Core C++ 2021

Obfuscation and beyond: securing your binary

by Alex Cohn

Reblaze

August 25, 2021

Introducti

2012

Founded

Hu

E

ly

96%

Average annual retention

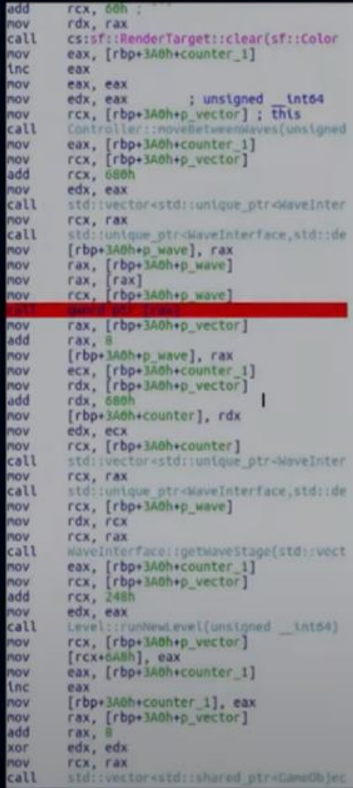
Note: Reblaze clients are month-to-month, with no contractual lock-in. They can leave at any time.

Alex C

@sash

Agenda

1. Why Obfuscation?
2. Why is C++ good for it?
3. How to hide my Java secrets in C++?
4. Is C++ secure enough?
5. White box cryptography
6. Active defence
7. Who can help?

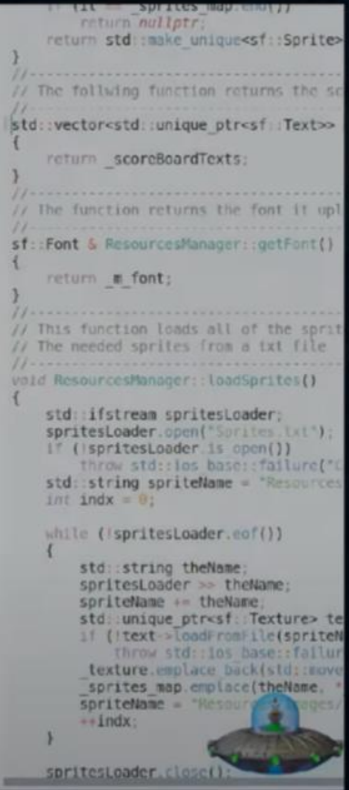


```
add     rcx, 60h ;
mov     rdx, rax
call    cs:sf::RenderTarget::clear(sf::Color
mov     eax, [rbp+3A0h+counter_1]
inc     eax
mov     eax, eax
mov     edx, eax        ; unsigned __int64
mov     rcx, [rbp+3A0h+p_vector] ; this
call    Controller::moveBetweenWaves(unsigned
mov     eax, [rbp+3A0h+counter_1]
mov     rcx, [rbp+3A0h+p_vector]
add     rcx, 680h
mov     edx, eax
call    std::vector<std::unique_ptr<WaveInter
mov     rcx, rax
call    std::unique_ptr<WaveInterface,std::de
mov     [rbp+3A0h+p_wave], rax
mov     rax, [rbp+3A0h+p_wave]
mov     rax, [rax]
mov     rcx, [rbp+3A0h+p_wave]
call    qword ptr [rax]
mov     rax, [rbp+3A0h+p_vector]
add     rax, 8
mov     [rbp+3A0h+p_wave], rax
mov     ecx, [rbp+3A0h+counter_1]
mov     rdx, [rbp+3A0h+p_vector]
add     rdx, 680h
mov     [rbp+3A0h+counter], rdx
mov     edx, ecx
mov     rcx, [rbp+3A0h+counter]
call    std::vector<std::unique_ptr<WaveInter
mov     rcx, rax
call    std::unique_ptr<WaveInterface,std::de
mov     rcx, [rbp+3A0h+p_wave]
mov     rdx, rcx
mov     rcx, rax
call    WaveInterface::getWaveStage(std::vect
mov     eax, [rbp+3A0h+counter_1]
mov     rcx, [rbp+3A0h+p_vector]
add     rcx, 248h
mov     edx, eax
call    Level::runNewLevel(unsigned __int64)
mov     rcx, [rbp+3A0h+p_vector]
mov     [rcx+6A8h], eax
mov     eax, [rbp+3A0h+counter_1]
inc     eax
mov     [rbp+3A0h+counter_1], eax
mov     rax, [rbp+3A0h+p_vector]
add     rax, 8
xor     edx, edx
mov     rcx, rax
call    std::vector<std::shared_ptr<GameObjec
```

Behind Enemy Lines Reverse Engineering C++ in Modern Ages

Gal Zaban

@0xgalz



```
    if (it == _sprites_map.end())
        return nullptr;
    return std::make_unique<sf::Sprite>
}
//-----------------------------------
// The follwing function returns the sc
//-----------------------------------
std::vector<std::unique_ptr<sf::Text>>
{
    return _scoreBoardTexts;
}
//-----------------------------------
// The function returns the font it upl
//-----------------------------------
sf::Font & ResourcesManager::getFont()
{
    return _m_font;
}
//-----------------------------------
// This function loads all of the sprit
// The needed sprites from a txt file
//-----------------------------------
void ResourcesManager::loadSprites()
{
    std::ifstream spritesLoader;
    spritesLoader.open("Sprites.txt");
    if (!spritesLoader.is_open())
        throw std::ios_base::failure("C
    std::string spriteName = "Resources
    int indx = 0;

    while (!spritesLoader.eof())
    {
        std::string theName;
        spritesLoader >> theName;
        spriteName += theName;
        std::unique_ptr<sf::Texture> te
        if (!text->loadFromFile(spriteN
            throw std::ios_base::failur
        _texture.emplace_back(std::move
        _sprites_map.emplace(theName, *
        spriteName = "Resources/Images/
        ++indx;
    }

    spritesLoader.close();
```

THE ACADEMIC COLLEGE of TEL AVIV-YAFFO 25 YEARS

Core C++ 2019

2. But ProGuard is free…

1. Please do use ProGuard

   ProGuard is a ***free* shrinker, optimizer**, obfuscator, and **preverifier** for Java bytecode

2. Your strings are visible

   How to store the Credentials securely in Android Hiding Secrets in Android Apps

3. How much code you want to maintain?

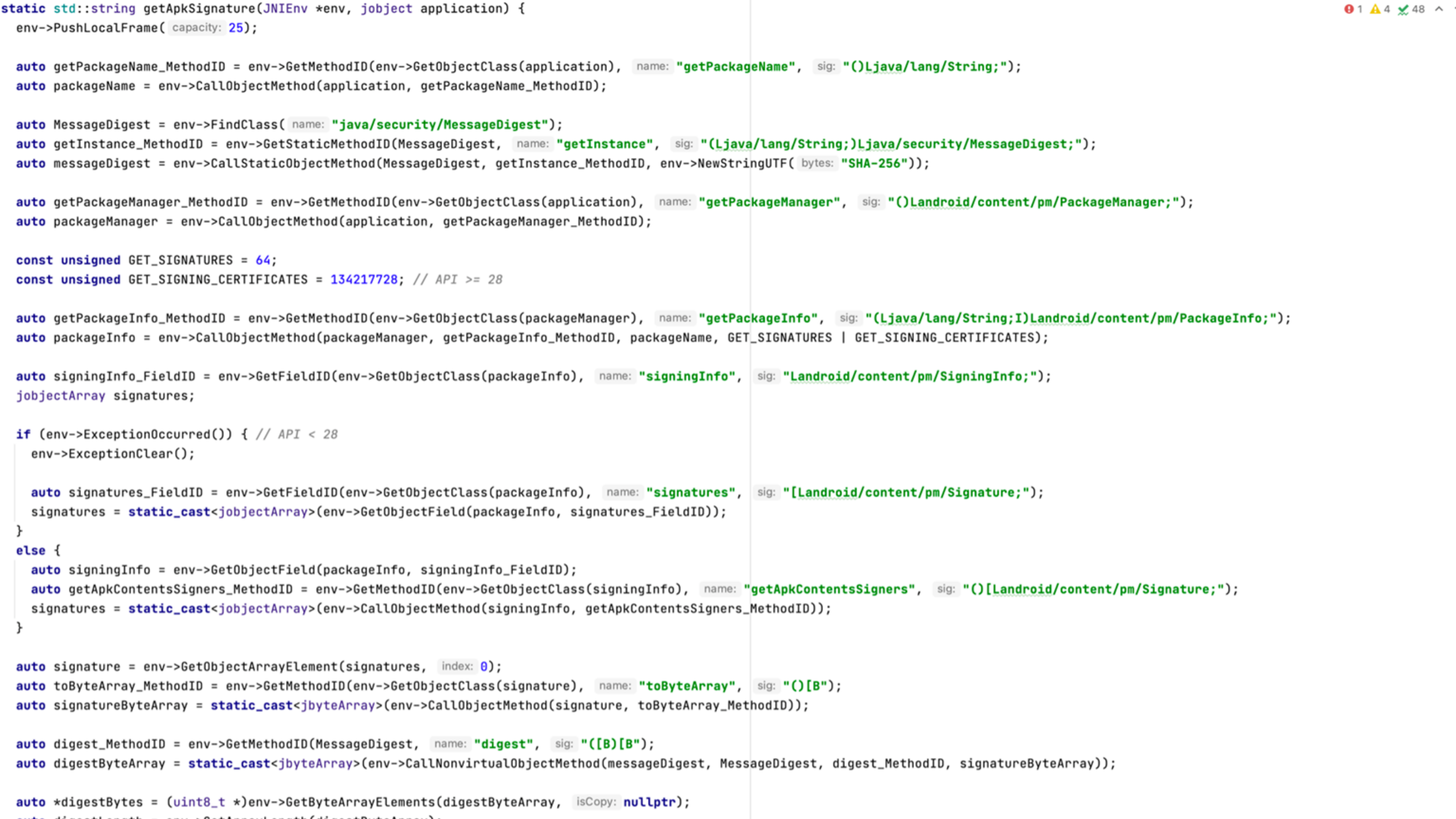
   StringCare Paranoid SwiftShield Objc-Obfuscator …

4. Is Swift compiler actually obfuscating?

   There is a lot of symbols and metadata just waiting to be explored!

5. Is Flutter compiler actually obfuscating?

6. Switch to C++, enjoy the modern language and profit!

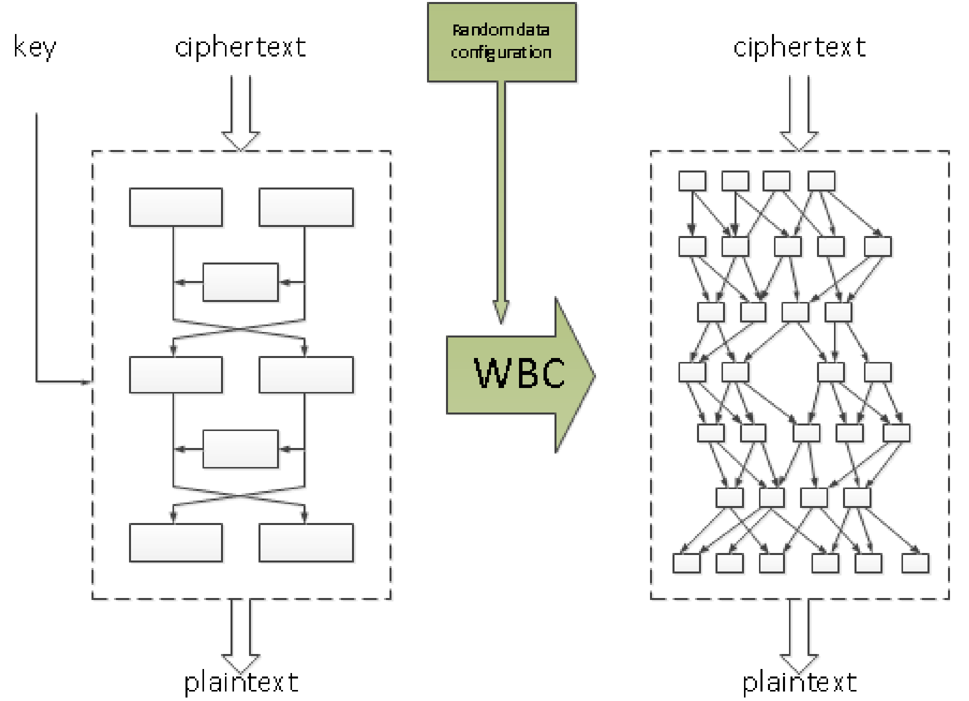


```
static std::string getApkSignature(JNIEnv *env, jobject application) {
  env->PushLocalFrame( capacity: 25);

  auto getPackageName_MethodID = env->GetMethodID(env->GetObjectClass(application),  name: "getPackageName",  sig: "()Ljava/lang/String;");
  auto packageName = env->CallObjectMethod(application, getPackageName_MethodID);

  auto MessageDigest = env->FindClass( name: "java/security/MessageDigest");
  auto getInstance_MethodID = env->GetStaticMethodID(MessageDigest,  name: "getInstance",  sig: "(Ljava/lang/String;)Ljava/security/MessageDigest;");
  auto messageDigest = env->CallStaticObjectMethod(MessageDigest, getInstance_MethodID, env->NewStringUTF( bytes: "SHA-256"));

  auto getPackageManager_MethodID = env->GetMethodID(env->GetObjectClass(application),  name: "getPackageManager",  sig: "()Landroid/content/pm/PackageManager;");
  auto packageManager = env->CallObjectMethod(application, getPackageManager_MethodID);

  const unsigned GET_SIGNATURES = 64;
  const unsigned GET_SIGNING_CERTIFICATES = 134217728; // API >= 28

  auto getPackageInfo_MethodID = env->GetMethodID(env->GetObjectClass(packageManager),  name: "getPackageInfo",  sig: "(Ljava/lang/String;I)Landroid/content/pm/PackageInfo;");
  auto packageInfo = env->CallObjectMethod(packageManager, getPackageInfo_MethodID, packageName, GET_SIGNATURES | GET_SIGNING_CERTIFICATES);

  auto signingInfo_FieldID = env->GetFieldID(env->GetObjectClass(packageInfo),  name: "signingInfo",  sig: "Landroid/content/pm/SigningInfo;");
  jobjectArray signatures;

  if (env->ExceptionOccurred()) { // API < 28
    env->ExceptionClear();

    auto signatures_FieldID = env->GetFieldID(env->GetObjectClass(packageInfo),  name: "signatures",  sig: "[Landroid/content/pm/Signature;");
    signatures = static_cast<jobjectArray>(env->GetObjectField(packageInfo, signatures_FieldID));
  }
  else {
    auto signingInfo = env->GetObjectField(packageInfo, signingInfo_FieldID);
    auto getApkContentsSigners_MethodID = env->GetMethodID(env->GetObjectClass(signingInfo),  name: "getApkContentsSigners",  sig: "()[Landroid/content/pm/Signature;");
    signatures = static_cast<jobjectArray>(env->CallObjectMethod(signingInfo, getApkContentsSigners_MethodID));
  }

  auto signature = env->GetObjectArrayElement(signatures,  index: 0);
  auto toByteArray_MethodID = env->GetMethodID(env->GetObjectClass(signature),  name: "toByteArray",  sig: "()[B");
  auto signatureByteArray = static_cast<jbyteArray>(env->CallObjectMethod(signature, toByteArray_MethodID));

  auto digest_MethodID = env->GetMethodID(MessageDigest,  name: "digest",  sig: "([B)[B");
  auto digestByteArray = static_cast<jbyteArray>(env->CallNonvirtualObjectMethod(messageDigest, MessageDigest, digest_MethodID, signatureByteArray));

  auto *digestBytes = (uint8_t *)env->GetByteArrayElements(digestByteArray,  isCopy: nullptr);
  auto digestLength = env->GetArrayLength(digestByteArray);
```

4. Harden your C++ code

1. Use defensive coding techniques.
2. Test your code including “unlikely” edge cases.
3. Check compiler warnings (e.g. they can discover a potential formatting attack).
4. `-D_FORTIFY_SOURCE=2` will add some extra checks e.g. some cases of buffer overflow.
5. `-fstack-protector`
6. `-fsanitize=self-stack`
7. `-fsanitize=cfi` for control flow integrity.
8. Use Address Space Layout Randomization.
9. Hide your symbols (may be tricky).
10. Protect your Global Offset Table (e.g. `-Wl,-z,relro`).

5. White Box Cryptography



5. Rely on compiler (it’s LLVM!)

Using **opt** command or rebuilding the whole toolchain.

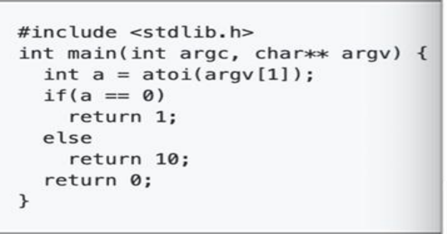
Apple LLVM is now open source – take it and add your passes.

I used DeClang in my work that inspired this presentation.

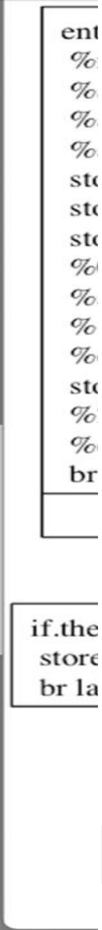
Control:

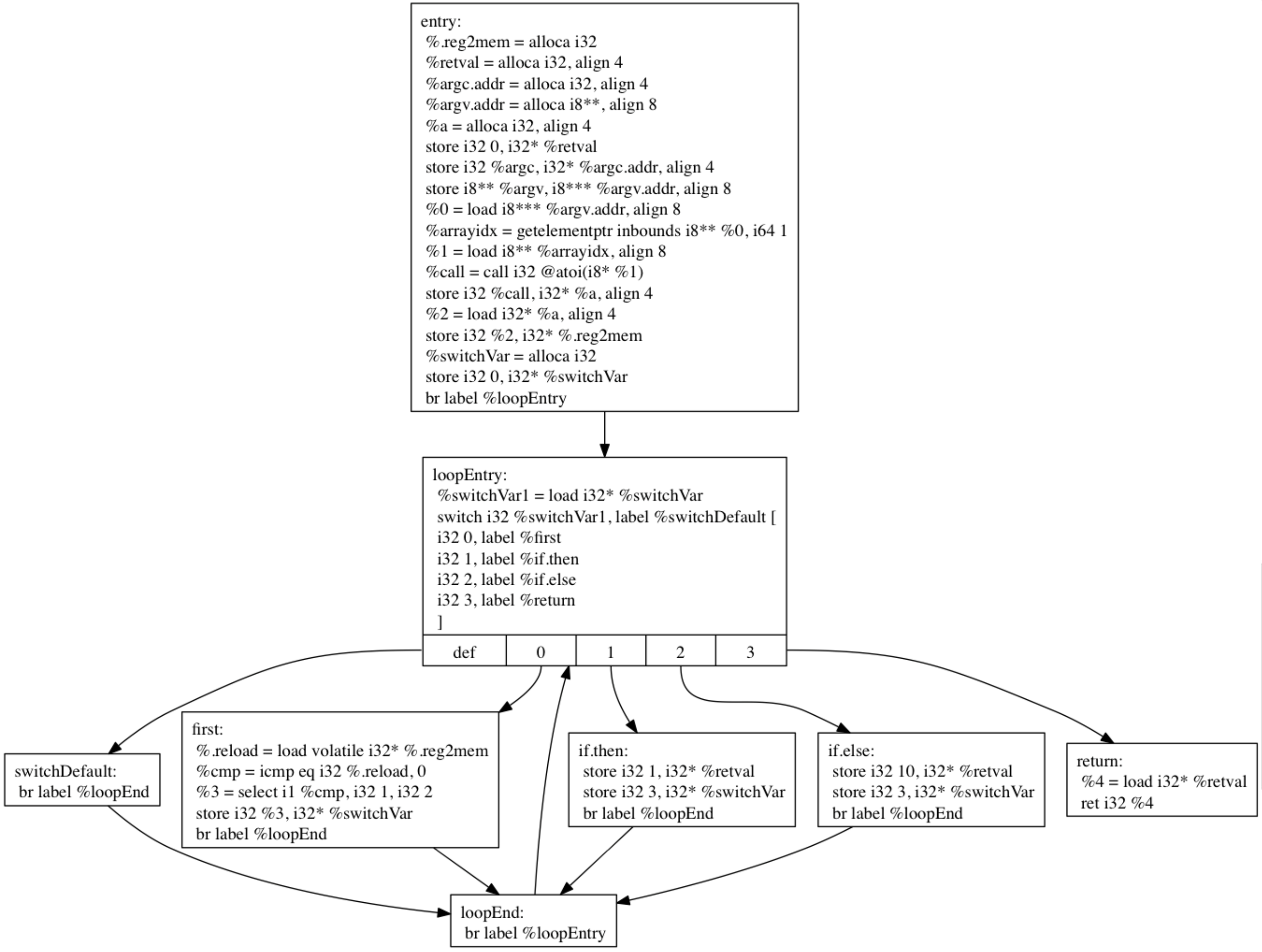
- Annotations embedded in your code.
- external configuration files, **yaml** or **json**.
- Or in-line parameters e.g. `-mllvm -fla`.

Check the results, tune performance vs. safety.



```
#include <stdlib.h>
int main(int argc, char** argv) {
  int a = atoi(argv[1]);
  if(a == 0)
    return 1;
  else
    return 10;
  return 0;
}
```





```
entry:
 %.reg2mem = alloca i32
 %retval = alloca i32, align 4
 %argc.addr = alloca i32, align 4
 %argv.addr = alloca i8**, align 8
 %a = alloca i32, align 4
 store i32 0, i32* %retval
 store i32 %argc, i32* %argc.addr, align 4
 store i8** %argv, i8*** %argv.addr, align 8
 %0 = load i8*** %argv.addr, align 8
 %arrayidx = getelementptr inbounds i8** %0, i64 1
 %1 = load i8** %arrayidx, align 8
 %call = call i32 @atoi(i8* %1)
 store i32 %call, i32* %a, align 4
 %2 = load i32* %a, align 4
 store i32 %2, i32* %.reg2mem
 %switchVar = alloca i32
 store i32 0, i32* %switchVar
 br label %loopEntry
```

```
loopEntry:
 %switchVar1 = load i32* %switchVar
 switch i32 %switchVar1, label %switchDefault [
 i32 0, label %first
 i32 1, label %if.then
 i32 2, label %if.else
 i32 3, label %return
 ]
```

```
switchDefault:
 br label %loopEnd
```

```
first:
 %.reload = load volatile i32* %.reg2mem
 %cmp = icmp eq i32 %.reload, 0
 %3 = select i1 %cmp, i32 1, i32 2
 store i32 %3, i32* %switchVar
 br label %loopEnd
```

```
if.then:
 store i32 1, i32* %retval
 store i32 3, i32* %switchVar
 br label %loopEnd
```

```
if.else:
 store i32 10, i32* %retval
 store i32 3, i32* %switchVar
 br label %loopEnd
```

```
return:
 %4 = load i32* %retval
 ret i32 %4
```

```
loopEnd:
 br label %loopEntry
```

CFG for 'main' function

6. RASP: anti-tampering

Runtime Application Self Protection

- Detect compromised environment (e.g. root)
- Detect debugger and hooks (e.g. **FЯIDA**)
    - Looking for the traces in memory map
    - Trying to sniff their sockets
- Detect code changes
    - Verify checksums, signatures
- Detect early (before the attacker grabs control)
    - Load your code on app start
    - Use C++ global var initialization

7. Commercial solutions

◉ **SDK**

❑ AppShield | Quarkslab

❑ DexGuard & iXGuard | GuardSquare

☁ **Cloud**

❑ SHIELD | Promon

❑ DexProtector | Licel

❑ ONEShield | Appdome