Harvesting Runtime Values in Android Applications That Feature Anti-Analysis Techniques

Siegfried Rasthofer, Steven Arzt, Marc Miltenberger, Eric Bodden

TECHNISCHE UNIVERSITÄT DARMSTADT

SECURE SOFTWARE ENGINEERING GROUP

Fraunhofer SIT

This we would still hope for…

```
@Override
 protected void onCreate(Bundle paramBundle) {
    SmsManager manager = SmsManager.getDefault();
    manager.sendTextMessage("3353", null, "798657", null, null);
}
```

FakePlayer 2010

But this is what we get…

```
public static void gdadbjrj(String paramString1,
    String paramString2) throws Exception{
    // Get class instance
    Class clz = Class.forName(
        gdadbjrj.gdadbjrj("VRIf3+In9a.aTA3RYnD1BcVRV]af") );
    Object localObject = clz.getMethod(
        gdadbjrj.gdadbjrj("]a9maFVM.9")).invoke(null);
    // Get method name
    String s = gdadbjrj.gdadbjrj(“BaRIta*9caBBV]a");
    // Build parameter list
    Class c =  Class.forName(
        gdadbjrj.gdadbjrj("VRIf3+InVTTnSaRI+R]KR9aR9"));
    Class[] arr = new Class[] {
        nglpsq.cbhgc, nglpsq.cbhgc, nglpsq.cbhgc, c, c };
    // Get method and invoke it
    clz.getMethod(s, arr).invoke(localObject, paramString1,
        null, paramString2, null, null);
}
```

SmsManager.sendTextMessage(...)

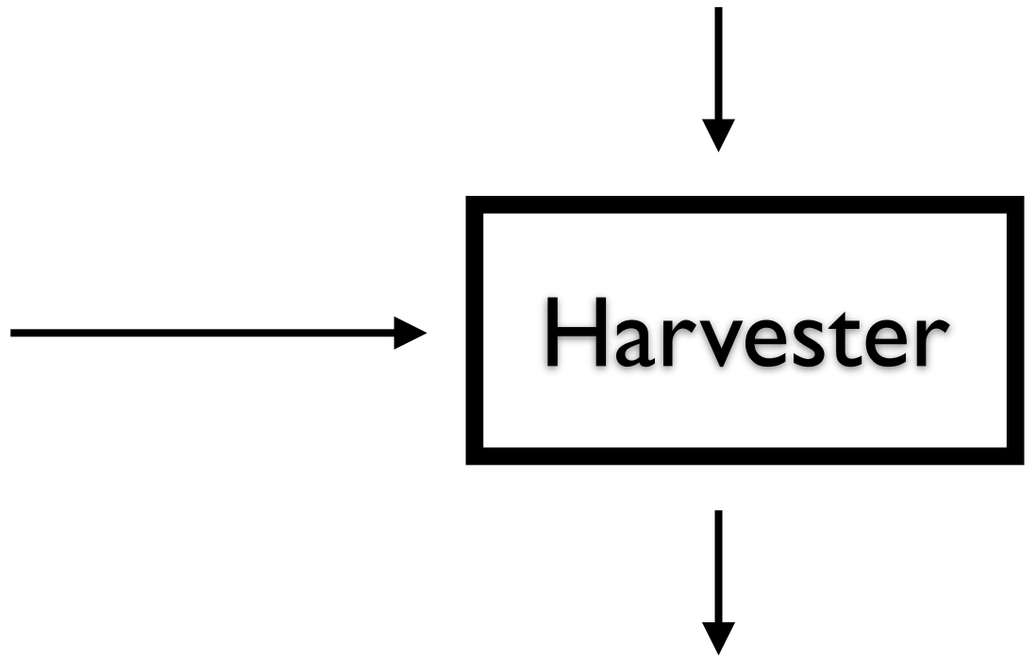
Contributions

C1: Fully-Automatic Extraction of Runtime Data

C2: Fully-Automatic Resolving of Reflective Method Calls

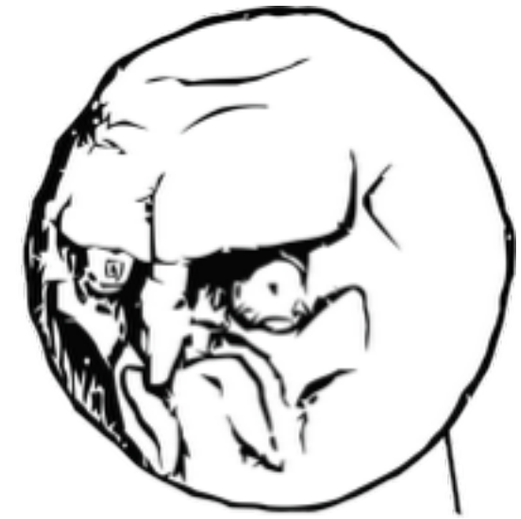
C3: Improving the Coverage of Existing off-the-shelf Static and Dynamic Analysis Tools

- sendTextMessage(num, text)
- Class.forName(className)



- sendTextMessage(“004242“, “loc_Other“)
- sendTextMessage(“008888“, “loc_US“)
- Class.forName(“SmsManager“)

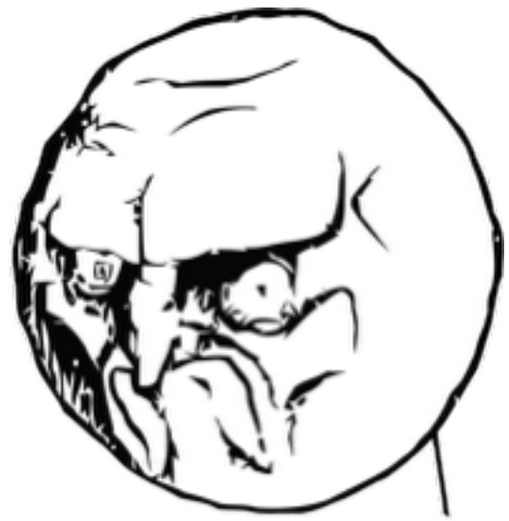
Static Analysis?



NO.

```
if(Build.FINGERPRINT.startsWith("generic"))
msg = AES.decrypt("1234","fri$ds\&S")
nr = "00"
if(simCountryIso().equals("US"))
nr += "4242"
nr += "8888"
sendTextMessage(nr, msg)
...
```

Dynamic Analysis?

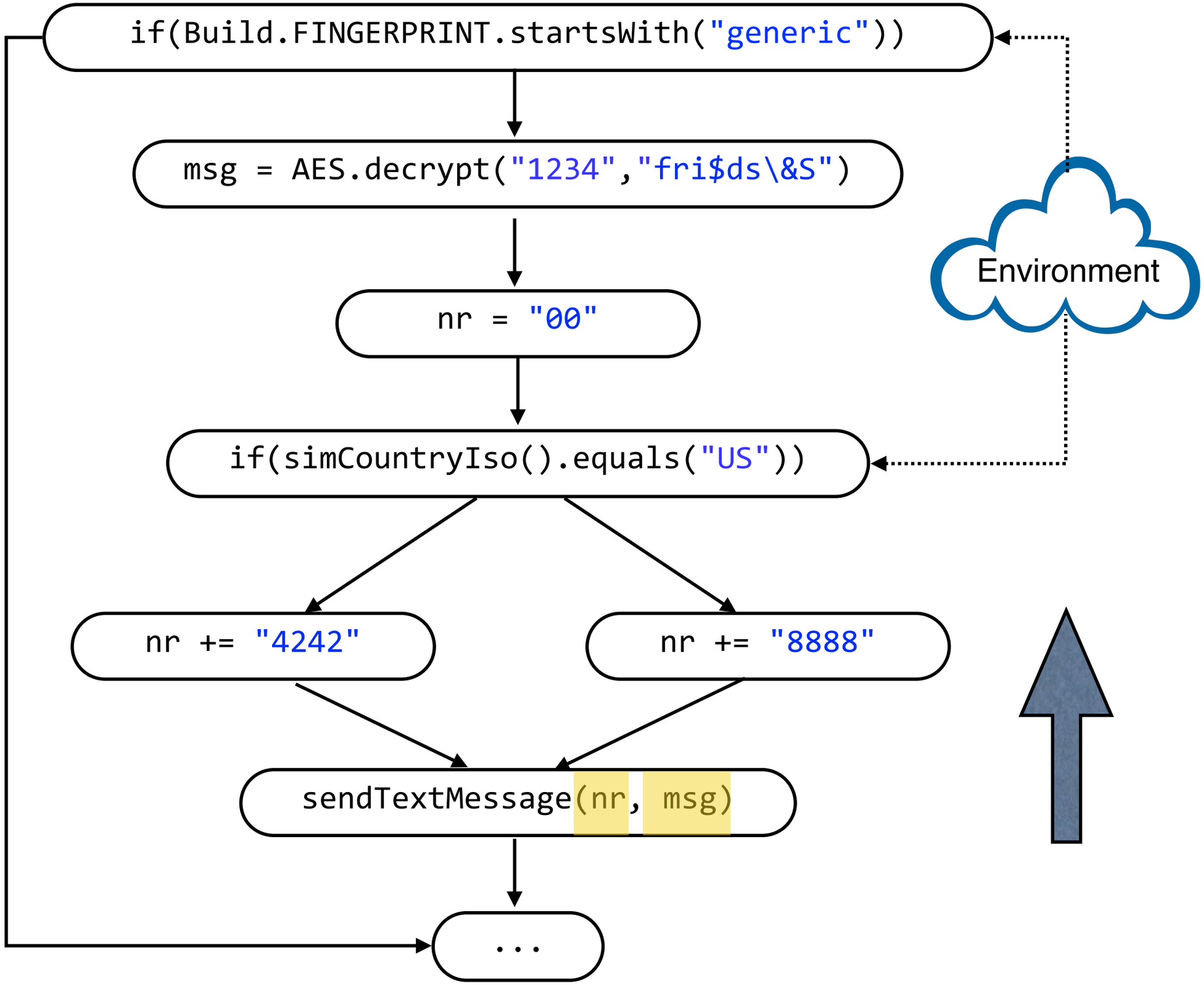


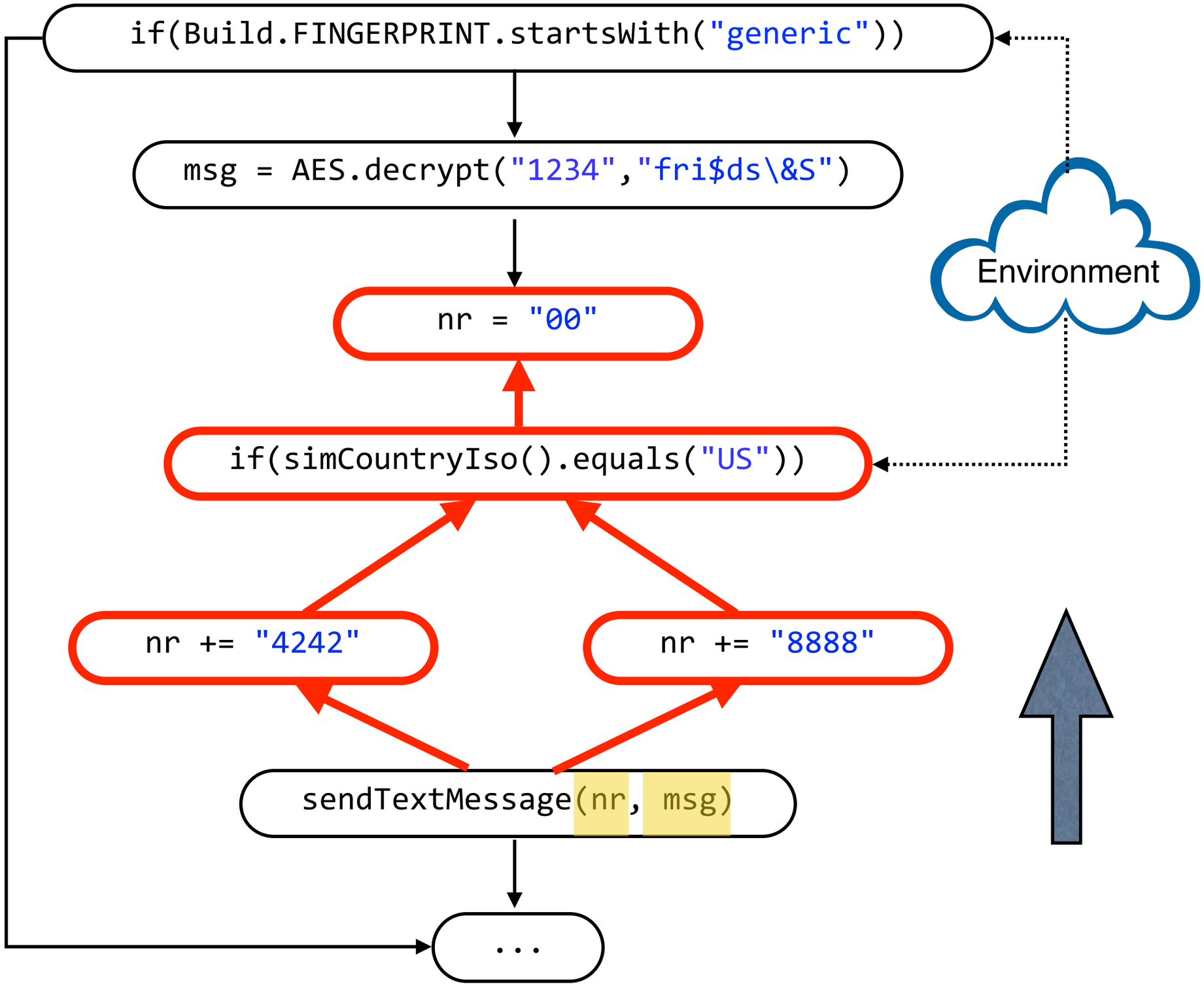
NO.

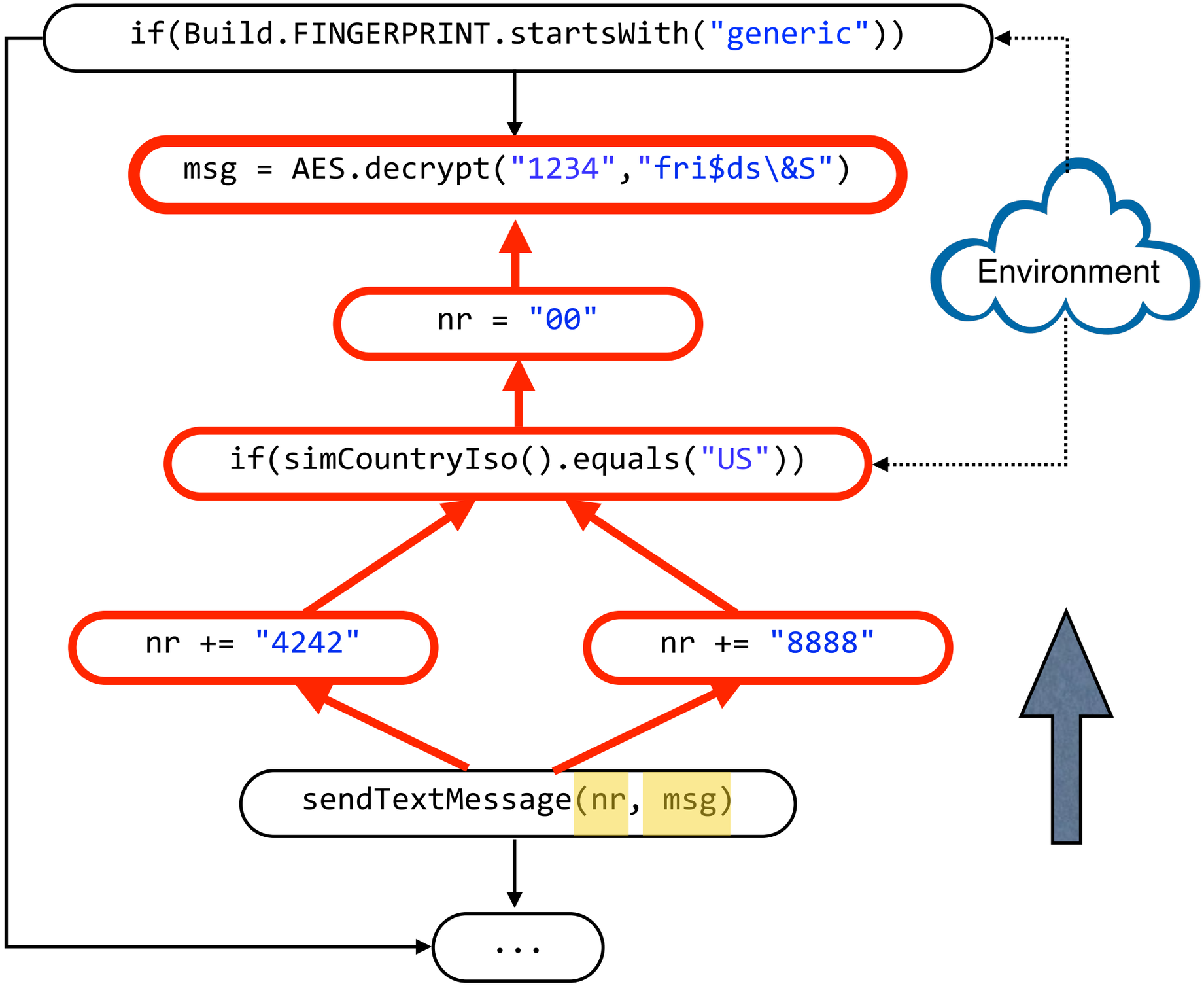
Static Analysis

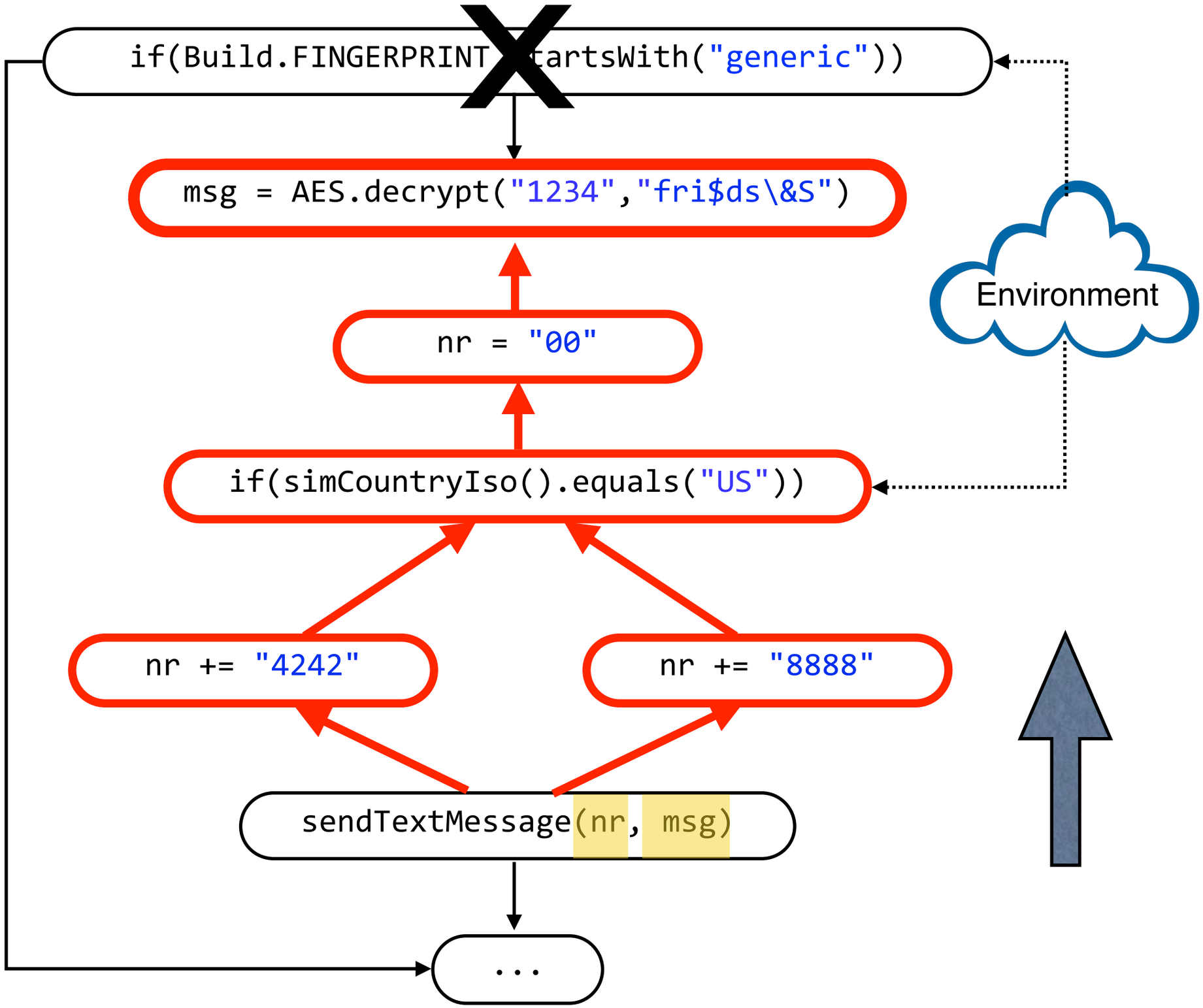
+

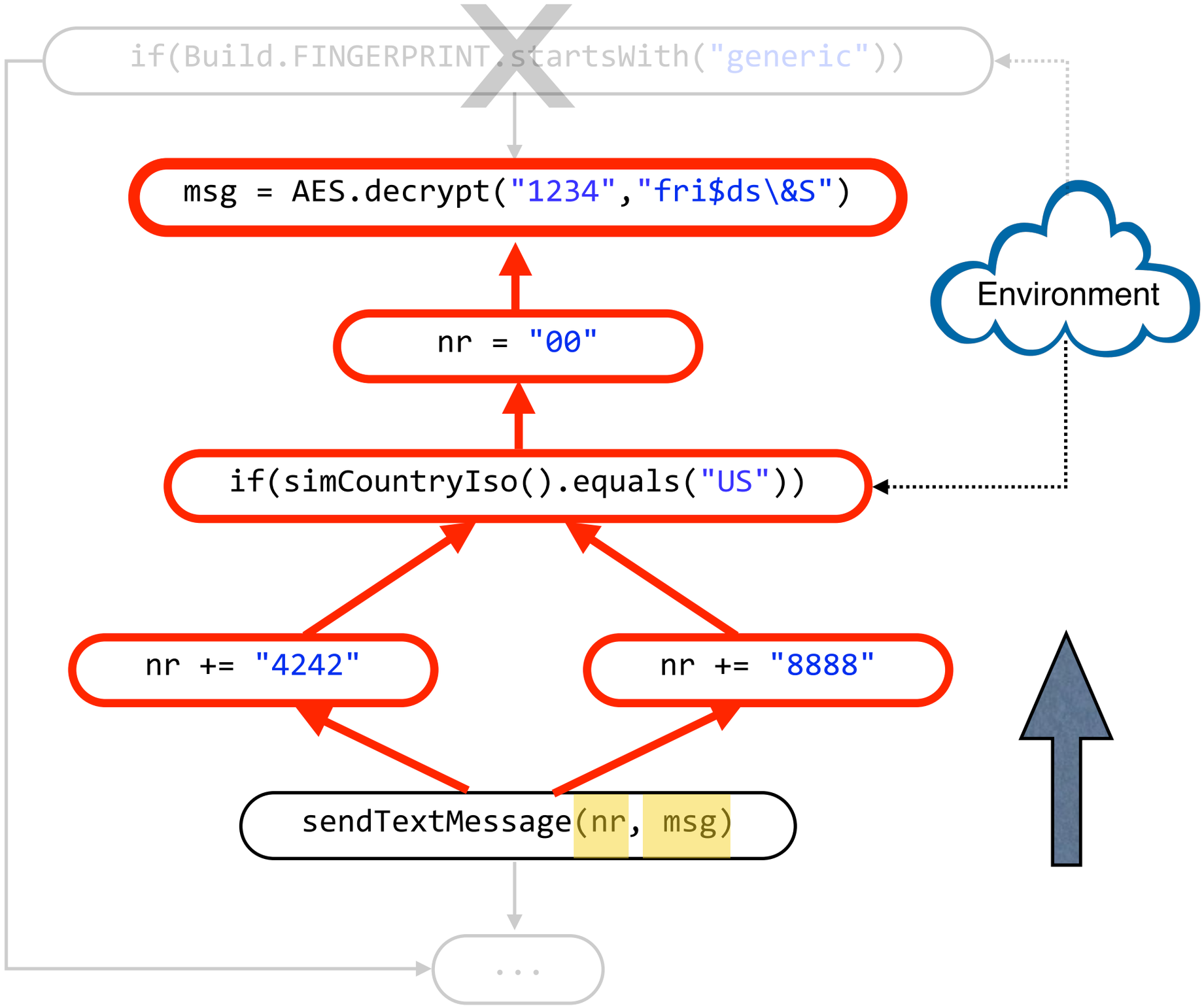
Dynamic Analysis

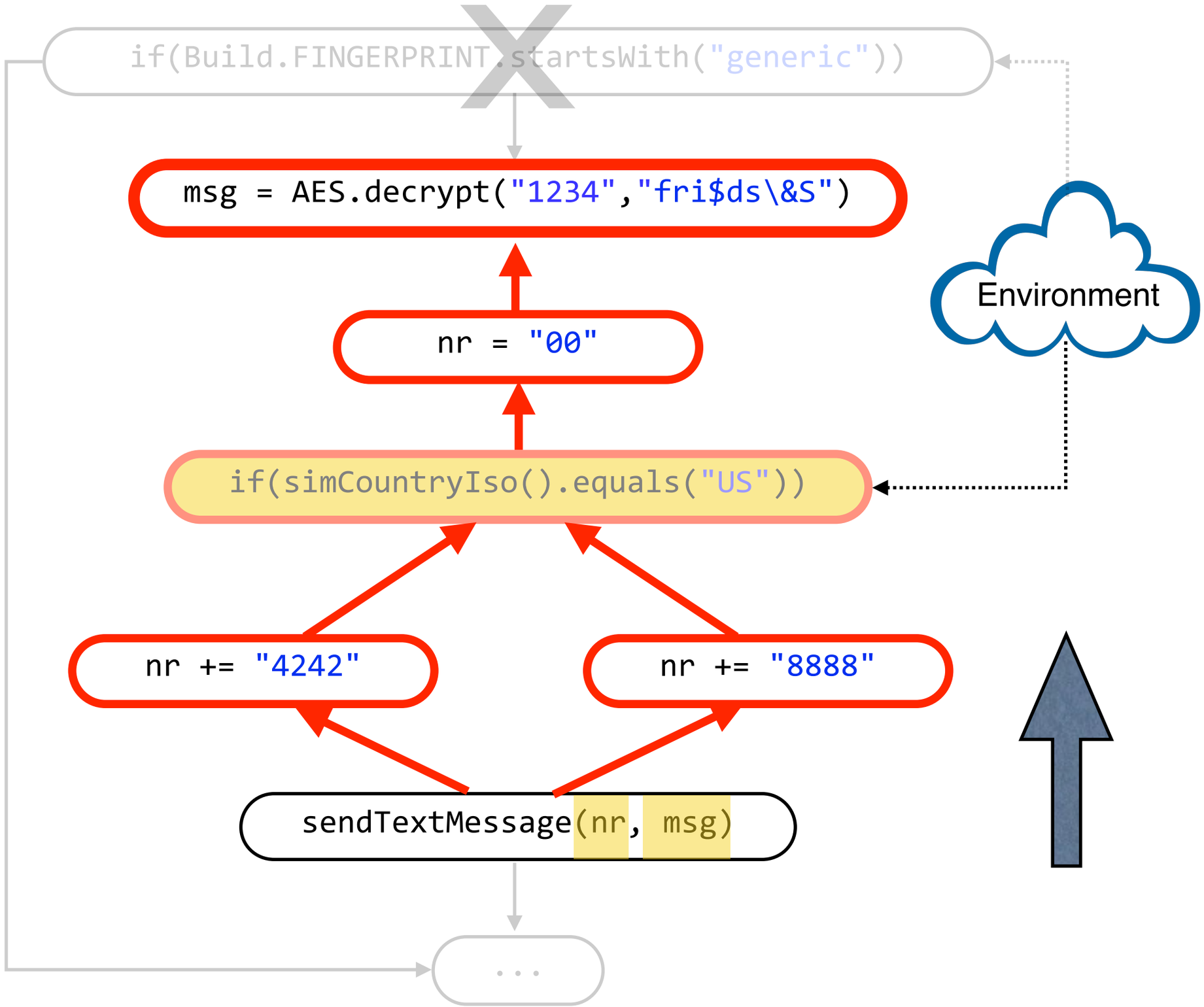

if(Build.FINGERPRINT.startsWith("generic"))
msg = AES.decrypt("1234","fri$ds\&S")
nr = "00"
if(simCountryIso().equals("US"))
nr += "4242"
nr += "8888"
sendTextMessage(nr, msg)
...
Environment

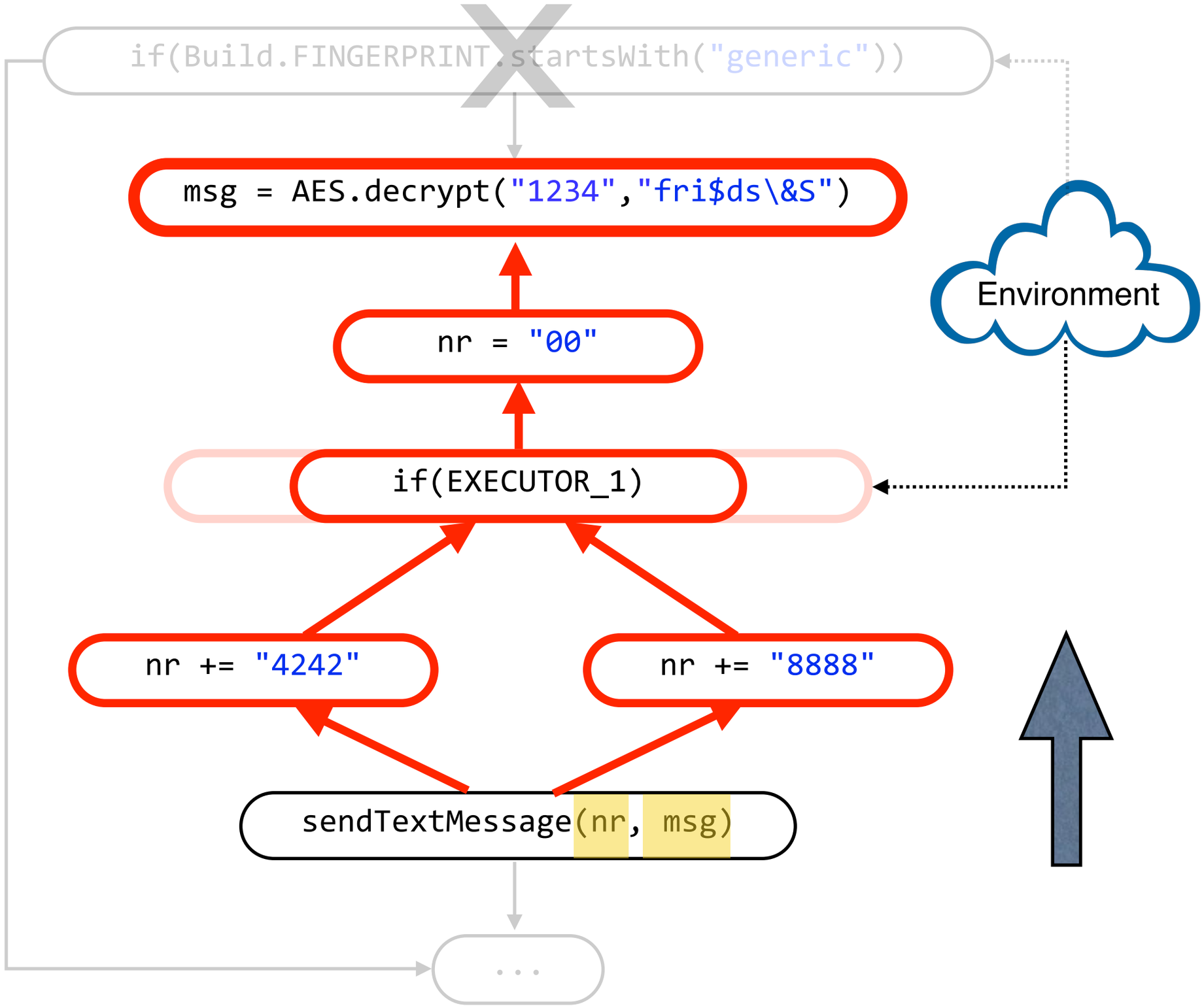

if(Build.FINGERPRINT.startsWith("generic"))
msg = AES.decrypt("1234","fri$ds\&S")
nr = "00"
if(simCountryIso().equals("US"))
nr += "4242"
nr += "8888"
sendTextMessage(nr, msg)
...
Environment


if(Build.FINGERPRINT.startsWith("generic"))
msg = AES.decrypt("1234","fri$ds\&S")
nr = "00"
if(simCountryIso().equals("US"))
nr += "4242"
nr += "8888"
sendTextMessage(nr, msg)
...
Environment


if(Build.FINGERPRINT.startsWith("generic"))
msg = AES.decrypt("1234","fri$ds\&S")
nr = "00"
if(simCountryIso().equals("US"))
nr += "4242"
nr += "8888"
sendTextMessage(nr, msg)
...
Environment

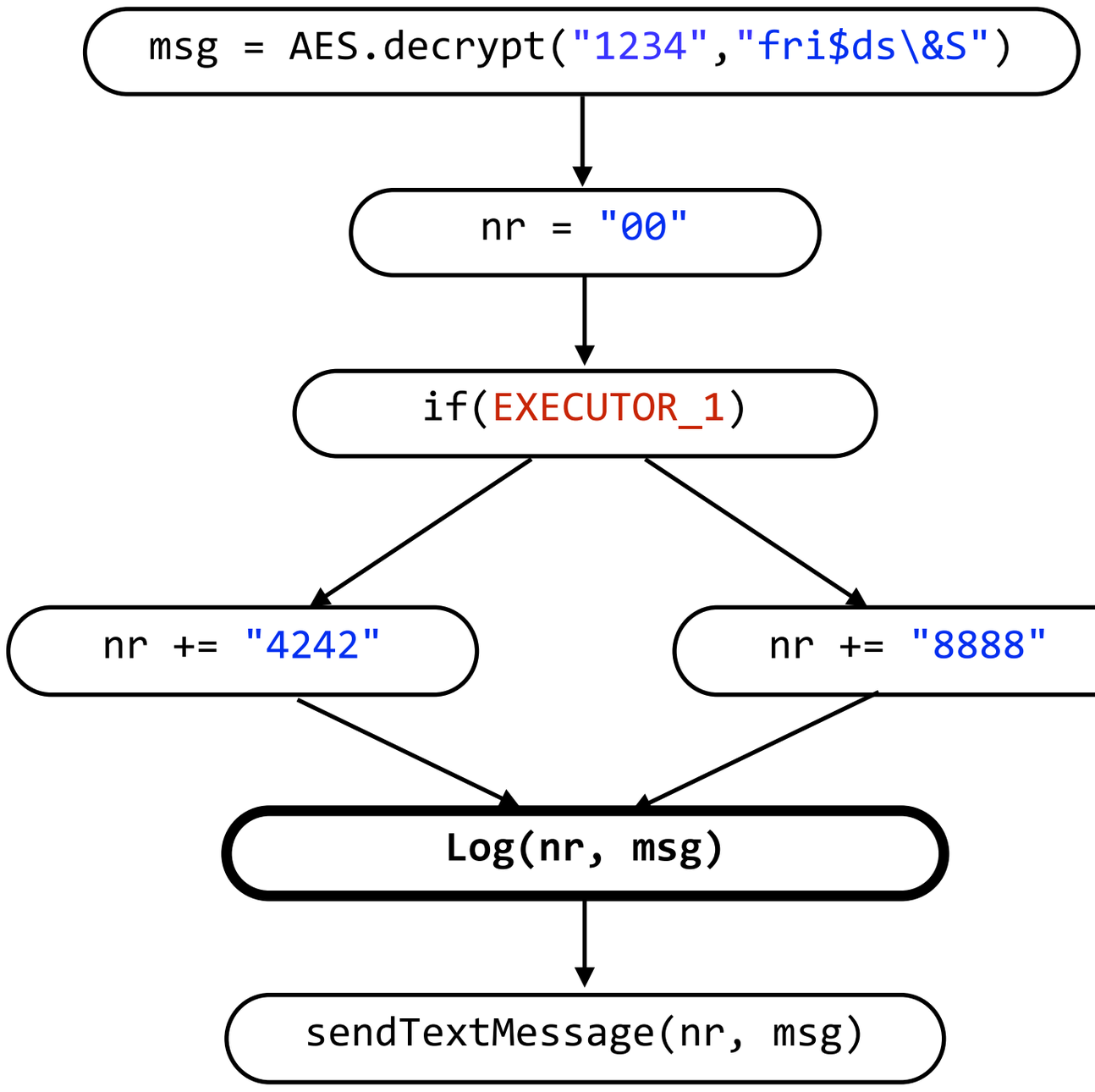

if(Build.FINGERPRINT.startsWith("generic"))
msg = AES.decrypt("1234","fri$ds\&S")
nr = "00"
if(simCountryIso().equals("US"))
nr += "4242"
nr += "8888"
sendTextMessage(nr, msg)
...
Environment


if(Build.FINGERPRINT.startsWith("generic"))
msg = AES.decrypt("1234","fri$ds\&S")
nr = "00"
if(simCountryIso().equals("US"))
nr += "4242"
nr += "8888"
sendTextMessage(nr, msg)
...
Environment


if(Build.FINGERPRINT.startsWith("generic"))
msg = AES.decrypt("1234","fri$ds\&S")
nr = "00"
if(EXECUTOR_1)
nr += "4242"
nr += "8888"
sendTextMessage(nr, msg)
...
Environment

```
main() {
  Callee1(false);
  Callee1(true);
}
```

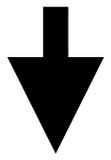
```
Callee1(boolean EXECUTOR_1) {
```



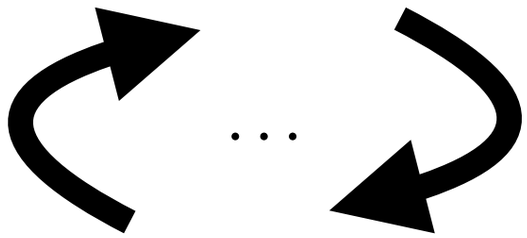

```
}
```

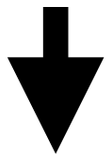
Harvester++

```
Class c =  Class.forName(gdadbjrj.gdadbjrj(„VRIf3+InVTTnSaRI+R]KR9aR9“));
...
```



```
Class c =  Class.forName("SmsManager");
...
```





```
SmsManager.sendTextMessage(a, b, c, d, e);
```

Recall: 87%

Precision: 100%

Efficiency:

< 3 minutes

16,799 Malware Samples

Interesting findings:

- Premium-rate numbers
- C&C messages
- URLs (URIs)
- Encryption key for WhatsApp data
- Backend-as-a-Service: 56 Million Sensitive User Data

```
public static void gdadbjrj(String paramString1,
    String paramString2) throws Exception{
    // Get class instance
    Class clz = Class.forName(
        gdadbjrj.gdadbjrj("VRIf3+In9a.aTA3RYnD1BcVRV]af") );
    Object localObject = clz.getMethod(
        gdadbjrj.gdadbjrj("]a9maFVM.9")).invoke(null);
    // Get method name
    String s = gdadbjrj.gdadbjrj("BaRIta*9caBBV]a");
    // Build parameter list
    Class c =  Class.forName(
        gdadbjrj.gdadbjrj("VRIf3+InVTTnSaRI+R]KR9aR9"));
    Class[] arr = new Class[] {
        nglpsq.cbhgc, nglpsq.cbhgc, nglpsq.cbhgc, c, c };
    // Get method and invoke it
    clz.getMethod(s, arr).invoke(localObject, paramString1,
        null, paramString2, null, null);
}
```

Static Analysis

+

Dynamic Analysis

Siegfried Rasthofer
TU Darmstadt/Fraunhofer SIT
Email: siegfried.rasthofer@cased.de

Blog: http://blogs.uni-paderborn.de/sse/
Twitter: @CodeInspect