Automated Vulnerability Analysis from the Cyber Grand Challenge

Michael F. Thompson mfthomps@nps.edu

April 11, 2017

Introduction

The DARPA Cyber Grand Challenge infrastructure included a real-time forensics harness to vet competitor submissions as one piece of a strategy to protect the integrity of the game infrastructure. In addition to vetting competitor supplied software, the forensics harness also allows an analyst to replay any game submission under control of the Ida Pro debugger client, and an IdaPython plug-in. The analyst tool includes reverse execution, automated identification of successful proofs of vulnerabilities (PoVs), and the ability to bookmark and return to execution to points of interest. This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S.

The forensics harness, known as the "CGC Monitor" is built upon the Simics full system emulator, and includes a high fidelity model of an Intel processor based on Intel's processor specifications. This note summarize the CGC Monitor Analyst tool.

Interactive Program Analysis with Reverse Execution

The analyst interacts with an emulated CGC game session under control of the Ida Pro GDB debug client. This "Ida Client" was extended using Ida Python scripts to interact with the CGC Monitor, supporting reverse execution to breakpoints and events of interest. A typical Ida Client session begins with the analyst naming a session to replay. The CGC Monitor executes the session and automatically pauses the simulation at a point of interest, e.g., reading protected memory as part of a Type 2 POV. The Ida Pro interactive disassembler launches and connects to the CGC Monitor's GDB server functions, displaying the execution point at which the simulation paused. The analyst can then use standard Ida debugger commands to interact with the program. We extended hot-key and menu selections to include operations such as:

- reverse step into (potentially stepping backwards into a function)
- reverse step over (stepping backwards over a function)
- uncall (reverse to the calling function)
- run forward or reverse to the next system call
- reverse until the highlighted register is modified
- reverse until the given address is written to
- reverse to the cursor
- reverse to the previous control branch (previous basic block)
- reverse tracking the source of a value at an address

The analyst can set and jump to bookmarks anywhere within the session. Initial, pre-generated bookmarks include the point at which the process begins execution, the point at which the simulation was initially paused, and events such as the first execution of an address from the stack. The final item in the above list of operations generates a set of bookmarks, one for each instruction that transfers the tracked value from/to memory and registers. The operation is similar to taint tracking, only in reverse and it halts upon any manipulation that is not a strict load, move or indirect memory transfer. An example use case tracks the providence of a faulting return address on the stack. When analyzing successful CFE POVs, this operation will usually halt at the "receive" system call that reads the address from the POV, leaving a series of bookmarks denoting execution points at which the faulting address was copied. This sequence of bookmarks, starting at the fault and ending at the ingest of the faulting address, gives the analyst specific locations in which to look for memory corruption.

While interacting with the Ida debugger, the analysis can alter memory content and play a session forward from the altered state. The tool also includes "what if" features, such as replaying a team's POV against some team other than the one it was targeted at. Filters can be removed to observe their effects on sessions, and POVs can be thrown against reference patched binaries.

Emulated architecture

Each team in the CGC Final Event had its own "defended host" upon which all of its services executed. Each team also has its own "POV Thrower", which is a server that executes the POVs submitted by that team's competitors, specifically targeting the team. Along side the POV thrower is a "poller" that sends service polls to the team's defended host and assesses responses to ensure the team's services are functional. All traffic that flows between a team's defended host and its POV thrower and poller passes through an "IDS". A competitor can submit IDS filters to block or modify traffic flowing to the services executing on a defended host.

The system emulated by the CGC Monitor includes three distinct simulated computers that correspond to specific per-team servers in the game infrastructure: the defended host; the IDS; and, the POV thrower. To reduce the quantity of emulated computers, service polls were originated on the emulated IDS server rather than a separately simulated server. Similarly, within the emulated system, the POV negotiation service runs on the IDS rather than a distinct server. The emulated computers run the exact operating system, (and custom hypervisor), deployed on the game infrastructure and the exact services, e.g., the programs that launch CBs and PoVs. While the CFE game infrastructure computers include multi-core processors, the emulated computers are configured as single-core processors because these are more efficient for Simics to emulate.

As per the CFE infrastructure, the emulated systems include ethernet links dedicated for use by in-game data and separate links for exchange of control information.

CGC Monitor System Requirements

The CGC Monitor executes on a set of Linux servers. The Ida Client used by the analysis functions run on either OSX or a Linux workstation. The client requires Ida Pro and a few client packages including a python MySQL client (pymysql), and the ZooKeeper python client bindings from Netflix (kazoo). The Ida Client services communicate with the CGC Monitor servers via ssh tunnels that are automatically established by the Ida Client package.

The CGC Monitor servers require Simics licenses, Apache ZooKeeper running on one or more servers, and a server that runs a MySQL server and an NFS server. The CGC Monitor also utilizes the python Fabric framework and tmux.

Multiple analysts can interact with the CGC Monitor concurrently, and while the monitor is also processing game sessions.