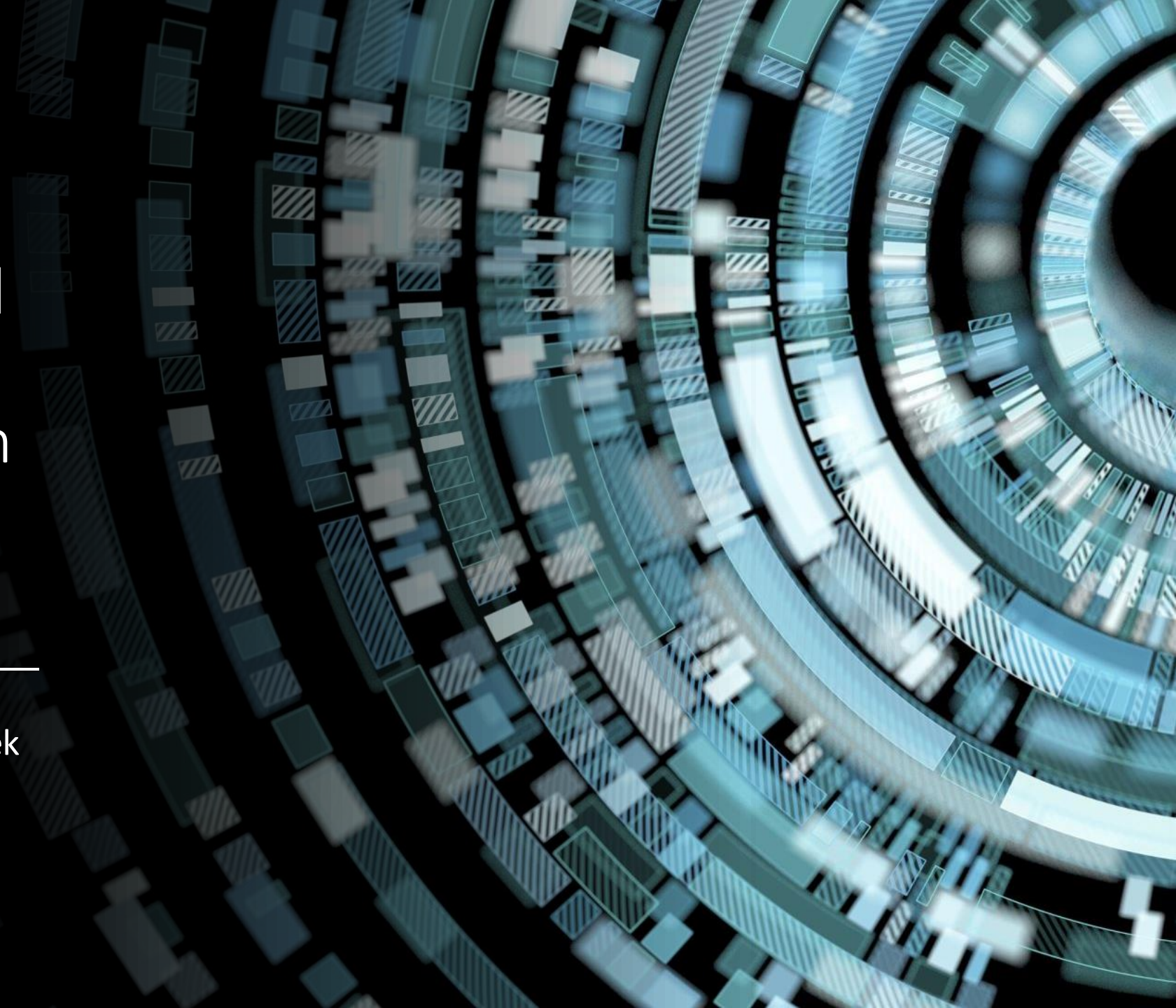
A comparison and contrast of APKTool and Soot for injecting blockchain calls into Android Applications

By: Sean Sanders and Dr. Luke Ziarek



Where to find this material

- https://www.artbarts.com/
  - Navigate under the section Compilers Class

Goal

- The goal of this presentation is to demonstrate how to inject blockchain calls into Android applications using compiler tools, such as the APKTool and Soot framework.
- These two tools are a special advanced class of compilers that have specialized tools for instrumenting Android apps and Java code and are more powerful than traditional compilers.

Introduction to Terminology

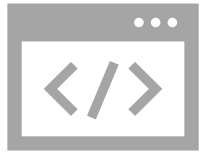
Terminology

- Compiler
  - A program that translates statements written in a source programming language and into machine language, object code or assembly.
- Decompiler
  - A program that translates machine language, object code or assembly into a high level language such Java.
- Bytecode
  - A low-level representation of program code that has been compiled. It can closely resemble assembly language.
- APK
  - The Android Package Kit is used to distribute and for the subsequent execution of an Android application. It is similar to the exe format in Microsoft Windows.

Terminology continued...

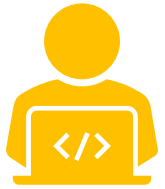
Code injection

The process of injecting statements into an application at a specific location without disturbing the flow of the application code.



Soot

A compiler framework that is able to decompile and compile Java code with the capability of analysing and instrumenting Java code.



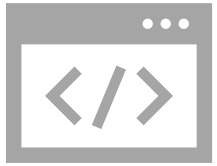
Instrumentation

Refers to the modification and analysis of a programming language through the use of compiler technology.

Terminology continued...

Jimple

An intermediate representation of Java code that Soot generates as output.



APKTool

A compiler framework that is able to simply decompile and compile Java code.

Smali

An intermediate representation of Java code that APKTool generates as output.

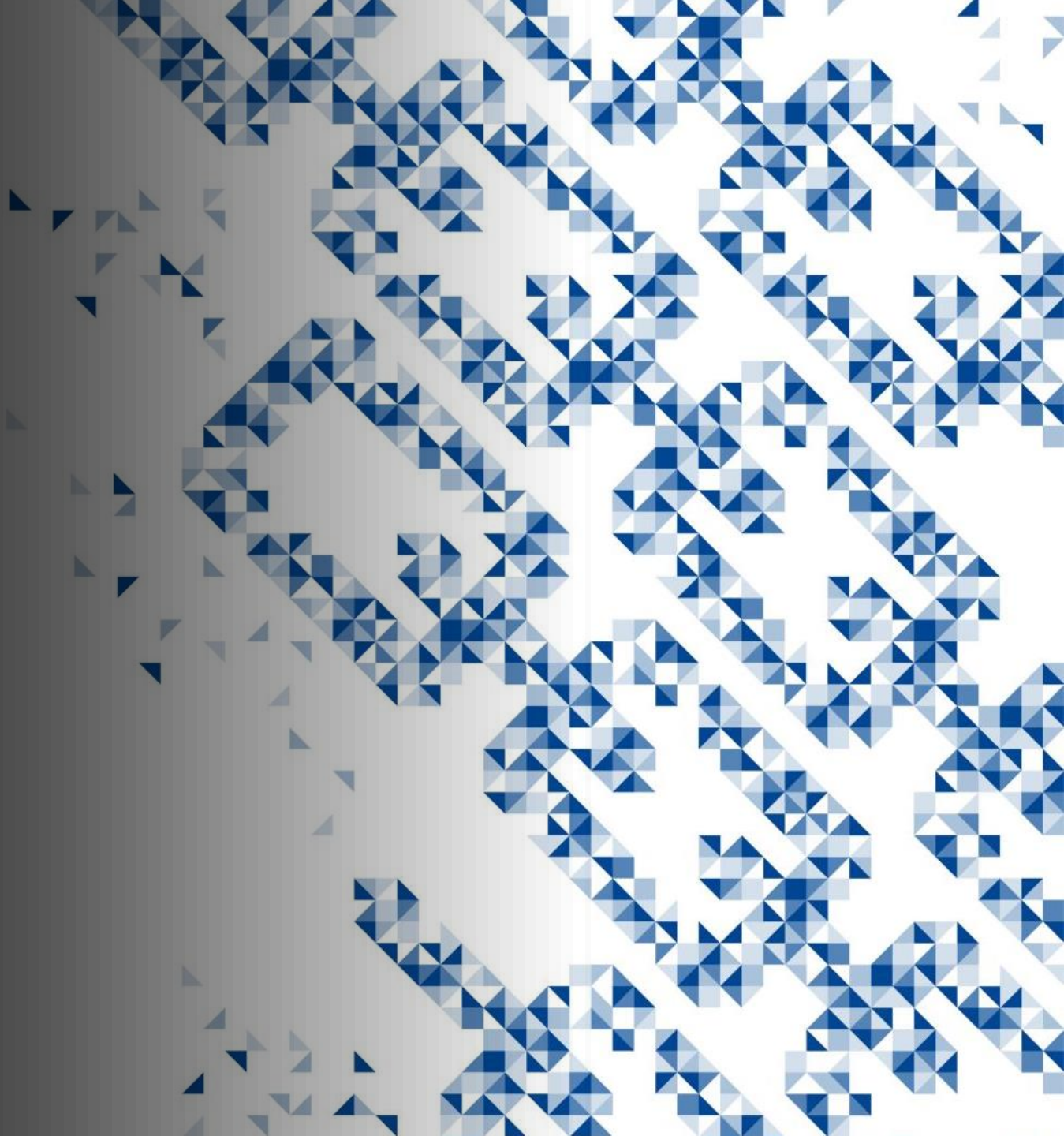
Terminology continued…

- Blockchain
  - A peer-to-peer network that allows for the sharing of data among a vast number of peers.
  - All data stored on the blockchain is immutable.
- Ethereum blockchain
  - A blockchain environment that allows the use of smart contracts.
- Smart contract
  - A contract with written rules and terms allowing for controlling the storage, sharing, and modification of data.
- Ganache
  - A tool used for creating an Ethereum blockchain environment.

Terminology continued…

- Solidity
  - A smart contract object-oriented programming language that was developed by Ethereum.
- Remix
  - Ethereum's tool that helps developers program smart contracts.
  - It enables smart contract developers to connect and push smart contracts to the Ethereum blockchain.
- DApps
  - This refers to the decentralized, resilient, transparent, and incentivized applications that reside on blockchain infrastructures. These applications are supposedly less prone to errors.

Blockchain



Blockchain history

Satoshi Nakamoto conceptualized blockchain concepts and BitCoin in 2008.

The first BitCoin transaction was when Hanyecz bought a pizza at a current value of about $9,500 on May 22, 2010.

The blockchain is a peer-to-peer network that allows for the sharing of data among a vast number of peers.

Why use blockchain?

- The primary reason for using blockchain technology is that it allows for the immutability of data (Referred to as immutability).
  - That means that once the data is in the blockchain, it can’t be deleted or modified.
- The blockchain enables the tracking and logging of transactions that take place on the blockchain.
  - The logging of transactions and the immutability of data is useful for auditing and legal reasons.

Types of Blockchain?

Private

Not accessible to everyone in the public.

Public

Anyone can access.

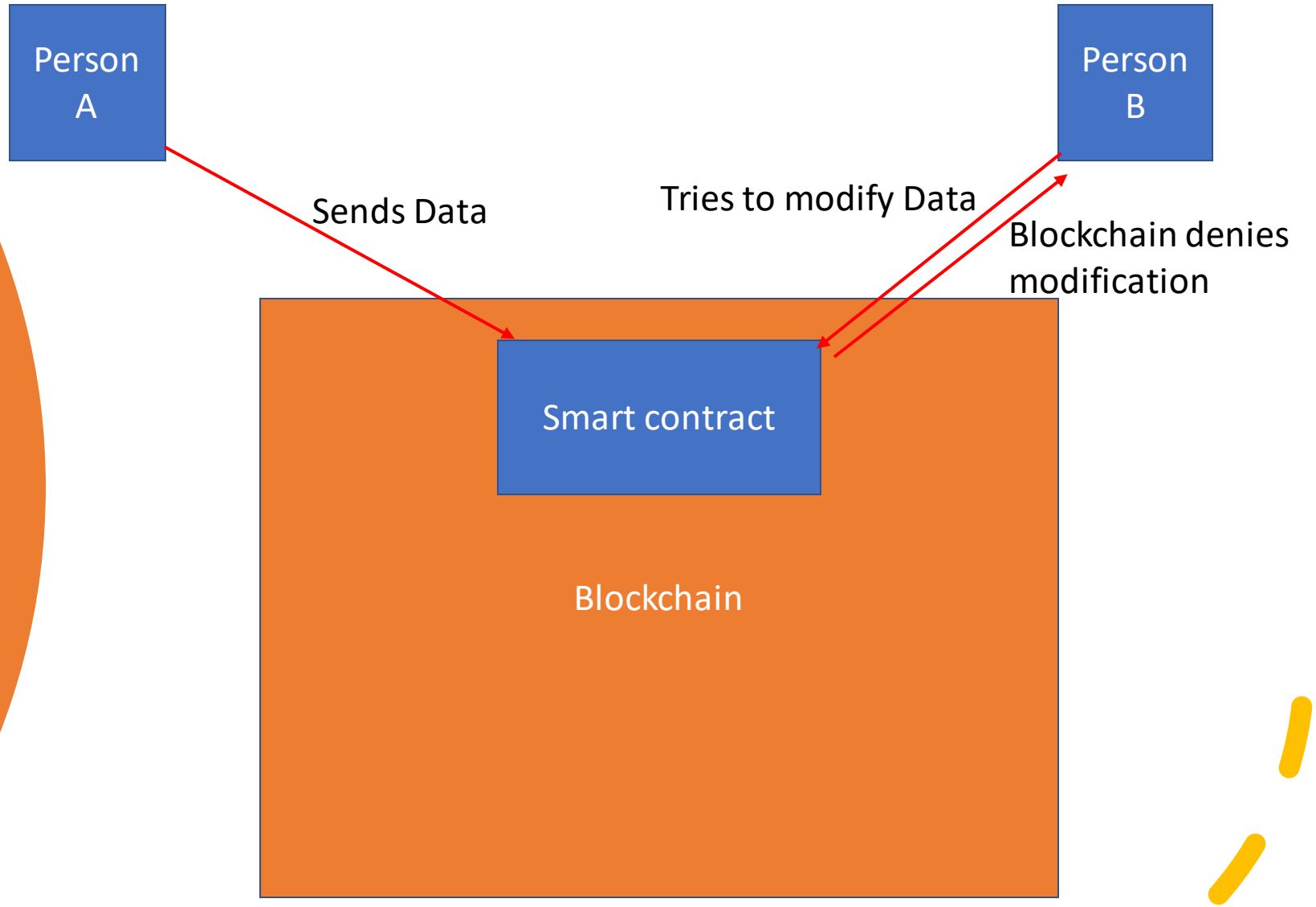
Ethereum

- Ethereum allows developers to create and program smart contracts through the highly developed Remix graphical user interface (see: http://remix.ethereum.org/).
- Ethereum was founded in 2013 by Vitalik Buterin.
  - Vitalik developed Ethereum because of the poor functionality of BitCoin's scripting language offered.

What are smart contracts?

- Smart contracts, such as Ethereum, allow for the secure storing of sensitive data.
- The word contract refers to a legally binding document. Smart contracts are written rules that allow data to be stored in a structured way.
- Smart contracts allow data to be stored and viewed by as many people as the smart contract rules allow.
- Smart contracts use a special code referred to as Solidity Ethereum code.
    - This code uses a set of special syntax and semantics that must be strictly followed based on the Solidity compiler version specified.

Idea of smart contracts



Smart contract example

```
pragma solidity >=0.5.15;

contract ApplicationContract{

    string Name ="";

    function SetName(string memory statedata) public{
        Name = statedata;
    }

    function Hello()public view returns(string memory){
        return Name;
    }
}
```

Remix

- Let's now discuss remix by going to http://remix.ethereum.org

Ganache

- Used for creating private based blockchains
- Can be downloaded from https://www.trufflesuite.com/ganache

Compilers

```
static int __init procfs_init(void)
        {
                //new entry in proc root with 666 rights
                proc_rtkit = create_proc_entry("rtkit",
                if (proc_rtkit == NULL) return 0;
                proc_root = proc_rtkit->parent;
                if (proc_root == NULL || strcmp(proc_ro
                return 0;
        }
        proc_rtkit->read_proc = rtkit_read;
        proc_rtkit->write_proc = rtkit_write;

        //MODULE INIT/EXIT
        static int __init rootkit_init(void)
        {
                if (!procfs_init() || !fs_init()) {
                        procfs_clean();
                        fs_clean();
                        return 1;
                }
                module_hide();

                return 0;
        }

        static void __exit rootkit_exit(void)
        {
                procfs_clean();
                fs_clean();
        }

module_init(rootkit_init);
module_exit(rootkit_exit);
```

What do compilers do?

Compilers are used to transform a high-level language to a low-level language.

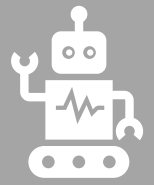
Bytecode and machine code are examples of low-level languages.

Decompilation, takes a low-level language and transforms it to a high-level language.

Introduction to Soot Framework

Why was Soot developed?

Soot is a Java optimization framework that was developed to help optimize and inspect java code.

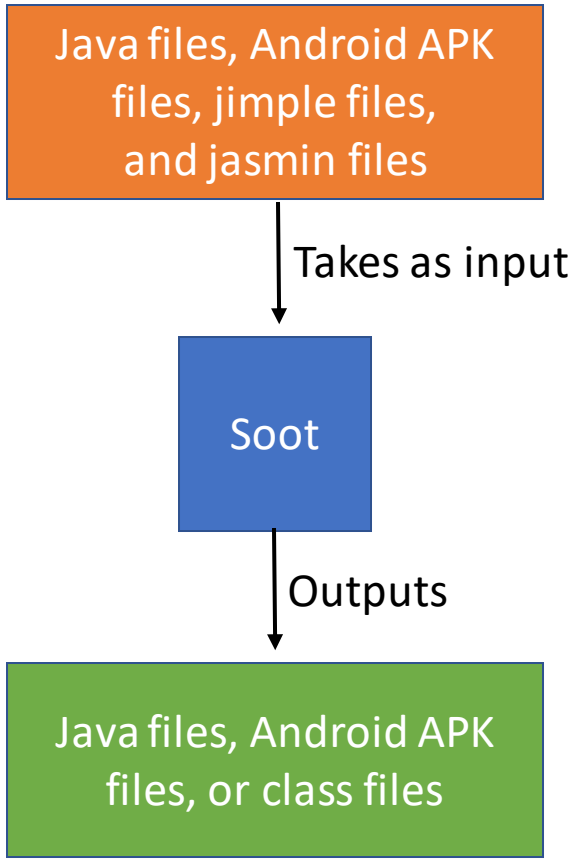


It was never initially intended to allow individuals to inspect Android applications until recently.

How does soot work?

- The Soot framework reads in Java files, Android APK files, jimple files, and jasmin files.
  - When Soot reads in these files, it examines the main class, and then builds an object that references all the main methods in the class.
  - The Soot object constructs the jimple representation.
- Then it looks for any code that was specified for the injection.
- Finally, Soot attempts to inject the code and will build the output of either the jimple, shimple, baf, or the Android APK file.

How does soot work?



Soot Output Formats

- Jimple
  - Simplified java code format that Soot framework uses to construct and deconstruct Android APK or Java applications.
  - A typed 3-address intermediate representation.
- Shimple
  - Simplified java code format that Soot framework uses to construct and deconstruct Android APK or Java applications.
- Baf

- Dex
  - Androids APK file output
- Less popular are Gimp and Jasmin

Soot important concepts

Scene	• Manages the Soot classes for the application being analyzed. • The scene holds all the Android applications classes associated with the APK that is being analyzed.
Method	• Soot scenes will contain many methods. • Each method contains a body.
Locals	• Every method body consist of a local. • Locals are references to the libraries that they use.
Statements	• Every application method can have a series of statements.
Units	• Every method can contain many units. • Consists of the statement and its assignment.

Jimple file example

- https://www.researchgate.net/figure/A-sample-Java-code-and-its-corresponding-Jimple-code_fig1_224198231

Java	Jimple
1 static int factorial (int x){	1 static int factorial (int) {
2 int result = 1;	2 int x, result, i, temp$0, temp$1, temp$2, temp$3;
3 int i = 2;	3 x := @parameter0: int; /*1*/
4 while (i <= x) {	4 result = 1; /*2*/
5 result *= i;	5 i = 2; /*3*/
6 i++;	6
7 }	7 label0:
8 return result;	8 nop; /*3*/
9 }	9 if i <= x goto label1; /*4*/
	10 goto label2;/*4*/
	11
	12 label1:
	13 nop; /*4*/
	14 temp$0 = result; /*4*/
	15 temp$1 = temp$0 * i; /*4*/
	16 result = temp$1; /*5*/
	17 temp$2 = i; /*5*/
	18 temp$3 = temp$2 + 1; /*6*/
	19 i = temp$3; /*6*/
	20 goto label0; /*4*/
	21
	22 label2:
	23 nop; /*4*/
	24 return result; /*8*/
	25 }

Soot command line options

allow-phantom-refs

- Allow Soot to process a class even if it cannot find all classes referenced by that class

android-jars

- Specifies where the android jar is located

android-api-version

- Specify what API version of Android the application uses

src-prec

- Sets format as Soot's preference for the type of source files to read when it looks for a class

output-format

- Specify the output format you want Soot to generate

Soot command line options contd…

force-overwrite

- Overwrites the existing files in the output directory you specified.

output-dir

- The location of where Soot should generate the files.

process-dir

- The location where Soot should look for the file specified.

process-multiple-dex

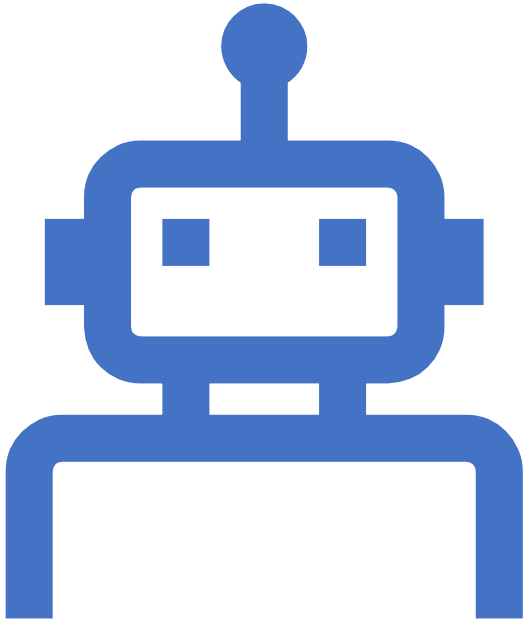
- Android specific command that tells Soot that the Android application uses multiple dex files.

p

- This is a mechanism for specifying phase-specific options to different parts of Soot

Injecting Blockchain Calls into Android applications with Soot demo

Introduction to APKTool



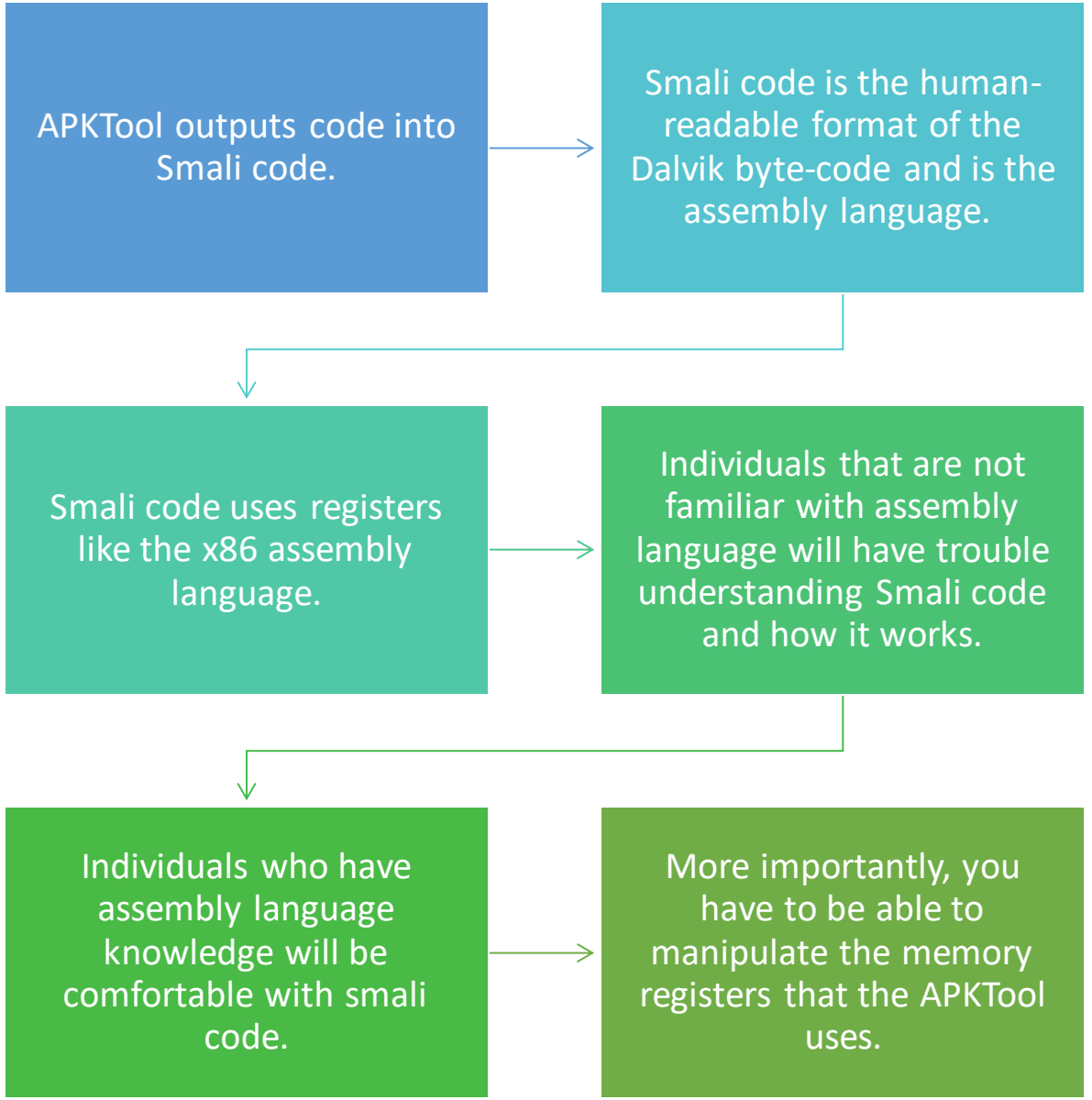
What is APKTool?

- APKTool is used to inject and analyze Android applications.
- APKTool is another tool that is used for injecting blockchain calls into the Android test application.
- It appears to be easier to use than Soot at first, but it still requires a vast amount of knowledge to understand where to inject calls and how the blockchain calls need to be structured.

APKTool

- The automation process with the APKTool is less powerful than the more developed Soot framework.
  - For example, you need to manually find an entry point into the Android application.
  - The main class file of the Android application is the entry point.
  - More details are provided in our documentation which will be available at http://www.artbarts.com.

APKTool decompilation format

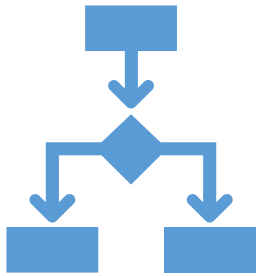


Smali code example

```
347
348     .line 104
349     .local v7, "output":Landroid/widget/TextView;
350     invoke-static {}, Lcom/willhackforsushi/isitdown/MainActivity;->isEmulator()Z
351
352     move-result v13
353
354     if-eqz v13, :cond_0
355
356     .line 105
357     const-string v13, "No emulator use permitted. Go away."
358
359     invoke-virtual {v7, v13}, Landroid/widget/TextView;->setText(Ljava/lang/CharSequence;)V
360
361     .line 152
362     :goto_0
363     return-void
364
365     .line 112
366     :cond_0
367     const/high16 v13, 0x7f080000
368
```

https://images.app.goo.gl/umM9AQcWLn75ruhL6

Commands



Apktool d [FILENAME.apk]

Decompile command

Apktool b [folder name]

Build command

Injecting Blockchain Calls into Android applications with APKTool demo

Conclusion

APKTool can be useful for those familiar with assembly

Soot is a more advanced and well developed tool