Generating Channel Ids in Virtual World Operating Systems (Transcript of Discussion)

George Danezis

Microsoft Research Cambridge

This is Michael Roe's work, but Mike has kindly asked me to present it so that I can be here today, and so that he can also present some different work tomorrow. You can ask questions, and check how well I understand the discussions we've mostly had by the coffee machine over the last years.

The world of gaming has changed since the 80s; a lot of us are familiar with pac-man, but things have progressed quite a bit. Games became more elaborate, then they went online, and as the games went online people started generating modifications of games, and there was a lot of creativity until today we've effectively stopped having games per se, and what we really have is online platforms, online communities.

The platform offers some basic services, but the content (the actual games being played online in the spaces) is generated by users or by third parties, and not the actual games platform itself.

So two key examples of this, Second Life and Metaplace, offer a virtual world, where you are represented by some 2 or 3-D character, and you go around and you interact with things that are created by other users. So in this setting we will see that there are some new security problems, but let's first offer some background with one of these examples, the oldest one, which is Second Life. So in Second Life you enter effectively a 3-D world that is hosted somewhere by Linden Labs, which runs the whole service, and each user is represented by a little 3-D character going around. Now there are details, there are two different places you can be, one for grown-ups and one for children, but since we're all grown-ups here let's assume you go straight for the adult part of the Second Life. And the world as it offered by Linden Labs is actually pretty boring, you know, there are some islands around, there is a sea, and there is this pretty basic boring person you are represented by; pretty much all the content (to a first approximation) is generated by users, which is amazing.

This is actually what made the web go crazy, because all content was not generated by the Tim Berners-Lee himself, but by everybody else, and the idea was that by allowing users to generate the content, including active content, which also contributed to the web becoming really cool rather than just static pages, it would become big. So effectively in this virtual world what is really important is that there are, in your impressions with the world, all the time three parties: yourself (represented by a client and a connection to the server), the actual server (which is the platform that runs all the logic), but then there are also the creators of the objects that you use. So if for example you have a car, you are the owner of that car but you are not the creator of that car, someone else has created the car, which involves the 3-D design, and the code that describes how this car goes around, and how you can race it with other people if it is a racing game, and so forth. And in this particular case, you are the owner, but you're not maybe allowed to do everything that goes in the car. Now you of course are attached to these objects, and sometimes these objects effectively encapsulate state that the owner should not really be allowed to interact with. So if we, for example, take a game, a shooting game, and we have two objects that facilitate the shooting game, one that represents effectively your level of energy left, and the other one that is a gun that you're allowed to shoot other people with, all the logic effectively in the state of the game object, despite the fact that you're their owner, is encapsulated in this object, and you shouldn't really be allowed to mess with it, otherwise you could cheat, for example, by aiming better, or by decreasing more energy from the other players as you're allowed to, or by increasing your own levels of energy, and so forth.

So how is this done in practice? What are the mechanics of having this kind of three parties effectively interact in this world? Everything happens through messages, all the objects in the world effectively communicate with each other on channels. So in the particular example of the shooting game that I described, the object that is the gun tells the object that is the energy of the other player, I have shot at you, reduce your energy level by ten, and if by the way it reaches zero, the player dies and goes back into the beginning of the game. And the players just buy these systems, buy these objects, and put them on effectively, and start running around shooting at each other without being able to modify their state. Now of course, as we understand, this messaging has to somehow be secure, because if anyone can write and read messages from these channels they would be able to pretend that they are shooting other players whereas actually they haven't, or they would be able to make other plays reset effectively and go to the beginning, and so forth. And in practice how is this done today on Second Life, it is done by using 32-bit channel identifiers, which are effectively just an integer. So if you can guess an integer, you can start communicating with these objects and modifying presumably through these channels that are secret state, and then doing all sorts of weird things to other players of these games.

Now so far this isn't a problem, because if you just have a relatively small number which has to be done online, so you cannot take it home and try all 2^{32} combinations, so it's kind of moderately hard but not impossible, but what mitigates really this problem is that for the moment the whole world is really run on the trusted clouds and servers of Linden Labs. But this is about to change. First of all it's going to change for scalability reasons: if the web were to only run on Tim Berners-Lee's servers probably it wouldn't be as big as it is today, so decentralisation would actually help scaling up. But there are also other reasons to run code on servers other than the servers provided by the infrastructure. One of them is policy: Linden Labs has very particular culturally-specific policies about how they interpret what pixels can do to other pixels, including games relating to adult themes, or gore, or violence, or whatever else, and they are under increasing pressures in all the jurisdictions, they are to change this according to the local customs, and so forth. So it would be a good idea not to have one big global infrastructure that manages all this, but actually allowed people to run their own local servers with their local games, and their local customs, as they wish. Actually the Linden Labs rules are a bit vague right now, and I'm not sure, maybe Mike you know, if they have actually planned particular games, but an interesting meta game would be to try to effectively build games that would get you back there.

Another reason why you might want to run your own servers is that Linden Labs servers give you very little control. So if you are McDonald's and you want to effectively create a virtual burger restaurant online, it is a bit embarrassing if you have to rely on this third party infrastructure that is slow and unreliable, and gives users a really poor experience, and you cannot fix it. And you cannot maybe have a VIP area where you can allow some people and not some other people depending on whether they pay you, or depending on whether they bought a Big Mac yesterday, and so forth. Similarly for authorisation, or if you are trying to have an intranet kind of virtual world as well, for example where only Microsoft employees can participate in meetings. You might want to run your own infrastructure for all these reasons.

Now what are the problems when you start decentralising the infrastructure where these things run? So far the security paradigm is that objects run code that contains secrets, and the particular secrets are the channel ids that these objects are communicating on. If now this code starts migrating onto untrusted platforms, and the source code reveals which channel ids are used to communicate between the different objects, it means that as soon as the code touches a rogue server, the channel ids will become known to the adversary, and then they can go to another world and start abusing them by sending messages that they shouldn't be sending. So we are not particularly concerned about what a rogue virtual world would do, because it can do anything that it wants. What we're really concerned about is that they learn information about third parties, objects and scripts, and then go off to the otherwise honest infrastructure, and abuse that information there.

So what is really our idea here is that, or our key security requirement is that, if you run your own infrastructure you should be able to get a lot of third party code, lots of third party content, and logic, and so forth, and then allowing your users to use it, but that shouldn't really give the adversary, if it is indeed the adversary that runs such an infrastructure, an advantage to go and violate security properties of code value on the honest infrastructure. And in particular, since we want to think of a way of keeping the same kind of security paradigm as Second Life has, namely objects communicating on channels as the main way of actually achieving secure communication, we better make sure that we won't have these kind of fixed channels ids that the adversary can learn and reuse to abuse effectively the objects somewhere else.

Now, this is opening a second problem here, which is that currently, because all the games are effectively hosted in one single place, players are allowed to roam around, and to effectively reuse their objects, and so forth. So if I have a gun and it has six bullets, which is the example here, and I shoot three of these bullets, then I have three bullets left. And this might be actually quite an important property in that a particular game designer might decide to actually charge me for bullets, so their particular business model might not be to sell me the gun, or to sell me the access to the game, but their business model might be that you have to pay for the bullets. So I would find basically I go and buy a bullet, it would give a unique id to the bullet that will be transmitted to the victim, and the victim will give it back to the server, and then make sure that the energy then goes down, otherwise it doesn't, I mean, there could be a scheme like this. And of course it would be really ideal if we decentralised the infrastructure effectively, and allowed users to run on different servers, to still maintain this property, but a simple thought experiment really allows us to see that this is very difficult. If I have an adversary server that is allowed to modify arbitrarily the state of objects (not following the logic that is embedded in the algorithm), how can I make sure that the state that comes from the adversary's servers is correct. This is a very difficult problem, and unless we go down the multi-party secure computation route, which is extremely expensive given the fact that we just want to have little avatar playing around, it is very difficult to accept state that comes from a different (untrusted) security domain into our servers.

So here for this solution that is being proposed. What we're saying is that effectively we're going to classify servers according to the security domain they're in. So we're going to say that some servers are effectively trusted within a particular domain, but effectively we're not going to allow any mutable state to go between the different clusters effectively of servers, effectively partitioning the world in terms of side effects, so any effects that exist in one world will not go anywhere else when it comes to access content. I mean, for images and such things, it's not such a big deal, except when it comes to intellectual property protection.

Bruce Christianson: Is there not a way out using proxies? As long as the mutable state itself doesn't actually move, why can't I just redirect the messages back so that they're always processed on the same server?

Reply: If you do different interfaces that might be the case. For reasons of speed the idea here is, and actually we have discussed exactly this architecture before, we could have an architecture where you effectively have the client, the virtual world server, and then a server that is a third party that manages and runs the scripts of the objects, and this is within the security domain of effectively the creator of the objects, so the guy who created the guns and the energy level indicators, and all that stuff. The problem there is that you automatically introduce a level of indirection that is a network operation, so if you fire a gun, you have to send a message to this third party saying, I fired my gun towards that direction, the other person will then have to check every so often, did anything hit me by the way, and then, you know, please decrease my energy; it could become very difficult to have that three-way indirection and still be able to play interactive games.

Bruce Christianson: I wouldn't advocate doing that generally, it's just that you just need to do it until the next time you reload.

Reply: So effectively have an audit, like have a trace that you then can give to this third thing, it is certified as being an honest trace, because it is indeed given the inputs and the outputs, the thing that the programme would run, and then go ahead, yes, that is maybe a way. And let's not forget, there are interesting sub-problems that may have efficient solutions, such as payment, right, I mean, so for some things it might actually be impossible.

Anil Madhavapeddy: The original Quake World when it first came out did a bunch of really good stuff on the basis that cheating was far rarer than actually just predicting stuff and redesigning it. So if the assumption here is that if there are too many attackers it would prevent the game being played, so if you did an audit on it that would actually provide quite a good incentive, right? Because let's say five seconds after I cheat the audit system catches up and exacts a penalty, or maybe it terminates the game if someone cheats, so there's no point cheating in the first place.

Reply: Yes, but let's not forget that these virtual worlds are not only about people shooting each other and getting hit. Sometimes you have objects that contain messages and such things, so you may have confidentiality issues also, because I might use it for having an online meeting within my company, because flying tends to be more expensive. Sometimes you actually have an integrity violation that might be difficult to reconcile later on as well. So it's not just availability and playability issue like in traditional gaming, these things are meant to be virtual world platforms for all sorts of interactions, so it's more difficult.

So the idea behind this protocol is really, how do we extract ids that allow very efficient user partitioning, and the actual technical scheme is rather straightforward. Each creator of objects effectively has a signing key and a verification key, and this is pretty much the same as for signing software patches. And of course, they use the signature key to sign any software components they give out as part of their objects, so the logic behind shooting, driving, and so forth. Then each cluster of servers within the same security domain actually share a symmetric key, KSi, and then what we say is that within that cluster any programme that communicates gets assigned effectively a security channel, so it asks for a security channel with a particular name, and then the actual id that this channel has is a hash of the signature key used to sign the programme, this is the multiplex across different provider effectively of code. The secret key of the cluster, this is to do multiplex between different security domains, and then the name, and this is to do multiplex between different channels within the object. And of course H is a key hash function, and actually it is the key that is KSi. And this way effectively we do multiplex across all these three dimensions, allowing each script to run and be tied in, and each id to be tied in per creator of script, per security domain, and per name of channel, so that we can actually use them for different things.

Now in practice this gives a really nice and simple interface for the game designers, because they don't have to care about any of the crypto, the only primitive they need to say is, could I please pull from the channel, fire bullet, and this is effectively the name, and everything else just happens behind the scenes, and at the time of the execution of the script, they don't have to care about anything else. Fire bullet is just the name of the channel, it's not the secret, so anyone that actually inspects the source code does not learn any information that could allow them to go then to another virtual world and be able to infer the channel ids that will be used there. Which means that additionally, since all of this binding happens at execution time, ...

Paulo Verissimo: But the time at which you fire the bullet should be a secret? Because, you know, if it's a game where you're trying to kill someone, if that is not a secret then ...

Reply: Yes, so what is still secret at this point (if the adversary has the KSi), is which is the correct key, the symmetric key that Macs can give as the channel id when this script is executed. So assuming that the adversary doesn't know that key, they will not be able to rewrite what is in some different channel, they will not be able to send a message on that channel on that server. Since the binding actually happens quite late, at the time of the execution of the script, this means that the KSi keys of the clusters are effectively private, and whether there is a key compromise, or a key rotation, and so forth, it can just happen transparently, it's just that people would have to just re-route all their channels and bring them up again, and everything else will just work fine. And this is the basic idea, and probably a lot of the questions will have to be redirected towards Mike, since it is his work, I'm just presenting it.

Bruce Christianson: This seems like a very nice generalisation of Li Gong's iCap system[1], where what he wanted was for capabilities to have different bit pattern representations in different domains, and that is precisely why caching the capability of the secret depends on the domain you were in. If you want to move from one server to another, do you have to reboot your channels, or is there a way that you can kind of push them through a firewall and have them mapped?

Michael Roe: The servers have their own virtual space, so you're never sending a message where the sender is on one server and the secret is on another, and so effectively across domains you have a separate instance, a new channel.

Reply: Migrating within the same security domain has a problem because the ids are the same, migrating across has a whole different set of problems as well, because then you run on state that is potentially corrupt.

Bruce Christianson: 32 bits doesn't seem like very much.

Michael Roe: So, easy to increase the number of bits. In Metaplace there's an arbitrary string.

[1] A Secure Identity-Based Capability System by Li Gong in Proceedings of the 1989 IEEE Symposium on Security and Privacy.

Bruce Christianson: Then it's not going to be sparse.

Reply: But just increasing the width is not sufficient, you also need some platforms to do with the multiplex, because if you rely on having a fixed, large, unguessable id, it's only going to be unguessable once the script runs on the trusted server.

Jonathan Anderson: This is also a usability improvement from the programmer's perspective, because then your time channel identifies to some sensible name like, the channel that I shoot bullets over, instead of a fixed concept.

Michael Roe: Yes, this is right. In Metaplace you can get yourself put in a slightly different security architecture perception to a certain amount. So the threat model is that the games don't trust each other, so one game wants protection, other games are nervous, the operators of server clusters don't trust each other, and operators of server clusters are usually only multi-party on a weaker service, so what you're really connecting to is someone who owns their own servers, then the secret is out of the range of using that user level access on somebody else's server to damage the integrity of the system.

Malte Schwarzkopf: I think the idea is that I, as the Second Life operator, get the source code and work out the channel ids, then I use that to write a book that goes around in another server, and it duly does something that benefits me in some way, kills people and takes their treasure.

Michael Roe: Yes, so that's supposed to be the protocol?

Bruce Christianson: But the crucial thing is that the person I'm shooting at is never on another server, right?

Reply: Yes.

Michael Roe: So you get to the second of these things called private islands where there's not so much sea as vacuum between you and the mainland, in other words, a part of the three-dimensional virtual space that nothing can get to.

Bruce Christianson: A cordon sanitaire.

Michael Roe: And so you have to have something similar to this, that when you're in a separate security domain you can't continuously move into some place where you would meet to compete in.

Bruce Christianson: My suspicion is this solution could actually be generalised to the multi-domain requirement.

Anil Madhavapeddy: So the whole virtual world context here is a red herring, you could apply this to a website, for example.

Bruce Christianson: This is the point, yes.

Anil Madhavapeddy: Yes, I mean, it sounds like it's essentially containing the threat.

Reply: I think that Facebook with its applications, sharing the same kind of security domain at the browser level, you know, similar things could be used there. You could be contained within the domain effectively, have a different key proved in and say, if you actually share a security domain you will end up with the same channel communicating with it.

Bruce Christianson: Or even if you just talk to a gateway that knows the translation protocol, then it will map you but enforce the security policies as it does so.