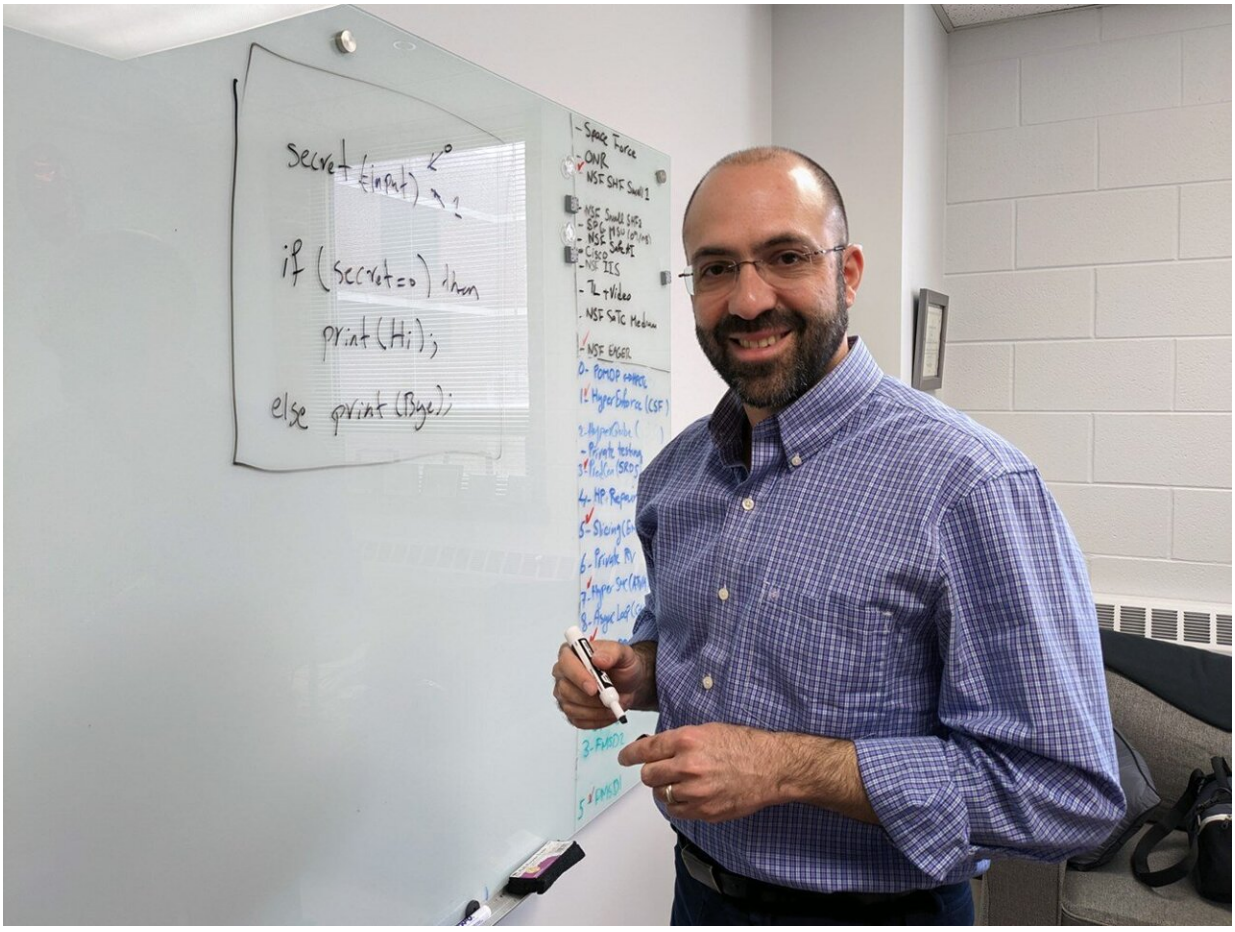
Q&A: Expert discusses how new research can help protect private data

June 14 2023, by Matt Davenport



Borzoo Bonakdarpour, an associate professor at Michigan State University, talks about information-flow security while drawing up an example on a whiteboard. Credit: Matt Davenport/MSU

With so much of our lives being stored online and in digital databases, it's reassuring to know there are researchers out there like Borzoo Bonakdarpour of Michigan State University.

Bonakdarpour, an associate professor in the Department of Computer Science and Engineering, works to prevent information that people want to keep private from leaking into the public.

Talk of data breaches probably calls to mind one (or several) of the well-publicized examples of people being hacked through phishing scams or lax security practices. But Bonakdarpour, a recipient of a 2023 Withrow Teaching Award, focuses on a more subtle aspect of data privacy that carries the same high stakes.

"We sort of trust that computer programmers don't make mistakes," Bonakdarpour said. "But they're still human, right? They make mistakes all the time. This can introduce bugs that are accidental, but they can still result in massive security breaches."

Bonakdarpour and his team recently won a grant from the National Science Foundation to develop what he calls "enforcers," programs that can automatically spot and remedy those digital gaffes before they do harm.

MSUToday sat down with Bonakdarpour to chat about cybersecurity and learn more about his work.

Where does your work fit into the big picture of cybersecurity?

The project we're talking about is just one aspect of cybersecurity. Outside of that, there's database security, network security—there are a

ton of different types of security.

My focus is on what's called information-flow security and on developing algorithms that can verify the correctness of computer programs with respect to information flow.

What is information-flow security?

Information-flow security is about how secrets can leak into observable public channels. Let me give you one small example.

When I first started studying this, I was writing papers for a conference with my students, and I could log into the conference management portal that showed the status of all our submissions in table form.

The status was color coded: there was one "accepted" submission shown in green, one "rejected" submission shown in orange, and there were two pending submissions shown in yellow.

Each entry also included a "Session" column. For the green accepted paper, that column said "not yet assigned."

Looking at that column for the yellow entries, one was blank, but the other said "not yet assigned." From that, I could guess that this paper was probably internally marked as accepted.

But this information was supposed to be confidential. I should not have been able to guess anything about what was happening internally while it was pending. We took a screenshot of that table and put it in the introduction of one of our papers.

So while this example isn't overly sensitive, it demonstrates how information can easily leak from a private channel to a public channel.

Where can you find information-flow concerns where the stakes are higher?

There are a lot of companies—from very big to small—that are paying attention to the correctness of information flow.

One area it's really important is in cloud services. For example, Amazon is very interested in this. If there's a bad flow of information from user to user on the cloud, that could mean a company disaster.

Paper title	Abstract or manuscript deadline	Status	Edit	Session
Example paper 1	February 2, 2015	Accepted		(Not yet assigned)
Example paper 2	October 18, 2014	Rejected		
Example paper 3	December 23, 2014	Pending		(Not yet assigned)
Example paper 4	December 23, 2014	Pending		

Borzoo Bonakdarpour of Michigan State University is interested in how bugs in systems can publicize private data. Serendipitously, Bonakdarpour found an example of this while submitting papers to present at a conference. The above chart provides a simplified example of what he saw when he logged onto the conference's paper management portal. One of his pending papers in yellow—that he shouldn't know anything about—looked similar to an accepted paper, whose information was no longer secret. This let Bonakdarpour infer the pending paper was also accepted. Credit: Michigan State University

Another big area is hardware design. Intel learned this the hard way in 1995 when, at the time, there was a bug in the Pentium processor code. They had to recall everything.

Then a few years ago, we found out there were different bugs in processor designs. They were called Spectre and Meltdown and they were very famous because virtually any computer could leak information.

Companies started developing software patches for that and the problem isn't bad now, but it just shows how subtle and intricate these problems can be.

Can you give an example of what that subtlety can look like?

For instance, how long it takes for a program to run can depend on a secret value. So, if you run the program twice, I can infer the value of the secret just by how long the program takes to execute each time.

There's an algorithm for encryption, and early implementations could not sense that your private encryption key determined its execution time. So, an attacker could guess your encryption key based on that time. The countermeasure is that the algorithm runs constantly, no matter what the key is, so an attacker cannot guess it.

And attacks can be even more sophisticated, like using the radiation or heat signature from your processor to reveal information.

What is your lab doing to help address these vulnerabilities?

We're developing algorithms to verify the correctness of computer programs with respect to information flow—sort of taking humans, who can introduce bugs, out of the loop. We're working to develop programs that can automatically generate other programs that we know are correct by construction.

With our new NSF grant, we're taking the first step. We're not generating those top-level programs, but we are generating what we call "enforcers." Enforcers look at what goes in and out of your computing system, and if it senses that something is not right, it starts addressing it.

It's kind of like a safety net, but more active. It doesn't just raise a red flag, it also tries to correct it. It doesn't fix the code—that's another cool problem called program repair that I have another pending proposal for—but it can take actions based on inputs and outputs.

I'll give you an analogy as an example. Imagine something goes wrong with the traffic light at an intersection. If there's a problem and both directions have a green light, there can be accidents.

An enforcer would force one or both of the lights to be red. It doesn't necessarily solve the root problem, but it prevents an accident until somebody can.

Do you have any advice on how those of us who aren't cybersecurity experts can keep our information safe?

This is not necessarily related to security, but my personal view is that we are sharing too much information with companies.

Two years ago, my wife Googled mattresses for our kids and the next day I'm getting Facebook ads for mattresses. We haven't declared that

we're married on Facebook, but somehow it knows we're connected.

I'm actually not a secretive person at all, but I think we should share as little as possible with these companies. At the same time, these companies and their services have privacy instructions. They have knobs and settings you can adjust to take care of your information, so be sure you're paying attention to those.

Provided by Michigan State University