Opening Remarks

Andrew Dutcher - angr

Félix Cloutier - The fcd Decompiler

Lunch on your own

Jamie Levy - Taking Memory Forensics to the Next Level

Mike Arpaia - Building successful open source security software

Break - refreshments in Pfizer Lobby

Patrick Hulin - Deterministic Differential Debugging: Finding Root Causes with Record and Replay

Jonathan Salwan & Romain Thomas - How Triton can help to reverse virtual machine based software protections

Ryan Stortz - Firing Rounds at the Analysis Shooting Gallery

Closing Remarks

CSAW Kick-off Reception & Keynote Address (Lucas Moody, CISO, Palo Alto Networks)

Opening Remarks

Andrew Dutcher - ANGR

Abstract: Binary analysis is daunting. Several years ago, very few usable tools existed to facilitate it, and the few that did required in-depth knowledge of OCAML or were early-stage research prototypes. Then, the UC Santa Barbara computer security lab released angr, the a next-generation research platform for binary analysis, and made everything even worse.

In all seriousness, angr was designed to offer the power for cutting-edge binary analysis to researchers, students, and enthusiasts an easy-to-use and extremely flexible package. It’s been used for automatic vulnerability discovery, automatic vulnerability exploitation, automatic vulnerability patching, binary rewriting, exploit development, exploit stealing, “Cyber Reasoning”, and the solving of many a CTF challenge. Thanks to community involvement, it has grown into an amazing system and is used by academics, security researchers, and hacker punks around the world.

Unfortunately, big systems, written by small teams of overworked graduate students and research interns, have warts. In our case, angr’s single biggest issue is documentation, and the philosophy behind the project, the points of interaction and expansion that angr provides, and the different ways of mitigating complications involved in binary analysis are not forthcoming from the existing docs. In this session, we will try to fill in the gaps. We’ll learn about angr’s design, angr’s flexibility, and angr’s failings and the art of addressing them. Coming out of this workshop, you will know enough to Not Panic the next time you need to reach for the angr within you.

FÉLIX CLOUTIER: THE FCD DECOMPILER

Abstract: There are very few open-source decompilers that target machine code in the wild, and most of them produce dishearteningly poor results. Fcd is a burgeoning decompiler that aims to have decent output out of the box for regular programs and provide enough extension points to complete one-off odd jobs. This talk discusses some challenges of decompiling and shows how fcd can be used and adapted to solve them.

JAMIE LEVY: TAKING MEMORY FORENSICS TO THE NEXT LEVEL

Abstract: You’ve probably heard of the Volatility Framework. Maybe you’ve learned about it, or you use it on a daily basis. If so, you’ve probably asked yourself how you could utilize it on an Enterprise scale. As DFIR investigations become more complicated, often spanning several machines, there is a need to employ some mechanisms in the memory forensics realm which are already heavily used in disk forensics. Some of these mechanisms include: whitelisting/blacklisting, indicators of compromise (IOCs), and profiling. This talk will show you how to take memory forensics to the next level.

MIKE ARPAIA: BUILDING SUCCESSFUL OPEN SOURCE SECURITY SOFTWARE

Abstract: Released in 2014 by Facebook, osquery is an open source operating system instrumentation framework and toolset. In this talk, I will reflect on some of the original motivations for creating osquery and discuss the concepts of openness in the information security industry. As a case-study, I’ll break down the attributes of high-quality open source security software by comparing osquery with some of my prior contributions to the open source host instrumentation ecosystem. Finally, I’ll share some tips and tricks when it comes to managing an open source project, gleaned from years of managing the most popular open source security software on GitHub.

PATRICK HULIN: DETERMINISTIC DIFFERENTIAL DEBUGGING: FINDING ROOT CAUSES WITH RECORD AND REPLAY

Abstract: When a developer encounters a bug, they try to understand why it happens and change the code to prevent it from happening again. Unfortunately, programs are stateful, so the immediate cause of a bug (e.g. a segmentation violation) might lead to another cause, which also has to be understood. Following this causal chain backwards in order to find the original error in the program can be difficult and time-consuming, especially in programs whose execution is largely driven by state. Furthermore, traditional debugging tools work forwards, but we want to execute backwards to find a bug’s root cause. Fortunately, a well-studied research area, deterministic record and replay, can help us solve these problems. At each step in a bug’s causal chain, we can use memory watchpoints and reverse execution to find the piece of code which most recently touched that part of program state. If we have a way of evaluating whether the program’s state is valid at any given point, we can do even better: a binary search over time to pinpoint the code which causes the first corruption. One way to approach this is by using a known-good version of the program and comparing state at any point. This approach can be automated, and I will demonstrate a tool which automatically uses one record-and-replay system to find root causes for bugs in another.

JONATHAN SALWAN & ROMAIN THOMAS: HOW TRITON CAN HELP TO REVERSE VIRTUAL MACHINE BASED SOFTWARE PROTECTIONS

Abstract: Triton is a dynamic binary analysis (DBA) framework. It provides several components like a Dynamic Symbolic Execution (DSE) engine, a Taint Engine, AST representations of x86 and x86-64 instruction sets semantics, SMT simplification passes, an SMT Solver Interface and, last but not least, Python bindings. The first part of the talk is going to be an introduction to the Triton framework to expose its components and to explain how they work together. Then, the second part will include a live demonstration on how it’s possible to reverse virtual machine based protections using taint analysis, symbolic execution and SMT simplifications.

RYAN STORTZ: FIRING ROUNDS AT THE ANALYSIS SHOOTING GALLERY

Abstract: DARPA spent hundreds of thousands of my tax dollars creating small C and C++ programs that include exploitable software flaws. We took those programs, ported them to Linux and OS X, and used them as a shooting gallery for static and dynamic analysis tools. Let’s see where each tool excels and where each tool fails.