Canadian Nuclear Safety Commission | Commission canadienne de sûreté nucléaire

CMD 15-H8.87

File / dossier : 6.01.07
Date: 2015-09-28
e-Doc: 4855765

Oral presentation

Submission from A.J. Kehoe

Exposé oral

Mémoire de A.J. Kehoe

In the Matter of

Ontario Power Generation Inc.

Application to renew the Power Reactor Operating licence for the Darlington Nuclear Generating Station

Commission Public Hearing
Part 2

November 2-5, 2015

À l'égard de

Ontario Power Generation Inc.

Demande concernant le renouvellement du permis d'exploitation pour la centrale nucléaire de Darlington

Audience publique de la Commission
Partie 2

2-5 novembre 2015

Canada

Mr. A.J. Kehoe

Participation via: Written_and_Oral

Scrutiny of Darlington Nuclear Generation Station is Currently Insufficient
==========

Date: 2015-09-28
##########

##########
Introduction

My name is A.J. Kehoe. People have been paying me for my I.T. knowledge and skills since 1994, and I've specialized in I.T. modernization services for mission critical systems since 2002.

My intervention is going to focus primarily on a major technical problem in the I.T. systems that OPG uses at Darlington. This is a problem that virtually anybody can understand, regardless of your technical knowledge. If you've ever followed a recipe to make food or used blueprints to build something, then you are probably fully capable of understanding this problem.
##########

##########
Recipe Analogy
-=-=-=-=-=-=-=-

A recipe typically has two parts: ingredients and instructions. Ingredients are the materials that you'll use to make your food, and the instructions are the steps you'll follow to cook or bake the ingredients. Assuming you consistently use good tools and good ingredients, following the recipe's instructions exactly should generate the same food every time.

Having a recipe allows you to know what's in your food, how it's prepared, and to change the recipe if necessary. If the ingredients include things that you won't eat, then you can make substitutions. If the instructions tell you to do things that could damage your kitchen tools, then you can alter your method.

In the computer industry, "source code" is what we call a recipe that's used to make a software program. Similar to how a recipe is written in a language that a cook can understand, source code is written in a language that a software compiler can understand. A cook uses a recipe to make food that you can eat, and a software compiler uses source code to make a program that you can use.

If you're somebody who cares about nutrition, then it's critically important that you have the recipes for the food you'll be cooking and/or serving. For example, the United States Secret Service won't allow food to be served to President Obama without knowing what's in it and how it was made. If the food satisfies the requirements of the Secret Service, but the President has a negative reaction to it anyway, then they can analyze the recipe and its kitchen to discover what went wrong.

Like the Secret Service requiring access to recipes to protect President Obama's nutrition, I.T. administrators require access to source code to protect their mission critical software. Without a software program's source code, there's no guarantee that you can anticipate everything that may go wrong, and if something does go wrong, there's no guarantee that you can fix it.
##

##
Nuclear Power Plants Aren't Video Games
-=-

A nuclear power plant is an environment where all software must be regarded as mission critical. Whether the software is for safety systems, monitoring, inventory, communications, or virtually anything else, it's needed in some way to keep the plant operating as expected.

A few years ago, I was flabbergasted to learn that neither OPG or CNSC have access to the source code for nearly all the software used at Darlington. This means that neither OPG or CNSC know how Darlington's software works. OPG certainly seems to know how to use their software, but they don't know exactly what's in it or how it was made.

When I raised this issue in 2012, OPG responded by saying that they hire professional software testing firms to do extensive tests of their software. Practical testing is a very important part of quality assurance, but there are many problems that are virtually impossible to detect without carefully inspecting that software's source code. Examples of such design flaws include mathematical errors, race conditions, poor input validation, buffer overflows, and many others.

In the 1980s, software design flaws in AECL's (Atomic Energy of Canada Limited) Therac-25 radiation therapy machine caused patients to receive extremely high doses, three of whom died as a result. An investigation concluded that AECL should have had their software's source code inspected by independent experts before the machine was allowed to operate. Standard IEC 62304 was created in response to nuclear software failures.
##

##
Open-Source Software is Necessary for Mission-Critical Environments
-=-

Software with publicly available source code is known as open-source software. Governments, military, corporations, universities, and even individuals depend on open-source software for many/all of their mission critical needs. The Internet is arguably humanity's most valuable and complex creation, and it is powered by open-source software.

Anybody can analyze the Internet's most mission critical software to find design flaws. When somebody discovers a design flaw in software and reports it to the organization responsible for maintaining it, the flaw is typically fixed very quickly, and the Internet improves. Decades of public scrutiny have made the Internet resilient to the point where we'll probably never hear a news reporter say "The Internet was down today.".

The hundreds of mission critical systems I maintain across Canada are all powered by open-source software. This software is regarded by experts around the world as the highest quality available, and yet people still manage to find the occasional flaw.

When software doesn't work precisely the way that I expect, I could ignore the imperfection or try to work around it, but I strongly prefer to fix it instead. I read through the source code, find the bug, correct it, test it, and then submit my revisions to the maintainers for review. The maintainers meticulously inspect my revision, make any necessary adjustments, and then release it to the world for additional scrutiny.

Software isn't the only technology that benefits from the open-source development model. Consider how open-source design has benefitted the automobile industry:

http://arstechnica.com/cars/2015/09/open-source-design-is-changing-the-way-we-make-cars/

Software that doesn't have its source code publicly available is called closed-source. With closed-source software, to fix its design flaws, you are eternally at the mercy of the people who wrote it. If the software's creator won't fix a bug because you can't afford their fees, or because they went out of business, died, or because of any other reason, you may have a serious problem.

The longer source code is publicly available for scrutiny, the more likely it is that someone will find a problem before something goes wrong. If the source code is never made available, then it's naturally much less likely that the problem will be detected in advance.

If Ontario Hydro had made the source code for the software at Bruce Nuclear Generating Station available before 1990, perhaps this may have been averted:

http://www.dcs.gla.ac.uk/~johnson/papers/seattle_hessd/georgechris-p.pdf

In my own experience, I have found more problems than I can remember in software, both closed-source and open-source. With open-source software, I've been able to fix design flaws with a 100% success rate, whereas with closed-source, my results have been dependent on the availability and willingness of its author.

In 2002, I began to phase out my support for closed-source software, despite the fact that nearly all my customers were using it almost exclusively. Today, I'm still looking after those customers, but because we're all now using open-source software, we're all much happier with the software we use and maintain.

##

##

OPG's Bogus Excuses

-=-=-=-=-=-=-=-=-=-=-

On many occasions, I've tried to explain to OPG that they need to treat their software with the same philosophy that safeguards the Internet. Most of the responses I've received have been from people who don't understand the problems I've described.

Various OPG people have thought that "source code" is synonymous with "access code" or "password". It isn't.

Some OPG people have told me that having their software's source code publicly available would somehow result in a security vulnerability. OpenSSH and LibreSSL are security products used by millions of people around the world to safeguard top secret information, and both of these software titles are open-source. The security keys and passwords are intended to be unique for each user, but the mechanisms for processing them are widely known and intensely scrutinized.

Knowing how something was built doesn't mean you have access to it. For example, if President Obama was to publish a book of his favourite recipes, you could make the same food that the President enjoys, but that doesn't mean you can access the food he eats.

OPG has claimed that even software in "protected areas" would somehow be vulnerable if its source code was released. It's surprisingly difficult to isolate a computer as they've described, but I know that this is possible to achieve if you have sufficient skill, time, and resources. If OPG is aware of vulnerabilities in these allegedly isolated systems, then why would they be afraid of revealing how they work "under the hood"? Is OPG lying when they say that these systems are completely separate from the outside world?

Several OPG people have told me that they don't need their software's source code scrutinized because of the practical testing that it undergoes. In reality, measly few people have tested it, and as I've explained already, practical testing alone is insufficient. Most of their software isn't available to the public for broader testing.

In what seemed like genuine efforts to understand my concerns, a few OPG people have been brave enough to have conversations with me. I consider such discussions to be much more reasonable than making statements that you can't support about a subject that you don't understand, so I appreciate their efforts.

In our conversations, two OPG people eventually came to realize what I was asking. "A.J., are you seriously saying that we should allow anybody to look for bugs in our software, and for us to fix any bugs that they find? Given our quality control procedures, do you have any idea how much this would cost us?" "Yes," I replied, "it would cost an insignificant fraction of what you'd lose if one of those bugs caused a nuclear disaster."

So, the real reason why OPG won't adopt the open-source model that's required for other mission critical environments is because they're trying to save money. CNSC has said repeatedly that financial cost considerations aren't part of CNSC's mandate, so if CNSC is being honest about this, then you'd have no problem telling OPG that they need to do a better job of maintaining their software.

CNSC's Ramzi Jammal has said repeatedly that nuclear power plants need to be as self-sufficient as possible in the event that something goes wrong. Currently, OPG depends on outside organizations to help them when a design flaw is found in the software they use, so they're completely dependent on the people with the "secret sauce" to respond in a timely manner.

OPG seems to understand hardware, but they don't seem to understand software. They'll gladly show off their hardware designs, introduce you to their staff, and talk about how humans operate their plant, but they're strangely afraid to say how instructions written by computer programmers are doing jobs that were previously performed by humans. They keep their software a secret because they're afraid of the costs of maintaining it responsibly.

Nuclear power plants are complex places, but the software running in them is exponentially more complex. Neither OPG or CNSC have remotely enough staff to analyze this software and/or respond to any problems, so you need to make it available for public scrutiny. Major I.T. companies like Google, Apple, and many others have conceded that there's a lot of things they can't reasonably hope to achieve correctly on their own, but OPG has acted like this reality somehow doesn't apply to them.
###

###
CSA (Canadian Standards Association) Shortcomings
-=-

OPG and CNSC have repeatedly said that Darlington's I.T. systems are compliant with CSA standards, but glaringly absent from these standards is any mention of the need for source code access. Having access to software's source code is an essential requirement of disaster prevention and emergency response, but no mention of this appears anywhere in the standards routinely cited by OPG and CNSC.

In 2014, I participated in CSA's standards process regarding cyber security for nuclear power plants (N290.7). Like CSA's other standards relating to this subject, this new one didn't include anything about the well-known fact that open-source software is essential for cyber security, so I commented accordingly:

-----BEGIN COMMENTS TO CSA-----
This draft offers a lot of prescriptive advice about how to secure critical systems, but glaringly absent is any review of the systems themselves. To secure anything, you need to have a very thorough understanding of what you're trying to secure, but this draft focuses merely on the periphery.

Modern computers are controlled using software, which is a series of instructions that can be fed into them electronically. If the software was written perfectly, then the machine can be expected to perform exactly as expected. If the software wasn't written well, then problems may arise, but they won't necessarily be evident to the human user.

In these comments, I will describe the importance of having access to software's "source code", which is the term for the computer programming instructions that get compiled into software. Certain CNSC (Canadian Nuclear Safety Commission) and OPG (Ontario Power Generation) officials assume that source code is synonymous with "access code" or "password", but this isn't the case. To use an analogy, "source code" is to software what "recipe" is to food.

Having access to a software program's source code will show you precisely how it works, which lets you do two very important things:

1. Anticipate when something can go wrong. A computer programmer can read a program's source code and detect limitations, design flaws, and other issues that could arise while using the software. This is like when a cook reads a recipe and finds a problem before they start cooking.

2. Correct any problems that may exist. A programmer can modify the source code to fix a problem, share their modifications with the appropriate authorities, and then the authorities can review the modifications to decide if they should be committed into their production environment. This is like when a cook changes a recipe to accommodate a patron's allergies or special diet.

I've been working professionally with computer software programs since 1994, and I have extensive experience dealing with I.T. security, systems administration, and software development. In my experience, when you have access to a program's source code, you can fix problems with the program 100% of the time, but when you don't have access to that source code, there's no guarantee. For video games and other applications that don't really matter, having access to the source code isn't usually important, but in mission critical environments like security, communications, accounting, et cetera, having access to the source code can save you from severe and costly disasters.

Without its source code, software often can't be substituted. This forces the operator to depend on the existence of a sole supplier (typically a commercial entity) for the continued life of the plant. Open source software would allow multiple providers to compete for service contracts, which is desirable from both cost and quality perspectives.

OPG has repeatedly assured the CNSC and the public that the software in their protected areas is completely isolated from publicly accessible areas, and that only personnel who are extensively screened and highly trusted can gain access to protected areas. Under these conditions, it is safe for the public to audit this software because any design flaws discovered shall not be exploitable by anyone other than trusted personnel in protected areas. It's the same as how a student can go to school, study the blueprints and a scale mock-up of a nuclear reactor, and not pose any risk of harm to a real reactor being used at an actual nuclear power plant.

Having dealt with OPG and the CNSC on numerous occasions, I am not convinced that their software is as safe as they've been led to believe by their suppliers. OPG and the CNSC are unable to sufficiently audit this software themselves because they don't have access to its source code, and there is no standard or regulation mandating that their software's source code be made available. OPG's current method for testing their software is to have some select individuals run practical tests on it, but this is no substitute for a thorough public audit of its source code.

I doubt that this is the case, but it's possible that OPG is using software that contains malicious code. Without access to the source code, only the culprit may know for sure. Such malicious software might not be remotely accessible, but its payload could be triggered by other mechanisms, such as clocks, keystrokes, et cetera.

Far more likely than a malicious attack is the existence of software errors. Some errors may be innocuous, but others might directly or indirectly result in a compromise of safety, security, emergency response, and/or disaster recovery. Practical testing is certainly important, but issues like race conditions, off-by-one errors, code injection, and others can be exceedingly difficult to detect without a

thorough source code audit. It's like trying to judge the safety of a car that has had its hood welded shut.

The Internet is arguably humanity's most complex and valuable creation, and the source code for all of its most critical software is subject to public scrutiny and extensive review. At OPG nuclear power plants, however, software is not subjected to the same degree of examination and testing that is used to safeguard the Internet. The Internet is obviously quite different from a nuclear power plant, but both need to be treated with the utmost care to prevent an international disaster, and I don't believe the current nuclear power cyber security standards are good enough.

Some people argue that having access to a program's source code is an inherent vulnerability, but for software being used in a completely secure place, this isn't necessarily true. It's the same as how having a cookbook of United States President Obama's favourite meals doesn't give you access to the food he eats.

Some have argued that a software program's source code is proprietary information like a secret recipe. OPG and CNSC can take apart and analyze any hardware component of their nuclear power plants, but they're incapable of giving their software the same degree of scrutiny. Given the risks of operating nuclear reactors, I find it absurd that the operators of these software devices are so disinterested in their inner workings.

In a private conversation, an OPG employee once asked me if I understood how much time and money it would cost to fix any bugs that might exist in their software. Such costs would be completely insignificant compared to the costs of a nuclear disaster.

This standard needs to mandate perpetual public scrutiny of the source code of all software that's used in protected areas. Without this amendment, the safety and security of nuclear power plants will remain questionable.

As discussed with Robert Reipas, I have two additional requests:

1. I request the identity of every member of this Technical Committee to be published in this document. This will allow reviewers to verify that the majority -- if not all -- of these people are unbiased and completely independent from the nuclear industry and the CNSC (a nuclear industry promoter).

2. I request to be provided with a copy of the Technical Committee's dispositions to my public review comments. I want to be certain that the Technical Committee understands my concerns, which are typically ignored or wrongfully dismissed by OPG and the CNSC.

I have tried to make these comments as clear and unambiguous as possible. Please don't hesitate to contact me if you have any questions.
-----END COMMENTS TO CSA-----

The response I received from CSA was absurd:

-----BEGIN FALLACIOUS RESPONSE FROM CSA-----

The N290.7 Technical Sub-Committee understands the security views of the open source community, and shares the goal of ensuring that nuclear systems are secure. The N290.7 Technical Sub-committee is not aware of any study that conclusively demonstrates that open source software is inherently more secure than proprietary software. On the contrary, while the committee was reviewing this submission the "Heartbleed" bug was receiving extensive public attention. This incident illustrated that open source code might have significant security vulnerabilities that are not discovered through inspection by the public. The N290.7 committee believes that the measures contained in the standard provide a strong basis for protecting nuclear systems. At this point the committee has finalized the N290.7 draft and agreed to send it to the editorial and balloting stages.
-----END FALLACIOUS RESPONSE FROM CSA-----

My retort:

-----BEGIN MY RETORT-----
1. The "Heartbleed" bug in OpenSSL affects the security of communication that takes place over the publicly accessible Internet, not the security of applications that are used in a "protected area". To quote my comments:

-----BEGIN QUOTE-----
OPG has repeatedly assured the CNSC and the public that the software in their protected areas is completely isolated from publicly accessible areas, and that only personnel who are extensively screened and highly trusted can gain access to protected areas. Under these conditions, it is safe for the public to audit this software because any design flaws discovered shall not be exploitable by anyone other than trusted personnel in protected areas. It's the same as how a student can go to school, study the blueprints and a scale mock-up of a nuclear reactor, and not pose any risk of harm to a real reactor being used at an actual nuclear power plant.
-----END QUOTE-----

2. At least one standard (IEC 62304) was created in response to nuclear software failures. To quote Wikipedia's AECL Therac-25 article (http://en.wikipedia.org/wiki/Therac-25#Root_causes):

-----BEGIN QUOTE-----
Researchers who investigated the accidents found several contributing causes. These included the following institutional causes:

- AECL did not have the software code independently reviewed.

- AECL did not consider the design of the software during its assessment of how the machine might produce the desired results and what failure modes existed. These form parts of the general techniques known as reliability modeling and risk management.

- The system noticed that something was wrong and halted the X-ray beam, but merely displayed the word "MALFUNCTION" followed by a number from 1 to 64. The user manual did not explain or even address the error codes, so the operator pressed the P key to override the warning and proceed anyway.

- AECL personnel, as well as machine operators, initially did not believe complaints. This was likely due to overconfidence.

- AECL had never tested the Therac-25 with the combination of software and hardware until it was assembled at the hospital.
-----END QUOTE-----

3. [This response] fails to address the problem of nuclear operators not knowing how their software works. They certainly know how to use their software, much like how any driver knows how to use a car, but there's nothing in the N290.7 draft that requires them to know how it works "under the hood".

I don't believe it's necessary for every nuclear operator to know their software as well as a mechanic knows a car, but at least with a car, you can open the hood to see how it works, whereas with software, you need the source code. If you don't have the ability to see exactly how something works, then there's no guarantee that you can anticipate when something will go wrong, and if something does go wrong, then there's no guarantee that you can fix it.

OpenSSL's "Heartbleed" bug was certainly disastrous, but because everybody had access to its source code, anybody had the ability to fix it on their own, without having to depend on the keepers of some secret recipe. Because of this, people were able to patch their code right away, and not have to wait days, weeks, or longer for something like one of Microsoft's "Patch Tuesdays".

Another advantage of open-source software is that you can replace it without risking the loss of critical but undocumented functionality. OpenBSD (completely unrelated to OpenSSL) created a fork of OpenSSL called LibreSSL, which is intended as a drop in replacement for OpenSSL, but with a less horrible code base. OpenBSD's Bob Beck gave an excellent and entertaining presentation on LibreSSL in Ottawa last month at BSDCan 2014:

http://www.openbsd.org/papers/bsdcan14-libressl/

Keep in mind that I'm talking about the software that's used in protected areas at nuclear facilities. As mandated in the N290.7 draft, this software shall not be accessible from outside of the protected areas, and the only people who may have access to protected areas are those who have been sufficiently scrutinized by trusted authorities.

Knowing how something works in a protected area isn't a problem if you're incapable of accessing it. I think we can all agree that unauthorized access to a nuclear facility's protected areas is an exponentially greater risk than knowing the intricacies of the software being used inside.

In their paper "Assessing the Quality of Scientific Software", Diane Kelly and Rebecca Sanders describe code inspection as a necessary part of the software assessment process:

http://secse08.cs.ua.edu/Papers/Kelly.pdf

Out of the twenty years that I've been working professionally with computer software, the past fifteen have been spent working almost exclusively with mission critical systems. I learned very early in my career that closed-source software does not belong in mission critical environments. I can regale you with stories about the countless occasions where -- had the software's source code been available -- costly problems could have been solved without extended downtime, or even prevented entirely.

[...] I strongly encourage you to amend this draft.
-----END MY RETORT-----

A message I sent in response to the ensuing reply:

-----BEGIN 2014-06-04 MESSAGE TO CSA-----
Robert Reipas wrote:
> Dear A.J.,
>
> Thank you for your comments, as the public review period has closed for this document the committee will not issue another response.
>
> All CSA standards are continuously updated, and any suggestions for their improvement will be referred to the appropriate committee. To submit a proposal for change for a published standard, please send the following information to inquiries@csagroup.org and include "Proposal for change" in the subject line:
> (a) Standard designation (number): (e.g. CSA N290.7)
> (b) relevant clause, table, and/or figure number;
> (c) wording of the proposed change; and
> (d) rationale for the change.
>
> All of our nuclear standards are available for free access through the below link. The N290.7 standard is scheduled to publish in December 2015 and will also be available there as well.
> https://community.csagroup.org/community/nuclear/nuclear-standards---view-access-
>
> Best regards,
>
> Rob

Why should I do all this if you're just going to ignore my recommendations?

Rob, I am extremely displeased with the way this public review transpired. I tried very hard to explain this problem to you, and to ensure that you understood it, but you've been unprofessionally evasive the whole time. You would never speak to me on the telephone, and getting unambiguous answers from you often required badgering, but was sometimes fruitless.

Your job was to facilitate the standards process, which includes ensuring that comments are received and understood by your committees. The fact that they clearly did not understand my comments shows me that you have failed, which is inexcusable given the effort I put forth to elucidate my concerns.

I find myself questioning whether I should trust CSA. I don't know if this was just one fluke incident, or if this is indicative of everyday practices at CSA. CNSC and OPG rely almost exclusively on CSA standards for their I.T. guidelines, but now I see that these standards can be deeply flawed.

This genuinely concerns me. [...] I have friends and customers who work at these facilities, plus a great many people whom I care about in the surrounding areas. The disregard you have shown for our safety is very unnerving.
-----END 2014-06-04 MESSAGE TO CSA-----

And another message I sent to CSA:

-----BEGIN 2014-06-21 MESSAGE TO CSA-----
Mary Cianchetti wrote:
> Dear Mr. Kehoe,
>
>
>
> I wanted to follow up on your email below and sincerely thank you for your time in preparing feedback to the N290.7 draft standard and also both Greg and Rob for addressing your comments while I was away. I am concerned that you feel your efforts were ignored.

"Ignored" was the wrong word for me to use. What I meant to say is that my efforts were dismissed fallaciously. And this is a fact, not an opinion.

> Our Federally accredited standards development process does not allow CSA Staff interference in the technical content of our standards. My project manager responsible for the development of this standard, Robert Reipas, cannot influence the outcome of a technical discussion - all technical discussion is the responsibility of the volunteer members that make up the committee. This committee consists of balanced representation from government, owner/operators, service industry, supplier/fabricators, and general interests experts in the field. The documents we publish are also voluntary standards and it is the responsibility of those who reference them to ensure that they meet their needs.

I recognize that the adoption of your standards is voluntary, but the reality is that CNSC and OPG almost consistently defer to your standards exclusively when it comes to I.T. matters. Members of the committees that drafted and reviewed N290.7 evidently have zero experience with debugging systems in mission critical environments, so I maintain my assertion that the compositions of this standard's committees are dubious. You (CSA) should have exercised much greater diligence in your selection process, given the dogmatic reverence of your standards by CNSC and OPG.

> I can confirm that all feedback received during the 60-day public review was taken seriously and considered formally by the committee; CSA staff have no influence on whether this feedback is persuasive to the committee. Rob did do his job by ensuring all public comments were given due consideration. Having reviewed all of his communication with you, I also feel that he responded to your queries professionally, with respect, and in a timely manner.

Rob definitely responded to my queries in a timely manner, but I disagree with his responses being characterized as professional or respectful. For example, I left two voicemail messages for him to call me back, but he returned neither call. Instead of calling me back like a civil person, he resorted to replying by E-mail, presumably because he wanted to choose his words very carefully (many people use this as a "cover your @$$" tactic). I suggested in my E-mail messages that he call me, but I suppose my wording was too subtle, so I guess I should have been more insistent.

Rob's unwillingness to speak to me is the first example of him being evasive. Let's review the other two from our correspondence:

-----BEGIN EXAMPLE 2-----

Me: Do you understand the concerns I've raised? I've been trying to explain this problem to OPG and the CNSC since 2012, and most of the time, they clearly misunderstand the issue. When they do understand, they cite CSA standards to say that their software is safe and secure, but these standards are insufficient because they don't require any real scrutiny of software internals. To use a car analogy, it's like you're describing almost everything that's necessary to ensure the safety and security of a vehicle, except for the part about being able to look under the hood for inspections and repairs.

Rob: CSA staff have do not have a say in the technical requirements of their standards. CSA staff facilitate the accredited standards development process and CSA's volunteer members, who are experts in their field, develop and vote on the standards.

Rob: The N290 Technical Committee and N290.7 Technical Subcommittee membership includes cyber security experts representing many different interest categories within the nuclear industry (i.e., owners/operators, regulators, service industry/consultants, suppliers/fabricators, and general interest/academic groups.) The committee is well equipped to address comments on the topic of cyber security.

Me: So, your job is to orchestrate the standards process, and to compile feedback in a comprehensive manner. Based on this answer, I'm guessing you're not a member of the technical committees, but do you understand the concern I've raised? If it's going to be your job to assemble the standard, then I want to make sure that you understand this issue so that you can verify that the technical committees are addressing it correctly. It's actually a very simple concept to understand that doesn't require any technical knowledge or experience, but I'm not sure how well I'm describing it.

Rob: Correct, I am not a member of the Technical Committee; CSA staff are a neutral third party which facilitates the standards development process and ensures the accredited standards development process is followed. CSA staff have do not have a say in the technical requirements of the standards; decisions on the standard are made by consensus from the balanced stakeholder representation on the committee. Please submit your comments to the committee for them to review through our public review website and they will be considered by them.

Me: I just submitted my comments on the website's title page. Can you please confirm that my comments are sufficiently legible and appropriate for the Technical Committee?

Rob: Thank you for submitting your comments, I can confirm I have received your comments on the overall draft.

Me: That's not quite what I was asking, but I'll accept this as a receipt confirmation.
-----END EXAMPLE 2-----

I find it absurd that Rob never admitted to not understanding my concern. I find it even more absurd that he would assume that the committees would understand my concern, given the fact I stressed repeatedly that this is something that CNSC and OPG have almost consistently misunderstood. As the standard's project manager, Rob should have been able to validate the competence of his experts, but he failed to do so for whatever reason, thereby causing you (CSA) to facilitate a poorly chosen group of "experts".

-----BEGIN EXAMPLE 3-----
Me: Who are the members of these technical committees? I want to verify that the majority -- if not all -- of these people are unbiased and completely independent from the nuclear industry and the CNSC (a nuclear industry promoter).

Rob: The committee members are a balance of individuals with expertise in the subject area from the interest categories which I outlined in my previous email. This committee is a balance of members from; the general interest category (including universities and retired professionals), the government/ regulatory category, the owners/operators category, the service industry/consultants category, and the suppliers/fabricators category.

Me: Unfortunately, your answer suggests that there will indeed be a strong bias among the Technical Committee. My experience with the nuclear employed is that they're much more likely to maintain the status quo than to enact changes that would make their operations less dangerous. I am very concerned that the Technical Committee will fabricate some absurd excuse to justify ignoring my comments instead of giving this matter the attention it correctly warrants.

Rob: No comments on the draft will be ignored, it is a requirement that the committee reviews and dispositions each comment received. The dispositions to your comments will also be provided you; you can expect a disposition to your comments from the committee mid to late this month.

Me: I hope their response will be relevant, and not completely miss the point like nuclear industry responses are prone to do. It would be great to read their dispositions by next month, but I will understand if it takes longer.

Rob: Pertaining to your request that the identity of the members be published in this document; the members of the Technical Committee responsible for approving the document as well as the Technical Sub-Committee which drafted the N290.7 standard will be published in the front pages of the document. [...]
-----END EXAMPLE 3-----

You'll also notice that I was absolutely correct about these committees fabricating absurd excuses to justify their fallacious dismissal of my comments. They completely missed the point too. Here are some car analogies to illustrate how ridiculous their responses sound to me:

-----BEGIN ANALOGIES-----
Me: If your car's hood is welded shut, it's impossible for you to know for sure that everything is working properly underneath, and if something goes wrong, there's no guarantee you can fix it.

Committee: Somebody elsewhere installed car door locks from a questionable manufacturer, and the locks turned out to be bad.

Me: Why are you worried about car door locks when your cars are always locked away in a protected environment?

Committee: We've not aware of any study that conclusively demonstrates it's better not to use cars with hoods welded shut.

Me: What about those studies that concluded your industry has had crashes caused by failures to properly inspect your engine compartments, and the international standard that was created in response?

Committee: We see no reason why we shouldn't continue using cars with hoods welded shut.

Me: But what about all the reasons why it's better to use cars that haven't had their hoods welded shut?

Committee: We will not discuss this further.

Me: You call this a discussion?
-----END ANALOGIES-----

> We work very hard to ensure that the standards development process remains transparent, fair, and inclusive. It is my hope that you continue to participate in our standards development process and share your valued feedback.

If the N290.7 standard development process was transparent, then you would have identified its committees' members before publishing the standard. If N290.7's process was fair and inclusive, then you would have had a forum open to discussions, and not demonstrated the elitism that I have seen here. In the case of N290.7, you have failed in your democratic aspirations.

I learned from Greg when we talked recently that it's a waste of time to comment on CSA draft standards if you're not a member of one of your committees. Greg explained that the only people who can expect to be included in an actual discussion (not just one-off statements) of a standard are members of these committees, and there's a long and tedious process to becoming a committee member. Greg encouraged me to become involved, but my time is very limited, so it would be exceedingly difficult for me to effectively prevent CSA from approving any more defective standards.

Again, I'm extremely upset by how summarily and fallaciously this very serious safety concern was dismissed. I genuinely tried to get the message across clearly while countering the misconceptions that I've heard previously from CNSC and OPG, and your committees responded with a new batch of preposterous claims and a seemingly psychopathic aversion to discussion. I'm keenly aware of why closed-source software doesn't belong in mission critical environments, so it deeply disturbs me that people I care about are using it in nuclear power plants that are surrounded by vastly more people whom I also care about. Rob and your committees don't seem to appreciate our safety like I do.

Before submitting my comments in April, I passed them around to some colleagues to review, along with a link to the draft. One person was highly critical of CSA's relationship with the nuclear industry, describing you as an industry puppet, and she also said that your public consultation process was limited to your posting of drafts on your website. Having dealt with many non-profits and government groups that don't have the funding or resources to conduct more publicized consultations, I wasn't dissuaded because I assumed that your website was merely the consultation's starting point. Now I know that your website postings are actually the extent of your public consultation. I feel like this has been a spectacular waste of my time.
-----END 2014-06-21 MESSAGE TO CSA-----

CSA's cyber security standard for nuclear power plants does have recommendations that I don't think anybody would consider bad, but its failure to address the extreme problem of software source code access is unacceptable. Based on my experiences, I think it's fair to say that this CSA standard should not be trusted implicitly, and that CSA's standardization process is flawed and irrational.
##

##
Questionable Emissions Reporting
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

OPG occasionally presents reports about their emissions. Since 2013, I've been asking them to explain how their numbers were generated, and OPG has consistently cited the use of software. I've consistently responded by asking for a copy of their software's source code so that I can verify the accuracy of their mathematical formulae and algorithms.

OPG has never agreed to divulge the source code for their environmental monitoring/modelling software. Sometimes, they say that this code is a trade secret or "proprietary information", but given their emissions' effects on human health and the environment, I won't accept this answer. Sometimes, they say that revealing this code would be a security vulnerability, but they've never explained how that can possibly be. Sometimes, they say that this software undergoes rigorous quality assurance tests, but without access to this software and its source code, I can't test their beliefs.

In 2015, automobile manufacturer Volkswagen was found using software to cheat on their environmental emissions tests from 2009 to 2015. Had their software been open-source, this scandal could have been discovered much sooner, and years of human and/or environmental harm would have been averted.

Similar to Volkswagen, OPG could also be using software to cheat on their emissions reports. OPG refuses to reveal publicly verifiable evidence that they aren't, so CNSC shouldn't trust these reports until they can be proven independently with source code access.

OPG claims to have the best software available, but why won't they prove it? Do they have something to hide, as was the case with Volkswagen?
##

##
Costs for OPG
-=-=-=-=-=-=-

Many independent studies have shown that -- despite the initial costs -- switching to open-source software can actually save you significant amounts of money in the long-term. For example, if a supplier becomes unreasonably expensive, you can switch to a different supplier that offers the same service, but for a better price. You can even resolve problems on your own, and not have to wait potentially long periods of time for potentially devastating problems to be resolved by an outside party.

Considering the exorbitant amount of money that OPG is planning to throw around with their soon-to-be-rejected Darlington rebuild plan, it would cost virtually nothing for them to switch to open-source software. Most of these costs would be borne by their suppliers.

At a meeting of DNHC (Durham Nuclear Health Committee) on 2015-09-11, an OPG official admitted that OPG has the authority to demand software source code from their suppliers. This means that -- for the vast majority of the software that OPG uses -- OPG can simply put something to the effect of "all software we use must be open-source" into their implementation requirements. If a supplier refuses to comply, I know from my own experience that there are other suppliers who would be eager to take their place.

Switching to open-source software could result in massive savings for OPG. At Pickering, for example, they use PDP-8 computers, which is a 12-bit computer introduced by DEC (Digital Equipment Corporation) in 1965. If OPG actually understood how their software worked "under the hood", they wouldn't be so dependent on ancient hardware.

I heard a story from the 1990s about OPG going to something analogous to a garage sale, searching for PDP-8 hardware. This story made me laugh and cringe simultaneously. Before ending my support for closed-source software, there were times when I had to track down old hardware to support customers' old software, but a PDP-8 is like an ancient artifact you'd find in a museum.

I personally have no interest in ever working for OPG, or to ever serve as one of their suppliers in any capacity. All I want is for them to stop treating their software like lowly video games, and to start respecting it as mission critical tools that may need to be taken apart and reassembled at any time.
##

##
No Backups
-=-=-=-=-=-

Darlington Nuclear Generating Station never should have been built. It is too close to a densely populated area worth many billions of dollars, and the area's population and property values will only increase. Exacerbating this problem is the woefully insufficient infrastructure required to accommodate a mass evacuation.

Before 2015, there were no safe routes for people to travel between Whitby and Ajax using bicycles or wheelchairs. A link was finally installed in 2015, but if it was severed for some reason during a nuclear disaster, such people fleeing Westbound from Darlington may get stuck in Whitby. Durham Region's Works Department has a history of prioritizing driver convenience over human safety, so it doesn't seem likely that an alternate link will be installed any time soon.

Ontario's Provincial Nuclear Emergency Response Plan is so vague and inaccurate that it's scorned as rubbish by people who are expected to take it seriously. A police traffic operations commander I spoke to described it as asinine, and he said that it would be as successful as trying to herd cats.

If Darlington suffers a catastrophic failure, what will we do? As far as I know, we don't have a backup Toronto and Ontario standing by to replace our current Toronto and Ontario. I haven't seen any 1640

cubic kilometre fresh water reservoirs lying around that we could use to replace Lake Ontario. My insurance company won't cover me in the event of a nuclear meltdown, and according to Canadian law, OPG won't be liable for any more than one billion dollars.
##

##
Too Many Vulnerabilities
-=-=-=-=-=-=-=-=-=-=-=-=-

When studying security, one of the first things you learn is that a determined attacker with sufficient resources will find a way to inflict damage. Emergency response and disaster mitigation are therefore very important, and planning requires an understanding of a worst case scenario.

Like any electricity generating station, nuclear power plants have many potential attack vectors. An attacker could:

- Manipulate staff or contractors to make them do dangerous things.
- Disrupt or corrupt the fuel supply.
- Cut off the electricity transmission towers.
- Crash an airplane into the power plant or nuclear waste storage buildings.

There are countless other ways to precipitate a disaster. A defender must counter every possible attack, whereas the attacker needs to succeed only once. Even an attack that poses no real risk of disaster can be considered a successful attack if it prompts operators to shut down the reactors.

False positives that lead to shutdowns are far more likely than any real attack. It costs a lot of money and time to stop and start nuclear reactors, and the effect of this on the power grid can be very problematic.
##

##
Extremely Toxic
-=-=-=-=-=-=-=-

Nuclear power generates excessive amounts of pollution:

- The mining and processing of nuclear fuel produces millions of tonnes of waste and emissions.

- In areas that host nuclear plants, ambient radiation levels are several times higher than what you'll find elsewhere. It's fascinating to roam around Durham Region with a Geiger counter and compare its readings with other parts of the province.

- Every year, Darlington's once-through cooling method kills millions of fish and other wildlife. Lake Ontario is the source of our drinking water, and the concentrations of tritium discharged into it are many times higher than what occurs naturally.

- Nuclear waste will remain radioactive and hazardous for thousands of years, and will need to be guarded for all that time. This will be our expensive and cruel legacy to hundreds of future generations, which is probably more lifetimes than anybody can comprehend.

All of this pollution exposes us to radiation unnecessarily. Like mercury, lead, poison, fire, dioxins, furans, and bullets, ionizing radiation is one of those things that has no safe level when it comes to interaction with the human body.
##

##
OPG Lies
Nuclear power is extremely dangerous, toxic, and expensive, but OPG routinely claims the opposite. Every time they make such bogus claims, there are people who get lulled into a false sense of security, and this compromises safety.

By law, the tobacco industry is forbidden from advertising their products in a positive light. The same laws need to apply to the nuclear industry's filthy, dangerous, and expensive electricity.
##

##
Instructions for CNSC
-=-=-=-=-=-=-=-=-=-=-

I instruct CNSC to do the following:

1. Decline OPG's request to renew Darlington's license for thirteen years.

2. Mandate perpetual public scrutiny of the source code of all software that's used in protected areas of all nuclear power plants.

3. Forbid OPG and all CNSC licensees from advertising nuclear power as clean, safe, or cheap.

4. Publish a report that accurately provides a description of the consequences of an INES (International Nuclear Event Scale) Level 7 disaster at Darlington.

5. Require OPG to distribute potassium iodide pills to all residences and businesses within a fifty-kilometre radius of Darlington by the end of 2016.

6. Require OPG to have at least one more link installed between Whitby and Ajax for people to be able to safely flee Darlington Westbound using bicycles or wheelchairs.

7. For the duration of their tenure, require OPG's president and vice presidents to live inside nuclear reactor buildings or in apartments that are attached to nuclear reactor buildings. This should entice OPG to act less irresponsibly.

8. For the duration of their tenure, require CNSC members to live inside nuclear reactor buildings or in apartments that are attached to nuclear reactor buildings. This should entice CNSC to act less irresponsibly.

9. Stop using GoDaddy.com for web server security certificates. GoDaddy.com kills elephants and doesn't respect their customers. CNSC is currently funding these abhorrently unethical activities.

10. Stop forcing the public to depend on closed-source and proprietary software to view webcasts and video archives. By failing to use open and royalty-free formats like Theora or WebM, CNSC is limiting Canada's ability to watch these hearings. Vatican City made a similar decision in 2015:

http://www.v3.co.uk/v3-uk/news/2407086/open-source-key-to-preserving-human-history-argues-vatican
##