Carnegie Mellon University
Software Engineering Institute

SEI Podcasts

Conversations in Artificial Intelligence, Cybersecurity, and Software Engineering

What Could Possibly Go Wrong? Safety Analysis for AI Systems

featuring David Schulker and Matthew Walsh as Interviewed by Thomas Scanlon

Welcome to the SEI Podcast Series, a production of the Carnegie Mellon University Software Engineering Institute. The SEI is a federally funded research and development center sponsored by the U.S. Department of Defense. A transcript of today's podcast is posted on the SEI website at sei.cmu.edu/podcasts.

Tom Scanlon: Consider this likely scenario. You are part of an organization that is concerned about data security, yet you also want to take advantage of the many benefits LLMs can offer. So, you decide to self-host an LLM system in your own environment for your research. This LLM can reach out to the internet to search for scholarly papers that will support your work. It can execute commands in a terminal, for instance if you want to run a calculation. You think you are protected because no one else can issue commands to your LLM but you. However, a common attack pattern is to leave an adversarial prompt on the internet, just waiting to be found by a search engine and pull it into your LLM context. Through this prompt your adversary could commandeer your LLM, get it to exfiltrate data using its terminal access. All it would really take for this to happen is a single HTTP request out to an attacker website, and you wouldn't even know that data is leaving your organization. If attacks like this can even affect a self-hosted, seemingly secure LLM, how can you really ever know whether an LLM is safe to use?

Welcome to the SEI Podcast Series. My name is Tom Scanlon. I lead the CERT Data Science Technical Program, which incorporates AI, machine learning, and statistical analysis to develop solutions for cybersecurity challenges. Today, I am joined by Dave Schulker and Matt Walsh, both who work as senior data scientists and the SEI CERT Division. They are going to discuss their work on System Theoretic Process Analysis, or STPA, which is a hazard-analysis technique uniquely suitable for dealing with the complexity of AI systems in assuring AI systems. Thank you both for sitting down with us today to talk about this important work. Before we get into this discussion, let's start by introducing yourselves a little bit more, and your unique paths that led you to the SEI, and the kind of work that you do here.

Matthew Walsh: Hi, everybody. My name is Matt Walsh. I am a senior data scientist here at the SEI. My undergraduate degree is in psychology. My PhD is in computational neuroscience. For the past 20 years, I have been studying human performance, training, and education. One of the coolest things about working at the SEI, and one of the most exciting things about being a data scientist today, is to see the new capabilities that AI systems afford and looking at the different ways that humans can interact with and leverage those capabilities.

Tom: And Dave?

David Schulker: Hi everyone, my name is Dave Schulker. I went to school to become an applied microeconomist. One of the things that I really struggle with in doing applied micro is that most of the information that we care about in the world does not exist as pre-quantified, numeric things that we can stick into models. Most of what we want to learn about the world is in these unstructured forms like text or images or things like that. And so, I have always been attracted to machine learning. A lot of my research for most of my career has been trying to apply machine learning to get better models of the real world so that we can use them. The best thing about working at the SEI is that we are not just doing analysis of static data sets, but we are trying to use these incredibly capable models to build real systems that solve national security problems.

Tom: Great, thanks, Dave. One thing I want to point out that you and Matt may not be aware that you have in common is we have a lot of veterans that work here at the SEI. We also have a lot of PhD researchers, but you happen to be both—folks that served in the military and went on to get PhDs and came to work here at the SEI. I was wondering if either of you wanted to comment on how that background affects the work you do here.

Matthew: Yes, absolutely. While I was an undergrad at Penn State, I joined the Pennsylvania Air National Guard, and I served in the Air Force for six years. So that led me to a postdoc at Air Force Research Lab and then from that to work here at the SEI. Given that background and experience, I have been very comfortable and eager to apply my skillset in the context of national security.

David: Just to build on what Matt was saying, I am a former Air Force officer and Air Force Academy graduate. Go Air Force, beat Navy. Really what I think that helps with in terms of our work for the SEI is you are constantly encountering national security challenges in your military operational missions that you want technology to be able to solve for you so that you can be more effective. Serving your country on the operational edge is rewarding and is helpful, but then being at a place like the SEI, where we can develop technology that can help those operators be more effective is also rewarding. I think my favorite thing is being able to still be involved in the mission from here, from this vantage point, post-military career, is very rewarding.

Tom: Great, thank you. I appreciate that background. Today we are going to talk about AI safety in assuring AI systems. Before we get into that, the technique that you all are using, I wonder if you could explain, how do AI systems differ from traditional software and hardware systems? At the end of the day, AI is software, right? Why can't we just use our traditional techniques? How is AI different?

Matthew: If we look back over the past 100-to-130 years, there has been an evolution in our technologies, and so we have gone from technologies that are primarily mechanical to electromechanical. Then beginning in the 1970s, the emergence of software. Now in the past 10-to-15 years, the addition of AI capabilities to those systems. And so, like you said, Tom, AI capabilities, machine learning, this is all implemented in computer code. So, it is an example of software, but it differs from traditional software in some key ways. The first key difference is how AI systems are built with traditional software, designers code explicit rules and logic that the program executes. In the case of AI systems, many behaviors are learned from training on data rather than providing the system with an explicit set of rules. The system's able to learn statistical patterns and generalize to new inputs based on the data that were used to train it. Some additional differences. This dependence on data leads to a dependency both on the code that implements the AI capability as well as the data that are used to train it. Another difference is the adaptability of an AI system. In a traditional software system, once it is deployed, its behavior only changes if a developer goes in and modifies a code. With some AI systems, they can learn online and continue to adapt and improve their performance as they interact with the world or an environment. Another difference concerns the behavior and predictability of these systems. So, with traditional software, the behavior is typically deterministic. Given the same input, you will always get the same output. AI systems, on the other hand, can be probabilistic, and so given the same input, it may produce different outputs. Another kind of related distinction is with the rise of generative AI, AI systems are now capable of producing complex outputs: things like images, videos, text passages. So that goes beyond the flexibility of the outputs that traditional software systems typically generate. Then the last difference that I want to point out is the flexibility in how they go about pursuing goals. With a traditional software system, the developer typically gives the system a sequence of steps to perform in different conditions. Whereas with an AI system, you typically give the system an objective, and then the AI system may pursue that objective in multiple and more flexible ways. To sum all of that up, differences between traditional software and AI systems kind of boil down to how they are built—whether you explicitly program the logic, or whether the system learns from data, the dependency on data, the adaptability of the system once it is deployed, the behavior and predictability of its outputs, and then lastly, the flexibility in how it pursues goals.

Tom: Let's apply what Matt was just saying and think about the system that talked about in the very beginning. You have this system that has a large language model, it is running entirely on your own network, and it is using that creativity, that complexity to solve problems by calling tools or using other things. But it is all on your network, and so you might think this thing is contained. But, at the same time, the flexibility potentially opens you up to new vulnerabilities that are going to surprise you. For example, if somebody is able to get an instruction to your LLM, then it might follow that instruction, and then it might use all of that creativity and all that access to tools in order to work against you without you even realizing it. These are some of the reasons why we might need to approach these machines learning-based or AI-based systems differently when we think about testing and assuring them.

Matt: Great, thank you both. I think you did explain that well. There are a lot of similarities with traditional software systems, and we still need to do traditional cyber protections, but there are also these differences for AI systems, and so we need to do other things to address those differences that are unique to the AI systems. Given that, that there are these differences as you explained so well, what have researchers, practitioners, what have people been doing to assure the behavior of AI systems?

David: There is this paradox when you think about these AI models because on the one hand, they are impossibly complex. They have hundreds of billions of parameters some of them. And so, you think it is completely unknowable in terms of what this system actually is. And yet, on another level, they are remarkably simple in terms of the system itself is fundamentally just a statistical model that is trying to predict probabilities about what comes next. So, it is either trying to predict pixels in an image, or it is trying to predict fragments of words that come into a sequence. And so, because they are statistical models, and they come out of the field of statistics, the default way to think about testing them is the way that you would test any other statistical model, which is you take a portion of your data, and you intentionally hide it from the model. Then, when it comes time to test, you give the model this hidden data, this held-out data, and you see how the model does with data that it has never seen before. As long as the new data that is coming in the future looks similar to the data that you held out, then you assume, *OK, well, if it performs all right on that test data, then it will perform all right in the real world.* That makes sense in statistics when you think about, usually you have a random sample from a population. So, if you have a random sample, then another random sample is going to be similar to the first one. Typically, that is where you start with approaching it. You think, *Alright, I have this large language model, I am going to create examples of tasks or questions I want it to answer or something like that and try to create a dataset that it is never seen before. Then once I am done creating the model, I'll give it that new dataset, and I'll see how it does*. That is generally the logic, but the problem is that that logic breaks down because of a lot of the factors that Matt was already talking about. When you think of the relatively infinite input space and the infinite potential output space, you can't necessarily make that assumption that if I tested it on this dataset, then that is going to really approximate what it might see in the future.

Matthew: Just to build on Dave's response and go back to the vignette that is motivating this discussion, an organization has developed a research LLM, and they are using it to answer questions and to generate code. The typical approach to test the behavior of that system would be to use some type of academic benchmark and some corresponding academic measures. For example, you might have question databases with reference answers. You can give those questions to the LLM system and see how closely its answers match the reference responses. And in this system, it is also generating code, and so there are code benchmarks that are publicly available where you can give the LLM a prompt and see how closely the code that it generates matches the functional performance of the reference code. Now, the issue with this, and there are examples, the LLM may be able to pass these academic benchmarks. And that says something about its knowledge base, but it doesn't tell us precisely how well the system will be able to apply that knowledge. There is this famous example from a few years ago where GPT passed the LSAT, but then a lawyer used GPT in practice and cited falsely generated legal precedent. So, the LLM was able to pass the benchmark but not apply the knowledge that it had.

Tom: I like the way you applied that to the vignette we have been talking about, Matt, appreciate that. But in doing so, you highlight that there is still a gap in the way people are approaching this problem. There are some shortcomings still there. So, what other types of safety analysis approaches might someone apply to help close that gap?

Matt: Yes, great question. The academic approach to test and evaluation of AI systems isn't wholly adequate to provide safety assurances about the behavior of those systems. And so, to close that gap, we can look at the safety engineering literature and the set of approaches that are used in that field. If we look back at the evolution of technology over time, the approaches used to evaluate the safety of systems have evolved. In early systems, they were primarily made up of mechanical components, and that evolved into electromechanical systems. The established view during that period is that accidents are caused by component failures, for example, a valve braking or a tire deflating. If the predominant view is that accidents are caused by component failures, it makes sense to use analysis techniques that focus on the ways in which a component can fail and to look for ways to mitigate the cascading effects of those failures and to increase the reliability of components. This early view of accident causation, which focused on component failures, emphasized solutions to increase the redundancy or the resiliency of components. Today's systems are more complex. They include software. They are made up of systems of systems, and they incorporate new technologies like AI. This shift toward greater complexity has also given rise to a new view of accident causation. And in this new view, accidents are caused not just by component failures but also by unsafe interactions between properly functioning components.

Tom: Dave, given what Matt just said, how would you apply this to the example vignette that I opened with, with the LLM agent system?

David: The flaws and the failure-based accident thinking are really apparent when we start to think about AI systems like the one that we are talking about with the agentic large language model that is searching and then performing actions using tools on your system. In the example, the agentic LLM searches the internet, it finds a relevant result, but that relevant result contains a prompt that is designed to mislead the LLM to take to take an action that the attacker wants that you don't want. The LLM follows the instruction, and then it sends internal data away through an HTTP request. Now, if you think about that chain with a failure mindset, which part of your system broke the way a valve would break? Did the search mechanism search the internet, and it found relevant results, just like it was designed to do? It gave those results to the LLM. It contained an instruction, and the LLM followed the instruction just like it was designed to do. But then it led to a harm or a loss in the real world. And so, the point is that that you can't use this failure mindset to identify which one of these components failed. None of the components failed. But what happened was, they all did what they were authorized to do, but they did it in a particular context where these authorized actions were, in this case, not secure or unsafe, if you want to say that in a general sense. So, we clearly need to think more broadly than the accident model that says that our components are going to break if we want to anticipate some of these problems and try to figure out, *Well, what is the safe envelope for this AI system to operate, and how can I constrain it so that it won't ever exit that envelope?*

Tom: STPA has its roots in physical mechanical systems, and so it makes sense when there's a component failure there, that I know I need to replace components or put on backup systems. But when I apply STPA to AI like you just explained, Dave, is it just a thought exercise, or can I get actionable outputs out of this STPA for AI, where I can actually do something about it?

David: Yes, it definitely leads to actionable outputs. The primary one is going to be an improved system design, because the fundamental thinking behind STPA is a better fit for the types of systems that you are trying to design. And so, one of the great things that the folks at MIT who developed STPA always point out is that if failure is the problem, if components breaking is the problem, then you want to add components in order to increase the probability of success. But if complexity is your problem, then adding components makes that problem worse. Now you have made your system more complex, and you have opened yourselves up to more potential problems. The high-level thing that you get out of STPA is a better design, but then concretely, I will give you a few specific examples of ways that when we have applied STPA to AI systems in particular, it is reliably helped us identify and prevent flaws.

The first one is it helps you immediately ground your problem in an actionable definition of safety or security. One of the things for anybody who is familiar with work and literature on potential AI risks or AI harms, there are lots of great taxonomies of all the potential risks that AI potentially introduces, but those taxonomies are vast. One taxonomy that Matt and I like to use a lot has hundreds of potential risks that an AI system could potentially introduce. If you sit down to design a system, and you say, *I need to avoid these 300 risks*. Well, that is intractable. It is very difficult to try to think about a system design there. So instead, in STPA the first step is to really understand what are the real-world harms—or in STPA parlance, you call them *losses—what are the real-world losses that I must avoid?* You usually get about 3-to-5 losses: high-level things like loss of life or loss of property or a financial loss or a loss of privacy, something like that. And so, you get about 3-to-5, typically, losses, and you say, *OK, that's my definition for safety. If the system doesn't result in these losses, then that is what we are trying to prevent*. *That is what we call safe*. So right off the bat, it helps you ground your design in an actionable definition.

But the next thing that it is really useful for is, once you have those losses, you unpack it. Before you even thought about implementing the system or coming up with an architecture or anything, you unpack the system into, *Well, what are the states that could result in those losses, and then at a high level, what do I need to prevent, essentially, to prevent the losses?* And then you take a step where you assign responsibility for those states to different parts of the system. And even though this sounds simple; we are just going to decide what we must not do, and we are going to make someone responsible for it. Every time we do this, we identify obvious flaws in our design, where we are assigning a responsibility to some part of the system that can't possibly execute that responsibility. Either because it doesn't have the power to do it, doesn't have the control to do it or because it doesn't have the information that it would need in order to be able to make the right decisions or actions. That step of assigning responsibility, before we have even gotten into any kind of detailed analysis, has reliably produced really good insights for us.

The last two things I will mention briefly are it helps you think holistically about mitigations because we mentioned that if you are taking the traditional statistical approach, I am going to test the model. That is all I am focused on. I want to try to make the model more reliable. But the challenge with some of these large language models is I can't make it 100 percent reliable. It is by design. I am not able to fully control its behavior. And so, I am stuck. If I take the traditional approach. When we use STPA, we can think about the whole system, and we can think holistically about how I control the risk. Assume the model might not be reliable 100 percent of the time. How can I design a system around that model to prevent the loss anyway? That helps you think holistically about mitigations.

Then the last thing is that it helps you solve problems when it comes to defining testing for systems this complex. When you think about functional testing, that is a little bit more straightforward because I specify what it should do, and I write tests that confirm that it does it. Got it. But with safety testing, it is a lot harder because I have this whole space of everything that it should not do, and then I have to somehow test that. Well, one of the primary outputs of STPA are a set of scenarios specifying the ways that you thought of using this process to identify where the thing might do something that it should not do. Those scenarios lead directly to either test cases or test datasets or situations that you need to test it to confirm that your system controls are effective in preventing those unsafe or those insecure conditions. Those are the main ways that we found that when we apply STPA to problems, hypothetical systems or prototype systems that have AI components, that it reliably delivers really useful insights and concretely can produce a better design.

Matt: To give an example of some of the practical artifacts that STPA produces in the context of the vignette that is motivating this discussion, an organization is using a research LLM to answer questions and to generate code. When we go ahead and perform an STPA on that, it gives us a set of loss scenarios. An example of a loss scenario would be a malicious user crafts an adversarial prompt leading to privilege escalation and access to unauthorized files outside of the sandbox environment. In this loss scenario, it is actually an attack scenario, so somebody with malicious intent is trying to get the system to behave in an unsafe way. From that scenario we immediately have test cases that we can include for security testing, and likewise, we could take the complete set of scenarios that we generate and come up with other scenarios to include in test and evaluation. We have a practical evaluation strategy that is rooted in the exact losses that we are trying to prevent. In addition, we can take each of those scenarios and look for design decisions that we can make to mitigate those scenarios for example, a malicious user crafting an adversarial prompt. We can seek to mitigate this by eliminating the hazard altogether through design selection. We could do this by hardening the sandbox environment in which the LLM can generate code. We can seek to do this by incorporating engineered features or devices. So, we can have a guardrail on the inputs to the LLM that seeks to detect potential adversarial prompts. We can include warning devices. For example, we can have a warning device that notifies security if the LLM tries to perform a privilege escalation operation.

Lastly, we can incorporate signage procedures or training. We can put warning signs on the LLM interface advising users not to input certain unauthorized behaviors. Just to kind of summarize here the space of options available to us, we can mitigate risk by alternating the system design. We can mitigate risk by providing signage. We can deliver training and education to users of the system, and we can also put organizational and administrative policies in place to mitigate those loss scenarios. By performing STPA and identifying those loss scenarios, we can then think about a multi-faceted way to mitigate those sources of risk.

Tom: Thanks, Matt. So far you and Dave both have done a great job of highlighting the gap in AI assurance that exists and how STPA can fill that void. It sounds exciting. But then that leads me to the question of how hard is it to actually do it? I want to use STPA. How challenging is it to actually do it?

Matthew: The answer is it depends. When performing STPA, the first thing that you have to do is to define the scope of the system, so the boundaries of the system that are included in the analysis. If you have limited resources, it is possible to focus on just a more narrow set of the controllers that make up the system. A second decision that you need to make is the degree of abstraction to employ. When you are conceiving of the system, you don't need to go all the way down to the computational hardware, necessarily. And you can employ higher degrees of abstraction to streamline the analysis. We find with both of these things, adopting a more narrow scope and employing abstraction, it is possible to use STPA to get meaningful insights with just several hours' worth of work.

David: One thing to add to that is even if a full formal STPA is something that might be beyond the reach of a particular organization, they don't have the personnel time to do it, or they don't have the folks that have been trained or have the backgrounds to do it. The core, the heart of STPA, is really the application of system theory and what are called control loops to your problem. It is to think about the whole system, or the behaviors of even a part of a system, in terms of a controller that is trying to execute control over a controlled process. There is a whole—we will talk about resources in a moment—there is a whole description of what a controller is and how that works, but the insights that you get from thinking about your system as a set of nested controllers, even if you don't want to do the full process, or you don't have training on the full process, is incredibly useful. That is where you identify those insights that, *Oh, the reason the LLM did this horrible thing, the reason it sent my data to an attacker, is because it thought that is what it was supposed to do*. *Why did it think that it was supposed to do that? Well, how did I design it, and how is the system set up? Well, it is designed to follow instructions. So of course, it thought that is what it was supposed to do. It was given an instruction to do it*. When you think about the LLM, even something as complicated as a controller, then you begin to uncover some of these insights. Even if a full STPA is too much, if that is a bridge too far, the application of system thinking, which is something that you can get with just a little bit of research and training, would really benefit anybody who is designing a system like this.

Matthew: Yes, so to keep building on Dave's response. This is the case. We have done this in as little as an afternoon with an organization. By employing abstraction and limiting the scope of the analysis, we still arrive at actionable, design recommendations within just a few hours of work. If you want to perform the full STPA, that is a heavier lift. This requires participation from engineers who are familiar with the system, managers who make decisions about how to deploy the system, and additionally, STPA facilitators and people who are trained in performing STPA. And so that complete analysis can take from days to weeks to perform. If your organization currently doesn't have expertise in STPA, there are many resources that are available. The MIT Partnership for Systems Approaches to Safety and Security has the STPA handbook, along with video tutorials and other educational material to learn about how to perform STPA. In addition to that, there are software tools available to support performing an STPA analysis. Finally, within our team here in the SEI, we have many people trained in STPA. We are available and routinely help organizations to get started with STPA.

Tom: Great, so you are mentioning some tools and resources, for STPA. As a federally funded research and development center here at the SEI. one of our core tenets of our mission is to transition our work and expertise to the public. If someone is interested in learning more about STPA, where should they go? Who should they reach out to?

David: I think the best place is definitely the home of STPA, if you want to think about it like that, MIT Lincoln Labs. Nancy Leveson and John Thomas are the folks that developed the technique, and they have a whole host of resources on MIT's website. They have video tutorials. They have written tutorials. They have published work applying STPA. They run a conference twice a year for anybody who is interested to get together and learn more about the technique and connect with other people who are trying to practice it and apply it. The technique has been around for a little over 10 years, which I think in system safety world makes it brand new. It is still growing in terms of its application, so I think that is probably the best place to connect. I think if people are more interested in the exact example that we were thinking about, with a security-related example of a system, then NIST publications, such as SP800-160, which focuses more generally on system security engineering and the approach to how do you secure a system by thinking through it via systems theory. Those and other related resources at NIST are also a great place to go.

Tom: OK. Great. We will include links to all those resources in the transcript of the podcast for our listeners today. Given everything you have laid out here today, and that you are involved with, what is the future direction going to look like? Where are we going in AI safety in the future?

David: I think first, we referenced earlier about how a lot of AI testing is still very model-centric. A lot of the headlines and the way people are thinking about models is, *Who has got the cutting-edge, state-of-the-art benchmark performance? Who can solve these problems with the highest level of accuracy?* So, it is still very model-focused. I think a big part of what we think needs to happen is we need to get more system safety and system security thinking over into the AI world. I think the biggest gap that we found in doing that are just more applied examples of what *right* looks like in this space. What we are trying to do is create examples that are relevant to national security where we think through the problem, and then we create the artifacts of STPA that are good, sort-of gold standard examples that other people can follow, so that they can learn the process, and they can begin to apply it to their problems. It is kind of a mass-based approach. We are trying to create the mass of examples that will be useful to try to get the word to the AI community and the people who are building these kinds of systems that the system thinking is a way that we can do a better job with safety and security.

Matthew: To add to that. STPA is a prospective safety analysis technique. So, it is a way to think about a system while it is being designed and to anticipate the ways in which it can fail in order to identify design alternatives to mitigate that risk. There is an alternate approach that uses system thinking called causal analysis using system theory or CAST. This is a retrospective analysis technique. This is done after the incident has occurred. As we see AI systems increasingly moving from research to deployment, there are incident databases that are increasingly documenting real-world examples of AI systems causing accidents. I think another important part in terms of the future of AI safety research is to continue to document those accidents and apply system thinking in order to learn from those accidents that have occurred in order to design safer systems in the future. One thing I will say, and this is kind of encouraging, and it makes me optimistic. When we look across those incidents using system theory, we are able to identify, substantive oversights that led to those accidents occurring. Many of them seem to be generalizable across different AI systems. And so, the lessons we learn from accidents can be applied successfully to future systems.

Tom: Yes. I am really interested to see where this goes in the future, particularly Dave and Matt your work and the work here at the SEI, but also just in the field. There is still a lot of future need in this space. I am really excited and interested to track this as we go along.

I would like to thank you both for talking with us today. For our audience, as I mentioned previously, we will include links in the transcript to all the resources that were mentioned during the podcast to help you on your own STPA journeys hopefully. Finally, a reminder to our audience, all our podcasts are available everywhere you find podcasts, including the SEI's YouTube channel. If you like what you see and hear today, give us a thumbs up. We really appreciate it. Thanks for joining us.