Research Statement

David J. Malan

My research interests span two realms primarily, security and education, with my most recent work focused on the latter, albeit informed by the former. Within the realm of security have I long been interested in cybersecurity specifically, which encompasses threats to networks and hosts as well as defenses thereof. Not only has this interest motivated my contributions to sensor networks, digital forensics, and botnets, it has also influenced quite a few of my courses' problem sets and lectures, which are now themed around that domain. Within the realm of education, in computer science specifically but STEM more generally, am I particularly interested in distance learning, collaborative learning, and computer-assisted instruction. Those interests are manifest across a range of recent research endeavors involving assessment, instructional technologies, and peer instruction.

Research in Security

Throughout graduate school, my research group's focus was economic and applied security. Among the group's goals were to develop new metrics for security and privacy, explore new threats to systems, and create and analyze new system and data safeguards, all of which characterized my own work.

Sensor Networks Upon their introduction, wireless sensor networks were proposed for such applications as habitat monitoring, structural health monitoring, emergency medical care, and vehicular tracking. However, those same networks were characterized by limited resources (CPU, RAM, etc.) and, accordingly, low costs. So limited were those resources that computationally expensive operations like public-key cryptography were initially ruled out. In [MWS04, MWS08], however, we contributed that public-key infrastructure was, in fact, viable for secret keys' distribution, presenting the first known implementation of elliptic curve cryptography for sensor networks.

Digital Forensics Privacy-preserving tools have long relied, at least in part, on creating one big file that fills up a storage medium in an effort to overwrite all of the "deleted" files that the medium contains. For some time, however, the effectiveness of this technique was largely unvalidated. In [GM06], we validated that one big file was effective on FAT32, NTFS, and HFS file systems (although not for file names). But we invalidated the technique as effective on Ext2fs, Ext3fs, and Reiserfs; on all three file systems did we demonstrate that not all user data was overwritten.

Botnets *Botnets* are networks of systems on which adversaries have installed software called *bots* that they can remotely control, typically unbeknownst to those systems' own owners. Detection of botnets reduces to detection of bots (e.g., worms) across systems. Detection of worm-like threats, though, conventionally relies on behavior-based defenses, whereby a host's behavior is deemed anomalous (and thus suspicious) if it differs from a host's prior behavior or resembles activity deemed worrisome a priori by some authority (e.g., McAfee or Symantec). The implication is that a host's behavior might be deemed anomalous simply because: some process follows a new, but benign, code path for the first time; some new, but benign, application is installed for which the host has no history of behavior; or some process behaves in a way that might be, but isn't necessarily, worrisome. In none of these cases would a host's user want to be prompted with an alert about suspicious behavior (i.e., a false positive). In [Mal07b, MS05, MS06], we thus defined *temporal consistency*, similarity over time in worms' and non-worms' invocations of system calls, as a proxy for hosts' behavior. And we offered an alternative definition of anomalous behavior. We proposed that a host evaluate some current action vis-à-vis its peers' current actions; a host's behavior should be deemed *anomalous* if it correlates all too well with other, otherwise independent, hosts' behavior. Worms can generally be distinguished from non-worms by their simplicity and periodicity: their design is to spread, and their execution is cyclical. Through cooperation among peers, then, we contributed that we can lower the risk of false positives by requiring that individual hosts no longer decide a worm's presence but a cooperative instead.

Research in Education

Throughout graduate school, I was fortunate to have opportunities to head courses at Tufts University, Harvard Extension School, and Harvard Summer School. By the end of graduate school, my research interests had broadened to include education itself, specifically in computer science but also in STEM more generally, sparked by those teaching experiences. Within the sphere of computer science education am I now particularly interested in distance learning, collaborative learning, and computer-assisted instruction. Those interests are manifest across a range of research projects that can be broadly categorized as relating to assessment, instructional technologies, and peer instruction.

Assessment Harvard's introduction to computer science, CS50, implements apprenticeship learning, whereby each student is apprenticed at term's start to a teaching fellow (TF) who is responsible for grading 12–16 students' programming assignments. While some aspects of grading are automated, some are deliberately manual so that grading itself is an opportunity not only to evaluate but to teach. The result, though, is that grading typically consumes upwards of 6 hours per week of each TF's (limited) time, and so optimization thereof is of perennial interest. In 2008, we set out to streamline the process by equipping the course's 27 TFs with tablet PCs (i.e., touchscreens plus keyboards). We hypothesized that such hardware would enable the TFs to mark up students' code qualitatively with ease (not unlike actual paper, which the course had long since abandoned) and to flip rapidly between students' executable code and PDFs thereof, thereby expediting their workflow. As

per [Mal09a], by this experiment's end, most TFs (63%) indeed reported that grading took less time, and nearly half (48%) reported that they provided students with more feedback because of the same.

In 2013, we set out to improve upon that result via software in lieu of (more costly) hardware [MM13]. We developed a web-based utility through which TFs could leave feedback for students via inline "sticky notes," deprecating altogether the course's reliance on PDFs and annotations thereon. Although less striking than our results with hardware, the software-only solution yielded a reduction of 10% in time spent on grading, while maintaining (and, in some cases, increasing) feedback per student, vis-à-vis PDFs' annotation. But the experiment, by nature of its online design, also revealed that, on average, 9% of the course's 715 students were not even reviewing their TFs' feedback, a result that has motivated us to experiment with in-person, interactive feedback loops. Similarly have we already begun to experiment with interactive assessments built into the course's own video player in order to improve learning outcomes [TMM13].

Instructional Technologies Long before there were massive open online courses (MOOCs), there were podcasts, feeds of audio, video, and other content that could be downloaded to clients like iTunes and devices like iPods. In 2005, we decided to podcast one of my own courses in order to provide students with more portable access to educational content and to involve them in the technology itself. To evaluate that experiment, we analyzed logs as well as surveys of students and found that students valued the podcast more as a vehicle for review (45%) than as an alternative to attendance (18%) [Mal07a]. We ultimately argued that podcasting, not unlike today's MOOCs, was but a marginal improvement upon technological trends long in progress and that it offered to extend universities' reach more than it offered to improve education itself.

In 2007, meanwhile, we set out to improve learning outcomes in introductory programming courses (CS50 among them) by way of Scratch, a graphical programming language that empowers students to program by dragging and dropping "puzzle pieces" that only snap together if "syntatically" appropriate. Although the language was intended for youths in after-school centers, we deployed it to students much older, 76% of whom reported that Scratch proved a positive influence on their subsequent experience with Java [ML07]. Later that year did we deploy Scratch to hundreds of undergraduates, albeit for only one week, before transitioning to C. A result, though, was a marked increase in retention: whereas in 2006, 68% of the students who "shopped" CS50 ultimately registered, in 2007, upwards of 95% registered (and remained registered).

In 2008, we began to experiment with cloud computing in courses in order to achieve pedagogical conveniences (e.g., root access) as well as to introduce students first-hand to the cloud. We not only told students about scalability, virtualization, multi-core processing, and cloud computing that year, we had them experience each [Mal10]. But that experiment was just one step toward a vision of scaling education itself. By 2011, we had transitioned those same courses off of the cloud and on to an "appliance," a client-side virtual machine designed to run on any student's computer. Not only did that appliance enable us to provide students with simpler, more pedagogically effective tools, it enabled us to bundle more sophisticated tools too [Mal13b]. And with execution of students' code finally distributed across students'

own CPUs, the appliance allowed us scale Harvard's introductory course from some 700 students on campus to more than 150,000 students online.

Peer Instruction Consistent with apprenticeship learning are office hours, opportunities for one-on-one help for students with TFs (most of whom are students themselves). In 2007, we opted to experiment with "virtual" office hours in CS50 so that students could meet with TFs at any hour from anywhere. Our goals were to lower the bar to interaction and to improve the efficiency and convenience of the same. Based on surveys of students, we ultimately found that virtual office hours were attended nearly as frequently as physical office hours, with 14% of students often attending the former and 21% often attending the latter. Among the subset of students (166 out of 282) who did not tune in for virtual office hours very much or at all, 74% explained that they simply preferred the physical. On the whole, we judged the integration of virtual office hours a net positive—another, imperfect but still useful resource—but not a revolutionary innovation [Mal09b].

In recent years have we focused more on improving and scaling physical office hours themselves. In 2011, for instance, we relocated them to residential dining halls after meal hours to create more social and collaborative workspaces, conducive to conversations and interactions among students from disparate fields. We also developed web- and iPad-based software with which to manage logistics. Overall, that new format proved a success. Attendance at office hours grew more than linearly, with an average of 120 students attending per night, up from 30 students a year prior, despite only a 23% increase in enrollment [MM12]. Although the software facilitated such scale, wait times for students still sometimes exceeded an hour, a reality that recurred in 2012. Not until 2013 did we determine empirically that we were over-engineering our solution to our problem of scale. We put aside all hardware and software that year, relying instead on more organic interactions among students and TFs (e.g., hands in the air). Preliminary review of 2013's data suggests these were perhaps the most effective office hours to date.

Future Work

While themselves more evolutionary than revolutionary technologically, insofar as distance learning has been available in some form for some time, MOOCs offer unprecedented opportunities to study at scale how students learn and how best to teach. Through CS50's own MOOC have we already begun to explore precisely those questions. Through ongoing surveys of students, fine-grained logging of students' interactions with tools, and automated evaluation of thousands of projects (coupled with hundreds of hand-graded projects as training data), we hope ultimately to design more effective educational experiences. Through the course's own appliance, for instance, are we able to observe (with students' consent) students' progression from blank files to working programs. Along the way, though, do most students encounter classes of bugs that we hope to model and cluster so as to provide more effective feedback. With the course's new sandbox environment, too, do we now have unprecedented eyes into students' progression over the course's duration that we plan to leverage as well [Mal13a].

Similarly do we hope to improve the quality of students' code even prior to submission,

as via automated qualitative feedback. Already have we deployed in CS50 (and its MOOC) tools for behavioral testing and static analysis that students and TFs alike can use.

In collaboration with the Science Education Department of the Harvard-Smithsonian Center for Astrophysics do we plan to study persistence and attrition in CS50's MOOC this coming year in hopes of characterizing factors that influence students' success.

In a higher-level course on software engineering (CS164), meanwhile, do we intend to continue our experiments with peer instruction. In 2014 will we ask students to collaborate in teams of four, semi-competitively tackling in parallel the same projects as other teams in pursuit of optimal designs.

References

[GM06] Simson L. Garfinkel and David J. Malan. One Big File Is Not Enough: A Critical Evaluation of the Dominant Free-Space Sanitization Technique. In *6th Workshop on Privacy Enhancing Technologies*, Cambridge, United Kingdom, June 2006.

[Mal07a] David J. Malan. Podcasting Computer Science E-1. In *38th ACM Technical Symposium on Computer Science Education*, Covington, Kentucky, March 2007.

[Mal07b] David J. Malan. *Rapid Detection of Botnets through Collaborative Networks of Peers*. PhD thesis, Harvard University, Cambridge, Massachusetts, June 2007.

[Mal09a] David J. Malan. Grading Qualitatively with Tablet PCs in CS 50. In *Workshop on the Impact of Pen-Based Technology on Education*, Blacksburg, Virginia, October 2009.

[Mal09b] David J. Malan. Virtualizing Office Hours in CS 50. In *14th Annual ACM Conference on Innovation and Technology in Computer Science Education*, Paris, France, July 2009.

[Mal10] David J. Malan. Moving CS50 into the Cloud. In *15th Annual Conference of the Northeast Region of the Consortium for Computing Sciences in Colleges*, Hartford, Connecticut, April 2010.

[Mal13a] David J. Malan. CS50 Sandbox: Secure Execution of Untrusted Code. In *44th ACM Technical Symposium on Computer Science Education*, Denver, Colorado, March 2013.

[Mal13b] David J. Malan. From Cluster to Cloud to Appliance. In *18th Annual ACM Conference on Innovation and Technology in Computer Science Education*, Canterbury, England, July 2013.

[ML07] David J. Malan and Henry H. Leitner. Scratch for Budding Computer Scientists. In *38th ACM Technical Symposium on Computer Science Education*, Covington, Kentucky, March 2007.

[MM12] Tommy MacWilliam and David J. Malan. Scaling Office Hours: Managing Live Q&A in Large Courses. In *28th Annual Conference of the Eastern Region of the Consortium for Computing Sciences in Colleges*, Galloway, New Jersey, November 2012.

[MM13] Tommy MacWilliam and David J. Malan. Streamlining Grading toward Better Feedback. In *18th Annual ACM Conference on Innovation and Technology in Computer Science Education*, Canterbury, England, July 2013.

[MS05] David J. Malan and Michael D. Smith. Host-Based Detection of Worms through Peer-to-Peer Cooperation. In *ACM Workshop on Rapid Malcode*, Fairfax, Virginia, November 2005.

[MS06] David J. Malan and Michael D. Smith. Exploiting Temporal Consistency to Reduce False Positives in Host-Based, Collaborative Detection of Worms. In *ACM Workshop on Recurring Malcode*, Fairfax, Virginia, November 2006.

[MWS04] David J. Malan, Matt Welsh, and Michael D. Smith. A Public-Key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography. In *Proceedings of the First IEEE International Conference on Sensor and Ad Hoc Communications and Networks*, Santa Clara, California, October 2004.

[MWS08] David J. Malan, Matt Welsh, and Michael D. Smith. Implementing Public-Key Infrastructure for Sensor Networks. *ACM Transactions on Sensor Networks*, 4(4), 2008.

[TMM13] R.J. Aquino Tommy MacWilliam and David J. Malan. Engaging Students through Video: Integrating Assessment and Instrumentation. In *18th Annual Conference of the Northeast Region of the Consortium for Computing Sciences in Colleges*, Loudonville, New York, April 2013.