local files.**

As a demonstration, I've written **BOHTTPD** for Netscape Communicator. BOHTTPD is a browser-resident web server and file-sharing tool that demonstrates these two problems in Netscape Communicator. BOHTTPD will serve files from a directory of your choice, and will also act as an HTTP/FTP proxy server. **[ed note: “open door”]**

Kestrel Institute

Two days later

[ed note: text taken from Dan Brumleve's website]

2000.08.05

Right now I'm at the internet cafe (Club I) at 850 Folsom in San Francisco (between 4th and 5th street). I'll be here until 2:00 a.m. showing demos to anybody interested.

A guy showed up here and made BOHTTPD multithreaded. This new functionality is live right now…

WHOA! I just saw a Windows 2000 system that was still running BOHTTPD even after Netscape had been apparently terminated. Even the "Task Manager" showed no trace. **[ed note: "door stays open"]**

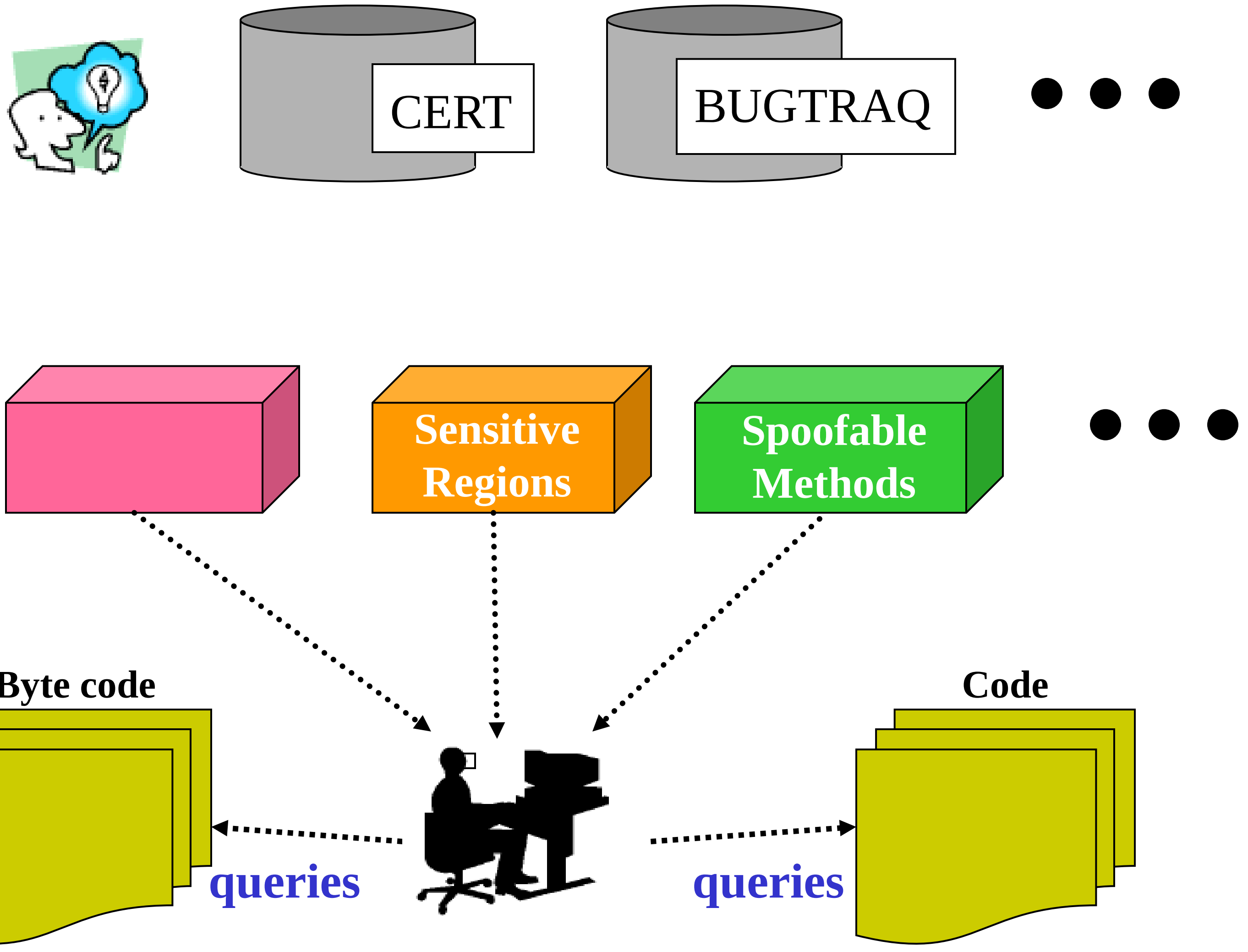
Kestrel Institute

Connecting flaw concepts to code

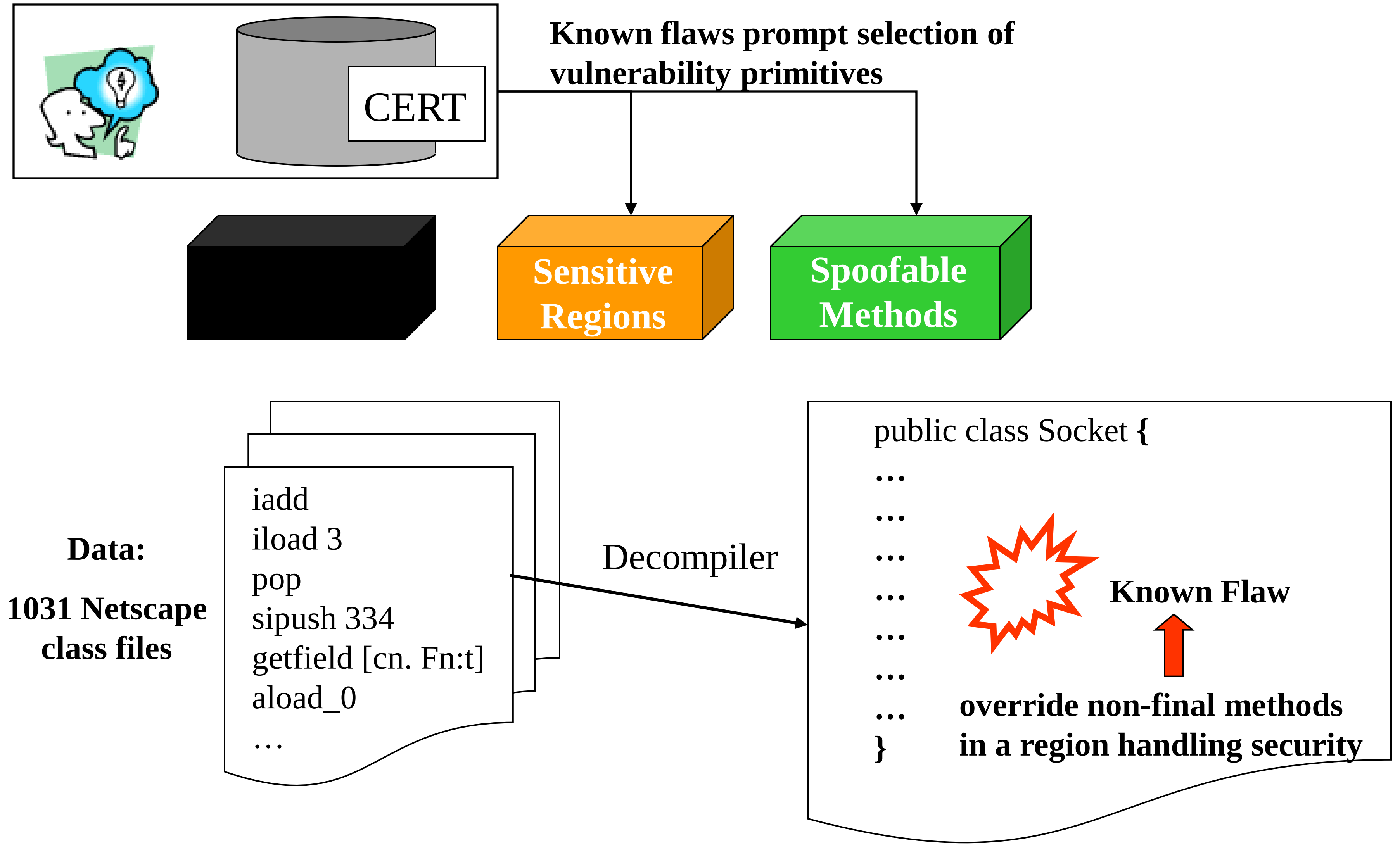
Mine rich sources of flaws

Select flaw primitives for a language

Create tools for making queries on (byte)code



Work with Brumleve's "BO" attack


CERT
Known flaws prompt selection of vulnerability primitives
Sensitive Regions
Spoofable Methods
Data:
1031 Netscape class files
iadd
iload 3
pop
sipush 334
getfield [cn. Fn:t]
aload_0
…
Decompiler
public class Socket {
…
}
Known Flaw
override non-final methods in a region handling security

Anatomy of the “BO” attack

```
public class BOHTTP extends Applet {
   …
   public void init () {
   …
   ess = new BOServerSocket(port);
   …
   }
   …
   public void run () {
      BOSocket client;
      …
      client = ess.accept.any();
      BOHTTPConnection ff = new BOHTTPConnection();
      …
    (new Thread(ff)).start();
  }
  …
}
```

Kestrel
Institute

Anatomy of the “BO” attack

```
public class BOServerSocket extends ServerSocket {
  …
  public BOSocket accept_any () throws IOException {
    BOSocket s = new BOSocket();
    try { implAccept(s); }
    catch (SecurityException se) { }    ← Does Nothing!
    return s;
  }
}

public class BOSocket extends Socket {
  public void close_real () throws IOException {
    super.close();
  }
  public void close () { }    ← Does Nothing!
}
```

Kestrel Institute

Anatomy of the “BO” attack

```
protected final void implAccept (Socket socket) throws IOException
{ try
   {  socket.impl.address = new InetAddress();
      socket.impl.fd = new FileDescriptor();
      impl.accept(socket.impl);
      SecurityManager securitymanager = System.getSecurityManager();
      if (securitymanager != null)
      { securitymanager.checkAccept(socket.getInetAddress().getHostAddress(),
                                    socket.getPort());
        return; }
   …
   catch (SecurityException securityexception)
   {
       socket.close();                  <-- Could be close from BOSocket!

       throw securityexception;         <-- accept_any from BOServerSocket can thwart!
   }
}
public void close () throws IOException
{ impl.close }
```

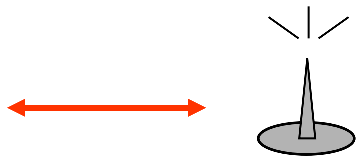
Kestrel
Institute

Anatomy of the “BO” attack

```
Class BOURLConnection extends URLConnection {
   …
   public BOURLConnection (URL u) {
      super(u);
      connected = true;
   }
}

Class BOURLInputStream extends URLInputStream {
   …
   public BOURLInputStream (URLConnection uc)
        throws IOException {
      super(uc);
      open();
    }
}
```

Anatomy of the “BO” attack

```
class BOHTTPDConnection implements Runnable {
   …
   euc = new BOURLConnection(uu);
   euis = new BOURLInputStream(euc);
   while ((b = euis.read()) >= 0) os.write(b);
   …
}
```

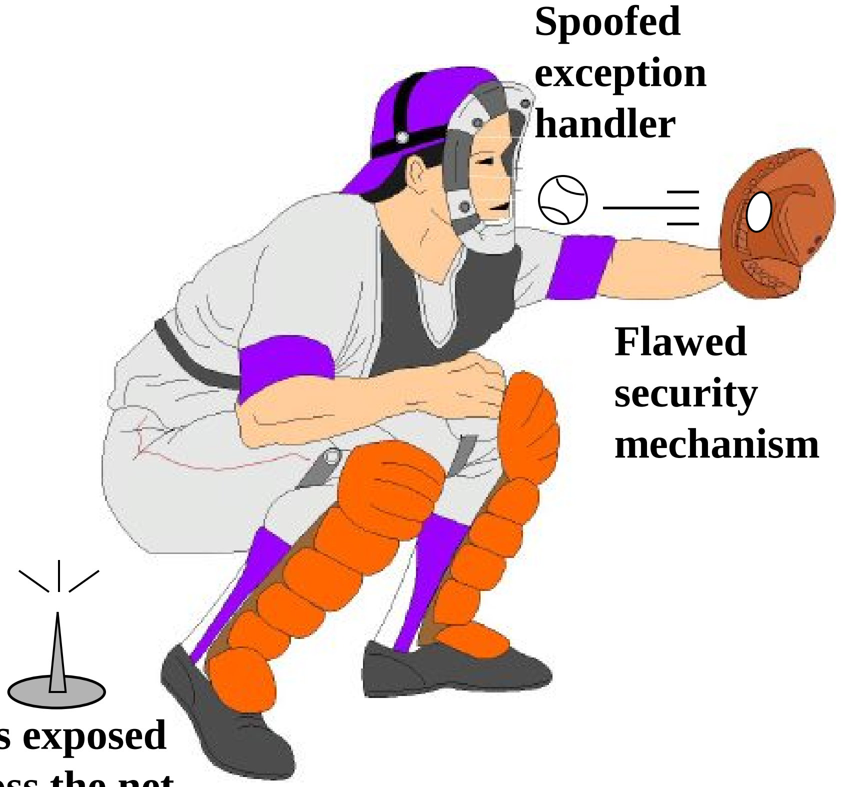


Files exposed across the net

Kestrel Institute

Concepts lead to queries

“Requisite” Sports Analogy

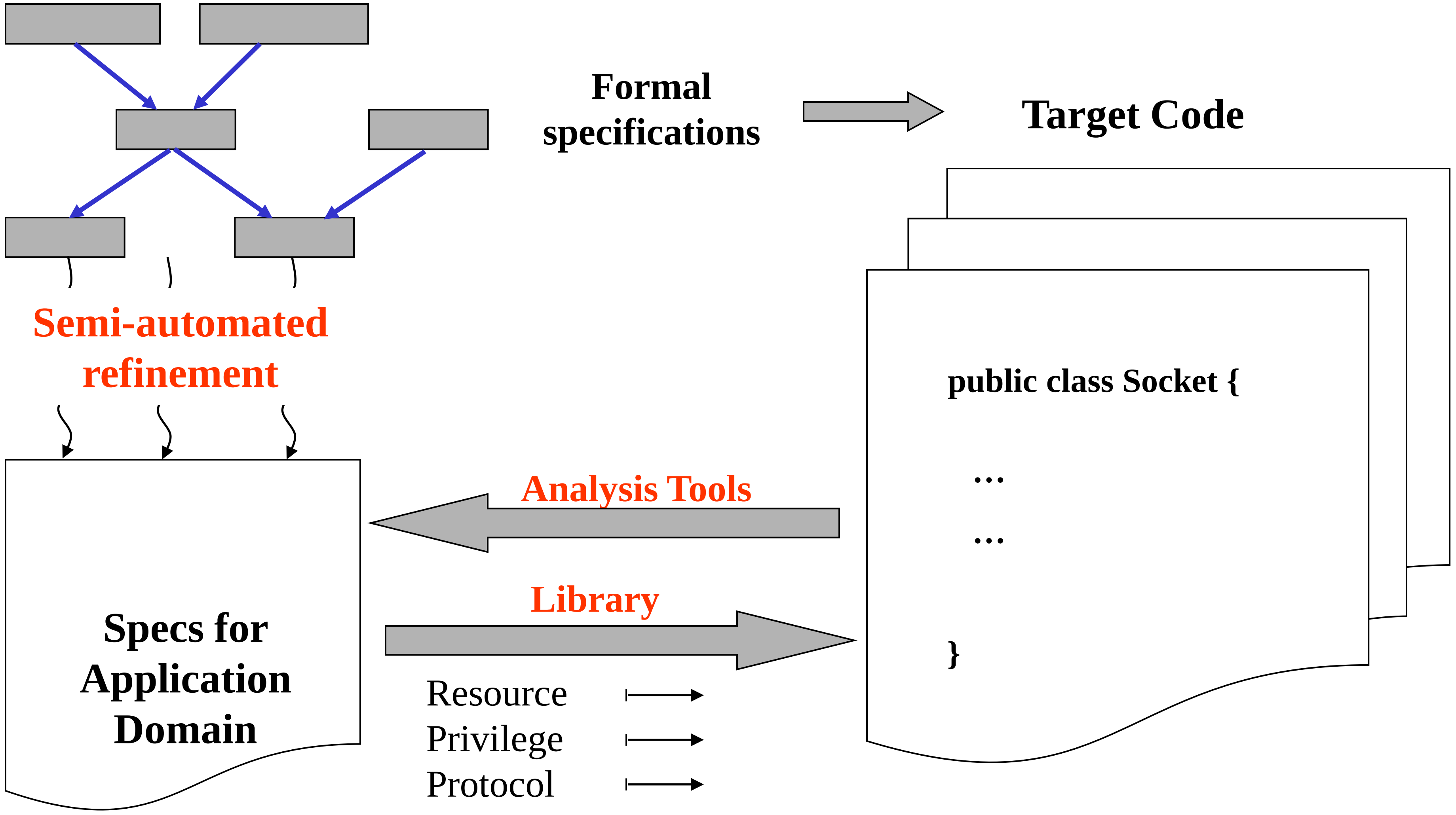

Spoofed exception handler
Flawed security mechanism
Files exposed across the net

1. *Find all methods M* that can be overridden; compute their traces†
2. *Find all sensitive regions R*; in this case, those handling security mechanisms
3. Look for *traces* of methods in M *that pass through R*

† Leverage from bytecode verifier tech base.

Kestrel Institute

Code ~~synthesis~~ analysis

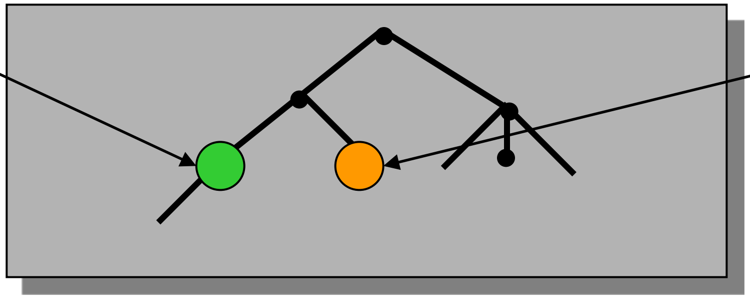


```
public class Socket {

   …

   …


}
```

Kestrel Institute

Formalizing the semantics

Spoofable invocations **Sensitive regions**

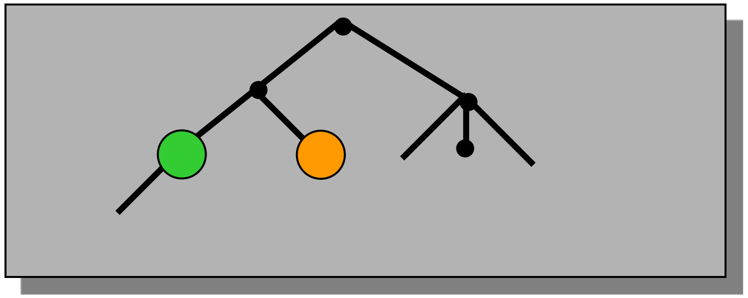


```
spec Spoofable-Invocation is
  op final?        : method     →  Boolean
  op virtual?      : invocation →  Boolean
  op spoofable? : invocation →  Boolean
  …
  end-spec
```

```
spec Sensitive-Region is
  sort CR-Attribute = | privileged
                      | …
  sort Code-Region =
         {context      : method,
          start        : pc,
          end          : pc,
          attributes  : set CR-Attribute}
  …
  end-spec
```

Kestrel Institute

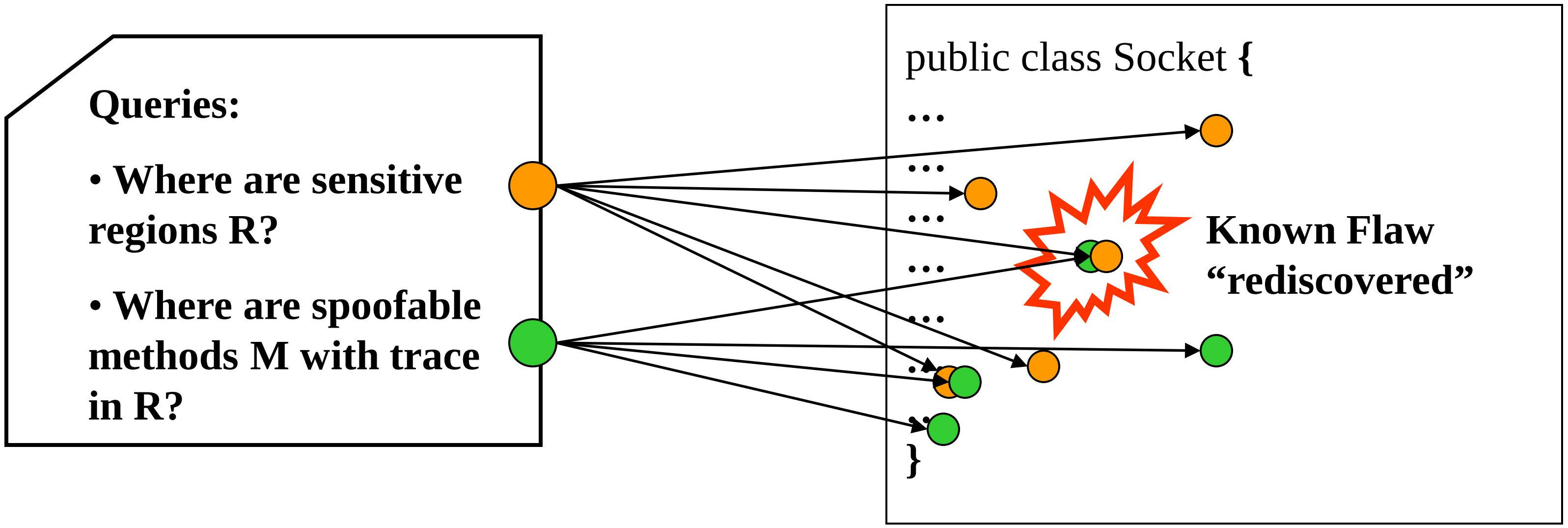
Initial queries on Brumleve's code



New entries for the semantic taxonomy

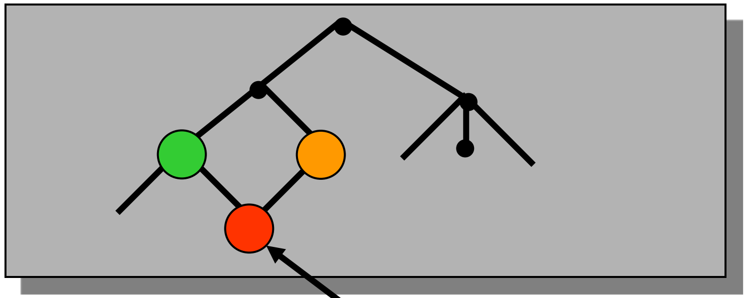
Queries:

- **Where are sensitive regions R?**
- **Where are spoofable methods M with trace in R?**



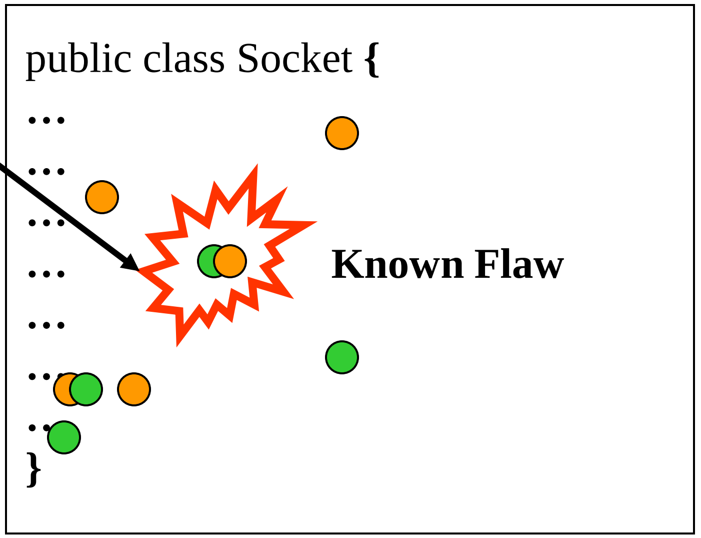
Kestrel Institute

Extending the taxonomy



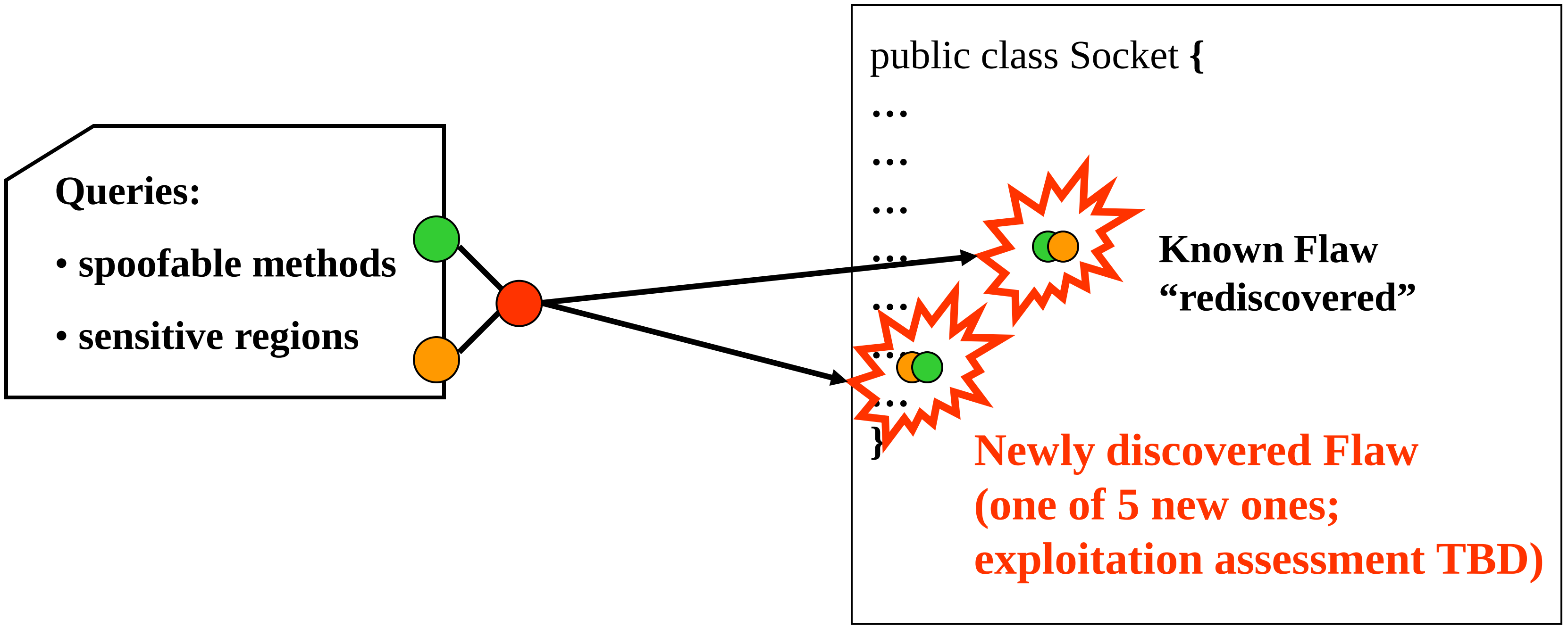
Refining vulnerability primitives

- Extend taxonomy
  - …with semantics
- Note: it is possible to use this information to construct an attack
  - Automatic construction possible
  - Can be stored with taxonomy for later use in testing, etc.



```
public class Socket {
…
…
…
…
…
…
…
}
```

Finding more than expected

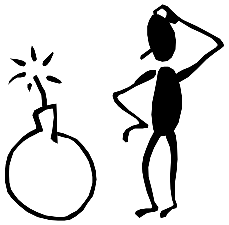


Kestrel Institute

Finding more than expected

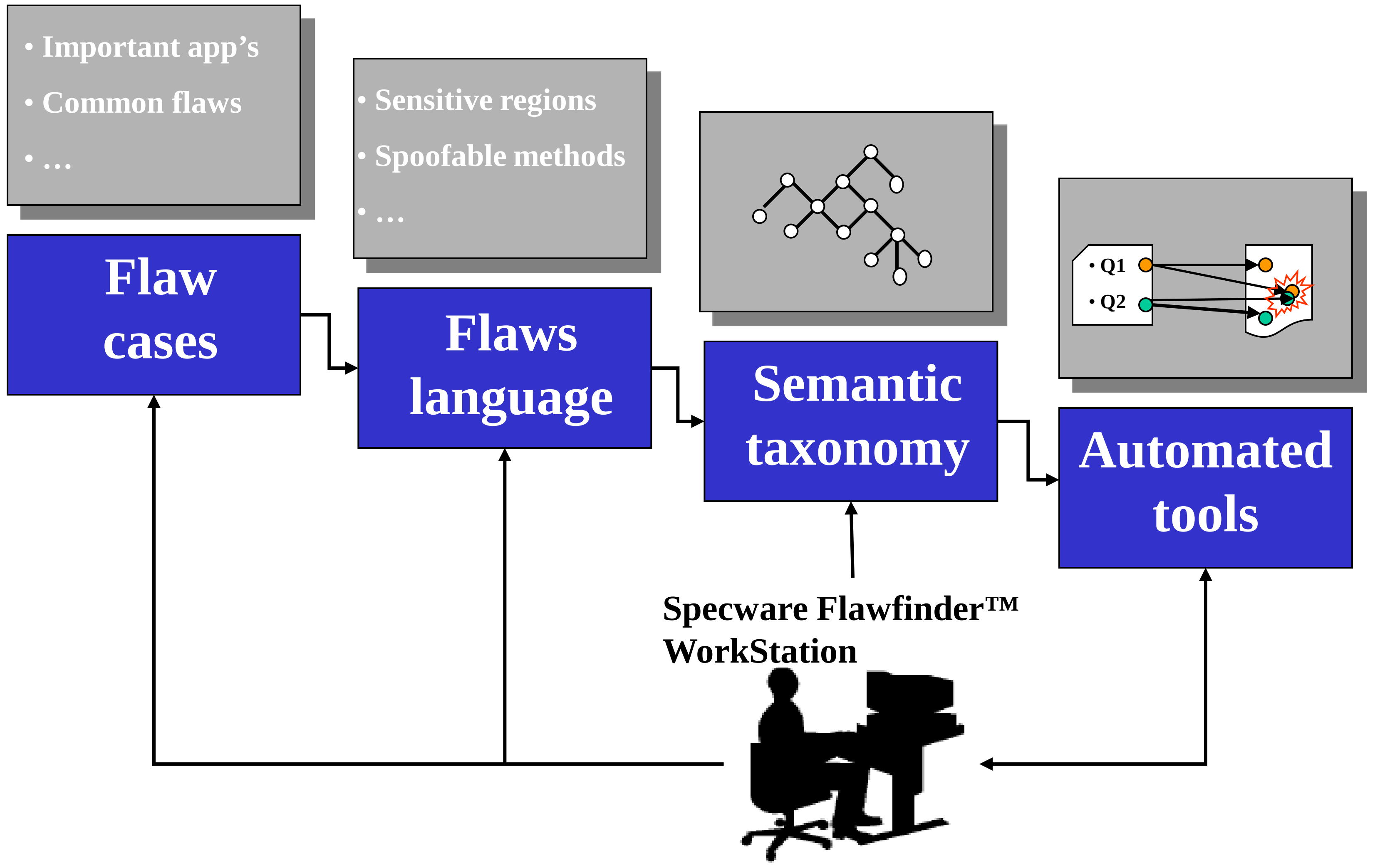
From java.net.DatagramSocket :

```
public synchronized void receive (DatagramPacket datagrampacket)
    throws IOException
{
   SecurityManager securitymanager = System.getSecurityMaganager();
   synchronized(datagrampacket)
   { if (securitymanager != null) do
      {  InetAddress inetaddress = new InetAddress();
          int I = impl.peek(inetaddress);
          try
           { securitymanager.checkConnect(inetaddress.getHostAddress(), I);
              break; }
          catch (SecurityException _ex)
           { DatagramPacket datagrampacket2 = new DatagramPacket (new byte[1], 1);
              impl.receive(datagrampacket2); }
       } while (true);
    impl.receive(datagrampacket);
   }
}
```



Kestrel Institute

Current vision



Kestrel Institute

Description of Demonstration

- Background:
  - Show infrastructure for analyzing Java byte code
- Ideas:
  - spoofable invocation – virtual invocation of non-final method
  - sensitive region – try/catch/throw involving security, etc.
  - Intersection is a vulnerability
- Demo:
  - Write specs to instantiate these ideas
  - Generate code to find and report vulnerabilities

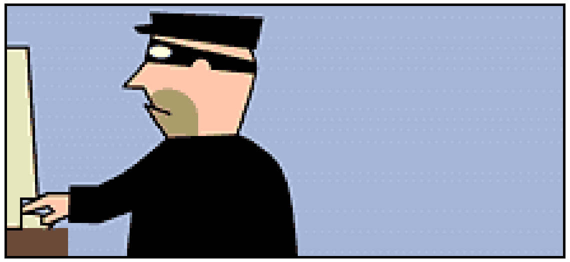
Kestrel Institute

SVA Project Plans

- Infrastructure optimizations
  - 10 hours →1 minute
- Enrich language for syntactic patterns
- Enrich language for semantic attributes
- Analyze tantalizing results
- Scan other target applications
- New CERT/BUGTRAC cases
- Construct taxonomy of vulnerabilities

Kestrel Institute

Backup slides follow



Kestrel Institute

How is this “not Norton”?

- Norton
  - Collection of fixed patterns matched against application
- No analysis
  - Won’t find new flaws
  - Won’t find variations on existing patterns

- Specware-based
  - Uses abstractions of known flaws
- Analysis aided by automation
  - Query synthesis to find flaws attributable to single or combinations of elements
  - Computer-based inference to find unprecedented flaw structures
  - Encourages expert initiative to find new flaws