Creating Infinite Possibilities.

SCTE
a subsidiary of CableLabs

# Bitcode Obfuscation

## Protecting Software Without Source Code Access

Rafie Shamsaasef

Director of Software Engineering
CommScope Inc.
rafie.shamsaasef@commscope.com

SCTE CABLE-TEC EXPO 22
SEPTEMBER 19-22 • PHILADELPHIA, PA

2022 Fall Technical Forum
SCTE • CABLELABS • NCTA

# Agenda

- Introduction
- Software Obfuscation Techniques
- Bitcode Obfuscation
- Example
- Conclusion

© 2022 Society of Cable Telecommunications Engineers, Inc. a subsidiary of CableLabs 

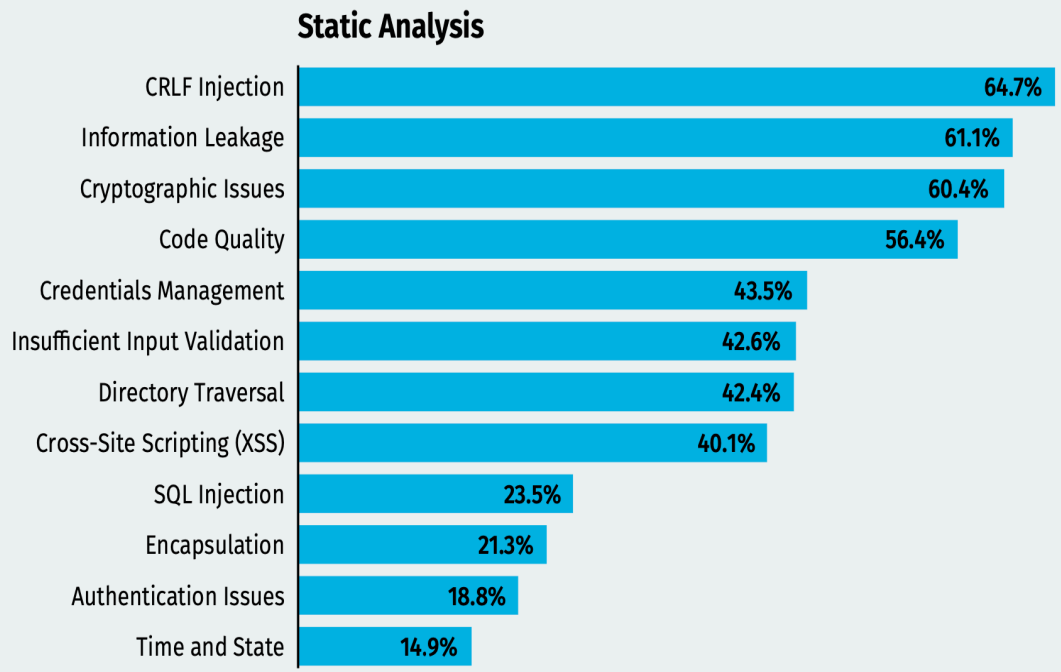
# What is software obfuscation?

- To generate source or machine code that is difficult for humans and automated tools to make sense of
- To protect applications against **theft of intellectual property, reverse engineering, tampering, software piracy and malware attacks**
- To achieve unintelligible code without introducing unacceptable levels of overhead
- To hide sensitive and secretive codes from static analyses
- **Obfuscation is not encryption**


© 2022 Society of Cable Telecommunications Engineers, Inc. a subsidiary of CableLabs 

# Protect against static analysis and reverse engineering

**Figure 10: Top software weaknesses discovered by scan type**




| Lack of binary protections | | | | | | | |
|---|---|---|---|---|---|---|---|
| Retail bank | Credit card issuer | Mobile payments | HSA bank | Retail brokerage | Health insurer | Auto insurer | Crypto-currency |
| **Insecure data storage** | | | | | | | |
| Retail bank | Credit card issuer | Mobile payments | HSA bank | Retail brokerage | Health insurer | Auto insurer | Crypto-currency |
| **Unintended data leakage** | | | | | | | |
| Retail bank | Credit card issuer | Mobile payments | HSA bank | Retail brokerage | Health insurer | Auto insurer | Crypto-currency |
| **Client-side injection** | | | | | | | |
| Retail bank | Credit card issuer | Mobile payments | HSA bank | Retail brokerage | Health insurer | Auto insurer | Crypto-currency |
| **Weak encryption** | | | | | | | |
| Retail bank | Credit card issuer | Mobile payments | HSA bank | Retail brokerage | Health insurer | Auto insurer | Crypto-currency |
| **Implicit trust of all certificates** | | | | | | | |
| Retail bank | Retail brokerage | Crypto-currency | | | | | |
| **Execution of activities using root** | | | | | | | |
| Retail bank | Credit card issuer | HSA bank | Retail brokerage | Health insurer | Auto insurer | Crypto-currency | |
| **World readable/writable files and directories** | | | | | | | |
| Retail bank | | | | | | | |
| **Private key exposure** | | | | | | | |
| Retail bank | HSA bank | Retail brokerage | Auto insurer | Crypto-currency | | | |
| **Exposure of database parameters and SQL queries** | | | | | | | |
| Retail bank | Credit card issuer | Mobile payments | HSA bank | Retail brokerage | Health insurer | Auto insurer | Crypto-currency |
| **Insecure random number generation** | | | | | | | |
| Retail bank | Credit card issuer | Mobile payments | HSA bank | Retail brokerage | Health insurer | Auto insurer | Crypto-currency |

**Source**: State+of+Software+Security+v12 https://www.veracode.com/state -of-software -security -report and https://threatpost.com/financial -apps-are-ripe -for-exploit -via -reverse-engineering/143348/

© 2022 Society of Cable Telecommunications Engineers, Inc. a subsidiary of CableLabs | expo.scte.org

# Approaches

- Wide range of **heuristic** approaches to manipulate source code and binary **without altering the logic**
- Source code to **source code obfuscation** techniques are **not effective**
  - Reverse engineering tools can easily de-obfuscate the code
- **Java obfuscations are not effective**
  - Unlike C++, de-compilation of Java programs is a much simpler task
  - Class hierarchy, high-level statements, names of classes, methods and fields can be retrieved from class files
  - It comes down to **renaming and hiding** some of the instructions
- Native languages like **C/C++ are more suitable for obfuscation**
  - Practically proven to be secure
  - LLVM based support  (Low Level Virtual Machine)
    - Obfuscate at binary level

© 2022 Society of Cable Telecommunications Engineers, Inc. a subsidiary of CableLabs 

# Control Flow Obfuscation

- **Instruction substitution** replaces assembly-level operations with randomly chosen code blocks that perform the same operation in different ways.

- **Bogus-control-flow** adds entry points that evaluate complex expressions to determine the outcome of conditional jumps: either to jump to valid program code or to randomly altered “junk” code blocks.

- **Control-flow-flattening** rearranges a program’s basic blocks in a randomized manner to give a program a uniformly random structure.

- **Virtual-machine interpreter** obfuscation incorporates known hard mathematical problems in the computation of the control-flow to further increase the cost of reverse-engineering attacks.

© 2022 Society of Cable Telecommunications Engineers, Inc. a subsidiary of CableLabs | expo.scte.org

# Data Flow Obfuscation

- **Randomized branch encoding** involves representing data-related logical and mathematical operations as a branching program composed of a sequence of permutations. Sequences of these branching programs are then concatenated together. When these programs are converted back to machine code, the result is uniformly randomized and unintelligible code that bears no resemblance to the original algorithm.

- **Randomized input, output encodings** can be used to make the obfuscated code even harder to reverse-engineer, as well as protecting constants and allowing seamless secure chaining to and from the obfuscate application.

© 2022 Society of Cable Telecommunications Engineers, Inc. a subsidiary of CableLabs 

Creating Infinite Possibilities.

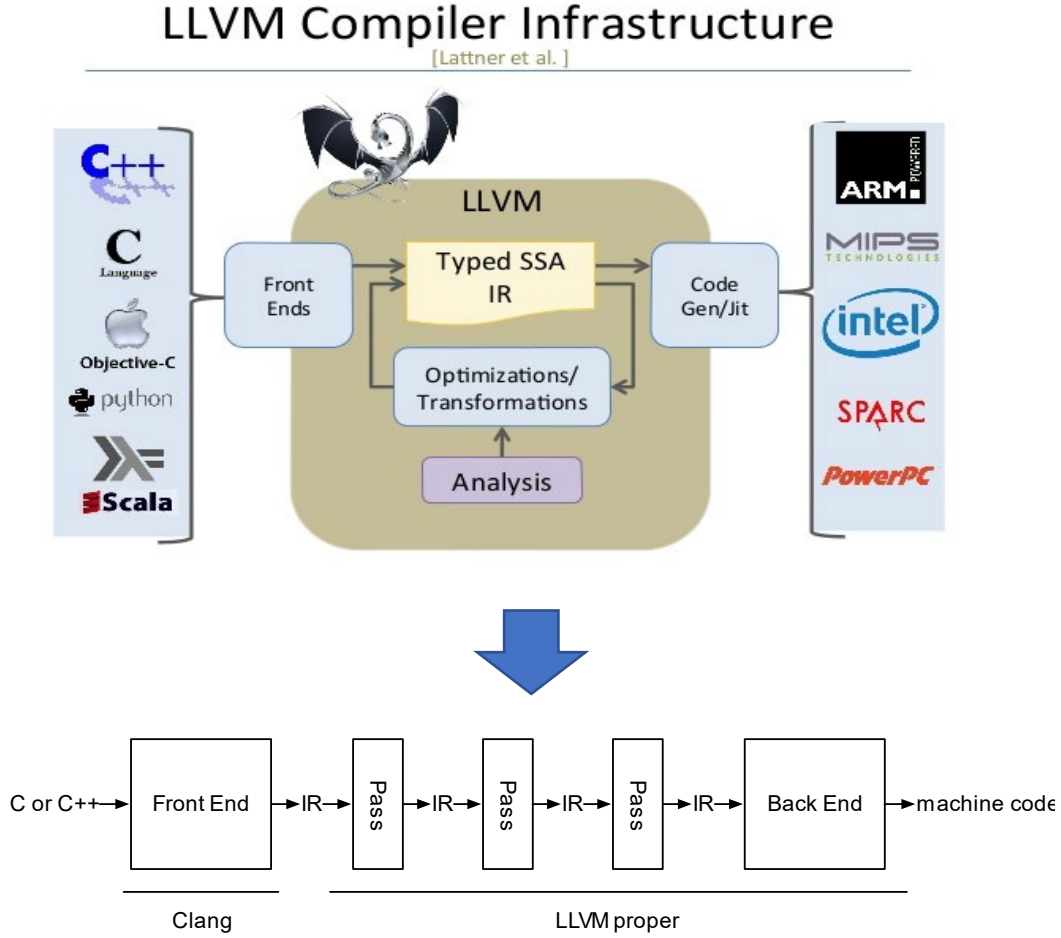
SCTE
a subsidiary of CableLabs®

# Bitcode Obfuscation

© 2022 Society of Cable Telecommunications Engineers, Inc. a subsidiary of CableLabs 

# Bitcode and LLVM

- LLVM is Low Level Virtual Machine:
  - LLVM is a set of compiler and toolchain technologies that can be used to develop a front end for any programming language and a back end for any instruction set architecture
  - Popular LLVM-based tools are Clang, Apple XCode, Microsoft Visual Studio, etc.
  - Supported languages are C/C++, Objectve C, Swift, GO, Rust, etc.
- Bitcode
  - **A platform-independent, universal low-level intermediate representation** (IR) used by LLVM compilers and tools.

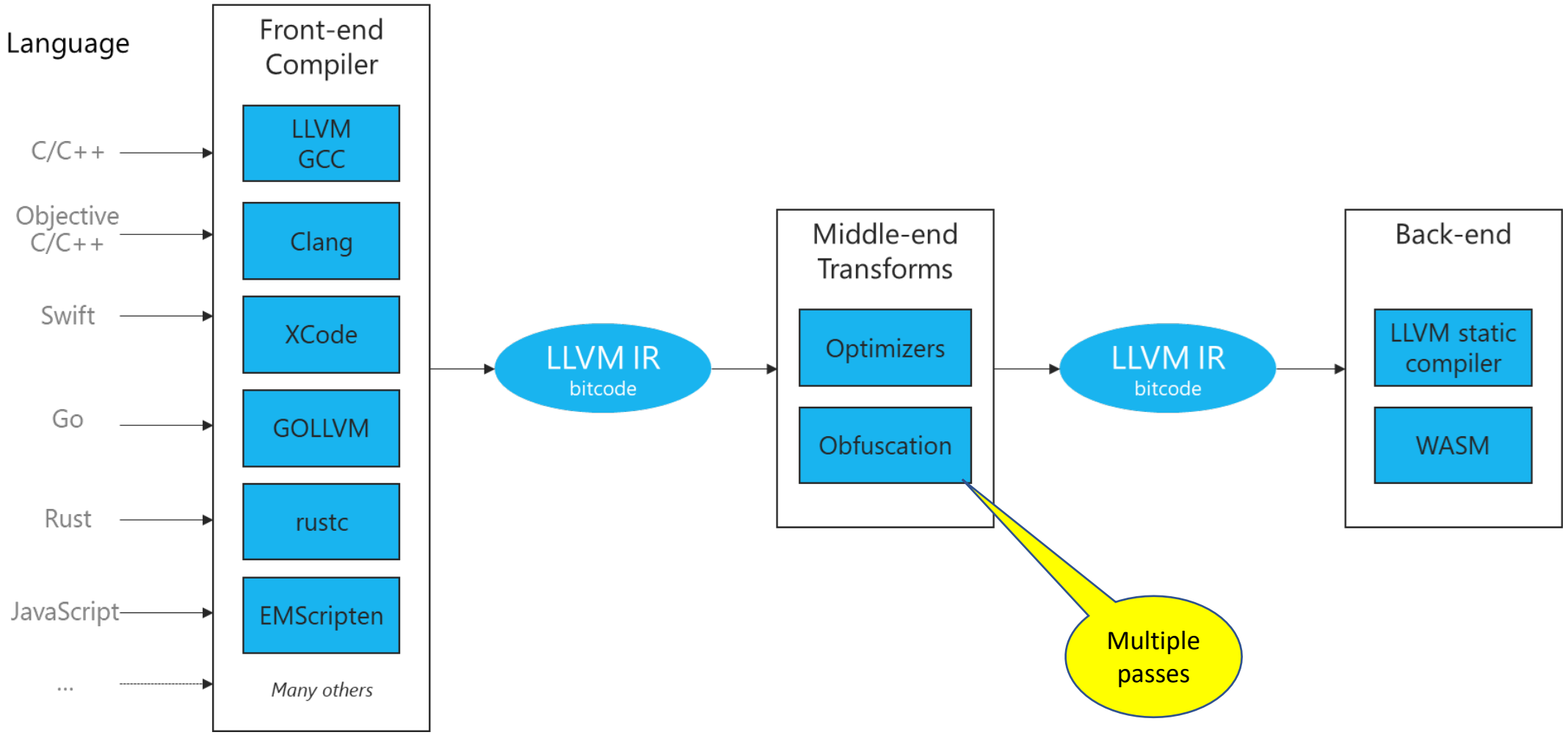



https://www.cs.cornell.edu/~asampson/blog/llvm.html

https://stackoverflow.com/questions/2354725/what-exactly-is-llvm

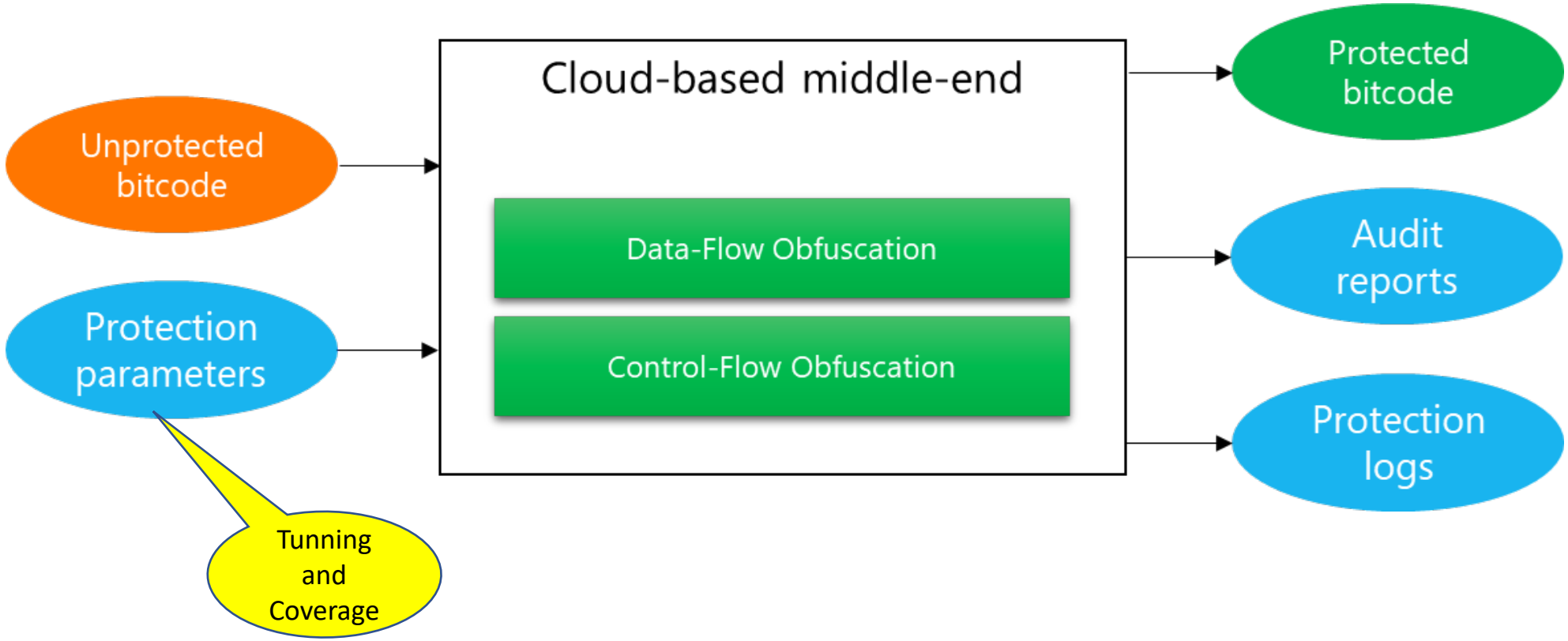
© 2022 Society of Cable Telecommunications Engineers, Inc. a subsidiary of CableLabs | expo.scte.org

# Bitcode Obfuscation with LLVM




© 2022 Society of Cable Telecommunications Engineers, Inc. a subsidiary of CableLabs 

# How does it work?




© 2022 Society of Cable Telecommunications Engineers, Inc. a subsidiary of CableLabs 

# Creating Infinite Possibilities.

SCTE
a subsidiary of CableLabs®

## Example

© 2022 Society of Cable Telecommunications Engineers, Inc. a subsidiary of CableLabs 

# Bitcode Obfuscation Example



## Output:

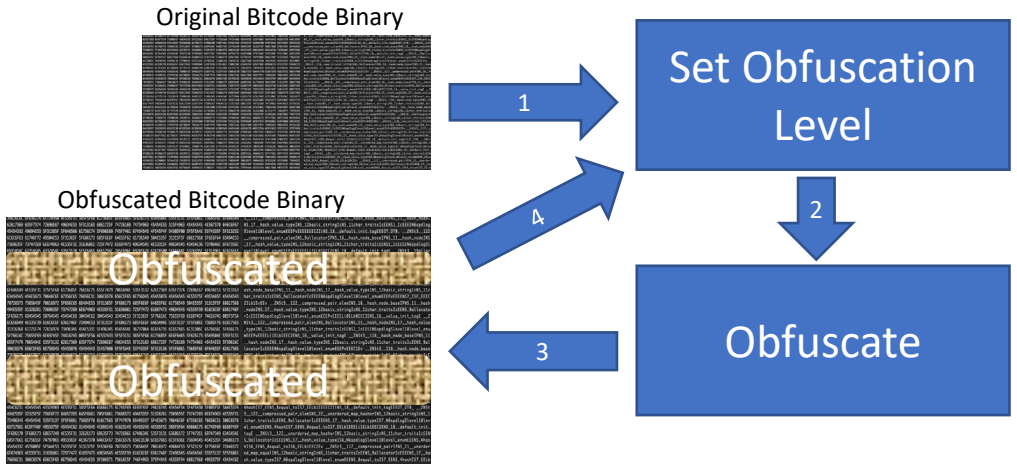
BEGIN TESTS

TEST1: 1 2 3 4 5 6 7 8 9

TEST2: 4362 8202 10506 8202 11274 8202 10506 14346 16650 14346

TESTS COMPLETE!

Program ended with exit code: 0

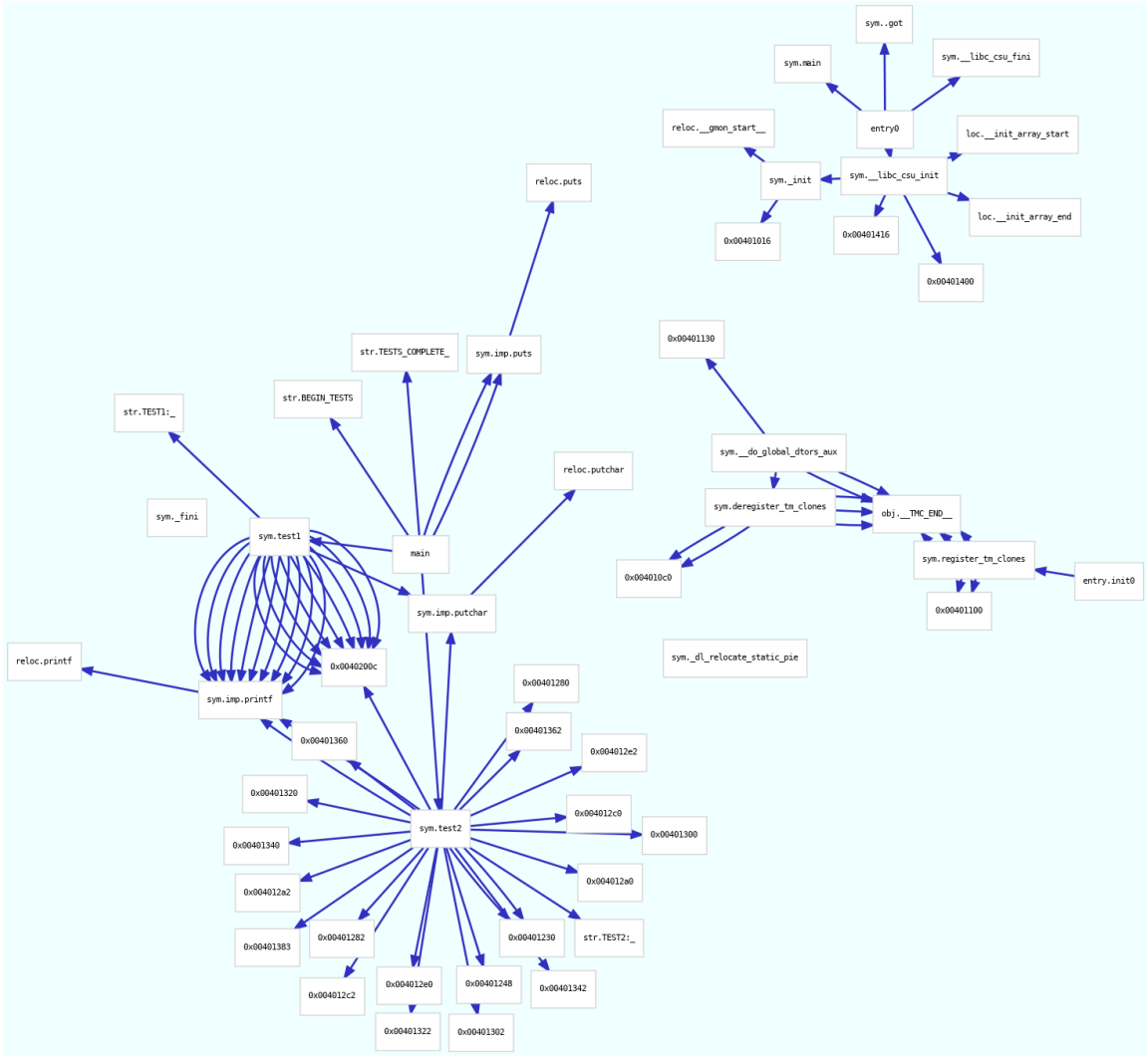



```
#include <stdio.h>

void test1() {
   printf ("TEST1: ");
   int a = 1;
   for (int i = a; i < 10; i++) {
      printf ("%d ", i);
   }
   printf ("\n");
}

void test2() {
   printf ("TEST2: ");
   int a[10], tmp;
   for (int i = 0; i < 10; i++) {
     a[i] = i;
     for (int j = 0; j < 10; j++) {
          tmp = (a[i] * 2) ^ ( j + 3);
          if (tmp % 3)
                tmp += 8;
           a[i] = tmp;
     }
     printf ("%d ", a[i]);
   }
   printf ("\n");
}

int main() {
   printf ("BEGIN TESTS\n");
   test1();
   test2();
   printf ("TESTS COMPLETE!\n");
   return 0;
}
```

© 2022 Society of Cable Telecommunications Engineers, Inc. a subsidiary of CableLabs | expo.scte.org

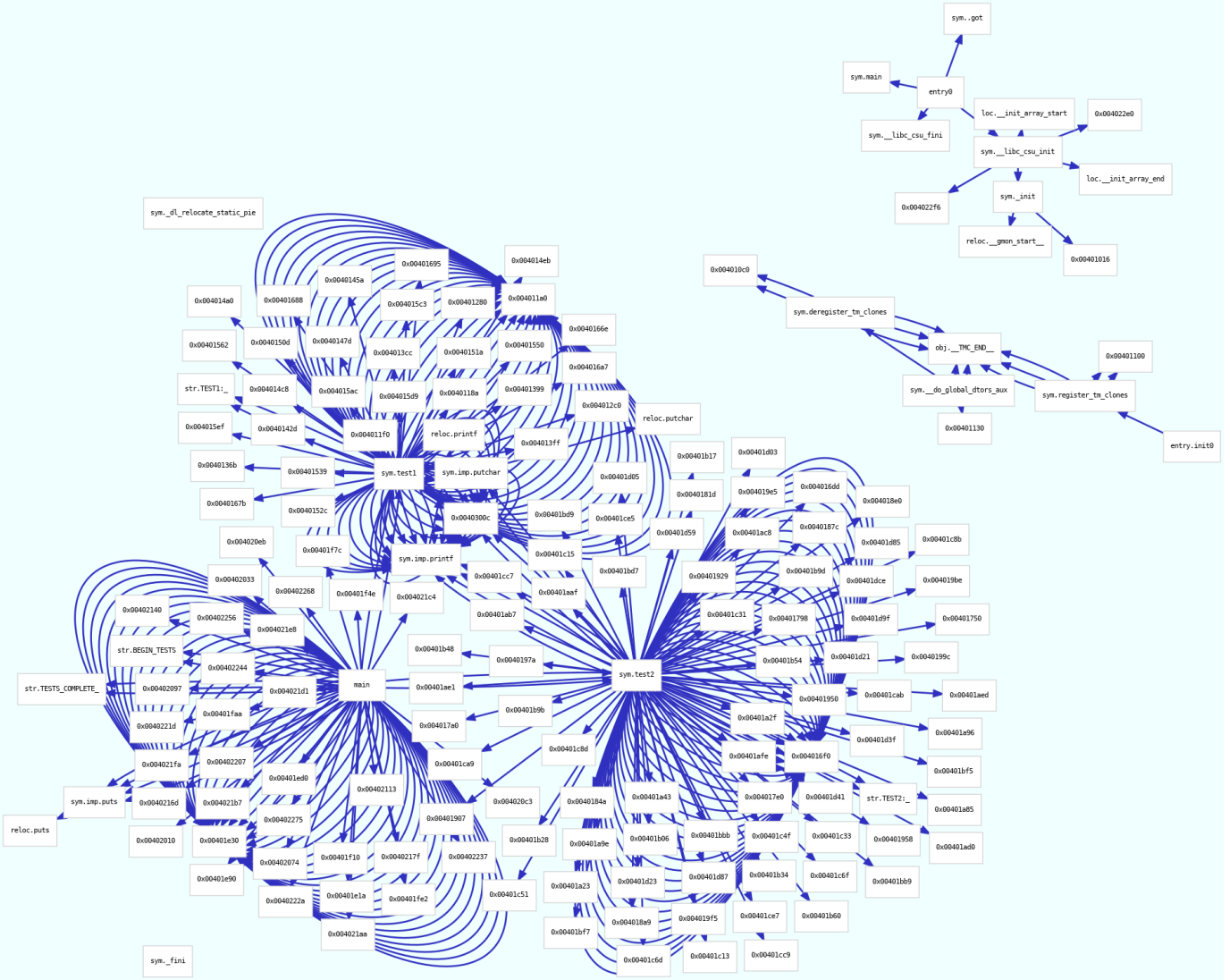
# No Obfuscation

File size = 5 KB



# 50% Obfuscation

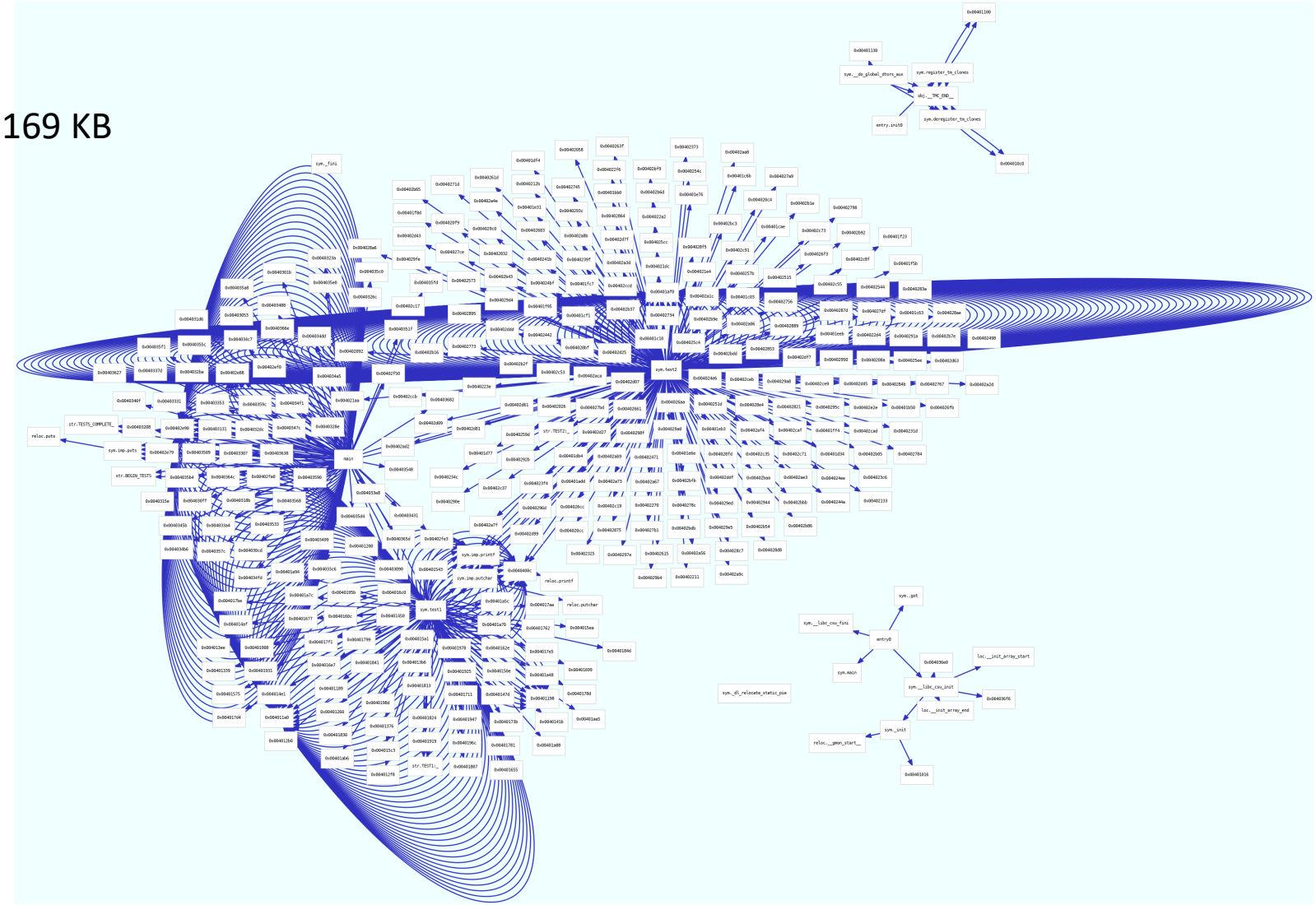
File size = 83 KB




© 2022 Society of Cable Telecommunications Engineers, Inc. a subsidiary of CableLabs | expo.scte.org

# 100% Obfuscation

File size = 169 KB



# Creating Infinite Possibilities.

## Use cases

Bitcode obfuscation techniques allow tunability to achieve a balance between security and performance; and can be applied to a wide range of software applications targeted to various platforms and devices.

© 2022 Society of Cable Telecommunications Engineers, Inc. a subsidiary of CableLabs 

# Use case examples



| | | |
|---|---|---|
| Desktop App | Mobile App | Cloud App |
| IoT App | Browser Plug-ins | SDK Libraries |

© 2022 Society of Cable Telecommunications Engineers, Inc. a subsidiary of CableLabs 

# Conclusion



Bitcode Obfuscation is a promising technology to **protect your binary** without altering the source code and has a wide range of usage in various type of applications in different industries.

With obfuscation, you can

- Guard your software against reverse engineering attack
- Hide your business logic and secret data
- Protect your intellectual property
- Balance your application security vs performance with tuning level
  - Overhead associated with obfuscation must be acceptable
- Audit and inspect your application protection strength

© 2022 Society of Cable Telecommunications Engineers, Inc. a subsidiary of CableLabs 

Creating Infinite Possibilities.

# Thank You!

**Rafie Shamsaasef**
Director of Software Engineering – Security Products
CommScope Inc.
1 (858) 404-2205
rafie.shamsaasef@commscope.com