Is There Value in Identifying Software Security "Never Events?" Transcript

Part 1: Errors in Medical Care as an Analogy for Errors in Software

Julia Allen: Welcome to CERT's Podcast Series: Security for Business Leaders. The CERT program is part of the Software Engineering Institute, a federally-funded research and development center at Carnegie Mellon University in Pittsburgh, Pennsylvania. You can find out more about us at cert.org.

Show notes for today's conversation are available at the podcast website.

My name is Julia Allen. I'm a senior researcher at CERT, working on security governance and software assurance. Today I'm pleased to welcome Bob Charette, founder of ITABHI Corporation and an internationally recognized expert in risk management, information systems and technology as well as systems engineering. Today Bob and I will be discussing the merits of a new idea he's kicking around, developing a list of software security "never events." Just by way of brief introduction the healthcare community uses this term to describe life threatening, serious and entirely preventable medical errors but Bob will tell us more. So welcome Bob, glad to have you with us today.

Bob Charette: Well thank you Julia for having me on today.

Julia Allen: Okay so just to do a little stage setting can you say a bit about what a "never event" is from whatever fields of endeavor it comes from and a little bit about how it came to be?

Bob Charette: Sure. As you mentioned the term is one that's been used first in the mid 1990s by the National Quality Forum which is a non-profit membership organization that was created to develop and try to implement a national strategy for healthcare quality measurement and reporting. And they wanted a way to describe errors in medical care. So what they did is they developed a list which has grown to some 28 what they call medical "never events" and some of them, to give you an example of a medical event, includes, say, surgery on the wrong part of the body like removing a kidney when you were really trying to remove a gallbladder, leaving a foreign object like a sponge or scalpel in a patient after surgery or receiving, say, the wrong dosage of medicine while you're in the hospital care.

So for something to be considered a "never event" from a medical standpoint, it must be clearly identifiable and measureable because it has to be able to be reported. It's usually preventable and it has to be serious enough in the consequence for patients which means having/causing a permanent disability or causing death (which is fairly serious) or to indicate a real problem in safety or credibility of a healthcare facility. So it also has an element of process.

In other words the purpose of developing this list of "never events" was really to try to bring up some level of public accountability to medical errors that under most circumstances should not by any reasonable standard or by looking at it by a reasonable person really occur. As a result of this list which is interesting is that the US government and many insurance companies now have decided that they're not going to pay for certain of these medical "never events" — not all 28. There's five or seven that the government won't pay for anymore and insurance companies won't pay for either.

Julia Allen: Well, we're talking about medical, healthcare, life threatening situations and that seems kind of far afield from software or software security so what made you make the connection and why do you think this might be a useful concept particularly now?

Bob Charette: Well for a fairly number of years, and you're aware of this because we've had conversations about it, I've been trying advocate for something similar to medical "never events" and have some type of list in regard to software, either in terms of development or process, etc.

Unfortunately I haven't really been able to make much headway because it was pretty hard to get agreement from my colleagues in either academia or in industry as to what we could really call a software "never event". First of all part of the reason is one of language and terminology. In the medical field, "never events" are very, very extreme events. They're things that really don't happen at all in any type of major statistical quantity. Whereas in software we're constantly surrounded, it seems, by problems, so a lot of my friends said we probably should call "In hopes of a software never event" or maybe better "Software ever events". In fact my friend Martin Thomas, who cofounded the UK software company Praxis which is a high reliability software company and he's a visiting professor at Oxford University Computing Laboratory, wrote me when I was talking to him about this that he's a strong supporter of having some type of initiative where we could eliminate things that shouldn't occur. But as far as he could recall, the software industry has never learned from a mistake and created a "never event." So part of it has been some push back in terms of how do we actually describe these things.

But recently it's come to my attention that a coalition of more than 30 US and international cyber security organizations led by MITRE and the SANS Institute and CERT jointly released a consensus list of what they're calling the 25 most dangerous programming errors that lead to security bugs and that are used to allow cyber crime and cyber espionage and cyber warfare. And they're calling this the Top 25 Errors Initiative.

And what they say, and I'm going to quote from their press release, is that "the impact of these errors is far reaching. Just two of them led to more than 1.5 million web site security breaches during 2008 and those breaches cascaded onto the computers of people who visited those web sites, turning their computers into zombies." They also said that "most of these errors, most of the 25, are not well understood by programmers. Their avoidance is not widely taught by computer science programs. And their presence is frequently not tested by organizations in developing software for sale."

So when I looked at that list I felt that well maybe this would be a good time to revisit the idea of "never events" and to see whether or not, given the increased frequency and severity of IT security incidents, that maybe this would be a good time to start a new conversation. The idea seemed pretty obvious in retrospect, especially if there's a list of agreed upon errors and ways to prevent them.

Part 2: Implications and Potential Consequences

Julia Allen: So clearly software is a young science, certainly much younger than the medical profession — software security even more so. So what I hear you saying is you're really taking the concept of something that, as ethical, high integrity practitioners, if we're developing software or acquiring software, that there's just a minimum set of things that we should just never do. Is that where you feel the two concepts connect?

Bob Charette: Yes but I think it's more than that too. To be honest I'm a little bit wary now, or weary is probably a better term, of us always continuing to say software is a young science; maybe in comparison to medicine. But I've been in the field for I hate to say it, but 37 years since the time I was 18 year old punch card operator at a university during the summer. And back then I heard that IT was a young field and here it is 37 years later and I keep hearing that same argument.

I think that it's not only in high reliability areas. I think that a lot of the errors that were on the list are ones that have been around for a long time that we know really shouldn't occur in software anymore. And I think it's time to start imposing at least some level of discipline onto the field and to get over this crutch that's really an immature field and it's acceptable to do things that we know that really are not acceptable.

Julia Allen: Well no, I definitely hear what you're saying. We know enough. As you said, some of these have been around for a very long time. It's probably a collective will issue or maybe it takes some kind of a catalyst where you use concepts from other fields like the medical field to raise awareness and attention.

So let's say we had such a list which would be a wonderful step forward. So if it could be developed and it could be agreed upon — as you said there's many differing points of view — what do you think some of the implications might be, perhaps legal or warranty or other types of implications, if we had such a list that the community grabbed onto and began enforcing, even as a grass roots effort?

Bob Charette: Well I think that's a pretty interesting question. As I mentioned earlier, the US government, insurance companies have decided not to pay for certain (and I said before, not all never) medical "never events.” I think in many ways having a list like that is a pretty blunt instrument but it has gotten the medical community's attention to pay attention to things that really shouldn't happen.

One could argue, and I do as I just said, that the software that contains some of the top 25 errors really having that type of software is not acceptable anymore. And I think that if it's not being acceptable, if we agree that it's not acceptable, then why should anybody really pay for this whether or not they be government or private contractors.

At the same time I think there's some who worry that this could open the door to a flood of litigation in the IT community and lead to a stifling [of] innovation. There have been arguments for as long as I can remember as well about having mistakes is the price we have for progress in our field. And I think that there is some merit to that argument. But I think again as a community we need to ask ourselves what are our responsibilities when it comes to developing software with known preventable risks in them.

I think that a lot of the companies, even the members who are part of the top 25 error coalition, might blanche at being held — having their software held to a level where they might not get paid or they might get held for consequential damages because of those errors. But I think again, we're at this point I think in the field that we really need to look at errors differently. And I don't think that we need to address all 25 at once. We can do something similar to what happened in the medical community which started with a very small list, three or five, and has slowly started to build that up. I think the real important thing is to get our conversation going in this area.

Julia Allen: Well we're certainly seeing some progress in specific market sectors like the Payment Card Industry standards. They've added some for application security and the consequences of not acting or implementing those practices is that you don't get to hold cardholder data. So we're starting to see some limited application of the idea you're suggesting.

Bob Charette: Well I think that that's true. And I think as we again, as especially in the IT security area where the damages are getting greater all the time and they're becoming very visible. And we're getting to a point now where people are afraid to start getting onto the web. And when you get to a point where the risk outweighs the benefit then I think we need to start putting in standards so that we can start to gain control again over our technology.

Julia Allen: I think we've tolerated a certain level of, if you will, sloppiness and defects in the software that we use and breaches and viruses, what have you. Do you think we're at a point where that tolerance is dropping and so there could be some momentum for this?

Bob Charette: Oh I think so. I think again let's go back in history just in the last 30 years. We've moved from a term that I'm quite sure a lot of people who will be listening have absolutely no idea what I'm talking about when we talked about a “closed shop” — where in programming, where it sounds like a union shop, but in fact where the programmers were the primary users of the code that was developed. And as we've moved from what was that old term “closed shop” to “open shops” where we had users but they were again tended to be technically savvy users of software and again internal. To where we're at today where grandma and Aunt Millie uses the software on the web and are demanded to use the web or computers to fill out forms or to check their bank statements, where you can't get instructions on a piece of product that you buy because you can only get it off the web.

Then I think what's happened is we've moved to that level of social impact where grandma isn't going to put up with this anymore, either is Aunt Millie, neither should anyone. And so we as a community, which has been very technically focused and inward focused and again, we're very forgiving. My father-in-law and mother-in-law and sister- and brother-in-laws are not. If we want the technology to move forward and be used then we have to pay attention to people who are going to be using it.

Part 3: Getting the Ball Rolling

Julia Allen: Well when we last spoke, as we were getting ready to do today's podcast, you said you're planning to do some research in this area, perhaps write an article to get the idea out into public conversation. So what are your plans for this work?

Bob Charette: Well right now I'm calling up lots of my — I was almost going to say old friends, my long time friends like you Julia and asking their opinion about the idea. And I'm talking to everyone I can from university researchers to practitioners about the feasibility and the practicality of creating such a list and whether or not the list that was created in the IT security space by the top 25 error initiative, whether or not that's a good place to start. Could we use some of those errors on that list as de facto "never events" or things that we should consider to be "never events." And what I'm going to be doing is hopefully talking to as many people pro and con on the subject and then hopefully have a major magazine story out by autumn of this year. And again if somebody has a strong opinion one way or the other they should contact you to get a hold of me.

Julia Allen: Well that would be great because I just think that if we can get some momentum, a lot of things happen just out of planting an idea, letting it run, and see what happens with it. But before I let you go, as you know at CERT, we're doing a lot of work in the area of software assurance. We tend to focus heavily on software security as you and I have been discussing but assurance also involves software safety and software reliability. So do you think this concept could have traction in those disciplines as well?

Bob Charette: Well I think so. I think it can have traction in a lot of different areas of software development. There are some similarities in safety and security obviously. You don't want to have race conditions in safety systems and you don't want to have buffer overflows in those areas either so there is a lot of cross fertilization.

I think too is one of the things that the medical community has done is also to take a look at process in terms of are there ways in hospital quality of care that can be improved? For instance, in the hospital areas, they're using a lot of military checklist approach to try to make sure that surgeons don't operate on the wrong limb or take out the wrong organ or leave scalpels in patients after they sew them up.

And so I think that just like what's happened in the medical community I think that we can take a look even in areas of software development. For instance, even simple things like ensuring that there are really good change control boards so we actually know what's being changed in software from version to version and that we don't allow patches to be put in; those types of things I think we can even detail out.

There's a lot of knowledge that we've developed over the last 35, 40 years of software engineering and systems engineering that we ought to be able to codify. We ought to be able to make measurable, visible, repeatable and say "If you don't follow at least some of these very, very basic practices the software — if it turns out that there's an error and it can be traced back to that and there should be some level of consequence for that."

Julia Allen: Right, because on the IT security side we'll sometimes call those minimum essential or security hygiene — kind of like brushing your teeth and going for your physical once a year, things like that.

Bob Charette: I don't think it's necessarily going to be easy either because again, there's a lot of different ways to develop a system. But I think that there are some core practices that maybe as a community as we start the conversation we can all agree that this probably or these approaches really should be ones that we all should follow and it's our bad as they say if we don't.

Julia Allen: So Bob while you're in the midst of your research and we're waiting for your article to get this ball rolling, do you have some places where our listeners can learn more about the idea?

Bob Charette: Well I think that at least they want to take a look at the background of the idea. They can take a look at something like the National Quality Forum site which describes medical "never events" and the history and how it came to be and some of the arguments pro and con which have application to software "never events" as well. I also think you could take a look obviously at the SANS top 25 most dangerous programming errors (you can find that on the web) as well as the OWASP which I'm —

Julia Allen: That's the Open Web Applications Security Project right?

Bob Charette: Thank you, thank you.

Julia Allen: You're welcome.

Bob Charette: Top ten list. It's a group that's looked at web application security flaws. And then there's the Software Assurance Forum for Excellence in Code or SAFECode which is a vendor consortium made up of EMC, Juniper Networks, Microsoft, Nokia, SAP and Symantec which also have outlined a core set of security development practices that can be used in a lot of different development environments, all toward improving software security.

Julia Allen: Well I'm really thrilled to have an opportunity to talk with you about this. I know your efforts in risk management broadly are quite well known and so taking that laser focus and applying it to this issue I think can really help us move the conversation along. So thanks very much for your time today.

Bob Charette: Well thank you for talking with me.