



# Stopping the bad guys: Toward a Server-Enforced Security Model

Given recent security exploits, it is time for Linden Lab to correct missing server-side security checks

by Gigs Taggart, Correspondent



Gigs Taggart (Jason Giglio) is an active member of the Second Life community as a scripter and contributor to the OpenSL project, land developer, and proprietor of GT HQ Store and Arcade.

Second Life software architecture [http://wiki.secondlife.com/wiki/Server\\_architecture](http://wiki.secondlife.com/wiki/Server_architecture) consists of an open source client viewer and back-end server code. Security checks are often written into the client code. Now that there are several client viewers—the official SL client, Windlight, and On-Rez, to name a few—these viewers may or may not share certain security constraints. This has resulted in serious security exploits.

When Second Life was a closed-source application, programmers often relied on the client to enforce certain constraints and security checks. This was always a risky security policy, as even a closed-source client can be modified. Now that Second Life (SL) is open source, this design is severely outdated.

Take the example of megaprims. Megaprims are prims (primitive objects in Second Life) larger than the normal 10-meter limit. While these are useful, they are a result of client software enforcing a 10-meter limitation on prim size – the server code does not enforce a limitation. Unscrupulous programmers can (and have) misused megaprims by setting unrealistic client-side security parameters. This was an early hint that the client-based se-

curity model has become obsolete. It is long overdue for an overhaul.

Other more recent examples of security exploits include: a) reading the source code to any LSL script regardless of ownership, b) reading a notecard without permissions, and c) taking free copies of any item straight from the contents of a vendor. This most recent exploit (fixed in the current version of SL) even allowed avatars to duplicate as many copies of a no-copy item as they wanted.

These exploits did not require any special knowledge to bypass basic security functions on the client. Even a beginning programmer can change a function named `is_this_allowed` to always return true.

Linden Lab developers are fully aware that client-side restrictions are not effective security. However, there does not seem to be much of a drive to eliminate the client-side checks, instead leaving them redundant with server-side checks. This is a risky strategy.

Quality Assurance (QA) testing at Linden Lab, for the most part, relies on testing the server using the official client. Other than LibSL and a few side projects, most open source developers and users are using the official client or some variant of it too. This creates a situation where a missing server-side check is difficult to detect. These missing security checks can go unnoticed for months or years because a redundant client-side check is covering it up.

Yes, most people won't be able to exploit the flaw because they won't even

notice it. But in reality this is a very bad situation because the few people that do discover the flaw may or may not be honest enough to report it. A flaw if not reported can be exploited over and over for years by an unscrupulous party. This is often called the "window of exposure" in security circles. A goal of software security is to reduce the duration of this exposure.

The window of exposure starts when the flaw is introduced in the software. Even though no one can exploit it yet, the flaw is there. Things escalate as soon as only one person discovers the flaw and figures out how to exploit it. The duration of time between the first discovery of the flaw and the final fix deployment is a critical segment of exposure. Having potentially undetected security checks in client code makes that segment of the window potentially very long. In other words, the "good guys" are less likely to find the exploit before the "bad guys" right now.

I urge Linden Lab to remove all client-side security enforcement. Always let the client attempt to do things that it knows are forbidden. The server can return an appropriate error code for the client to display. This is the only reasonable strategy to allow QA and user testing to be effective, and prevent these serious exploits from lingering in the server code for months or years. ☞