Where Next for Formal Methods?

(Transcript of Discussion)

James Heather

University of Surrey

Tuomas Aura: Why do you need to model the medium separately, why not just merge Alice and the medium?

Reply: You could merge them. If you were doing this in the model checker, that's certainly what you would do, because you increase the state space by having the medium done as a separate process, you're right. When you're doing it in PVS you don't have the problem of increasing the state space, so it doesn't really make a lot of difference whether you merge Alice with the medium or not. But yes, you could easily do that.

There's no intruder modelled in there either because what we're going to do is to get Alice to act as the intruder. The purpose of something like an authentication protocol is to protect the two participants from malicious behaviour coming in from a third party. A non-repudiation protocol doesn't work like that, you're trying to protect Alice from malicious action by Bob, and Bob from malicious action by Alice.

The enemy's, the intruder's, actions in the standard model are incorporated into Alice. So we've got no guarantee that she'll play by the protocol rules, but we assume that Bob will. Now this only gives you half a proof, because this only proves that the responder of the protocol gets a fairness guarantee if the initiator plays by the rules, but you could equally move the intruder model to B, and get the same thing the other way round.

Mike Bond: I must just say I'm curious, you really had to prove over a thousand lemmas to get yourself up and running with the system?

Reply: It depends what you mean by lemma, I mean, yes, you prove something and it decomposes and...

Mike Bond: But because you're proving only positive properties of protocols, it doesn't matter if you happen to have forgotten something in that library?

Reply: Yes, that's right, so there might be obvious bits that we just haven't remembered, and some of what we did was to go through lists of CSP laws and prove them regardless, so I don't really mean to suggest that everything that we've proven was necessary to do this work.

Mike Bond: And if that library's deficient, that doesn't damage the correctness of the proofs?

B. Christianson et al. (Eds.): Security Protocols 2006, LNCS 5087, pp. 59–61, 2009.

Reply: No, absolutely not.

Bruce Christianson: It may damage the liveness of the proof process though.

Reply: Yes, it just makes it take a bit longer.

Mike Bond: On one of the slides you said, PVS had been chosen for its "Expressive libraries plus wealth of support." Did you find it easy to use?

Reply: It's not something where if you've never used it before you can just sit down and get to grips with it within 20 minutes. Once you are up and running with it, and you know what you're doing, it works quite nicely. It's a bit like moving from Windows to Linux, you might be glad three years later that you did it, but you've got quite a headache getting there.

Mike Bond: Three years? [Laughter]

Peter Ryan: So the reduction to finite state, I wasn't quite clear quite how that was happening, is data independence the kind of trickery you're using in that process?

Reply: Yes, well, it depends on quite what you're trying to prove, but things like the data independence will help you get there. There's no automation of the reduction from infinite state to finite state, but what you can do is construct theorems and lemmas that apply fairly neatly, and fairly quickly, to get you from something that's infinite state to something that's finite state. So, you can prove, for instance, that in certain cases it doesn't matter if a key is reused, and so you can shrink the set down so you've only got a finite number of keys in it, and prove that it doesn't matter that it's only a finite number of keys.

Peter Ryan: Is that a paper proof, or a mechanised proof?

Reply: It's mechanised, but it's not automated. You have to do it in PVS, so it's formally constructed, and rigorous, but it's not automated, it's not until you get to FDR that you get the automated support.

Michael Roe: How automated is proof sharing? The right thing to do, is to have every protocol accompanied by a machine checkable CD Rom, which proves everything to the server. Can you do that for security protocols, every time somebody suggests a protocol, that there comes a CD Rom which has a fully formal machine verifiable proof that it works? The Zhou-Gollmann type protocols would be a good target because of the number of times I have to referee papers proposing variants on that scheme.

Peter Ryan: And then find it's broken.

Ross Anderson: Better still, don't let anybody patent a protocol until they provide 100 gigabytes of machine checkable proof.

Reply: Yes, that's the beauty of using a theorem, that you do then get a machine checkable proof at the end of it. So what you'd get out of this actually is two things, you'd get a machine checkable PVS proof that would have some assumptions built

into it, and then you'd get an FDR script that would contain those assumptions, and that FDR script would be checkable as well. Now that doesn't quite get you machine checkability of proof of the whole thing, because the translation...

Michael Roe: Right, that's what I'm worried about.

Mike Bond: Well why should we trust the translation less than we trust FDRs to give us the right answers?

Reply: Well there is that, and then you've got to trust your hardware that it's doing the right FPU operations that you're throwing at it, and things like that, you can go as far as you like.

Michael Roe: I usually trust the verification software, but on the disk that's given to me are two different statements of the theorem: one in PVS and one in FDR. If I just have to trust that these really are the same theorem, then there's a problem. But if there's some tool that churns on and says, that one is just a direct translation of the other...

Reply: Yes, OK, you've got to trust that translation tool to the same extent that you've got to trust the PVS proof checker, and the FDR checker.

Tuomas Aura: The point is that someone else from another university can write another translation checker, and check the same proofs and the same translations, independently.

Reply: Yes, that's right.

Michael Roe: But if there's any kind of human element that can't be verified automatically in the translation, then you've got a problem, for example if there are any handwaving arguments in the translation.

Reply: Yes, that's absolutely right. Getting a formal proof of the translation mechanism would be a massive amount of work, but there's nothing to say that in principle it couldn't be done.

Bruce Christianson: You only need to verify the translation though. You don't need to produce it automatically, you only need to verify it automatically.

Reply: I don't want to be too dogmatic because we haven't finished writing the translation stuff yet, but I don't see why in principle it shouldn't be doable. It might be possible to go a slightly different route and get the model checking effectively to be done within PVS. If you could find an automated way of turning the finite state assertion into 100 Meg of theorem about each state of the system, you then have something that you could put through your PVS proof checker. If all you want is machine verifiability, then you might be able to go down that route. It would be much, much slower than FDR to run the check, because it's doing something it's not really intended for, but it would work, and in principle you could do that. There's nothing that I've talked about that couldn't in principle be proved in PVS, it's just not what it's designed for, it's not designed to let you go through everything state by state. But, yes, it could be done.