A Protocol’s Life After Attacks...

(Transcript of Discussion)

Giampaolo Bella

Computer Laboratory, University of Cambridge, UK

I am going to be speaking about protocol verification again; I’m going to take a rather different perspective from the one we normally take, and I’ll be talking about what happens after an attack takes place. Is there a life for a protocol beyond the attacks?

We all know about verification. On the one hand we have the model checking community trying to find a witness of an attack, trying to find if something went wrong and why the specific property of interest failed. On the other hand we have the opposite approach, assuring that there’s no such witness therefore the specific property holds. But the question here is, is this the whole story? It appears that everything is about finding the attack: is there an attack, is there no attack against confidentiality or authentication? It appears kind of weird. Is it only the attack we are really interested in? Is this really all we should look at? I’ll try and convince you that there’s something more.

So, let’s suppose for a minute we own a jewellers, and one day we find that the main window has been completely smashed by someone. In the worst case there is no-one around and basically all we can do is suspect anyone, any passer-by, because there’s really no evidence against anyone. If we’re luckier, we could find the people there while they’re still at work carrying away the stuff. Basically we detect who actually mounted the attack and we’re kind of happy with that, because we saw them. But we can even do more than that, maybe we have time to call the police, and the attackers will be caught, punished, and sent to jail, so we basically retaliate against them. If you move this to a different context perhaps retaliating means that I go up to the attacker’s window and smash the window. Anyway, this is just a general notion of punishment and retaliation.

This is certainly about the best we can do. This line attempts to convince us that there are some measures we normally take after an attack takes place in the real world. So the idea here is to apply these very same concepts to the world of security protocols and see what we can get out of it.

So how should we do this? Well in the first place we shouldn’t just stop in the face of an attack as we find it, as we are used to doing up until now, we should continue the analysis after the attack and investigate further. Take the model checking perspective; once I find an attack, I may wonder whether there is another attack in the history. Perhaps if I continue the history after I find the first attack, I’ll find that in fact there’s another attack whereby I can retaliate against who mounted the first attack. Or, from a theorem proving perspective, I could assume the existence of the first attack and wonder whether I could have
the same attack mounted twice by two different sets of bad guys. In this case, I wouldn’t really know who the real attacker is, whether the set that mounted the first attack mounted the other one, and if you have the same attack mounted by more than one set of people then it gets more complicated.

Another related issue is why I should do this. When we find an attack nowadays, it means that the protocol designer failed. A successful attack means the protocol is flawed, we need to redesign, to go back to the drawing board and start from scratch, or fix the protocol somehow. That’s what we normally think. But suppose you are the bad guys; you consider mounting an attack to be less convenient according to some criteria if you know that there is a sufficient threat that you can be retaliated against. Would you still mount the attack? In simple words, would I steal a thousand pounds from you today if there’s a realistic threat that tomorrow some of you will steal two thousand pounds from my own account.

Perhaps if we find that a protocol allows this notion of retaliation, we could still live with a protocol that we consider flawed. We know that the bad guys should be kept from mounting the attack by balancing the benefits of the attack with the threats of the possible retaliation against them. So the issue I am addressing here is whether it’s really convenient to attempt to mount attacks, and clearly in this scenario we should allow principals to change their behaviour.

What I’ve seen nowadays in protocol verification papers is that we tend to give people a fixed role so if they’re good, they’re going to be good forever, and if they’re bad, they’re going to be bad forever. But, realistically, I might decide to change my mind and become a bad guy if that’s convenient. And on the other hand we may find the bad guys repent and change their minds. So we should allow for principals to change their behaviour. Certainly this notion of retaliation wouldn’t have made sense if we didn’t allow principals to change their behaviours.

So having said this, we all know those middle person attacks, where we see this middle person who can perhaps steal money from A’s account if the B’s are bad. However several people last year at the workshop showed that if this is possible, then retaliation is possible too; namely, B basically can retaliate against C, and steal money from C’s account in the case where C mounted the first attack. So this is an example of retaliation.

So that’s the general view, we have good guys, bad guys who are certainly trying to subvert the protocol, and ugly guys who normally conform to the protocol but might collaborate with the bad guys. Crucially, as I said, they can change their behaviour. One principal cannot be in two different sets, but can jump from one set to another as the protocol goes on.

We should all be acquainted with the notion of trace, it’s like the log, or history, of the network (a list of messages, or emails, or similar). Then we also have the notion of projection over a set of agents. The projection only includes the details of the trace that pertain to that set. An attack is some function of the trace, and of the people involved.