Digital Investigation 26 (2018) S127–S135

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

DFRWS 2018 USA — Proceedings of the Eighteenth Annual DFRWS USA

Cyber Grand Challenge (CGC) monitor: A vetting system for the DARPA cyber grand challenge

Michael F. Thompson [a, *], Timothy Vidas [b]

[a] Naval Postgraduate School, 1 University Circle, Monterey, CA 93943, USA
[b] Secureworks, One Concourse Parkway, Atlanta, GA 30328, USA

Keywords:
Full system simulation
Digital forensics
Proactive forensics
Introspection
Vulnerability analysis

A B S T R A C T

The DARPA Cyber Grand Challenge (CGC) pit autonomous machines against one another in a battle to discover, mitigate, and take advantage of software vulnerabilities. The competitors repeatedly formulated and submitted binary software for execution against opponents, and to mitigate attacks mounted by opponents. The US Government sought confidence that competitors legitimately won their rewards (a prize pool of up to $6.75 million USD), and competitors deserved evidence that all parties operated in accordance with the rules, which prohibited attempts to subvert the competition infrastructure. To support those goals, we developed an analysis system to vet competitor software submissions destined for execution on the competition infrastructure, the classic situation of running untrusted software.

In this work, we describe the design and implementation of this vetting system, as well as results gathered in deployment of the system as part of the CGC competition. The analysis system is implemented upon a high-fidelity full-system simulator requiring no modifications to the monitored operating system. We used this system to vet software submitted during the CGC Qualifying Event, and the CGC Final Event. The overwhelming majority of the vetting occurred in an automated fashion, with the system automatically monitoring the full x86-based system to detection corruption of operating system execution paths and data structures. However, the vetting system also facilitates investigation of any execution deemed suspicious by the automated process (or indeed any analysis required to answer queries related to the competition). An analyst may replay any software interaction using an IDA Pro plug-in, which utilizes the IDA debugger client to execute the session in reverse.

In post-mortem analysis, we found no evidence of attempted infrastructure subversion and further conclude that of the 20 vulnerable software services exploited in the CGC Final Event, half were exploited in ways unintended by the service authors. Six services were exploited due to vulnerabilities accidentally included by the authors, while an additional four were exploited via the author-intended vulnerability, but via an unanticipated path.

1. Introduction

Proactive forensics often blurs the lines between traditional forensics, pedantically requiring the application of a legal system, and similar techniques that may not ever be used to a legal end. In the digital space, such processes are increasingly common, with the use of similar, sometimes identical tools and procedures as those found in digital forensics, but to pursue a wide range of situations from corporate policy violations to complex computer administration troubleshooting. In many cases, adopting the relatively high standards demanded by legal systems early, even before demanded by circumstance, leads to not only a smooth transition into a digital forensics case, but in some cases the a priori action enables a case to form that otherwise might be impossible. For instance, if no digital artifacts were created as a result of some computer crimes, prosecution may be difficult, indeed in some cases the offense might even go unnoticed. However, observation and prosecution may be straightforward if the entity had previously put in place proper logging, netflow collection, and/or host-based software security agents. The level of preparation any entity might undergo varies drastically, with some proactively collecting data that may eventually become evidence (Shields et al., 2011), others physically installing hardware to enable future evidence collection (Carrier and Grand, 2004), or preparing tools and procedures in effort to achieve a state of readiness (Rowlingson, 2004).

* Corresponding author.
E-mail addresses: mfthomps@nps.edu (M.F. Thompson), tvidas@secureworks.com (T. Vidas).

https://doi.org/10.1016/j.diin.2018.04.016

The US Department of Defense Advanced Research Projects Agency (DARPA) created the Cyber Grand Challenge (CGC) to push the boundary of technology in "autonomous cyber defense capabilities that combine the speed and scale of automation with reasoning ability exceeding those of human experts (DARPA, 2016)." The challenge was framed around vulnerabilities in binary software. To encourage focused research, a concentrated, representative software environment was created: the DARPA Experimental Cyber Research Evaluation Environment, or simply DECREE. Concomitant with this focused environment, and due to the competitive nature of the challenge, many architectural and implementation design decisions made throughout program aimed to ensure the highest standards of integrity (Vidas et al., 2017). The mechanics of the challenge, in particular the final portion of the competition, was modeled heavily after attack-defend style cybersecurity Capture-the-Flag (CTF) competitions. The competition was structured as a "brokered" environment. That is, instead of competitors directly administering the hosts that required defense, or directly leveraging offensive actions upon one another, competitors uploaded software and minor metadata on how the software should be used (e.g. targeting information) to competition infrastructure services. This mediation afforded many desirable properties, among them more organizer control over the competition operations and the ability to catalog and inspect every input into the competition.

The CGC infrastructure development team built an analysis system to vet competitor submissions as one piece of a broad strategy to protect the competition integrity. The challenge organizers did not expect any of the CGC Final Event (CFE) competitors to attempt to subvert the competition infrastructure. Even so, there was desire for convincing evidence supporting the assertion that competition integrity was not compromised in violation of CGC rules (DARPA, 2016), and in the event that competition forensics were required, the unusual environment and relative urgency for results required investigative tools and processes to be foresight, not afterthought.

The competition consisted principally of two events, a qualifying event and for those that progressed, a final event. Participation in the CGC Qualifying Event (CQE) was open to any applicant who met a relatively open set of criteria (DARPA, 2014), and this accessibility motivated the vetting of CQE submissions. The analysis system, known as the *CGC Monitor* is built upon a full system simulator. An early goal was to simulate the entire competition infrastructure software execution environment and execute competitor-provided software within the simulated system prior to its introduction onto the actual competition infrastructure. The simulator is instrumented to detect attempts to compromise the operating system execution control paths or its data structures, (e.g., credentials used to identify a process and its permissions). For CQE, the goal of vetting all software prior to its reaching the infrastructure was realized. For CFE, time constraints required vetting of competitor software concurrently with execution on the actual competition infrastructure. Even though this vetting was not a prerequisite for introducing software into the competition, all submitted software was vetted prior to the naming of the CFE winners.

In addition to automated vetting of competitor-supplied software, an analysis system was developed to facilitate investigation of any executed competition submission. That is, an analyst can investigate any particular execution that occurred during the contest using a novel IDA Pro debugger client alongside an IDA-Python plug-in (Eagle, 2008). The analysis tool was available to further investigate sessions flagged during automated vetting, and was utilized after CFE to better understand which flaws were successfully exploited during the competition. The analyst tool includes reverse execution, automated identification of successful proofs of vulnerabilities (PoVs), and the ability to bookmark and return execution to points of interest.

During CQE, vetted submissions included each replacement challenge binary (RCB), and each proof of vulnerability (PoV), which in CQE was an XML file that controls network traffic sent to vulnerable Challenge Binaries (CBs). For CFE, every PoV was run against the RCB and IDS rule-set pair specified by the Cyber Reasoning System (CRS), that submitted the PoV. And every RCB was exercised using a sample of the service polls created for that CB. The competition infrastructure provides CBs with a repeatable source of simulated entropy, potentially leading to divergent execution between two sessions for the same service poll. The entropy sources used within the competition infrastructure were re-used in the simulated system to align code paths followed on each system.

For the purposes of this paper the term "malicious" is defined as attempts to subvert the competition infrastructure or scoring system, ultimately in violation of one or more of the CGC Rules (DARPA, 2016). The remainder of this paper first describes the simulated architectures in section 2 and the implementation of the CGC Monitor using the Simics full system simulation including strategies for detecting specific malicious activity in section 3. The analyst support functions and implementation of reverse execution features are then described in section 4. Details from at scale experiments of the CGC Monitor are provided in section 5. Experimental results are presented along with a discussion of performance in section 6. Finally, related work is described in section 7 and concluding remarks are provided in section 8.

2. Emulated architecture

CQE scoring occurred on a set of isolated virtual machines (VMs), each of which ran a single session, e.g., a reference PoV thrown against a competitor-provided RCB. Each VM included the two primary infrastructure software components: a player that replayed XML files to simulate client network traffic, and a launcher that spawned RCBs or CBs. CQE did not include an IDS component (DARPA, 2016). The CGC Monitor for CQE was similarly constructed, with individual simulated computers running the very same player and launcher software to replay CGC sessions.

Each team in the CFE had its own defended host upon which all of its services executed. When a team's CRS submits a RCB, that program eventually executes on that team's defended host. 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. Alongside 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 CRS can submit IDS filters to block or modify traffic flowing to the services executing on a defended host. Each team's suite of components included a negotiator with which an executing PoV negotiates attributes of the proof of vulnerability. These negotiated attributes include whether the PoV is Type 1 (controlled crash), or Type 2 (memory disclosure).

For CFE, the system simulated by the CGC Monitor includes three distinct simulated computers that correspond to specific per-team servers in the competition infrastructure: the defended host; the IDS; and, the PoV thrower. To reduce the quantity of simulated computers, service polls were originated on the simulated IDS server rather than a separately simulated server. Similarly, within the simulated system, the negotiation service runs on the IDS rather than a distinct server. The simulated computers run the exact operating system, (and custom hypervisor), deployed on the