Class Notes Monday

October 4th 2021

**Reducing performance overhead for security in code:**

1. Reduce the instrumentation
2. Improve the efficiency of the instrumentation to detect security vulnerabilities

Better analysis means we can find more objects that are safe. The amount of tools doesn’t matter, its how well tuned the tools actually are.

**Write-What-Where Vulnerability:**

Where an attacker can manipulate any value in memory at any point in time.

**Architecture Security Techniques:**

-X86-64 -> Information hiding with ASLR

ASLR hides the return address of the safe stack inside an address space that’s 2^48

**Hardware Enforced Isolation:**

* Targets ARM v7/8 -M
* Safe Stacks
* Silhouette

Soundness & Completeness on the opposite ends of the security spectrum.

You can’t have both, must choose one.

Things like the safe stack hope to raise the security bar high enough to justify the cost and to reduce the amount of attacks overall that are possible on a system. It’s not designed to be a one stop solution to everything. It’s about reducing the attack surface.

JAVASCRIPT

* Client-Side Execution
* DOM Manipulation
* Cookies
* Provides State for websites to make them interactive
* Third-Party/External JS inclusions
* Used for website analytics

**Why 3rd party JavaScript is dangerous:**

* Daisy chaining of JavaScript sources that depend on other sources

**The Equifax Attack in 2017:**

* Used 3rd Party JavaScript Analytics called FireClick
* The javascript was sourced from netflame.cc
* 2016 netflame.cc domain expired, leaving it open for registration by someone else
* Even after netflame.cc was expired, Equifax kept the javascript in their website code
* The attackers bought the netflame.cc domain and redirected the javascript to malware downloads