# Security Analysis of Compiler Optimization Techniques– CRG, SERB

## Dr. Chandan Karfa -Department of Computer Science and Engineering

**Theme:** Compiler transformations can be correct but not secure. The objective of this project is to analyze the security aspects of modern day compilers.

## Deliverables:

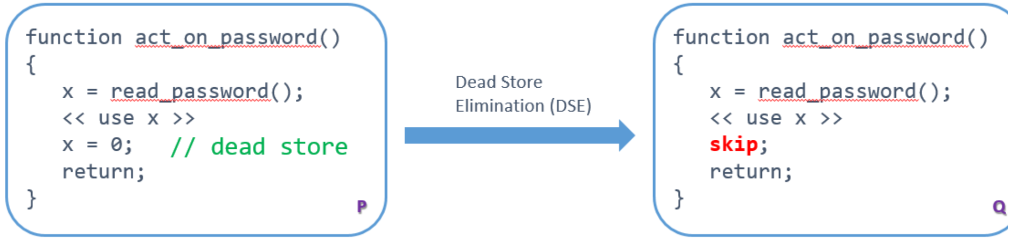
- Security analysis of important compiler optimizations with respect to **information flow attack:** Code motion, Register allocation with spilling and splitting, Retiming
- With respect to the other **target level attacks,** security analysis of compiler optimization techniques such as Dead Store Elimination (DSE), Register Allocation, Code Motion, SSA, Retiming
- Translation validation of Compiler Security

## Current Status:

- We have proved that register allocation in LLVM leaks information. We came up with a secure register allocation scheme.
- We have developed an translation validation framework for checking security of complier optimizations

## Societal Impact:

- Security in software and hardware is a biggest concern in recent days. This project will help to fix certain security loop holes in software and hardware generation process.

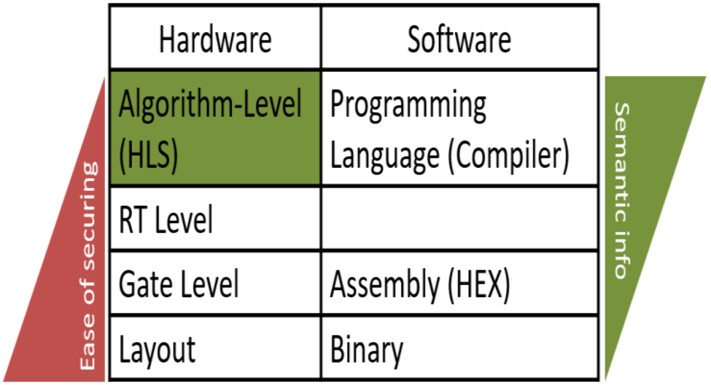


```
function act_on_password()
{
    x = read_password();
    << use x >>
    x = 0;    // dead store
    return;
}
```

```
function act_on_password()
{
    x = read_password();
    << use x >>
    skip;
    return;
}
```

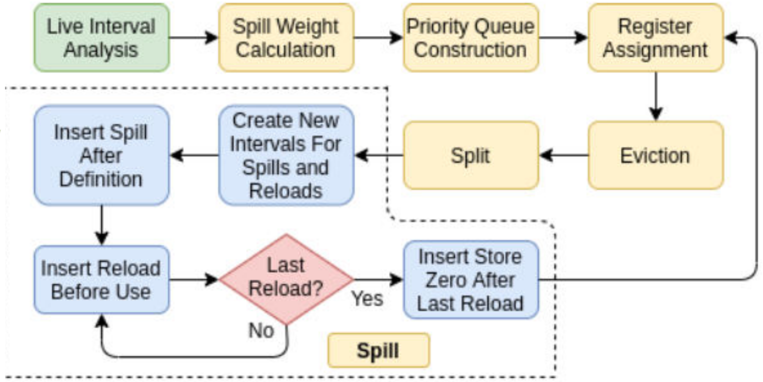
Motivating question: Is it secure ?  No

- **Insecure**, as the secret password is leaked through the stack in Q.



| Hardware | Software |
|---|---|
| Algorithm-Level (HLS) | Programming Language (Compiler) |
| RT Level | |
| Gate Level | Assembly (HEX) |
| Layout | Binary |

Ease of securing
Semantic info



Live Interval Analysis
Spill Weight Calculation
Priority Queue Construction
Register Assignment
Eviction
Split
Create New Intervals For Spills and Reloads
Insert Spill After Definition
Insert Reload Before Use
Last Reload?
Yes
No
Insert Store Zero After Last Reload
Spill

Secure Greedy Register Allocation in LLVM