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

**Abstract**:

Multithreaded programs have become a mainstream of software due to the rise of multi-core hardware. Unfortunately, despite much effort, multithreaded programs are still notoriously difficult to get right; they are plagued with concurrency bugs (e.g., data races on memory). Worse, recent research reveals that numerous real-world concurrency bugs have led to “concurrency attacks”: hackers leveraged the memory corrupted by concurrency bugs to bypass security authentications and constructed various vulnerabilities (e.g., privilege escalations). Existing security defense techniques are weakened or even completely bypassed by concurrency attacks 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 to describe how concurrency attacks manifest. A central question for this model is: how does the memory corrupted by concurrency bugs propagate to vulnerable instructions in program source code? We will conduct an extent study on real-world concurrency attacks, summarize general elements on how bugs propagate to attacks, and leverage our expertise on precise program analysis methods to develop our model.

Second, following the model, we will construct a systematic approach for detecting attacks during testing phase. A major challenge on this detection approach is: for several types of vulnerable instructions, we must precisely identify which parts (e.g., functions) of program source code may contain these instructions. To tackle this challenge, we have presented a preliminary algorithm to identify one type of vulnerable instructions in our ASPLOS 2013 paper.

Third, to deal with concurrency attacks that may be missed by detection tools, we will construct a runtime defense infrastructure to protect general programs during their deployment phase. This infrastructure requires several reliability techniques, including replication techniques for tolerating program failures and checkpoint techniques for recovering programs’ states (e.g., memory) from failures. In preparation for this infrastructure, we have presented an easy-to-use replication technique in our SOSP 2015 paper.

We envision that our expertise on program analysis algorithms and our preliminary results on replication techniques will help us achieve the objectives of this project. By greatly improving the reliability and security of multithreaded programs, this project will benefit almost all computer users and software service providers. Our techniques developed in this project will also broadly spur new ideas and tools in reliability, security, and concurrency fields.

**Long-term impact:**

Multithreaded programs, ranging from real-world applications such as Apache and MySQL, to OS kernels such as Linux and Android, are already ubiquitous. Unfortunately, despite decades of effort from both academia and industry, these multithreaded programs are still extremely difficult to get right. Because multithreaded programs can run into too many possible thread interleavings at runtime, concurrency bugs can easily hide in the untested thread interleavings but show up at deployment phase, causing severe program behaviors such as wrong outputs or program crashes.

Worse, recent research shows that numerous concurrency bugs have led to “concurrency attacks”: hackers first triggered a concurrency bug to corrupt a program’s global memory (e.g., a Linux user ID) with crafted inputs, and then they leveraged the corrupted memory to construct various vulnerabilities (e.g., call to setuid(corrupted user ID)). Once a concurrency attack succeeds, even fixing the concurrency bug by upgrading the program can’t recover from the attack, because hackers have already broken in.

Concurrency attacks are extremely difficult to tackle due to two main reasons. First, most existing security defense techniques are weakened or completely bypassed by concurrency attacks, because these techniques are mainly designed for single-threaded programs and they are not good at reasoning about concurrent behaviors. Second, existing concurrent defense techniques, which mainly prevent TOCTTOU (time-of-check-to-time-of-use) attacks triggered by illegal file access, are ineffective on tracking concurrency attacks. The reason is that the concurrency attacks we target in this proposal are triggered by much broader types of instructions (e.g., general memory access) and these attacks have led to much broader types of security vulnerabilities (e.g., privilege escalation and malicious code injection). In sum, state-of-the-art neither has a rigorous model to describe how the concurrency attacks manifest, nor has effective detection or defense approaches.

This FALCON project takes a systematic, thorough methodology to tackle concurrency attacks with three objectives, including developing a concurrency attack model, constructing a detection approach, and building a runtime defense infrastructure. Our weapons to achieve these objectives are our recent advances on precise program analysis methods and our preliminary work on fault-tolerant replication techniques.

The success of this project will greatly improve the reliability and security of real-world multithreaded programs, potentially benefitting almost all computer users and 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 hunt prey. We envision that our models, algorithms, and software implementations will spur following research questions, ideas, and tools from widespread research areas, including security, reliability, and concurrency. Our techniques developed in this project can also be broadly applied to relevant research topics that also suffer from both concurrency and security, including TOCTTOU attacks, timing channel attacks, and byzantine fault-tolerance.

**Objectives:**

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

We will conduct an extent study on real-world concurrency attacks, summarize general elements on how bugs propagate to attacks, and leverage our expertise on precise program analysis methods to develop the first concurrency attack model.

[To construct a systematic concurrency attack detection approach].

With the developed concurrency attack model, we will design and implement a concurrency attack detection approach and its software tool for the software testing phase. This approach will be able to systematically explore a program's execution paths relevant to concurrency bugs and identify various types of vulnerable instructions that these bugs may lead to.

[To build a runtime defense infrastructure].

We will design and implement a runtime infrastructure for the software deployment phase. This infrastructure will leverage recent advanced replication techniques for tolerating concurrency attacks and checkpoint techniques for recovering program states (e.g., memory and files) from concurrency attacks.