Improving Interoperability in Coordinated Vulnerability Disclosure with Vultron

featuring Allen Householder as Interviewed by Suzanne Miller

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.

Suzanne Miller: Welcome to the SEI Podcast Series. My name is Suzanne Miller, and I am a principal investigator in the SEI Software Solutions Division. Today, I am joined by Allen Householder, a senior vulnerability and incident researcher in the SEI CERT division. We are here to talk about a protocol that he and his team have created for vulnerability disclosure and response called Vultron. Welcome, Allen.

Allen Householder: Thanks. Glad to be here.

Suzanne: Glad to get a chance to talk to you again. I know you have done other podcasts with us, but for those members of our audience who aren't

familiar with your work, let's start by having you tell us a little bit about yourself. What brought you to the SEI and the work that you are doing now?

Allen: Sure. I have been with the SEI for a little over 20 years now, minus a couple of years in between where I went and did something else. Basically, I started in 2001. Prior to that, I had been a network engineer and had gotten into firewalls and network security things back in the 90s, came to CERT as an incident analyst, which was almost coincidental with all of the worm outbreaks in 2001 and thereabouts. So, Code Red and Nimda and Slammer and Slapper and a few others whose names I have forgotten at this point were all things that I was involved in analyzing from a network data perspective. Those were some of the events that highlighted that the internet was getting bigger than CERT could really be the incident response team for at that point. We had a hotline.

You could call us if there was an incident, and we might be able to help you, sort of like the A-Team. That was getting too big, so we started to move on to other things. I did some software engineering stuff for a while, then got into malware analysis. Then I left the SEI for a few years, went and did web infrastructure stuff. I came back and was getting engaged in the vulnerability analysis side of the house where I worked on some of our fuzzing tools and vulnerability discovery, which led directly to, we were finding so many vulnerabilities that we couldn't really coordinate them in a practical way, which highlighted the idea that there was a coordination problem that we knew there were more vulnerabilities than could be coordinated. We saw very early on about 10 years ago, *We are going to need to fix this bottleneck*, which started into a few research projects where we were modeling the ecosystem, and how do vulnerabilities move from discovery through getting fixes through assigning CVE IDs and prioritization and all those problems? That has branched off into a couple different research avenues within the SEI. There is vulnerability prioritization. There is the actual coordination platform, and we can talk about all of these things as we go on. Then there is the broader goal of having interoperability between folks who were trying to fix vulnerabilities, and that is where Vultron falls into. That brings you up to approximately the current day.

Suzanne: OK, so you have given us a little bit of a history. Let's talk about the cyber landscape that leads us to be worried about vulnerability coordination and disclosure. Obviously, vulnerabilities are on the rise, consequences, ransomware, et cetera. How do you describe the current landscape for

organizations today?

Allen: It is an undeniable fact that the absolute count of CVE IDs that have been assigned every year is continuing to go up. They ran into a bottleneck themselves a few years ago where they couldn't count past 10,000, and they had to increase the number space so that CVE could handle more than 10,000 vulnerabilities a year. I think there is somewhere around 20,000 right now, possibly more, and the year is not over yet. There is an increasing volume of vulnerabilities, but there is also obviously a proliferation of more and more software that you need to track and do vulnerability management on. As things have migrated into…There are different platforms. When we started CERT in 1988, there were computers. They were either desktops or servers. You had two kinds of computers, but they kind of ran the same operating systems, and it was Linux and Windows and various UNIXs. I guess Linux came along a little bit later. But it was all still the same sort of software and things. Then, as [the] Internet of Things [IoT] came along, you have now smaller devices that maybe don't get patches as often. You have more cars and airplanes and other things that now have a network port or a Wi-Fi thing attached to them. Those become things that also have vulnerabilities attached to them. We keep telling people—and this has been CERT's message for years—if it has software, it has vulnerabilities.

Suzanne: Right.

Allen: You have to plan for how you are going to deal with that when it happens, which means that if you deploy something that depends on software that can't be patched, you have a problem that just hasn't occurred yet. There is always need for vulnerability management, but it is also spread out across so many different platforms and so many different ways of delivering software now. Very rarely does it come on a CD anymore, but the model for how you update a cloud service is very different than how the software is distributed to update critical infrastructure, the thing that controls the valves on the dam or the thing that controls your power grid. Those are very, very different models for how you can deploy things, but they all have vulnerabilities. So we have to build a system that can handle migrating both the knowledge and the fix for those vulnerabilities out into those systems, regardless of where they are.

Suzanne: When the landscape really changed for me in some ways was when I saw a movie that malware/software was going to be put everywhere

and in the vehicles and all this other kind of stuff. One of the differentiating aspects of the hero was they had an old car that had no software. That said to me, *Oh, my gosh. Even Hollywood is recognizing that this is something that really is going to make a difference in how we manage these things*. This is one of the reasons that this kind of a protocol that helps to get through those bottlenecks is I think really important and very timely in terms of getting it out to the world, but you have got to deal with these multiparty issues, because you are talking about a car. You are not just talking about the car manufacturer. You are talking about all of the repair centers that deal with that particular car. And you are dealing with all the parts manufacturers that contribute to that. The whole supply chain aspect of this becomes a network of its own very quickly. That brings us into the Vultron idea that you have that we need more direct interaction between involved stakeholders, because everybody calling CERT, as you have said, the hotline is overwhelmed. Even if you have lots of CERTs, they are overwhelmed. We need the interaction between the involved stakeholders, the people who discover the vulnerabilities. Sometimes, not always, the general public. So this shift, how are people reacting to this need for …*I work in a beauty salon, and I have got to have some vulnerability coordination, because I am using software to manage how I mix my colors for my customers and things*. I mean, it is getting to that point. How do people react to this idea that they have got to be part of a vulnerability coordination ecosystem?

Allen: I think for a lot of folks, especially for end users, where you don't necessarily have a sysadmin staff to call on to apply patches and do all the monitoring stuff. On the plus side, many software vendors have gotten more into automating the update process. You may notice on your phone that occasionally you will get a prompt that says, *We are planning to install new software tonight*. Or if it is the apps, very often the apps just update in the background, and you don't even notice that you got a new version of it this morning, and maybe you got a new version two days ago. That part of the end consumer product has gotten better in those niches where—for mobile phones or smartphones and for desktops and laptops and things—a lot of that consumer-grade software does get updated fairly automatically. Many of the IoT devices are not there yet. Like my thermostat downstairs, it is possible to update the software, but I think it involves a USB stick, copying something to a USB stick and plugging it in and doing the right thing. There is a wide disparity of…And some of it is, the companies that produce phones and computers, they know that they are in the software business, and they have known this for a long time. They have watched this thing evolve. Car

manufacturers only figured out that they were software vendors in the last decade or so. Somehow network appliances that you buy, some of those vendors haven't figured that out yet. They still have this idea that they sell a fixed product, and once that product goes off the shelf, then there is not really a good update process.

Suzanne: Security cameras have become the Christmas idea of the year. That is one that I worry about. Vulnerabilities there can lead to all kinds of interesting things that I don't want to think about.

Allen: Yes, and so this actually becomes one of the sort of hallmarks of, are the vendors doing a good job. It's not that they don't have vulnerabilities. The goal is not to find vendors that don't have any vulnerabilities. The goal is to find vendors that have good processes for dealing with it when it happens. My home router is one that automatically does get updates from the vendor, the manufacturer, and I trust them to actually do that and roll it out. I don't necessarily have to think about it too much, but I know that it's getting updates. Versus some off the shelf things you can just go buy, and they are what they are when they came out of the box, and they'll never change.

Suzanne: So the software vendor community is changing the way they think about things, and the Internet of Things is starting to think about that too. But let's talk about sort of how do people think about vulnerability disclosure? Because it's one thing to have a vulnerability and to fix it and get a fix out there. But then it is another thing to let other people know that there is a vulnerability that they may not be on the same networks that I am doing automatic updates on. They still need to know that there is a vulnerability there. How is that part of the equation changing over time?

Allen: If you go back to the mid-20-teens, most people would hear about, *There is a vulnerability in this product, and you should go apply a patch*. That is sort of where the status quo was. Then, around 2018 or so, there was the Meltdown and Spectre vulnerabilities, which were actually problems with the chips, the CPUs, and how they interact with their processing memory and things. And that became a problem because the fix was maybe software, but sometimes it was to replace the hardware, and it was, every computer on the planet was affected that had a certain set of chips in it. Because of that, it affected not just Intel, it affected Apple and Google, and Microsoft. And every operating system had to make some updates because sometimes you could work around it in the operating system, but you had to modify the operating

system, even though the problem was actually in the chip. But it also affected AMD, and it affected all of the CPU manufacturers. That brought the idea of multiparty coordinated disclosure to the forefront, because it was such a big deal that every computer was affected. And this group of vendors had gotten together. They did a reasonable job of coordinating amongst themselves to produce the fixes. But, inevitably, there were some vendors who got the information later than others, and, you know, there were some hard feelings and things like that. In the summer of 2018, it wound up that there were Senate hearings about vulnerability disclosure, which was weird. For someone who like, when I started this job, it was hard for me to explain to my parents like what I do. I work with computers. And all of a sudden like, *Oh, there are Senate hearings about the thing, and I got to write some of the testimony for it, because one of my colleagues actually gave a testimony at the hearing*. All of a sudden vulnerability disclosure was at the top of mind for people who were thinking about cybersecurity at national scales.

Coincidentally, a few months before that, we had published the CERT Guide to Coordinated Vulnerability Disclosure. We published it in 2017. We updated it in 2019, and that has sort of served as the narrative of here's how you do this process, but it's really, it's really more of a field guide, it's not, it's not a strict step-by-step process. *Here are the things you are going to encounter. Here are some of the problems you might run into along the way. Here is what you can do to address those problems*. It turned out that was a fairly long document. It's turned into, when the vendors were called to the Senate to go testify, they cited that document as, *This is the process we are doing*. Which is when I realized, *Oh, I just wrote this down*, or, I was the lead author on it, but we wrote this stuff down. It turned into like now all of these vendors are just citing this as the way you do it. I didn't think I was writing a standard. We thought we were just writing a how to guide. Then a year or two later, it starts to show up in like the Center for European Policy Studies. The European Parliament has cited this stuff as part of how they are addressing vulnerability disclosure. There are ISO standards, and there is a whole bunch of very high-level global things that have grown up around vulnerability disclosure. While that message was getting out there, and it was obvious that we had someplace to have some influence on how people around the world thought about this, we also realized that the way we had been doing this process, which was essentially email driven, you would send us an encrypted email. We had a decent contact management system where we could blast an email out to a bunch of vendors and send encrypted mail and things back and forth. But essentially it was, it was really email-centric and not much

better than just an email thread where you are cc'ing, or you are bcc'ing everybody. Because we had this idea of, if it's a vulnerability, CERT should be talking to vendor A, and vendor B, and vendor C in distinct threads, and those are separate conversations. So, they don't even know about each other. We realized, with Meltdown and Spectre and other vulnerabilities, as things got deeper, we couldn't afford to have that, us be the store-and-forward hub for all of that. We needed to go to a shared communication model. Which is where we started to develop VINCE, which is the Vulnerability Information and Coordination Environment platform, which we have open-sourced. It is available on GitHub, but that has become our environment for where we do coordination now. That changed it from being email threads where it us talking to a whole bunch of folks in different email threads to more of a shared forum per vulnerability. Vendors can talk to each other. They can talk to the researcher. They can talk to us. We are all in the same space where we can have a conversation about this vulnerability and coordinate all of the efforts. There are still some issues with that.

Suzanne: What you have is more of a knowledge network instead of a hub-and-spoke kind of communication.

Allen: Right. We actually described it in network terms. Instead of hub-and-spoke, we needed a shared bus. We went from hub-and-spoke to this shared environment, but there is still a bit of a problem with that in that in order for you to coordinate with us via VINCE, you have to have an account on VINCE. You have to be able to log into VINCE. VINCE has to be available. There is one VINCE instance currently that runs, and that is it. And if it is down, it is down. Like all systems, we have had some outages. We have had performance issues at times. We have had things where, you know, multifactor authentication breaks for a while, and then this comes back up, typical system administration stuff. But the fact is we can't have the whole planet depending on a particular system to make sure that things are getting coordinated and getting fixed. So the problems of having a centralized system, having more accounts for vendors to manage, and for researchers to manage as well, we realized that the vendors already have a problem-tracking systems for tracking bugs. They already have ways of internal processes that they need to integrate with; a report comes into a vendor, and they have to get that out to their development team. They also have to deal with whatever. Maybe they have legal issues, or maybe they have marketing or messaging communication issues that they are going to have to get out.

There is a lot of coordination that goes on inside a vendor when they have a vulnerability that they are working on. It has to get worked into the deployment plan and the development plan and the roadmap or whatever and all of those things. But they already have systems to do those things.

Now we are asking them to also log into this other system, VINCE, to coordinate with the other vendors on those things. What we really want is interoperability between the processes. I don't care if they log into my system to do any of this, or your system to do any of this. I want the process to work regardless of which platform is hosting any particular case. That is really where the idea for a Vultron, or for the protocol, comes in. We need to get to an interoperability of the process before we can talk about even syntax or any of the details of what are the message types, and should this field be four bytes or eight bytes or whatever. I don't care about data formats and all that stuff yet. We need to get there, but for right now where we are at with Vultron is this idea of, *Can we talk about standardizing the process so that we all have the same words to talk about where we are in this process*? If we can get that to work, then the rest of it we can figure out. The rest of it is implementation. We need to get there and do all those details. But my argument when we started this was, *I want a process that can work just as well with note cards and envelopes and paper and stamps and a Rolodex, where we can still talk about it the same way. Or, we can implement it as a complete technical protocol where there are robots doing the whole thing, and humans very rarely get involved in*. That is an optimization for the process. The process has to make sense, and we have to be able to talk about the process and understand what it means.

Suzanne: Vultron, depending on what kind of movies you go to, and comic books you read, Vultron is a familiar concept. But if you don't go to those kind of movies or read comic books, the anime comic books, what is the meaning of Vultron that makes it so appealing as a name for this protocol?

Allen: I am not sure if you can make out the Star Wars figures and the Mach Five on the shelf behind me here.

Suzanne: I can.

Allen: I am a child of the 70s and 80s. When it comes to giant robots and the things that they have to go off and fight. So Voltron with an O is an anime series. It was based on the idea of having multiple robots. In one case, they

were lions. In another case, they were cars. It depends on which series you are talking about. But they were independent robots that each had a pilot, and those robots would come together to form another giant robot that could then take on bigger, bigger monsters. It is very much in the genre of Godzilla movies and Ultraman and a lot of those ideas, Robotech and all of these things. But Voltron, the robot, is the real concept that I wanted to map from that to this protocol, the idea of autonomous independent decision makers coming together in a coordinated way, using technology, and coordinating their efforts to defend something. That is really the core of what Vultron, the protocol, is intended to embody. We changed the O to a U because this is a really niche thing. In most cybersecurity places, they talk about, they will shorten the word vulnerability to *vuln*, with an n. CERT has always, it's a cultural thing for us, we shorten it to *vul* without the n. Our argument has basically been, well, if you draw a line between where the syllables are, the n is on the wrong side of the line, so it should be *vul*. I personally wanted to instantiate *vul* is a way of talking about vulnerabilities instead of *vuln*, because I just think the n is ugly at the end of that. That, coupled with the idea of Voltron, this coming together and using both human decision makers and technology sort of independently operating in order to coordinate to do something, had this nice…

Suzanne: And something important, right? That is the other piece of Vultron is, *We wouldn't put all these robots together if it wasn't something that was more than any one of us could deal with*. That is the other piece of this that I think the willingness of people to engage in a process that allows them to defend against something that is nasty or bigger than what they can deal with themselves is a big part of the appeal of this, and the timeliness. We talk about Vultron as a protocol. That involves us thinking about it from a process viewpoint as we have talked about. Can you walk us through an example of a vulnerability discovery? How would Vultron guide that coordination and response differently than what we would have done before? I would like you especially to say something about the embargo step and why it's important, because I think that is the real differentiator between Vultron and other mechanisms for coordinating vulnerabilities.

Allen: The process of getting a vulnerability fixed usually starts with someone has to find it and discover it first. But in order for that to get fixed, usually the individual that finds the vulnerability is often not the person who can actually change the code to fix it. That means that the finder has to report it to someone, in which case they become a reporter. They have to tell

someone. Eventually the goal is that that information needs to get to people who can actually change the code and create the fix. That is really where vulnerability coordination starts at: somebody knows something and they need to tell somebody else to make the fix happen. The Vultron protocol, when we talk about protocols, I mean both the technical idea of a protocol, which is the formats and the structure and the state machines and all of the pieces that have to fit together for how processes and computers interoperate and interact. The other meaning of protocol is sort of from the medical field or diplomacy of, *These are the set of rules and behaviors and expectations that we have on how people are going to do things*. Because we are talking about process integration or process interoperability, the thing that we started with here was really talking about the process by which these things are handled. There are three different state machine models that are part of the Vultron protocol. There is a report management process. There is an embargo management process, and there is the case state. The case state is a little bit complicated, so I am going to defer that for a moment. Report management is very simply a trouble ticketing system. There is IT service management processes. You have a report, you do something with it, you prioritize it, you validate that it is actually a report, and you go do the work. Eventually you check to make sure that the work was done correctly, close it out, or whatever.

Embargo management is the process of deciding how long you need to keep a secret when you are dealing with a security problem. There are lots of schools of thought in terms of, *Should vulnerabilities be disclosed immediately? Should they be never disclosed?* and anywhere in between. So zero to infinity, or pick a number in between, and that is how long you should keep the secret before you tell anybody about it. The simple fact is that for most vulnerabilities, it is not possible to just silently deploy the fix without anyone knowing about it. You have to tell somebody some time that the fix is available. At the very least, you have to tell the vendor that they have to do something. Depending on which software distribution model the product has, that might be, *We need you all to click your update button so that*...When Chrome has that little pop-up that says restart Chrome to apply updates. You at least need to make it public that there is something to be done.

Sometimes, in a cloud-service environment, the vendor might just be able to fix it and deploy it and then they can tell people later if they want to that, *Hey, we fixed this problem, and you are no longer affected because we already fixed it*.

Embargoes come into play when we need to coordinate among different parties who may have different distribution models. They may have different needs and different policies for that matter of how long they are willing to keep secrets. There are vendors who basically say, *I am looking out for my customers, and that means that when I receive a report, I will fix it as quickly as I can. As soon as I am ready for that to go out in my product, I will put it out there, and I'm not waiting for anyone else*. There are other vendors who see value in cooperation and coordination where maybe they are willing to wait a little while, while another vendor readies their fix as well so that it can all go out in synchronization. But they are not willing to wait indefinitely. There is a good bit of negotiation here that goes on. The underlying insight that we have in the Vultron protocol is that it is really a calendaring problem. We are not trying to coordinate who is going to publish what exactly when. We are not trying to schedule the publication. We are trying to schedule the end after which we don't expect you to keep the secret anymore. If that makes sense, the idea of an embargo is, from the time that you propose it to the time that it ends, the expectation is, and this is the behavioral part of it, that no one is going to publicize this information. After that embargo ends, anyone can publish, and if you choose to coordinate and make an announcement at noon on Thursday in a certain time zone, you can do that. But the Vultron protocol right now doesn't actually address that. It just says that, *As of 8:00 a.m. Thursday the embargo comes off, and then you can publish*. Which, if you think about how news embargoes work, it is very similar. You put an embargo on a thing that says you can publish any time after this point. It is not necessarily saying that the New York Times and the Washington Post have to have their articles up at the exact same time. They are not trying to synchronize at that level. They are just saying, *Please don't publish this before 1:00 a.m. on Tuesday*.

Suzanne: I think that news analogy resonates with people, because there are some things that, if it gets published, there is risk to people being hurt sometimes or some negative consequence of it going out too quickly, which is what the embargo is really meant to prevent. You can't wait indefinitely, or else now you have other harm that happens.

Allen: Embargoes are not without controversy even in scientific publishing. There are some who argue that it is providing an opportunity for some to have advanced information, but not others. There is a bit of fairness argument going on there of who should have access to that information. Again it comes down to, sometimes the most fair thing to do is just put the

information out there and let it fall where it may. Sometimes, if you think you can have a good handle on it, you might want to keep it secret a little bit longer. The goal of embargoes, in the context of vulnerabilities disclosure, is almost always to protect users because knowing about a vulnerability when there is nothing you can do about it doesn't help you.

Suzanne: True.

Allen: Or it doesn't help you much. It gives you some advantage in that you might be able to build some workarounds. You might be able to take some systems offline if you need to. But it doesn't actually give you a positive action to fix the problem. But it does inform. If it is public, we assume that the attackers know everything that the public knows. If the attackers know that there is a vulnerability, sometimes just knowing that a vulnerability exists is sufficient for them to go find it and know what to do and they can have an exploit. You generally want the race to be won by the blue team rather than the red team. You want the fix to be available and the fix to be deployed and all that. One of the things that we built into the Vultron protocol is this idea of there are certain events that happen in every case. We want those events to happen in a certain order. The more of those orderings we can get right, the better we are doing. We actually have a way of measuring this process, which we have never had before. If the fix is ready before the public knows about it, that is a good thing. If the fix is deployed before the exploit is public, that is a good thing. If the fix is deployed before attacks are observed, that is the best thing. The whole goal of this is that if you beat the attacker to… If the fix is deployed, they can attack you all they want, and it doesn't affect you. There are other things like, does the vendor know about it before the public does? Because usually they have to take an action before the public can take any action. So that is necessary. Is the public aware before the exploits are available? Is the public aware before attacks are happening? There are these 12 criteria written into the protocol that basically say, *These are the 12 things we want to happen, and we want more of, it is better that more of them happen than fewer*. That gives you a yardstick where you can measure your process and say, *Are we doing a good job?* Well, if you are getting more of those things happening most of the time, you are doing well. If you can get more of those things to happen over time, you are doing better. We give you a mechanism to sort of measure that.

Suzanne: Then you have got the last state is the case state is the last part.
Allen: Right. So the case state is really where those measurable things come

in, because the case state talks about the six events on which all of those events are ordered off of. Those six things are, *Is the vendor aware of the vulnerability? Is a fix ready for the vulnerability? Is the fix deployed?* That is the vendor fix part of that model. The other part is, *Is the vulnerability public? Is there an exploit public? And have attacks been observed?*

Ideally, you want the first part of that, you want the vendor pieces of that to be done before the public pieces of that happen. You don't always get it that way. Sometimes it is necessary for the vulnerability to become public before you can deploy it, because that is just the way that you distribute the software. The ideal case is for the cloud provider, or for those phone apps as I mentioned before, if you can update the software before the user even knew that the problem was there, well, you win. You win the whole thing. That is the best possible outcome. Especially if that happens before the exploits are out there and before there are attacks happening. You don't always get that. Also, you don't get to control all of those. Even if you are a coordinator or you are the vendor or you are the researcher, you don't get to decide when someone else finds out about this vulnerability and decides to attack it. The attackers can do their own thing, and they are acting independently. Some of this is about a race that we are always in because it is an adversarial situation.

Suzanne: OK, so I get the protocol. I am thinking about being in an organization that a vendor organization in particular that has to deal with the vulnerabilities on a regular basis. What are some of the potential roadblocks to an organization like that in implementing Vultron? Have you thought about solutions to help them overcome those barriers?

Allen: Probably the biggest barrier right now is that Vultron is a paper, it is a PDF file. You can go read it, and you can understand it and hopefully understand the process. But it is not an implementable technology yet. A lot needs to happen to make it implementable. I mentioned earlier that eventually we do need to get to data formats, and we have to figure out how that process is literally going to turn into code. The main thing that a vendor, or anyone who is interested in coordinating vulnerability disclosures, could do is by understanding the process that we lay out in a paper, and understanding the arguments we are making for—one example I forgot to mention when we were talking about embargoes is the idea that the shortest embargo proposed wins always. That is the idea that let's say that you and I are negotiating a vulnerability disclosure, and I want to keep it secret for 30

days, and you want to keep it secret for 10. Well, one way to deal with that is I say, *No, I want 30*, and I just reject your offer of 10 days, and do that. There is lots of game theory you can build around this to really figure out whether or not that is the best decision to make. But our proposal within Vultron is if you say 10 and I say 30, we agree on 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10. We should agree to what we agree to, which is we take 10 days. So if you propose 10, and I say, *OK, fine, we take 10. Then I can immediately counter-propose, but I think it should really be 30*. Then I can take the next 10 days to negotiate with you about why we should do it longer. Maybe we don't get to 30, but at least we agree on 10.

Again, this comes down to the idea that embargoes are not a schedule for publication. It is a, *How long do you want to keep this thing secret?* If we all agree it should be secret for at least 10 days, let's just agree to that.

Suzanne: Let's start there and see if we need to go farther.

Allen: Right. Start there. That gives us 10 days of secret keeping that we all agreed to, during which we can then figure out should it be actually 30, or should it be 90, or whatever.

We specifically don't recommend any specific disclosure timeline within the protocol because even today our policy is different from Google's policy is different from Microsoft's policy is different from…There are lots of different policies out there that say, *It should be this many days*. They don't always agree. When you are talking about this multiparty problem of coordinating all those, there will be logical incompatibilities there, and so we can't rely on any particular answer to be the right answer by default. So we just say take the shortest one, and then adapt. And the protocol has some mechanisms in it for how to, how to kind of walk through that process. It's just a simple logic thing of, If we all agree on 10 days, then start off with 10 days as the assumption, and then work to extend it. It doesn't seem to make sense to have it be a lot of arguing back and forth. *Oh you want 20 and I want 10, but we're just going to go with zero because we couldn't agree*, when we actually agreed on 10.

Suzanne: What I am hearing you say is that we are really looking, at this point, Vultron is ready for innovator and early adapter types of adopters, people that don't typically rely on turn-key solutions, *Here is a whole package of things and all the data formats and everything, and just turn-key installation*.

The people that you are really looking to consider Vultron at this point are ones that are willing to overcome some of those things, figure out what some of the mechanisms might be for optimizing the communications among the parties that they work with at least, and then sharing some of that information back with us. Is that pretty accurate in terms of sort of who you think would be good Vultron adopters at this point?

Allen: Yes. I think if you are involved in coordinating vulnerability disclosures over and over again, so maybe you are a large vendor, maybe you are a coordinating, a CSIRT [Computer Security Incident Response Teams] either a National CSIRT or another large CSIRT that happens to do vulnerability coordination, going through the document, there are a lot of math and equations and stuff in there because we wanted to be formal about it. But there are diagrams, there are lots of diagrams. If you can understand the process that those diagrams represent, and they are pretty straightforward I think. If you understand the process those diagrams represent and try to envision how your process that you already have—because I am going to assume that everyone already has a process for doing this stuff. What we are trying to do is map this Vultron schema on top of whatever your process is. The first question is, *Does your process map into the way we have modeled the world?* And if it doesn't in a significant way, that is feedback we really need, because chances are if we have missed something where your process is very different than what this is trying to model, you are not the only one that we have missed. We really need that level of feedback. The second piece is there is also a section in the paper that looks at some of the process logic: *If this condition is met, then do this*, and *If this condition is met, then do that*. Some of those things sort of describe an automation that you might build. Sometimes it is not necessarily a full automation. Some of those pieces can be automated and some pieces are human, so think cyborgs, not robots.

Again, it is the concept of Vultron. It is the humans combined with machines that are building this process, but we have tried to model out and sketch out at least as much of the process that we think points toward automation in the future. We are still in the early days of this. The paper was published over the summer. We are still in the process of getting the feedback mechanisms in place and figuring out what kind of community needs to be built around this. Part of our process right now is we are sort of in the awareness phase, and letting people know *This thing exists, it has a name, here's the concepts it embodies*. That is what this podcast is about is just introducing people to the concept. We haven't really gotten to the point where folks really understand

what it is or how it can fit into their process. But I think that is the next step; understand you have a process, how does this thing fit into your process, or does it, and if it doesn't, you know, we need to know about that. If it does, we would like to know that too, because that starts to build toward an interoperability problem that we are trying to address.

Suzanne: What kinds of users or vendors are you looking at embodying the group that you really want to work with? What does that community look like that you want to work with to extend these ideas? Have you got some sort of, *Well, I really want people from the critical infrastructure community, but we are really not ready for weapon systems yet*. Do you have any ideas about what some of the types of adopters would be that would be most useful for giving you that feedback?

Allen: Yes, so there are obviously big software vendors and cloud-service providers. Any of the big names that people recognize that do software-related things. They all have processes we are interested in knowing whether or not this fits their processes, because most of the vulnerabilities that are going to happen are going to happen in one of their products. It would be really cool if they were onboard with us. Beyond that, there are also the vulnerability disclosure platform providers may also be recognizable as bug bounty providers, but they are not strictly bug bounty. Vulnerability disclosure is the process of whether or not you pay somebody money as a reward for that is optional. They tend to be called bug bounty platforms, but I think it's more accurate to call them vulnerability disclosure platforms. Those platforms, if this protocol becomes the way that we do this, then we want those platforms to be able to think about how this process works for them. At least within the U.S. government, and I suspect other governments around the world, but I know about the U.S. Government side of things, there are vulnerability disclosure programs that were mandated by… I have lost track of whether it was an executive order or a binding operational directive, some way that the government has to tell all of the partners and agencies, *You will do this thing.* They are all mandated to have an ability to receive vulnerability reports in their systems, and to deal with those and address them. From the outside world, if somebody notices the problem, they should be able to report it to the government agency on the website or whatever. The most well-known of these is the DoD Vulnerability Disclosure Program run out of the Defense Cyber Crime Center. They have been at it for a few years. We think they would be a good candidate for this as well. In fact, we have been talking with them about the ideas embodied in Vultron to try to get this

transitioned out conceptually to…If this is how we understand the process… At a government level, if the government agrees that this is how we track the process, that gives some visibility back to the folks that need to watch across all of those agencies, like at CISA, who might need to be able to tell, *OK, which cases are having problems and which ones seem to be operating smoothly?* A lot of the information in Vultron would help sort of make that judgment, because we can tell, based on what state the case is in and what the history of that case has been, we can tell is this one making progress towards good resolution or bad resolution and that sort of thing.

Suzanne: OK, all right. You have got some transitioned planning to do to get this CERT out into the world in a broad way. To get to the broad, you have got to get to some specifics first. Organizations that are listening today that are thinking, *Wow, this would really help*, you need to get in touch with Allen. I get very excited when I see these early protocols processes, you know, that are about to sort of be introduced into different aspects of the work. I think there are some really interesting aspects of Vultron that I certainly…The embargo thing I have never really thought about formalizing before. I think that is going to be something that is actually going to start a whole different kind of conversation in terms of how vulnerabilities are coordinated and dealt with. I am really looking forward to seeing on our next podcast seeing some of the results that you are getting from people using this process.

I want to thank you so much for sharing all this with us today. You have given us people a lot of detail. I know that there is a lot more in the paper. In the transcript we will have all the resources that we have talked about in the conversation will be available to people. I know that anybody that needs to get ahold of you can do it through our info@sei.cmu.edu and that will get to you. Hopefully, you'll have a whole bunch of people contacting you to get going with Vultron. I am also looking forward to seeing what you come up. You have got to have a logo for this. If you haven't got one already, you need to be talking to somebody in communication services, because we need a cool logo for Vultron.

Allen: When I have presented it so far, what I've done is coincident, coincidentally to this paper coming out, there have been a number of AI image generation systems that were available. You can give it some words and describe the thing that you want, and it will give you an image. I've used a few of those in my presentations because I generated the words, and, therefore, because of the rules of the site, I had the copyright on those

images so I can use those as the slides in the presentation without technically violating any larger trademarks or anything. We want to be sensitive to that too.

Suzanne: Yes, we do have to be sensitive to that.

Allen: Yes.

Suzanne: I look forward to seeing that, and I look forward to another conversation down the road to talk more about how this is going along. I do want to remind our viewers that you can get this podcast all the places that you get all the rest of your podcasts, whether it's SoundCloud or Stitcher, or my favorite, the SEI YouTube channel. If you like what you see here today, feel free to give us a thumbs up. Allen, a joy to talk with you as always. And I want to thank everyone for joining us today and listening to this very important topic.

Suzanne: Thanks.

Thanks for joining us. This episode is available where you download podcasts, including SoundCloud, Stitcher, TuneIn Radio, Google Podcasts, and Apple Podcasts. It is also available on the SEI website at sei.cmu.edu/podcasts and the SEI's YouTube channel. This copyrighted work is made available through the Software Engineering Institute, a federally funded research and development center sponsored by the U.S. Department of Defense. For more information about the SEI and this work, please visit www.sei.cmu.edu. As always, if you have any questions, please don't hesitate to email us at info@sei.cmu.edu.