



SP2025 Week 02 • 2025-02-06

Java & Android Rev

Henry

Announcements

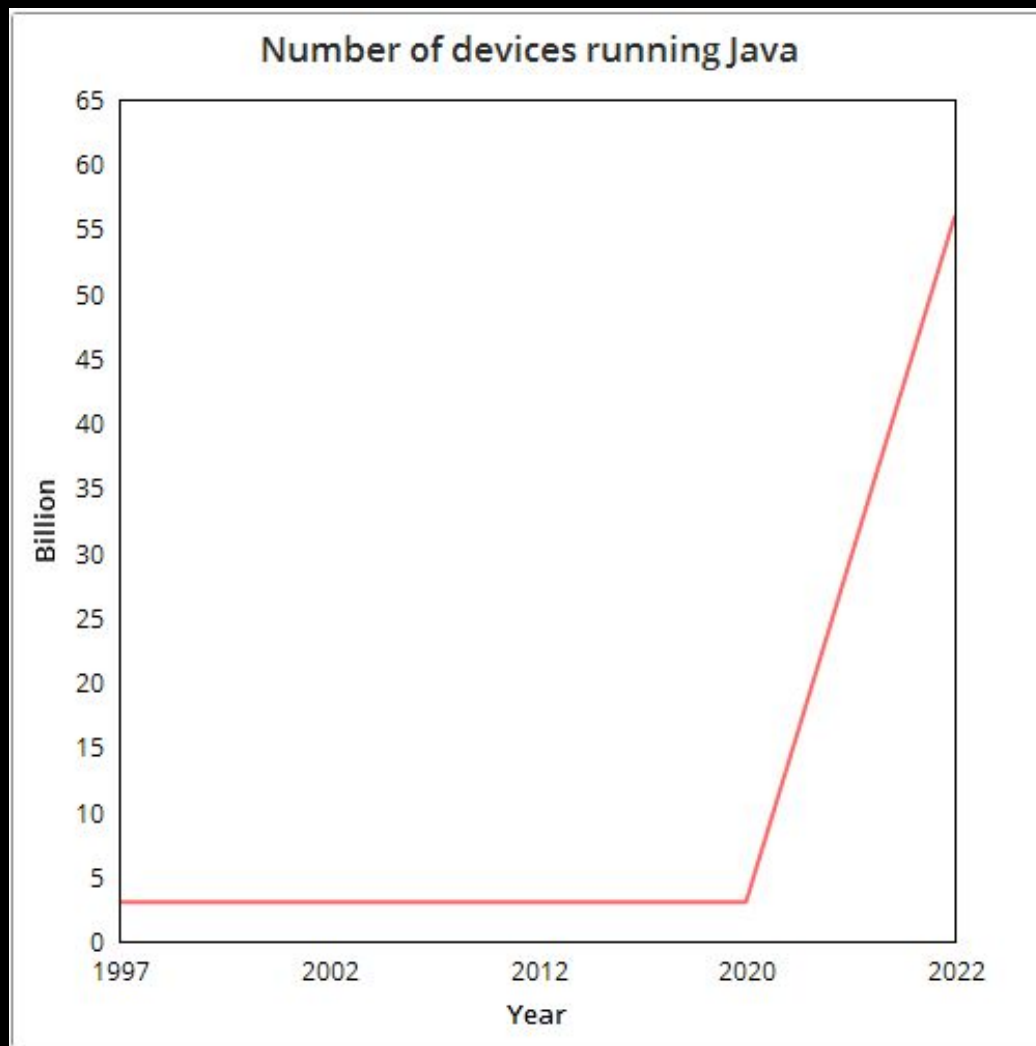
2025-02-07 • This Friday

- Our first CTF of the semester, LACTF!
- CTF starts at 10 pm CST.
- Many beginner friendly challenges available.

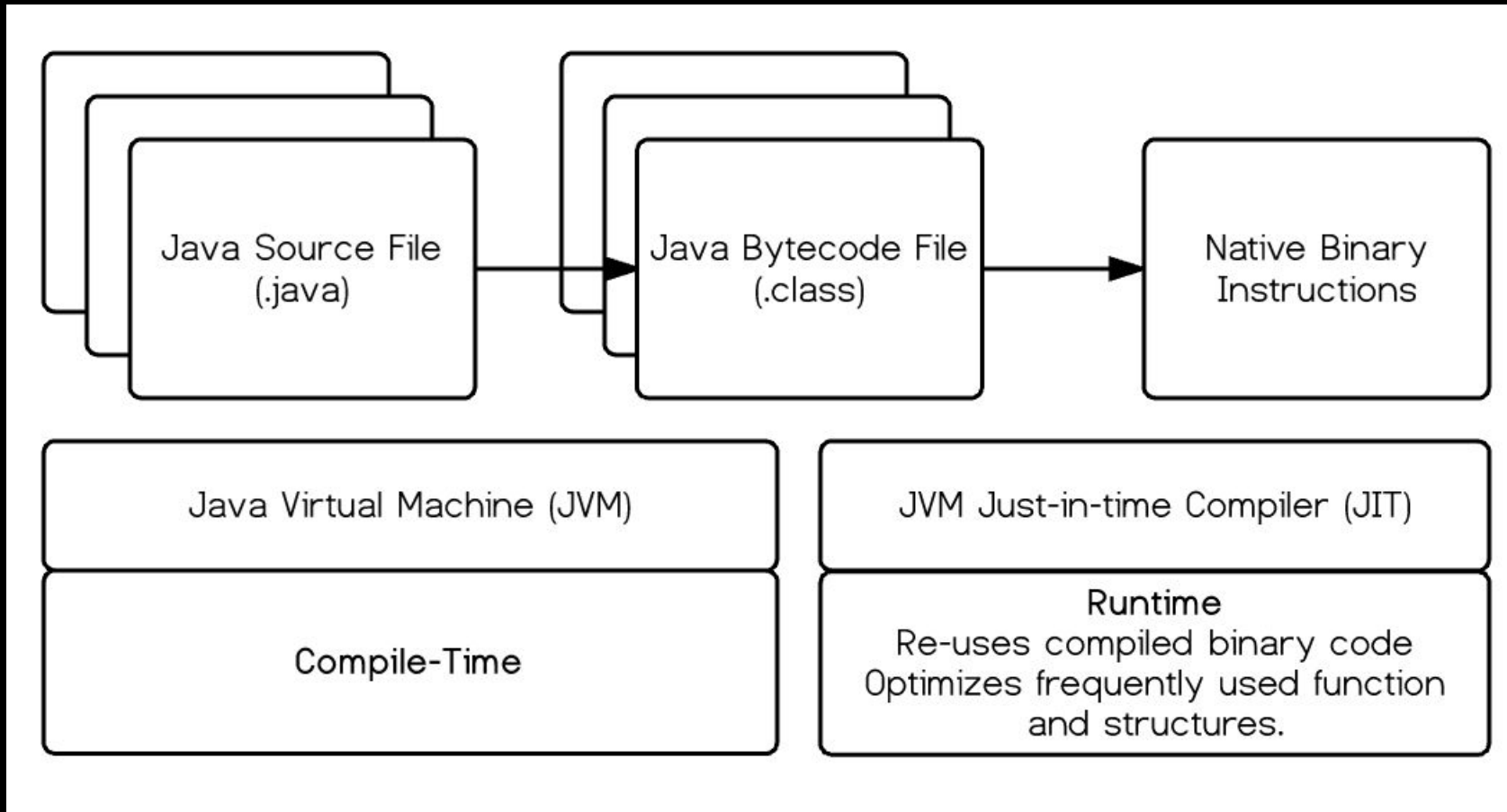


ctf.sigpwny.com

sigpwny{3_billion_devices}



Java Overview



Java File Formats

- .java files are just source code
- .class files
 - Represent a compiled .java file that contains JVM bytecode
 - Uses javac to compile
- .jar files
 - A zip file with .class files and META-INF folder
 - This is the file executed by JVM
- META-INF
 - A folder with all the Java metadata
 - Most important file is MANIFEST.MF, which tells JVM where to start



Android File Format

- Kotlin can also compile to `.class`
- `.apk`
 - Originally extends from `.jar`, now is its own format
 - Is also a fancy ZIP with similar metadata, with extra assets
 - instead of many `.class` files, it now contains a single `.dex` file
- `.dex`
 - stands for Dalvik Executable, used only by Android
 - dexer(dx) tool is used to convert a collection of `.class` into one `.dex`
 - Can be easily converted to jar with tools like dex2jar



Java and Dalvik Virtual Machine

- The JVM is a **stack-based** virtual machine
 - All instructions take arguments using push and pop instructions onto a global stack
 - JVM for all platforms exists to run Java on *3 billion devices*TM
 - Very easy to compile, and also very easy to decompile
- The DVM is a **register-based** virtual machine
 - Probably because ARM has a lot of registers
 - DVM only exists on Android, and performs JIT
 - Google invented Android RunTime to do improve efficiency
 - Why not just compile to ARM assembly? 🙄



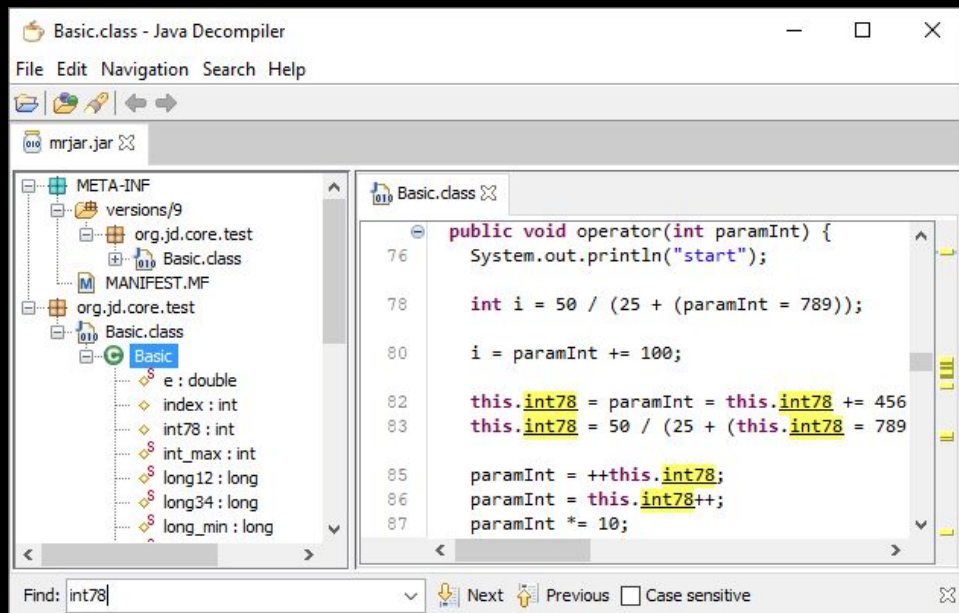
Java Compilation

- You can keep:
 - Class names
 - Function names
 - Class variable names
 - Line numbers (for stack trace printing)
- You only lose
 - Local variable names (you will see a lot of local_x)
 - Comments
- Similar story with DEX



How to decompile

- Many tools to decompile .jar and .dex
 - JAR: jd-gui, Bytecodeviewer (with two decompilers), asmtools (cli),...
 - DEX: JADX, APKtool, Ghidra, dex2jar, ...



The screenshot shows the JD-GUI interface. The left pane displays a file tree for 'mrjar.jar' with 'Basic.class' selected. The right pane shows the decompiled Java code for 'Basic.class':

```
public void operator(int paramInt) {
    System.out.println("start");

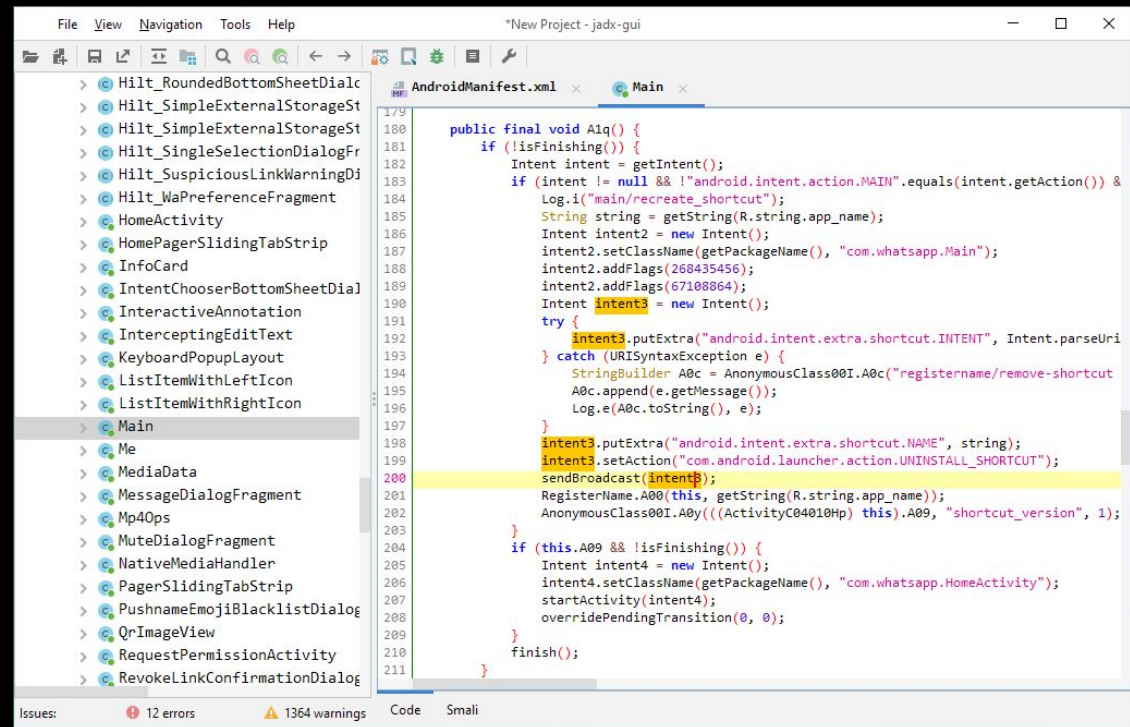
    int i = 50 / (25 + (paramInt = 789));

    i = paramInt += 100;

    this.int78 = paramInt = this.int78 += 456;
    this.int78 = 50 / (25 + (this.int78 = 789));

    paramInt = ++this.int78;
    paramInt = this.int78++;
    paramInt *= 10;
}
```

The search bar at the bottom contains 'int78'.



The screenshot shows the JD-GUI interface with an AndroidManifest.xml file decompiled. The left pane shows a list of classes, with 'Main' selected. The right pane shows the decompiled Java code for 'Main':

```
public final void A1q() {
    if (!isFinishing()) {
        Intent intent = getIntent();
        if (intent != null && !"android.intent.action.MAIN".equals(intent.getAction()) &
            Log.i("main/recreate_shortcut");
        String string = getString(R.string.app_name);
        Intent intent2 = new Intent();
        intent2.setClassName(getPackageName(), "com.whatsapp.Main");
        intent2.addFlags(268435456);
        intent2.addFlags(67108864);
        Intent intent3 = new Intent();
        try {
            intent3.putExtra("android.intent.extra.shortcut.INTENT", Intent.parseUri
        } catch (URISyntaxException e) {
            String builder A0c = AnonymousClass00I.A0c("registername/remove-shortcut
            A0c.append(e.getMessage());
            Log.e(A0c.toString(), e);
        }
        intent3.putExtra("android.intent.extra.shortcut.NAME", string);
        intent3.setAction("com.android.launcher.action.UNINSTALL_SHORTCUT");
        sendBroadcast(intent3);
        RegisterName.A00(this, getString(R.string.app_name));
        AnonymousClass00I.A0y(((ActivityC04010Hp) this).A09, "shortcut_version", 1);
    }
    if (this.A09 && !isFinishing()) {
        Intent intent4 = new Intent();
        intent4.setClassName(getPackageName(), "com.whatsapp.HomeActivity");
        startActivity(intent4);
        overridePendingTransition(0, 0);
    }
    finish();
}
```

The status bar at the bottom indicates 12 errors and 1364 warnings.



Java Obfuscation

- Generally, obfuscators can:
 - Change control flow
 - Rename class, variables and functions
 - Remove or Add functions
 - Encrypt strings
- Popular Obfuscators: ZKM, Allatori, ProGuard, and many open source options
- Android Studio, Obfuscapk, ProGuard, etc can be used to obfuscate apks.



Java Deobfuscation

- General Strategy: Try different deobfuscators
 - Typically the obfuscators have some patterns, or is open source
 - Many deobfuscators are developed to reverse this process
 - Control flow and strings may be fixed to be more readable
 - Class, Variable and Function Names are still lost
- Many deobfuscators available, use [this](#) for starters
- Nathan's tutorial [here](#)



Java Deobfuscation

Original

Obfuscated

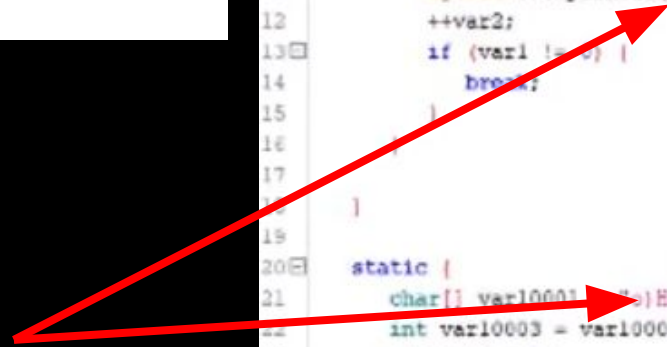
Deobfuscated

```
Fernflower Decompiler - Editable: false
1 package com.sigpwny;
2
3 public class Main {
4     public static void main(String[] args) {
5         for(int i = 0; i < 5; ++i) {
6             System.out.println("Hello world: " + i);
7         }
8     }
9 }
10
11
```

```
Fernflower Decompiler - Editable: false
1 public class a {
2     public static int a;
3     public static int b;
4     private static final String c;
5
6     public static void main(String[] var0) {
7         int var2 = 0;
8         int var1 = b;
9
10        while(var2 < 5) {
11            System.out.println(c + var2);
12            ++var2;
13            if (var1 != 0) {
14                break;
15            }
16        }
17    }
18
19
20    static {
21        char[] var10001 = "H2R\nEH>B:\u0007\n".toCharArray();
22        int var10003 = var10001.length;
23        int var0 = 0;
24        char[] var10002 = var10001;
25        int var2 = var10003;
26        String var1;
27        char[] var4;
28        int var10004;
29        boolean var5;
```

```
Fernflower Decompiler - Editable: false
1 public class a {
2     public static int a;
3     public static int b;
4     private static final String c = "Hello world: ";
5
6     public static void main(String[] var0) {
7         int var2 = 0;
8         int var1 = b;
9
10        while(var2 < 5) {
11            System.out.println("Hello world: " + var2);
12            ++var2;
13            if (var1 != 0) {
14                break;
15            }
16        }
17    }
18
19
```

Encrypted
"Hello world"



Tackling Custom Obfuscation

- We can use ObjectWeb ASM to manually parse and manipulate bytecode
- Since Java standard is constantly updated, new constructs like invokedynamic can create new obfuscators
 - See [writeup](#)
- Worst case scenario, just treat this like an assembly rev and read the bytecode with documentations



Challenges

- Java Rev Suite
- DroidVault



Next Meetings

2025-02-13 • Next Thursday

- Crypto Block Cipher with Sagnik
- Come and learn how to exploit AES!

2025-02-16 • Next Sunday

- SIGPolicy Colab



ctf.sigpwny.com

sigpwny{3_billion_devices}

**Meeting content can be found at
sigpwny.com/meetings.**

