**Title: Modeling, Detecting, and Defending against Concurrency Attacks**

**Abstract**:

Driven by the era of multi-core hardware, multithreaded programs have become a mainstream of software. Unfortunately, despite much effort, multithreaded programs are still notoriously difficult to get right; they are plagued with concurrency bugs (e.g., data races). Worse, both recent study and our own preliminary study have revealed that real-world concurrency bugs can lead to “concurrency attacks”: hackers can leverage concurrency bugs to corrupt global memory of a program and construct security vulnerabilities (e.g., gaining OS root privileges). Recent study has also shown that concurrency attacks can weaken most existing security defense techniques because these techniques are mainly designed for single-threaded programs. State-of-the-art neither has a rigorous model on how concurrency attacks manifest, nor has effective detection or defense approaches.

This FALCON project tackles concurrency attacks by pursuing three objectives. First, we will develop a general, rigorous model on how concurrency attacks manifest. A key question for this model is: how does the memory corrupted by concurrency bugs propagate to vulnerable instructions in source code? We will leverage our recent advances on precise data-flow and control-flow analysis methods to address this question.

Second, leveraging this concurrency attack model, we will construct an automated approach that can detect these attacks during testing phase. One major challenge for this detection approach is that we need to precisely identify which parts of program source code (e.g., which functions) may contain vulnerable instructions relevant to a concurrency bug, then we can safely prune out the irrelevant parts and save testing time significantly. To address this challenge, we have presented a precise pruning algorithm and its preliminary implementation in our ASPLOS 2013 paper.

Third, to deal with concurrency attacks that may be missed by detection tools, we will build a runtime defense infrastructure to protect general programs during deployment phase. To implement this infrastructure, we need reliable runtime techniques that can tolerant minor program or deployment machine failures. To prepare for this infrastructure, we have presented a set of fault-tolerant techniques in our SOSP 2015 paper.

We believe that our expertise on precise program analysis algorithms and our preliminary results on fault-tolerant runtime techniques can contribute to achieving the three objectives of this project. By greatly improving the reliability and security of real-world multithreaded programs, this project will benefit lots of software service providers (e.g., PCCW and Amazon) and will spur new ideas and tools in reliability, security, and concurrency areas.

**Long-term impact:**

Multithreaded programs, ranging from real-world applications such as Apache and MySQL, to OS kernels such as Linux and FreeBSD, have become increasingly pervasive and important in order to harness the power of the multi-core hardware. Unfortunately, despite decades of effort from both academia and industry, these multithreaded programs are still extremely difficult to get right. Our previous study reveals that the key reason of this difficulty is that multithreaded programs may run into “exponentially many” thread interleavings at runtime depending on the various execution order of inter-thread communications (e.g., acquiring mutex locks or accessing global memory). It is extremely challenging for developers or existing tools to analyze all these threading interleavings and make sure that they are free of concurrency bugs (e.g., data races). Therefore, concurrency bugs can easily hide in the untested thread interleavings but show up at production runs, easily causing severe behaviors such as wrong outputs or program crashes.

Worse, recent study on diverse real-world applications and OS kernels has shown that concurrency bugs can be exploited by hackers to construct “concurrency attacks”: hackers can choose specific inputs to trigger a concurrency bug, corrupt global memory of a program (e.g., a Linux user ID) with this bug, and leverage the corrupted memory value to construct security vulnerabilities (e.g., call to setuid(corrupted user ID)). After concurrency attacks succeed, even fixing the concurrency bugs by upgrading the program can’t help, because hackers have already broken in. Moreover, both recent study and our own preliminary study have shown that concurrency attacks can weaken or completely bypass existing security defense techniques, because these techniques are mainly designed for single-threaded programs.

State-of-the-art neither has a rigorous model on describing the concurrency attack patterns, nor has effective detection or defense approaches. This proposed FALCON project takes a systematic, thorough methodology to tackle concurrency attacks with three objectives, including developing an attack model, constructing a detection tool, and building a runtime defense infrastructure. Our key weapons to achieve these objectives are our recent advances on developing precise program analysis methods and our preliminary work on building practical fault-tolerant runtime techniques.

The success of this proposed project will greatly improve the reliability and security of real-world multithreaded programs and benefit lots of software service providers (e.g., PCCW, Alibaba, and Amazon). Due to this benefit, we name our project after falcon, one of the strongest birds that help human to hunt prey. Our models and systems implementations will spur new research questions, ideas, and techniques on defending concurrency attacks. Our techniques in this project can also be broadly applied to other research topics which also suffer from both concurrency and security, including achieving byzantine fault-tolerance and mitigating timing channels in clouds.

**Objectives:**

[To develop a general, rigorous concurrency attack model].

We will leverage our expertise on program analysis methods to build a model that describes how the corrupted memory caused by concurrency bugs propagate to vulnerable instructions in program source code.

[To construct an automated concurrency attack detection approach].

Leveraging the concurrency attack model, we will build a concurrency attack detection approach and its implementation tool for developers in the testing phase. This approach can systematically explore a program's execution paths relevant to concurrency bugs and identify potential exploits that these bugs may lead to.

[To build a concurrency attack defense infrastructure].

We will design and implement a runtime infrastructure for software deployment phase. This infrastructure leverages our preliminary work on state-machine replication, a powerful fault-tolerance technique to tolerate and recover from attacks.