

Interview

Silver Bullet Talks with Greg Morrisett

GARY MCGRAW
Cigital

Greg Morrisett is the Allen B. Cutting Professor of Computer Science and associate dean for computer science and engineering at Harvard University. His current work is on applications of advanced type systems, model checkers, proof-carrying code, and inline reference monitors for building efficient and provably secure systems. He's also working on languages for sensor networks.

Hear the full podcast at www.computer.org/security/podcasts/ or www.cigital.com/silverbullet.

Gary McGraw: Your early career focused lots of attention on the relationship between security and programming languages. Why do you think programming languages are important to software security?

Greg Morrisett: Like it or not, humans make a lot of mistakes when they code, and having tools that automatically rule out—or at least in a fail-stop way, check for classes of errors—makes a lot of sense. Something like a buffer overrun just couldn't happen in a properly implemented programming environment.

McGraw: Making things just not able to happen in the first place.

Morrisett: If you go back and look at the early days of type abstraction and abstract data types, back to the '70s as [they were] being developed, people were really thinking about systems and had the same vocabularies in the design of programming languages that they had in the design of security systems. A lot of those ideas are being dusted off and revisited as a way to enforce or provide a model for security in software today.

McGraw: There's this notion of type safety that seemed important, and if you think about Java or C#, it's an important characteristic of those languages. Can you explain what type safety is and why it's important, and then what other programming languages' properties can help with security?

Morrisett: Type safety is a pretty fuzzy notion because it's very relative, and there are technical definitions that aren't all that useful. At a rough level, what it means is some notion of object integrity—that I can only access an object, either to read or write it, using a proper capability. I don't have to worry about somebody changing the state of some object in a type-safe language unless they have an accessible type-safe path to that object. That's important [because] when you go to reason about programs, either using a tool or even

informally, you make assumptions about how that code is going to execute. Type safety is the bottom line in a security policy. It's the minimal amount that you need to get any kind of notion of integrity because without it, you can't, for example, even ensure that the code you thought you were executing is what you're executing.

McGraw: I guess that's one of the problematic aspects of C and other very low-level languages.

Morrisett: As you know better than just about anybody, just because you've written code in a type-safe language doesn't mean the security problems go away. It does get rid of a set of baseline things, or at least transform them into different sets of issues, but the point is that all of a sudden, we have a firm footing to stand on when it comes to doing things like static analysis. If you, for example, do a static analysis on C, you have to be extremely conservative—you don't know when you call this function that the bytes making up the function definition haven't changed. That's an assumption that an analysis tool has to make, which could be a faulty assumption that an attacker can exploit.

McGraw: This is a ridiculous question, but in your opinion,

what's the best programming language from a security perspective that's widely available?

Morrisett: Oh, that *is* a ridiculous question. I don't think it matters so much as the developers [matter].

McGraw: A bad developer can really screw up a good language.

Morrisett: Yeah, and vice versa. Really talented, good developers can take just about anything, and through a whole lot of discipline, produce something that is very robust, but it's not very cost-effective today to do that because it doesn't scale at all.

McGraw: Let's assume we have a whole bunch of mediocre developers, and we can choose whatever language we want to aim at them. What would you pick?

Morrisett: It depends on context. If I had the luxury of a clean-slate development where there were no constraints, I would probably pick something like ML or Haskell because they have really powerful type systems that let you capture more security-relevant properties in a language, but they're not widely used and may not provide the libraries or the functionality that you need. So then I would drop back to something that runs on the JVM or the CLR.

McGraw: Is the choice of a reasonable VM more important than, say, code review?

Morrisett: I think it is because code reviews—at least manual code review—just do not manage to catch off-by-one errors, no null references, all those.

McGraw: So in some sense, some of these code review engines for C that are becoming popular are making up for the fact that C doesn't have a type system and en-

About Greg Morrisett



Greg Morrisett is the Allen B. Cutting Professor of Computer Science and associate dean for computer science and engineering at Harvard University. He's well known for his work in programming languages, in particular, for the work he did while at Cornell on Cyclone. Morrisett's current work is on applications of advanced type systems, model checkers, proof-carrying code, and inline reference monitors for building efficient and provably secure systems. He's also working on languages for sensor networks. Morrisett has a PhD in computer science from Carnegie Mellon University. He served on the US National Science Foundation's CISE Advisory Board and on DARPA's ISAT Board. He lives outside of Boston with his wife and two children. Contact him at greg@eecs.harvard.edu.

forcement is poor and null references happen all the time and so on.

Morrisett: Yep. But what's interesting is even languages like Java have not been well designed so that these tools can give them the most bang for the buck. A good example is null pointers. They're the most common error in Java, and they can lead to a denial-of-service attack if you identify an unexpected null pointer exception that you can trigger. There are perfectly good language designs where not every reference has to have a nullable type, and you can actually end up with explicit checks in the type system that say, "Hey, you need to check for whether this thing is null before you de-reference it," which would be completely unworkable as a design in Java because every method invocation, every object reference would require a null check. But in languages like ML or Haskell, you don't have null as a default value for every single type. In fact, you rarely use it, so the design allows you to rule out a whole class of errors by default in a way that makes type checking more effective.

McGraw: Some aspects of your work have involved foisting formal systems into more widespread use—in particular, I'm thinking about proof-carrying code and your classic debate with Dave Ev-

ans at the InfoSec Research Council. What role do formalisms have to play in software security?

Morrisett: My current research is asking the question, "Twenty years from now, is it feasible that when you have a system that needs to be high assurance, can you mechanically prove that the code will respect a detailed security policy?" It's not at all clear to me that [this] goal is achievable, which is why it's research. But I do think that amazing progress has been made in the last five to 10 years on a whole bunch of fronts that make it within the realm of feasibility for at least some classes of systems. The level of assurance that you're getting out is incredibly

CLASSIFIED ADVERTISING

SUBMISSION DETAILS: Rates are \$110 per column inch (\$125) minimum. Eight lines per column inch and average five typeset words per line. Send copy at least one month prior to publication date to: Marian Anderson, Classified Advertising, IEEE Security & Privacy, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314; phone: 714-821-8380; fax: 714-821-4010.

LUNA INNOVATIONS INCORPORATED is seeking Senior Research and Software Engineers for our Secure Computing and Communications division. See job postings at: <http://www.lunainnovations.com>.

high when compared to the kind of processes that we use today.

McGraw: What do you do personally to try to influence the state of the practice in the commercial world? I know, for example, that you serve on a few advisory boards, including Fortify's, but what do you try to do to push things in that direction?

Morrisett: Well, I mostly serve on those boards because I feel like I keep in touch that way with what the real problems are. It's easy, as an academic, to go off and get interested in technically interesting problems that aren't practically relevant. Sometimes I think all we do is provide an outside, fresh view on the way that they're thinking. Sometimes we can point people

to new advances that they might be able to take advantage of, you know, immediately, and I think part of that is, as academics, we end up reading and going to conferences and networking in a way that's different than people do typically inside of companies.

McGraw: And I suppose you also seed them with various students.

Morrisett: Yeah, actually that's the real influence—we work people's brains in college or in grad school, and then they go off to become developers and hopefully carry some of the ideas forward.

McGraw: I wanted to shift gears a little bit and ask you some questions about what you're currently working on. What security and in-

formation control issues are posed by sensor networks?

Morrisett: Oh, lots of stuff, [but] sensor networks, at least in the setting that I've been working with Matt Welsh, are hard enough to program without even thinking about security.

People have not been thinking about this. A good example is, on some of these sensors, you don't have the power budget to even do reasonable crypto for some of the communications that you're doing. And then there are the privacy issues. Matt has set up a really interesting network around Cambridge of a whole bunch of sensors on light poles, and we'd like to turn those sensors loose on the community and let people write applications. Maybe they could do weather monitoring or traffic or pollution monitoring.

McGraw: Or tracking their friends or stalking people.

Morrisett: Exactly, right? I think we are going to have these platforms, and people need to be thinking about these security and privacy issues from the very beginning.

McGraw: Well, we're counting on you, Morrisett. Come on.

Morrisett: Well, no, my focus there has shifted, too, because now we're working on RoboBees, which are moveable sensor networks. [A RoboBees is] a lightweight, about the size of a dragonfly, [a] robotic insect that can fly around and do various tasks as a form.

McGraw: Like spy on people in the shower. I like it.

Morrisett: There we go. No, this is a big, \$10 million grant from NSF that's really exciting because it draws across a range of fields from biology—we have biologists studying how insect flight actually

ADVERTISER INFORMATION • SEPTEMBER/OCTOBER 2010

Advertising Personnel

Marian Anderson: Sr. Advertising Coordinator
Email: manderson@computer.org
Phone: +1 714 821 8380 | Fax: +1 714 821 4010

Sandy Brown: Sr. Business Development Mgr.
Phone: +1 714 821 8380 | Fax: +1 714 821 4010

IEEE Computer Society
10662 Los Vaqueros Circle
Los Alamitos, CA 90720
USA
www.computer.org

Advertising Sales Representatives

Western US/Pacific/Far East: Eric Kincaid
Email: e.kincaid@computer.org
Phone: +1 214 673 3742
Fax: +1 888 886 8599

Eastern US/Europe/Middle East: Ann & David Schissler
Email: a.schissler@computer.org, d.schissler@computer.org
Phone: +1 508 394 4026
Fax: +1 508 394 4926

works—to neurology, what insect brains look like, how we can replicate them—to swarm computing to security and reliability issues.

McGraw: So, at this point, you aren't really taking into account rogue sensors and misinformation and misuse and all these sorts of awful things you can think about doing if you had a network of little tiny bugs everywhere?

Morrisett: No, not yet. It's kind of like the early Internet days. The things that are wired up in these sensor networks are for typically scientific applications like volcano monitoring or pollution monitoring, that sort of thing.

McGraw: I think it would be great to make people believe a volcano was going to erupt when it wasn't. You'd get the population to leave, and then you'd take all their stuff while they're gone.

Morrisett: Oh, that is good.

McGraw: Hey, that's a new idea for a grant—or at least a grant application. Greg, you've provided more than your share of service to the research community, and we're all grateful for that, but we're a little confused because we're not sure exactly what this CISE Board does for the NSF or what's involved in a DARPA ISAT. How do those things work? What are they like?

Morrisett: Well, they're very different because NSF and DARPA couldn't be more different agencies for funding research. They're very complimentary. For NSF, one of the things that the advisory board does is review the programs and make sure that NSF is doing its job with respect to computer science research and trying to identify new areas where NSF might provide funding for research. A good example is the big

push into computational economics and game theory. That's a new area that NSF has decided to invest in. At DARPA, you get a very different model because NSF hands out lots of little grants for the most part, and they're all peer reviewed, which is very good in many ways, but has the effect of also making a lot of research somewhat incremental. For the same reason that getting a paper published can be hard, getting a peer-reviewed proposal funded is also very hard. Places like DARPA seem like they want to do more high-visibility, high-risk kinds of things, but they don't spread the money around as thinly as NSF does.

McGraw: Do you think that the United States will catch back up in funding science now that things are changing in Washington?

Morrisett: It certainly has improved a lot, but it's hard to tell whether that's a function of the recovery act, [meaning] it will go away next year, or whether it's going to really be sustained.

McGraw: What's the most embarrassing story that you can tell about our joint childhood fun? And even more importantly, how much will you pay me to keep the other one secret?

Morrisett: Ah, yeah, I was going to say, is it embarrassing for you or me?

McGraw: This is your chance to decide.

Morrisett: Tell your listeners to call me up, and I'll be happy to tell them many stories about you. Many, many stories.

McGraw: You can't think of any?

Morrisett: Oh, I can think of many, but I'm not sure that it would be appropriate for me to...

McGraw: You'd have to admit that you were there.

Morrisett: Exactly. Maybe that's it. You always had really good parties, and there was a really good party one time where there was a bonfire outside. And some idiot decided to pour gasoline on...

McGraw: That was John Rees.

Morrisett: On the fire. Was that John?

McGraw: Yeah, it was.

Morrisett: And then I told everybody to step back, that I would light it, and I struck the match, and a huge fireball happened. I burned off my eyebrows and arm hair.

McGraw: Well, we're glad that it's grown back by now.


Morrisett: I'm not sure it has, but...

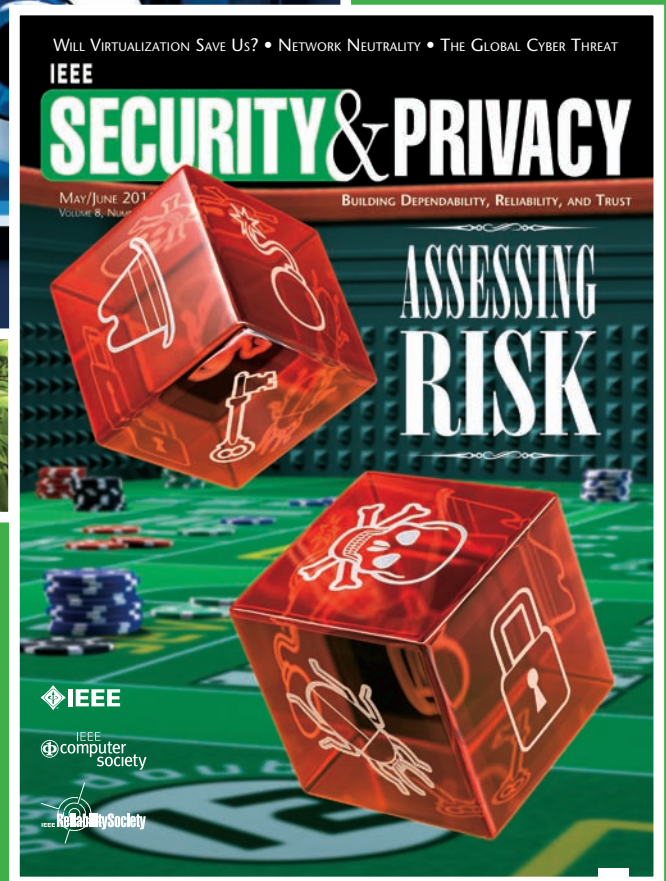
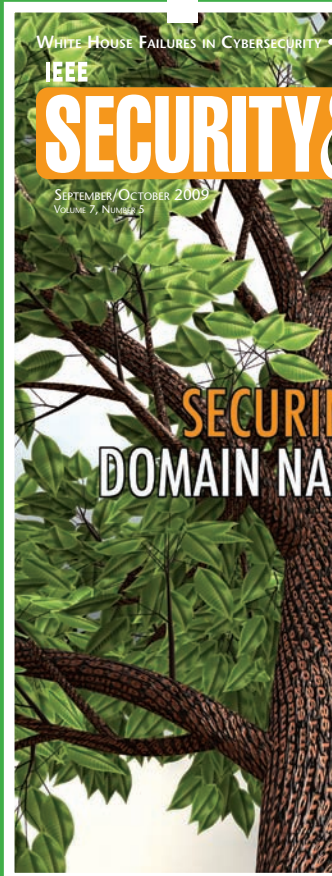
McGraw: Well, thanks Greg. It's been a fun interview.

Morrisett: Thank you.

See the full text of this interview at www.computer.org/cms/Computer.org/dl/mags/sp/2010/05/extras/msp2010040006s.pdf. □

Gary McGraw is *Cigital's* chief technology officer. He's the author of *Exploiting Online Games* (Addison-Wesley, 2007), *Software Security: Building Security In* (Addison-Wesley, 2006), and seven other books. McGraw has a BA in philosophy from the University of Virginia and a dual PhD in computer science and cognitive science from Indiana University. Contact him at gem@cigital.com.

 Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.



■ Subscribe Now!

IEEE Security & Privacy magazine is the premier magazine for security professionals. Each issue is packed with information about cybercrime, security & policy, privacy and legal issues, and intellectual property protection.

Watch for these special issues! ■

Mobile Device Security ■ Sharing Sensitive Information ■ S&P of Cloud Computing
Reliability of Embedded and Cyberphysical Systems ■ Engineering Secure Systems

www.computer.org/security