MIT CSAIL Alliances | Mengjia Podcast Export 2

Welcome to MIT CSAIL Computer Science and Artificial Intelligence Labs alliances podcast series. My name is Steve Lewis. I'm the Assistant Director of Global Strategic Alliances for CSAIL at MIT. In this podcast series, I will interview principal researchers at CSAIL to discover what they're working on and how it will impact society

Mengjia Yan is an Assistant Professor in MIT's Electrical Engineering and Computer Science Department Her research interests lie in areas of computer architecture and security with a focus on side channel attacks and defenses. Her group works on exploiting new micro architectural vulnerabilities and designing comprehensive and efficient defense mechanisms.

She is the recipient of multiple awards, including the NSF Career Award, Intel's Rising Star Faculty Award and multiple micro top picks in computer architecture. Before joining MIT, she earned her PhD in computer science from the University of Illinois at Urbana-Champaign. Mengjia, thanks for your time today.

Tell me about some of the biggest problems right now in hardware security.

I think the biggest thing is that used to be the case that people think hardware security, you must have an attacker get close or have the possession of your device and they try to use some very fancy techniques to probe your device and get some secret out of it. Well, the biggest change today is that now hardware attacks can be conducted via software exploitations.

That hackers may only need to send you a malicious website link or it launch attack by running some looks benign applications on your device. It can exploit hardware vulnerabilities in a very similar way as exploiting software vulnerabilities, which make hardware security a much more serious problem because it is much more accessible to the attackers nowadays than before.

So the biggest security vulnerability we have been knowing about chip like processor vulnerability a Specter meltdown, which was discovered in 2018. And why it is a big problem because it violates the interfaces between the hardware and software. People used to think the interfaces between the software and hardware is the instruction set architecture like an assembly instructions you write code with that.

Nowadays people realize it's not-- you need to expose more vulnerabilities at the micro architecture level to the software people. I would say that is the biggest problem. I know it's a bit abstract but a lot of security vulnerabilities nowadays pose a serious security threat because they violate the boundary between soft and hardware.

I see and I think people would generally think that a hardware or chip is very much locked down. There it's burnt into the firmware and it's not readily accessible. And of course Specter proved that all wrong. But let's talk about a side channel attack. Can you tell our audience what a side channel attack is?

Yeah, so a relatively formal definition of a side channel is really it is a form of information leakage via some unintended communication mediums. So let me give a very simple example. We have this network nets and you need to connect to network-- outside network using some routers. Let's say, you share this router with your roommate or your family members. A direct information leakage, as opposed to side channel, a direct information leakage is that oh, you may have some malicious application, monitor the packages going through the network and the probe inside the packet to figure out what the secret happening.

While a side channel is doing something very different. That is, if you share the router with your roommate, you will see how fast your network speed is because if your roommate is using a lot of network bandwidth like streaming videos, et cetera, you will see your bandwidth is limited, reduced, so your network latency will be increased. While on the other hand, if your roommate is not using the network, or is just using some very lightweight web browser surfing, et cetera, then you have all the network bandwidth belonging to you so you will see your network speed is very fast.

By just looking how much bandwidth you have and how slow or fast your network speed is, you can figure out what your roommate is doing. So that is an example of side channel in the sense that you kind of use this bandwidth as an unintended communication media to leak some information.

While, of course, the example I give is a very coarse-grained information leakage, but people see that you can bring it with some fine tuned stuff and you can do some very fine grained secret leakage.

So does this mean that a hacker via side channel keeps a communication channel open with the hardware so that they can then steal information?

Yes.

Was it more just to degrade the system performance of that piece of hardware?

So both-- while you are-- the example I give is more like an active attacker, where the attacker need to probe the system, but this probe is very non-intrusive because it just using the network in a very normal way-- I'm sending in the normal request to some YouTube website, et cetera, so it's very difficult to be distinguished from a normal user. In this active attacker case, you do need to attacker need to proactively access the resource, the communication channel.

While there does exist some passive attack strategy, for example, some power side channel tags EM, Electronic Magnetic side channels, for these attacks, attacker will need to just passively monitor the power utilization of the device, et cetera. But of course, this requires you to get closer to the device and try to have some [INAUDIBLE] equipment to receive the signal, and some acoustic side channels where you just try to distinguish some secrets based on the sound generated by your hardware devices.

In that sense, it's quite passive. But usually-- not 100% many passive attackers require the attacker getting closer to the hardware device. While a lot of active attackers, they just require you to run a software application on the victim platform.

Can you tell us a little bit about what a Pac-Man attack is?

Yeah. So Pac-Man attack is-- the unique thing is that it is a software-hardware co-attack. Basically it takes the side channel attack idea and they use it to exploit some software vulnerabilities. And you know that software vulnerabilities are all over the system, and it's very difficult to eliminate all of them so there's already existing a lot of defense mechanisms.

And one of the very important and useful mechanism is called ARM pointer authentication, which was introduced by ARM and Qualcomm in around 2017. The idea is really that you have very important data structures in your program, and if you have a software vulnerability, this data structures may be tampered, crafted by attackers. Attack pointer authentication says, let's add some metadata to this data structures.

I will compute a hash, or some message authentication code of your important data structure, and I will verify the integrity of the data structure to see whether it has been tampered or not. So they use this to protect pointers which can block-- control for hijacking attacks, et cetera, quite effective attack-- quite effective defense mechanism. And the Pac-Man attack is showing that by exploiting some hardware vulnerabilities, we can actually bypass this very strong security perimeter of enforce a software level.

So the idea is really that if an attacker tampered some important data structure, and it also tampered the message authentication code, the program will crash. However, the idea here is that you could-- attacker could guess the message authentication code and speculatively check whether this guessed authentication code is correct or not. And the biggest thing here is that by doing all these things under some speculation window, like what Spectre Meltdown did, all these things will not crash your program. You can brute force trying all these-- guess the values-- under speculation and you will find the correct one. And then you'll see the correct one into the program to bypass the security check.

So it's-- I think the unique thing is really that we combined software attacks and hardware attacks, which used to be considered in isolation before. Software researchers, hardware researchers you look at probably separate, now the [INAUDIBLE] kind of a bridge each other, now, we really require some more advanced and more broad defense mechanism, consider both sides.

So how can machine learning assist in stopping these types of attacks?

Great. Machine learning, from my perspective, it's a very useful technique. Unfortunately, I think from what I have seen in the community, the research community, there are more uses of machine learning to assist attacks than stop attacks. It's a sad fact, but I think it's really that because it's very handy, machine learning techniques, they are very good at finding correlations between two signals.

So that's why I've seen a lot of work trying to incorporate, to use machine learning to do post-signal processing of the side channel traces to find out which traits correspond to which secret, and also people try to use machine learning to generate attack code because many times, we need to tune our tech towards a specific machine, some microarchitecture structures, et cetera. And the machine learning is very good at doing that. So that is why the fact that there's more work on using machine learning to design more advanced attacks, which is also a concerning trend.

On the light side, bright side, there are also exists some work using machine learning to mitigate, to block these attacks. But I would say the work on the defense side is very limited. Because for attack, you find one attack, bypass the system, you declare success. While for defense, it's very difficult to deal with like, false positives, false negatives. My sense is really that we need more work on this direction. There is some progress, but the progress is not satisfying enough and we should definitely push more on the defense side using machine learning.

I see. And how easy is it to patch these vulnerabilities? And how responsive are the chip manufacturers once they're exposed?

It's a very good question. I have to say that compared to the software community, the hardware community is much lagging behind in dealing with the software-- dealing with these vulnerabilities, bugs, et cetera. But since 2018, the whole community, the industry has been evolving a lot. So there's a lot of open bounty programs and whenever you have some vulnerabilities, you report to the hardware vendors, and they will try to have detailed discussion with you, and they will quickly try to release either software patches, because a lot of these hardware vulnerabilities which require you run some specific code pattern on the machine, they can be patched using some software techniques, which comes with some high performance overhead, but that is the immediate mechanism patch you can take.

Second is that, a lot of companies, they can modify your hardware by releasing new code soft microcode patches. So nowadays, hardware is so complex. So it is not everything is baked into the circuit-- silicon. There is some flexibility there such as microcode. You can think about that is even lower level of code below the firmware, so you can patch, change some behaviors of the hardware by releasing some microcode patches. Finally, we will need to wait for new hardware to be released. I think the community has been evolving so much, so this trend of process is being improved so much over the last few years. So we do have some good promise on that.

My own experience with Pac-Man attack, because Pac-Man attack comes from our group, we talked to Apple and also ARM about this vulnerability, Apple kind of reacts in the sense that, because it exploited the software vulnerability, if you patch your software to mitigate as much-- as many software vulnerabilities as possible, then you can also mitigate Pac-Man attack. And ARM, I really liked ARM's response. ARM wrote a very in-depth, detailed, thorough response about how Pac-Man works, and how they can react to these attacks. They have some software mitigations and some very clever, very clever software-hardware co-design mitigations posted on their blog post.

Specifically, they have some versions of the point application which is implemented slightly different than the version we exploited and they can reposition that version to mitigate Pac-Man attack, which is a very clever idea. I would say, that to conclude, just about the response is from industry, it's not an end world if you have a hardware vulnerability nowadays. You still have software patches and the microcode patches can go, and you can reposition some variations of hardware information for better defenses.

And are these vulnerabilities sort of more prevalent in say, like an ARM architecture, or Intel architecture, or RISC architecture? Or are they sort of just any chip platform can be exploited?

Let's put it this way. It really depends on what types of attacks you are talking about. So if you are talking about Spectre Meltdown, this very generic form of speculative execution vulnerabilities, because it is such a classic and prominent feature in almost every microarchitecture, so all the processors will be vulnerable. Unless your processor very old and very low performance. So there's some very low performance-- in older cores who use it in microprocessors, then that will not be vulnerable but, all the others will be vulnerable. As long as you have this Spectre exclusion features.

But there does exist some marginal small differences between different processors. For example, Intel processors their engineers are most prominent. And also, Apple, the reason-- the processors, they are very powerful, like monsters-- the more powerful they are, very likely the more optimizations they have to integrate into the chips. That's why you get all the benefit, performance benefit, right? But it's also very likely that these optimizations may introduce new vulnerabilities which were not discovered before. So that is unknown.

But the comment I'd like to make is that it is very great research topic for us because we really enjoy digging deep into processors to find all these vulnerabilities. But I would say that many of these vulnerabilities that are deeper, or more complex, than Spectre Meltdown, that they are actually also very difficult to be carried out by an attacker in the real world for practical setup.

So as researchers, we really want to do some clinks like, some security mechanism with perfect security properties and in that sense, we need to consider all the tricky attack vulnerabilities, but for practical deployment, I have to say, maybe business people should really focus on the very basic Spectre Meltdown of vulnerabilities.

The other differences in this more deeper vulnerabilities, they are more like marginal difference, honestly. That is the different view from research versus I think industry.

So some of the steps businesses can take to protect themselves from security risks, is this all about educating the users about not clicking on emails with links, and things like that? Or what would you say?

Some generic tips I would like to give is to upgrade your system as early as possible, because the companies frequently release some patches for both software vulnerabilities and hardware vulnerabilities.

Another interesting phenomena I have observed recently is maybe people should take more conservative measurements because recently, like I think it just happened the last week, Google Zero project. They find a new vulnerability because of such an interesting scenario. One Spectre Meltdown came, it is such a serious problem. So the first reaction people have is to patch the software because that is the only thing we can do, and the people have this KPTI which is separate your kernel and the user space. So that user space cannot directly access kernel even speculatively, which is a very superlative, very effective but introduce very high performance overhead.

Now after a few years, now Intel introduces-- and also other companies-- introduce hardware-based meltdown fixes. So with this fixes you can mitigate meltdown with much lower performance overhead. Then the Linux community said, so now we have the hardware fix, so we should like, disable this KPTI feature, the software fixes which introduces unnecessary performance overhead. And guess what? In the last week, the Google Zero project blog post posted something that, if you disable the KPTI feature, the original attack may still go through with some variations.

So I'll just say that nowadays, we are shifting back and forth between software fixes and hardware fixes towards the same attack, and the because they are different companies, different organizations, the coordination may not be at that fine granularity. So these features may not fix the problem at the same level, and if you just very aggressively disable these features, you get some performance benefit, but maybe it will leave some doors, some narrow gap for security breaches.

I guess this is something the community, hardware-software community need to be coordinated better, to see whether their patches really overlap, or whether they are not complete overlap, so that people can be better advised. But for now, I guess we should be more conservative . Just according to this lesson I've seen last week.

Does your group get sort of advanced chips design so that you can test? Or do you have to wait till it's commercially released in order to find these vulnerabilities?

Usually, we need to wait for it to-- is commercially released, then we find this vulnerability. That is the current state.

And what do you see as sort of the next great security problem facing industry?

I think what I've described so far, about the interface between hardware and software, how to coordinate them, will be a long-lasting problem for a very long time. So this could include the Spectre Meltdown, side channel, microarchitecture attacks. It also include rogue hammer attacks, physical side channel attacks, and even reliability issues nowadays people facing about fault injection attacks. All of these will eventually interact both software and hardware, because both the software and hardware becomes more and more complex as the days goes by.

This interface issue, how to expose a better contract between the hardware-software is the very challenging problem for both researchers and industry.

And where can people go to find out more about your research?

So we have our own-- I have my own personal website. You can easily find it on MIT, that will be the main place to find my research. In addition to that, I would also be very excited to announce there's a hardware security course at MIT, and a lot of senior undergraduate students and master's students take these courses. We have a lot of useful materials.

It covers the broad topic about how to read security and some selected topics, advance topics, and we also have some detailed hands-on lab assignments. Actually, a lot of institutions are actually using our materials, including some government agencies and some other universities. So if people are interested, they can also visit that website, that course website, which I would be very happy to provide a link.

Great. Well, Mengjia, thank you very much for your time. It's a fascinating topic and obviously, your research is vital to improve security for our computing platforms.

Thank you. Thanks for having me.

[MUSIC PLAYING]

If you're interested in learning more about the CSAIL Alliance Program and the latest research at CSAIL, please visit our website at cap.csail.mit.edu and listen to our podcast series on Spotify, Apple Music, or wherever you listen to your podcasts. Tune in next month for a brand new edition of the CSAIL Alliances Podcast and stay ahead of the curve.