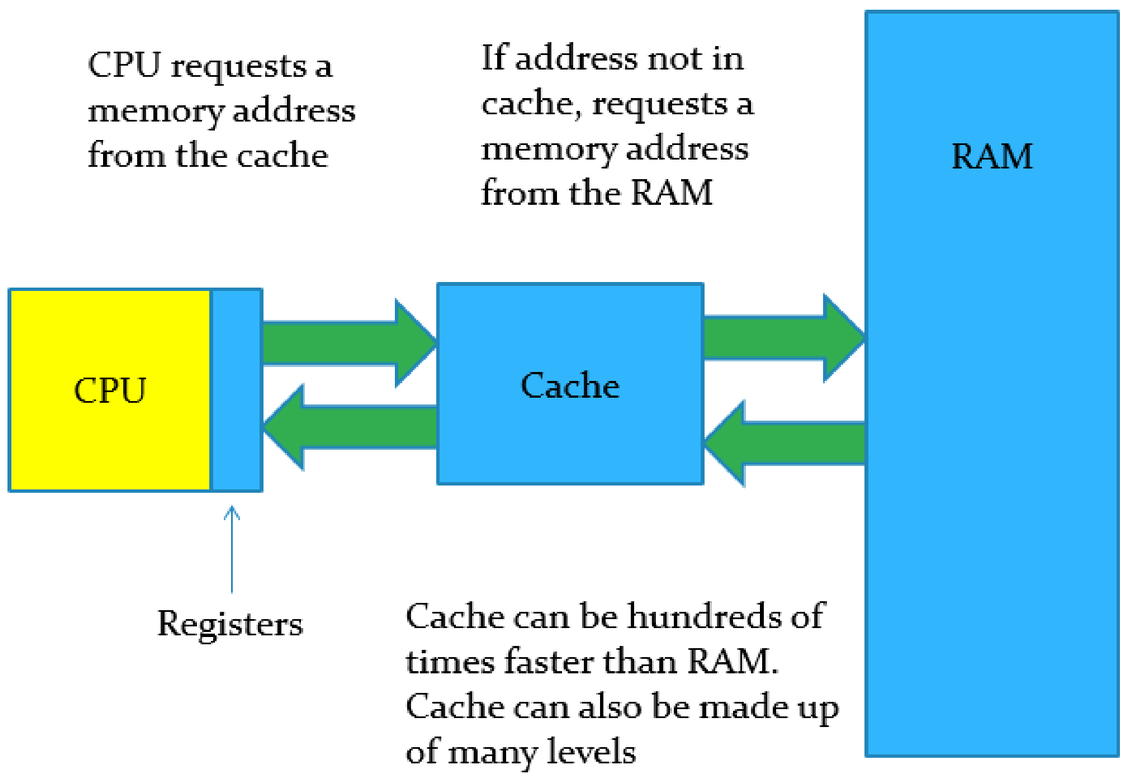
Automatic Side Channel Detection and Prevention

Robert Brotzman-Smith, Gang Tan, Danfeng Zhang

The existence of side-channels has been known for many decardes. They exploit various mediums, such as power, network, memory, etc. All of these side channels require that the adversary have access to a shared resource. With the advent of cloud computing, it makes it more likely an attacker will have access to the same physical hardware as a victim. A popular choice is to attack the CPU cache since it is often shared amongst multiple users and can contain sensitive information, such as an encryption key. Since CPU caches can leak a great deal of information quickly, it is imperative that easy to use techniques are developed to mitigate this threat.

The goal of this work is to provide developers with an easy to use tool to remove side channels. So far, we are able to automatically detect side channels in a basic programming language. The goal is to expand this to encompass an entire language such as C as well as automatically remove the side channels.

How Cache Works



Results

```
int main()
{
  int a,b,key;
  if (key>0)
    a = 1;
  else
    b = 2;
}
```

Figure 1: Side-channel Present

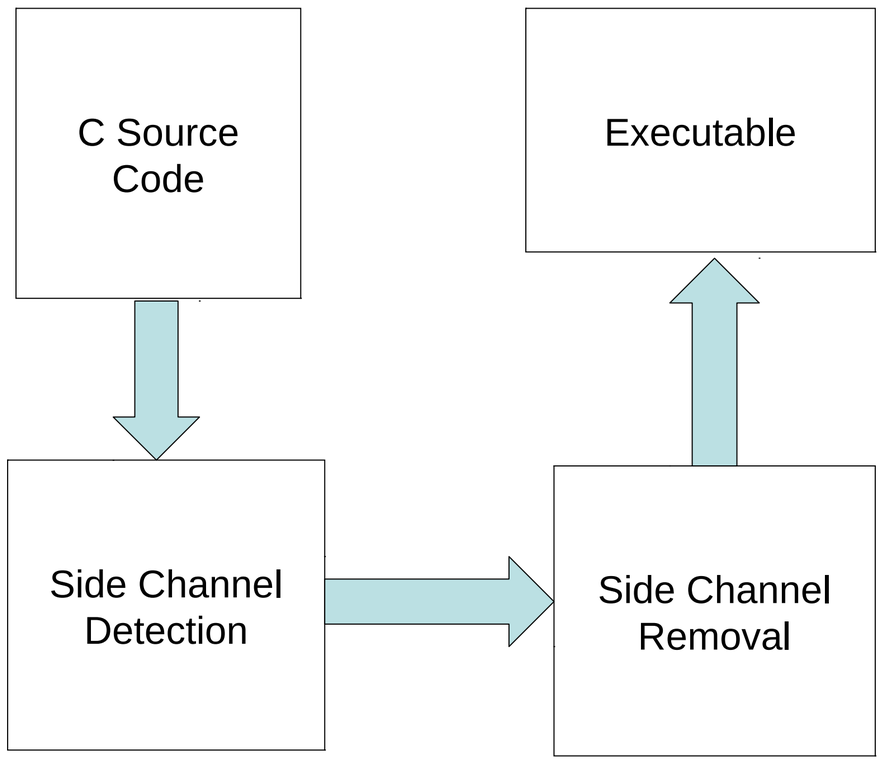
```
int main()
{
  int a,b,key;
  if (key>0)
    a = 1;
  else
    b = 2;
  if (key>0)
    b = 3;
  else
    a = 4;
}
```

Figure 2: No side-channel

Formalized a technique on a basic programing language to detect the potential for a cache side channel. Our model currently assumes that the CPU cache size is infinite, this allows us to ignore cache replacement policy in our evaluation. The technique essentially determines if two or more cache states can exist given a program.

If more than one cache state is possible, there is the possibility of a side channel. We implemented this idea as an LLVM pass. Currently, it can take a constrained C program and determines if a side-channel could exist.

Overview



Goals

- Detect potential side channels
  - Point out what code causes the side channel
- Eliminate the side channels
  - Fix them via a compiler pass
- Do this automatically
  - Done with little to no programmer intervention

Applications

- Cryptography
  - Ensure keys cannot be leaked
- Applications using private data
  - Credit card numbers, SSN’s, etc.

Related Publications

[1] Goran Doychev, Dominik Feld, Boris Köpf, Laurent Mauborgne, and Jan Reineke. 2013. CacheAudit: a tool for the static analysis of cache side channels. In *Proceedings of the 22nd USENIX conference on Security* (SEC'13). USENIX Association, Berkeley, CA, USA, 431-446.

[2] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. 2009. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security* (CCS '09). ACM, New York, NY, USA, 199-212.