Scribe notes 838g 01/30/2014

Andrew Ruef

1 Summary

This paper is a high-level overview with information about the history of low-level memory attacks and a systemization of flaws in software. It shows a large attack tree that describes the steps attackers take to exploit vulnerabilities in low-level programs. The paper also discusses differences between spatial and temporal safety, where spatial safety is a violation of allocated object bounds while temporal safety is the use of memory after its allocated lifetime. Through the chart, the paper also formalizes dependencies between attacker exploitation techniques and memory-safety vulnerabilities in software.

2 Questions

1. Kris: Don't garbage collectors mitigate the temporal safety risks? When free is a no-op, how can you have a use after free?
2. Matt: What is the psychology of choosing defenses and languages? What makes people decide that they can use ASLR but not CFI?
3. Aseem: Why is the overhead of these checks bad? Shouldn't your program, to be safe, do spatial safety checks anyway?
4. Xiao: What's the divide between javascript and C/C++? Isn't javascript memory safe? Why are there memory errors with javascript?
5. Alex: We've seen alternatives to C recently from rust and go, but java and c# not good compeditors, will they make a change?
6. Kris: Can we use symbolic execution to find inputs that produce bugs in low level programs?
7. Kris: What if we structured heap allocators such that objects of a type are only allocated in a particular zone, would this remove some of the impact of temporal safety bugs?

8. James: How does ISA-R work? XOR instructions with a key? New key after every program restart?

3 Criticisms

There didn't seem to be many criticisms of the paper itself.

1. Matt: Security can't be tacked on? Why add all of these probabalistic features at the end of a programs development?

4 Ideas

1. Take ideas this paper presents as too slow, see if we can make them faster
2. Xiao: Use safe languages for security-centric processes, unsafe languages for everything else
3. Survey SF.net and github, find projects that were written in C/C++ that could have been written in something safe, ask why
4. Luis: JIT compile the main program, shift the form and location of code on every invocation, make it difficult for the attacker to predict the program they are attacking