

# Compilation using LLVM

**Juan Manuel Martinez Caamaño (@jmmartinez)**

Quarkslab

## **Last course**

- Symbolic Execution
- Dynamic Symbolic Execution

# Today's objective

- Protecting Data
- Whitebox Cryptography

# Protecting Data

# Protecting Data vs Code

Until now we seen how to protect the code.

- The input/output of a function still can be instrumented
- Functions can be debugged to trace intermediate values

# Protecting Data

Different ways of protecting data:

- OpaqueConstants
- Encrypt constant data
- Encode constant data

# Protecting Data

Encrypt constant data and decrypt at program startup

- Log strings leak a lot of information
- Once the data is decrypted everything is clear in memory

# Protecting Data

Encode constant data applying a bijective function on every element

- Data is never completely clear in memory
- All the uses of the data must be available
- The values can still be seen when operating with the data



# Protecting Data

```
static const uint8_t secret[256] = { 0xa, 0xb, ... };

uint8_t work_with_secret(uint8_t value, uint8_t i) {
    return value + secret[i];
}
```

# Protecting Data

```
static const uint8_t secret[256] = { rol(0xa, 3)+3, rol(0xb, 3)+3, ... };

uint8_t work_with_secret(uint8_t value, uint8_t i) {
    return value + ror(secret[i]-3, 3);
}
```

# Protecting Data

```
static const uint8_t _add_secret[256][256] = { 0+0xa, 0+0xb, ... };

uint8_t work_with_secret(uint8_t value, uint8_t i) {
    return _add_secret[value][i];
}
```

## Protecting Data

```
static const uint8_t _bij[256] = { ... }; // random permutation
static const uint8_t _bij_inv[256] = { ... }; // bij inverse
static const uint8_t _add_secret[256][256] = { ... };

uint8_t work_with_secret(uint8_t value, uint8_t i) {
    return _bij_inv[_add_secret[_bij[value]][_bij[i]]];
}
```

# Whitebox Cryptography

# Whitebox Cryptography

Tailored cryptographic algorithms for a particular key

- The key is distributed in the tables that encode the operations
- The key is never clear in memory
- Immediate results from operating with the key are never clear in memory

# Whitebox Cryptography

Tabulate all operations in an algorithm that work with products of the key

**This is not enough !**



# Differential-Fault-Analysis

# Conclusions

- Tailored algorithms for a particular key
- Static and Dynamic versions
- Greybox attacks
- Encoding the operations is not enough!