New tool automatically finds buffer overflow vulnerabilities

July 9 2021, by Daniel Tkacik

Credit: Pixabay/CC0 Public Domain

In 1988 when the internet was still in its infancy, a piece of malware known as the Morris Worm infected nearly 10 percent of the internet over the course of two days, eventually instigating between $100

thousand and $10 million in damages according to the Government Accountability Office. The Morris Worm would eventually be known as the 'Grand Daddy' of a specific cyberattack common even to this day: the buffer overflow.

Put simply, a typical buffer overflow occurs when a computer program receives a request to process more data than its physical memory is capable of handling all at once and places the excess into a 'buffer'. The buffer itself has a finite capacity, so if the buffer can't handle the excess, it 'overflows,' or crashes. Imagine pouring three gallons of water into a two-gallon bucket; things get messy.

"The goal is to automatically find memory bugs that lead to security vulnerabilities in Rust libraries," says Jia. "Manually checking for these bugs is inefficient and time-consuming."

Their tool works on software libraries written in the increasingly popular Rust programming language, which brands itself as both safe and efficient.

"It's a superior language, but it only works if you write in the strict idioms of Rust," says Jia.

Rust developers often need complex data structures for their software. But these complex data structures and their operations typically are written using 'unsafe' Rust, which are not checked by the Rust compiler for memory safety bugs. This is where SyRust comes in; the tool can automatically generate unit tests for library APIs and test these library implementations for memory bugs.

"We applied SyRust to 30 popular libraries and found four new bugs," Jia says. "Given that these libraries were written in Rust already and have been tested, meaning that the programs themselves were very robust to

begin with, we expect a small number of bugs to be discovered."

While the tool isn't yet perfect, Jia says, it's a step in the right direction. For instance, the tool does not generate enough tests to elicit all possible behaviors to ensure a bugless program.

"If I knew that I enumerated all possible behaviors and I don't find any bugs, then I'm happy," Jia says. "That would mean the library truly has no bugs, but right now I don't know how much I've tested, and I don't know how much more I should be testing."

Moving forward, Jia says the team is trying to improve their method of what they refer to as 'improved courage' of the testing. This 'improved courage' would ensure more ground has been covered in the testing process, giving the user more confidence that most, if not all, of the bugs have been found.

More information: Yoshiki Takashima et al, SyRust: automatic testing of Rust libraries with semantic-aware program synthesis, *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation* (2021). DOI: 10.1145/3453483.3454084

Provided by Carnegie Mellon University