Securing the Hypergrid

Michael Heilmann M.S.C.S
University of Central Florida

Douglas Maxwell, Ph.D.
US Army Research Laboratory
September 29, 2016

Introduction

The idea of a formalized foundation to accept the responsibility of OpenSimulator is an interesting concept and gaining traction with the community. In another post, we outlined the rationale for why such a foundation is necessary for the continuation and longevity of the community. We mentioned security, but were pressed for more details after much discussion. We need to deal with a number of problems with the OpenSimulator from a security standpoint before it can become a viable platform for significant investment and support a robust economy. This article is intended for a technical audience; however, the case studies are written in such a way that a non-technical audience can relate.

We will discuss case studies that cover four major areas of vulnerability and possible remediation strategies. Case Study #1 is an overview of how content theft is accomplished and precisely where in the OpenSimulator the vulnerabilities lie. Content theft is a major problem and it will surprise you how easy it is to accomplish. Case Study #2 discusses something called a "User Trust Model" and how the hypergrid is broken in this regard. We discuss why it is risky for grid owners to allow unauthenticated avatars to come and go. Case Study #3 discusses how lazy or incompetent OpenSimulator administration can be a disaster for grid owners. In short, since there is no user authentication and the OpenSimulator's powerful scripting abilities are unchecked, a malicious visitor can easily drop code into your operating system and bring your grid down. Lastly, Case Study #4 discusses how easy it is to take your harvested content and launder it through the OAR and IAR system. You can remove all permissions and creator/owner fingerprints from content using the current archival methods. To take the discussion further, you can also open the OAR, insert your own malicious scripts, and repackage the OAR for an unsuspecting victim. In the conclusions I discuss one last vulnerability that should send a shiver down your spines…

In a future article, we will discuss the more theoretical attack vectors we have thought of. Mike and I think that just discussing the five basic ones are enough to get your attention for now.

It is our hope that this article will provide context to why the MOSES project is so reluctant to touch hypergrids and why InWorldz/Halcyon technology has not adopted. You will come away with an understanding of why we believe the only way to move forward is to fix these flaws. We believe the correction of the security flaws will also break backward compatibility.

WARNING: This article is controversial and pulls no punches. We discuss details about the vulnerabilities we have detected in the OpenSimulator. None of the vulnerabilities discussed in this document are theoretical. When reading this article keep this in mind: we have manually tested them all.

Hypergrid 2.0 Background

The hypergrid is a technology used by OpenSimulator to allow for visitation rights between several OpenSimulator grids. It has evolved and improved over the releases. Michael had originally set out to apply his background in architecture and security to improving the model through the judicious application of JSON Web Tokens and encrypted pipelines. We have reviewed the current implementation of hypergrid and have come to the following realization: Better encryption cannot improve hypergrid, because the hypergrid design is flawed. Hypergrid has a broken trust model.

Hypergrid had originally been released as 1.0, and has since been re-released as 1.5 and 2.0. The current hypergrid implementation, Hypergrid 2.0, has been in OpenSimulator since release 0.7.5. Hypergrid 2.0 brought many improvements, such as the idea of separating internal and external assets by running multiple databases, and restricting outside asset access to the “suitcase” folder in a user’s inventory. These are improvements, but they are only minor bandages on a large wound.

Hypergrid is designed to allow for the following interactions:

- A user without an account on your grid may enter and interact with your content.
- Depending on region permissions, that user may rez objects from other grids.
- Depending on region permissions, that user may take, or take copies of, objects from the destination grid.

In order to support this interaction, both grids must allow read/write access to their databases from each other's region processes. Even ignoring the issues of copybots, and rogue region hosts running malicious regions, a new attack vector is opened through these mutually granted permission.

Hypergrid access may be restricted based on source or target domain, and users may be individually blacklisted from the grid or from regions, but this is assuming that the domain remains trustworthy, and that the users are always acting in the best interest of the grid. Examples of bad actors include users that generate tremendous load, abuse the system, or abuse other users. We are proposing that this is a primary flaw in how hypergrid is implemented.

Case Study #1 – Content Theft

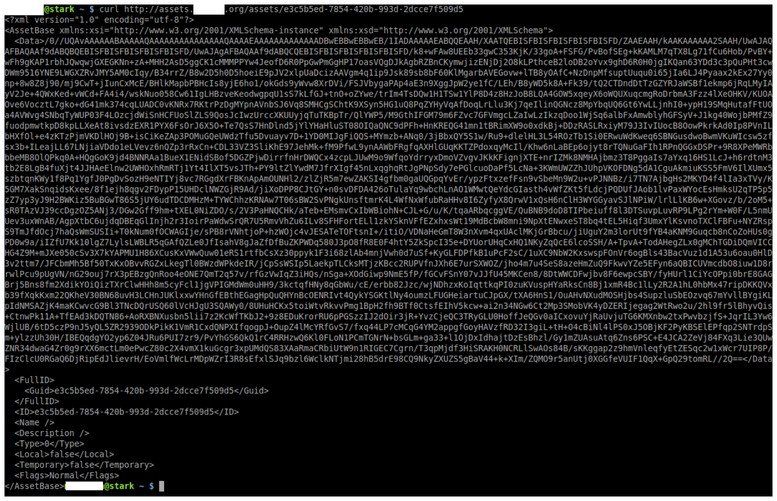
Content theft is at the forefront of every content creators mind. A user wishes to maintain their ownership over objects, over their creations, and most importantly over their sales. To support this, OpenSimulator has enacted permissions models that mimics Second Life. However every new function or feature is also a potential attack vector.

The normal asset pathway is for regions to serve all assets to all users themselves. This protects an asst behind two UUID codes. A UUID as OpenSimulator uses it is a randomly generated 128 bit identifier that hopefully never has a duplicate. The session Id is unique to the avatar session, but asset UUID codes are permanent.

Problem 1: The normal asset pathway is a mild protection, given that it is not clear. The pathway is not encrypted by default, so any client-server communications and any network interceptor can gain the session ID, and any asset IDs in the current region. Any mesh or texture can then be downloaded as long as the session ID is still valid.

The normal asset pipeline has poor performance, especially when users log connect with an empty cache. As it is a common fix for users to empty their caches for a number of bugs, users will connect regularly with an empty cache. The solution within OpenSimulator to fix this is to use the asset functions GetMesh and getTexture to serve assets from a process outside of the region. This dramatically improves the responsiveness of the region and is highly recommended.

Problem 2: The use of external services for asset service is dangerous. Regions do not share the session Id externally. These asset services, whether third party or built into the grid services cannot use the session Id, and only protect the assets through the unknown asset ID. An example to download a texture asset using readily available tools (this URL was taken from a Google-indexed forum post):



```
[redacted]@stark ~ $ curl http://assets.[redacted].org/assets/e3c5b5ed-7854-420b-993d-2dcce7f509d5
<?xml version="1.0" encoding="utf-8"?>
<AssetBase xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Data>/0//UQAvAAAAAABAAAAAQAAAAAAAAAAAAAAAQAAAAEAAAAAAAAAAAAAADBwEBBwEBBwEB/1IADAAAAAEABQQEAAH/XAATQEBISFBISFBISFBISFBISFD/ZAAEAAH/kAAKAAAAAA2SAAH/UwAJAQ
AFBAQAAf9dABQBQEBISFBISFBISFBISFBISFD/UwAJAgAFBAQAAf9dABQCQEBISFBISFBISFBISFBISFD/k8+wFAw8UEEb33gwC353KjK/33goA+FSFG/PvBofSEg+kKAMLM7qTX8Lg71fCu6Hob/PvBY+
wFh9gKAP1rbhJQwqwjGXEGKNn+zA+MHH2AsD5ggCK1cMMMPPYw4JeofD6R0PpGwPmGgHP17oasVQgDJkAgbRZBnCKymwjizENjDj2O8kLPthceB2loDB2oYvx9ghD6R0H0jgIKQan63YDd3c3pQuPHt3cw
DWm9516YNE9LWGXZRvJMY5AM0cIqy/B34rrZ/B8w2D5h0D5hoeiE9pJV2xlpUaDcizAAVgm4q1ip9Jsk89sb8bF60KlMgarbAVEGovw+lTB8yOAfC+NzDnpMfsuptUuqu0i65jIa6LJ4Pyaax2kEx27Yy0
np+8w8Z8j90/mj9CwT+jIunCxMcE/BHlkMapbPBHcIs8yjE6ho1/okGds9yWvw8XrDVi/FSJVbygaPAp4aE3n9XggJpW2ye1fC/LEh/B8yWD5k8A+Fk39/tQ2CTDndDtTzGZYRJaWSBf1ekmp6jRqLMyIA
yV2Je+4QWxKed+vWCd+FA4i4/wskNuo058Cw61IgLHBzveKeodwgpqU1sS7kLfGJ+tnO+oZYwe/trIm4TsDQw1H1TSw1YlP8D4z8HzJoB8LQA4GOW5xqeyX6oWQUXuqcmgRoDrbmA3Fzz4lXeOHKV/KUOA
Ove6VocztL7gko+dG41mk374cqLUADC0vKNRx7RKtrPzDgMYpnAVnbSJ6Vq8SMHCgSChtK9XSyn5HG1uQ8PqZYHyVqAfDoqLrLlu3Kj7qeIlinQGNcz8MpYbqUQ6Gt6YwLLjnhI0+ypH19SMqHutafFtUO
a4AVWvg4SNbqTyWUP03F4LOzcjdWiSnHCFUoSlZLS9QosJcIwzUrccXKUUyjqTuTKBpTr/QlYWP5/M9GthIFGM79m6FZvc7GFVmgcLZaIwLzIkzqDoo1WjSq6albFxAmwblyhGFSyV+J1kg40WojbPMfZ9
fuodpmwtkpD8kpLLXeAt8ivsdzEXR1PYX6FsOrJ6X5O+Te7QsS7HnDlnd5jYlYHaHluST08OIQaQNC9dPFh+HnKREQG41mn1tBRimXW9o0xdkBj+DDzRASLRxiyM79J3IvIUocB8OowPkrkAd0Ip8PVnIL
bHXfOl+e4zKTzPjmVKDlHOj9B+isCiKeZAp3POMuGQeUWdzTfu5Dvuayv7D+1YD0MIJgFiQQS+MYmzb+ANq0/3jBbxQY5S1w/Ru9+dlelHL3L54ROzTb1Si0ERwuWdKweq6SBNGusdwoBwmVKuWIcsw5zf
sx3b+ILeajLL67LNjiaVDdo1eLVevz6nQZp3rRxCn+CDL33VZ3SliKhE97JehMk+fM9PfwL9ynAAWbFRgfqAXHlGUqKKTZPdoxqyMcIl/Khw6nLaBEp6ojyt8rTQNuGaFIh1RPnQGGxDSPr+9R8XPeMWRb
bbeMB8OlQPkq0A+HQgGoK9jd4BNNRAa1BueX1ENidSBof5DGZPjwDirrfnHrDWQCx4zcpLJUwM9o9WfqoYdrryxDmoVZvgvJKkKFignjXTE+nrIZMk8NMHAjbmz3T8PggaIs7aYxq16HS1LcJ+h6rdtnM3
tb2E8LgB4fuXjt4JJHAeElnw2UWHOxhRmRTj1Yt4IlXT5vsJTh+PY9ltZlYwdM7JfrXIgf45nLxqghqRtJgPNpSdy7ePGlcuoDaPf5LcNa+3KWmUWZZhJUhpVKOFDNg5dA1CguAkmiuKSS5FmV6IlXUmx5
szbtqnKWy1f8Pq1YgfJ0PgDvSozH9eNTIYj8vc7RGgdXrFBKnApAmOUNHl2/zlZjR5m7ewZAKSI4gfbm0gaUQGpqYvEr/ypzFtxzefFsn9vSbeMn9W2u+vPJNNBz/i7TN7AjbgHs2MKYD4f4lIa3xTVy/K
5GM7XakSnqidsKxee/8f1ejh8qgv2FDypP15UHDclNWZGjR9Ad/jiXoDPP8CJtGY+n0svDFDA426oTulaYq9wbchLnAO1WMwtQeYdcGIasth4vWfZKt5fLdcjPQDUfJAob1lvPaxWYocEsHmksU2qTP5p5
zZ7yp3yJ9H2BWKiz5BuBGwT86S5jUY6udTDCDMHzM+TYWChhzKRNAw7T06sBW2SvPNgkUnsftmrK4L4WfNxWfubRaHHv8I6ZyfyX8QrwV1xQsH6nClH3WYGGyavSJlNPiW/lrlLlKB6w+XGovz/b/2oM5+
sR0TAzVJ39ccDgzOZ5ANj3/DGw2Gff9hm+tXEL0NiZDO/s/2V3PaHNQCHk/aTeb+EMsmvCxIbWBiohN+CJL+G/u/K/tqaARbqcggVE/QuBNB9doD8TIPbeiuff8l3DTSuvpLuvRP9LPg2rYm+W0F/L5nmU
Uev3uxWnAB/AgpXtbC6ujdqDBEqGlInjh2r3IoirPaWdwSrQR7U5RmvVhZu6ILv8SFHFortELl1zkYSknVFfEZxhxsWt19MdBcbW8mni9NpXtENwxeST8bq4tEL5Hiqf3UmxYlKsvnoTXClFBFu+NYZRsp
S9TmJfdOcj7haQsWmSUSIi+T0kNum0fOCWAGIje/sPB8rVNhtjoP+hzWOjc4vJESATeTOFtsnI+/itiO/VDNaHeGmT8W3nXvm4qxUAclMKjGrBbcu/jiUguY2m3lorUt9fYB4aKNM9Guqcb8nCoZoHUs0g
PD0w9a/iIZfU7Kk10lgZ7LylsLWBLR5qGAfQZLe0JfIsahV8gJaZfDfBuZKPWDq580J3pO8fR8E0F4htY5ZkSpcI35e+DYUorUHqCxHQ1NKyZqQcE6lcoSSH/A+TpvA+TodAHegZLx0gMChTGDiDQmVICC
HG4Z9M+mJXe050cSv3X7kYAPMU1H86XCusKxVWwQuw01eRS1rtfbCsXz30ppyk1F3i68zlAb4mnjVwh0d7uSf+KyGLFDPfkB1uPcF2sC/1uXC9NbW2KxswspFOnVr6ogBls43BacVuz1d1A53u6oau0HlD
3v2ttm7/JFCbmMh5Bf50TxKxOBvvRGZxLkegTl0BWzdWPkdeIR/jCpSsWSIp5LaekpTLCksMTjzKBcc2RUPVfnJXh6E7urSXWOZ/jho4m7u4SeS8azeHmZuQ9FkwvYZe5EFyn6aQBICUVmcdbO8iuw1D8r
rwlPcu9pUgVN/nG29ouj7rX3pEBzgQnRoo4eONE7QmT2q57v/rfGzVwIqZ3iHQs/nSga+XOdGiwp9NmE5fP/fGCvFSnY07vJJfU45MKCen8/8DtWWCDFwjbv8F6ewpcSBY/fyHUrl1CiYcOPpi0brE8GAG
Brj5Bns8fm2XdikYOiQizTXrClwHHh8m5cyFcl1jgVPIGMdWm0uHH9/3kctqfHNy8qGbWu/cE/erbb82Jzc/wjNDhzxKoIqttkqPI0zuKVuspHYaRksCn8Bj1xmR4Bc1lLy2R2A1hL0hbMx47ripDKKQVx
b39fXqkKxm22QKheV30BN68uvH3LCHnJUKlxxwYHnGfEBthEGagHpQuQHYnBcOENRIvt4QykYSGKtlNy4oumzLFUGHeiartuCJpGX/tXA6HnS1/OuAHvNXudMOSHjbs4SupzluSbEOzvq67mYvllBYgiKL
pIdNMSAZjK4maKCwvcG9Bl3TNcDQrUSQ60lVcHJgU3SQAWy0/8UHuHCKx5toiWtvRkvvPmg1BpH2fh9BTf0CtsfEIhV5kcw+ai2n34NGw6Ct2Mp3SMobVK4yDZERIjegag2WtRwo2u/2hl9fr5lBhyvQis
+CtnwPk11A+TfEAd3kDQTN86+AoRXBNXusbn5lii7z2KcWfTKbJ2+9z8EDuKrorRU6pPGSzzIJ2dOir3jR+YvzCjeQC3TRyGLU0HoffJeQGv0aICxovuYjRaUvjuTG6KMXnbw2txPwvbzjfS+JqrIL3Yw6
WjlUB/6tD5czP9nJ5yQL5ZR2939ODkPikK1VmR1CxdQNPXIfqogpJ+OupZ4lMcYRfGvS7/fxq44LP7cMCqG4YM2appgfGoyHAVzfRD32I3giL+tH+O4cBiNl4lPS0xJ5OBjKF2PyKBSElEPfqp2SNTrdpS
m+ylzzUh30H/IBEQqdgYO2yp6Z04JRu6PUI7zr9/PvYhGS6QkQ1rC4RRHzwQ6Kl0FLoN1PCmTGNrN+bsGLm+ga33+l1OjDxIdhajtDzEsBhzl/Gy1mZUAsuAtq6Zns6PSC+E4JCA2ZeVj84FXq3Lie3QUw
ZNR34dwaG4Zr0g9rXX6mctLm0ePwcZ80c2X4vmX1kuGcgr3xpUMdQS83XAaRmaCRbiUtW9n1RIGEC7Cgrn/T3qpMjdf3HiSRAKH0NCRLlSwAOs84B/sKKggap2z9hmVnleqfyEtZESqc2w1xWcr7UIP8P/
FIzClcU0RGaQ6DjRipEdJlievrH/EoVmlfWcLrMDpWZrI3R8sEfxlSJq9bzl6WclkNTjmi28hB5drE98CQ9NkyZXUZS5gBaV44+k+XIm/ZQMO9r5anUtj0XGGfeVUIF1QqX+GpQ29tomRL//2Q==</Data
>
  <FullID>
    <Guid>e3c5b5ed-7854-420b-993d-2dcce7f509d5</Guid>
  </FullID>
  <ID>e3c5b5ed-7854-420b-993d-2dcce7f509d5</ID>
  <Name />
  <Description />
  <Type>0</Type>
  <Local>false</Local>
  <Temporary>false</Temporary>
  <Flags>Normal</Flags>
</AssetBase>[redacted]@stark ~ $
```

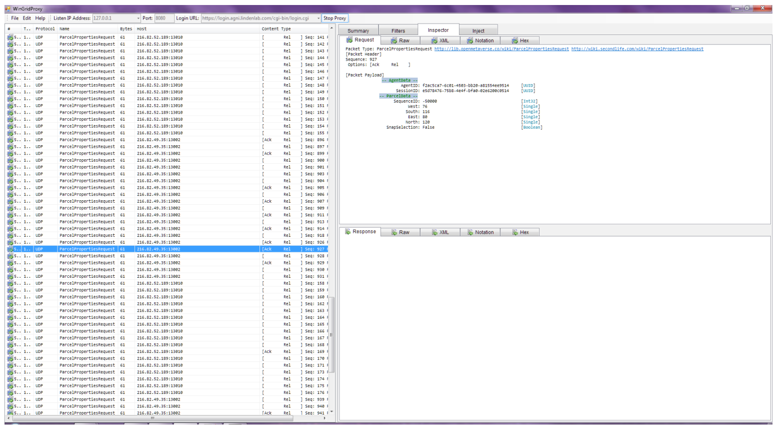
Figure 1. Screenshot of a Content Scraping Activity using a Command Line

Using the "curl" command, it is possible to access a target grid's asset database and copy content directly. No copybot or client required. An alternate command to use is "wget." This works in Linux, Windows, and MacOS. We redacted the identity of the target grid so as not to embarrass anyone.

The screen shot above demonstrates that in order to harvest a grid's assets, all you need is the IP address of the target grid and the UUID of the asset you desire to acquire. Someone may use hypergrid to hop to the target, use a harvester script to slurp up all the UUIDs of objects in a scene, save that information to a text file and then leave the region or grid. The unauthorized content harvest can be automated.

Problem 3: Unless region owners are prudent and well-informed, not all assets may have restricted permissions. A visitor may cause any permissive asset to be copied to their home asset service by taking it, or taking a copy. Assets are free to leave the grid.

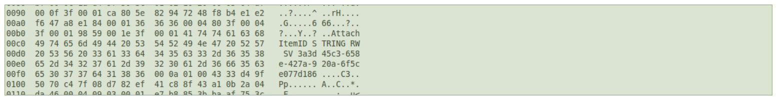
Problem 4: Even with appropriate permissions on, if a visitor can scrape asset IDs from their viewer or by script, asset textures and meshes can be downloaded at their leisure. This can readily be done by examining viewer caches, using a copy-bot enabled viewer, or a rogue viewer that spoofs its identity. All asset IDs are broadcast across the wire as well. Wireshark and WinGridProxy are easy-to-use applications that can record all of the messages between a local client and a remote grid.



```
Packet Type: ParcelPropertiesRequest http://lib.openmetaverse.co/wiki/ParcelPropertiesRequest http://wiki.secondlife.com/wiki/ParcelPropertiesRequest
[Packet Header]
Sequence: 927
 Options: [Ack     Rel    ]

[Packet Payload]
               -- AgentData --
                         AgentID: f2ac5ca7-6c01-4503-bb20-a81554ee9514     [UUID]
                       SessionID: e5d78476-75b8-4e4f-bfa0-02e6200c0514     [UUID]
              -- ParcelData --
                      SequenceID: -50000                                   [Int32]
                            West: 76                                       [Single]
                           South: 116                                      [Single]
                            East: 80                                       [Single]
                           North: 120                                      [Single]
                   SnapSelection: False                                    [Boolean]
```

Figure 2. WinGridProxy Session and Agent ID Surveillance and Capture.



```
0090  00 0f 3f 00 01 ca 80 5e  82 94 72 48 f8 b4 e1 e2   ..?....^ ..rH....
00a0  f6 47 a8 e1 84 00 01 36  36 36 00 04 80 3f 00 04   .G.....6 66...?..
00b0  3f 00 01 98 59 00 1e 3f  00 01 41 74 74 61 63 68   ?...Y..? ..Attach
00c0  49 74 65 6d 49 44 20 53  54 52 49 4e 47 20 52 57   ItemID S TRING RW
00d0  20 53 56 20 33 61 33 64  34 35 63 33 2d 36 35 38    SV 3a3d 45c3-658
00e0  65 2d 34 32 37 61 2d 39  32 30 61 2d 36 66 35 63   e-427a-9 20a-6f5c
00f0  65 30 37 37 64 31 38 36  00 0a 01 00 43 33 d4 9f   e077d186 ....C3..
0100  50 70 c4 7f 08 d7 82 ef  41 c8 8f 43 a1 0b 2a 04   Pp...... A..C..*.
```

Figure 3. Wireshark Network Surveillance of Asset UUIDs

When we discuss in public that network traffic is sent in the clear, this is exactly what we mean! It is even HUMAN READABLE. This needs to be locked down and encrypted immediately!

Problem 5: Briefcases can help a user prevent an external grid they visit from scraping all of their assets, but an abusive user could easily bring dangerous assets into MG using theirs. Any asset rezzed by a visitor is copied into the local asset service, and will persist after they have left. Assets can be injected into the grid. There is nothing stopping malicious assets with scripts to be injected into a grid.

Problem 6: Assets are not checked for duplicate uploads on an OpenSimulator grid. There is work in place to deduplicate an asset service to save disk space, but any asset that can be uploaded is treated as a new original asset.

Any asset that can be moved to a process outside of the grid operators' control should be considered lost, as any number of bad actors, or even OAR and IAR files, can launder the asset for reuse on a different, or even the same grid.

CASE STUDY #2 The Broken User Trust Model

Closed grids have the opportunity to limit and police their own user populations. Abusive avatars can be banned, and multiple re-registrations may be tracked and limited. But once hypergridding is enabled, users who are not vetted by any local policy are allowed into the grid.

Problem 7: Even if region permissions are locked down perfectly, and visitors may not build or rez from their briefcases, they are unrestricted from wearing malicious attachments or HUDS. One such abusive example from older viewers is a hat with multiple crashing web browsers on it.

Problem 8: As said by Justin Clark-Casey: “any version of OpenSim older than 0.7.6.3 should be considered unsafe.” This is due to a bug in how LSL treats network connections. An LSL script can make and receive HTTP connections from the machine that the region is running on, and they discovered that a well-crafted script could make calls against the grid services to alter user inventories and worse. This fix includes blocking HTTP calls to local IP addresses, to prevent users from scanning the network behind a grids firewall and discovering host IP addresses, database instances, etc. The specific user inventory fix was introduced in OpenSimulator version 0.8.1-rc2.

Problem 9: Speaking of networking calls from LSL scripts, any HTTP call made from an LSL script will be traceable to the host machine that it is calling from. A script can form a crude HTTP proxy, allowing your grid to participate in denial of service attacks, botnets, or a poor-man's tor. Unscrupulous or even illegal activity can take place and the grid host will be the first and primary suspect. Add to the mix that OpenSimulator LSL can allow for prims to receive networking calls or email traffic. This can happen from a worn attachment on a region with strict build permissions.

Problem 10: In addition to problem 8, where user actions will reflect poorly on the grid operator, these networking calls also allow scripts to perform calls against other grids. This can be used to bypass the domain white-listing feature of Hypergrid 2.0. It is not spoofing your IP address or domain, they are using it freely, as it is a feature.

Problem 11: As user creation is no longer completely under the control of the hosting grid, it is fact that abusive accounts can be created and utilized faster than any administrator can identify and ban them. An organized effort could overwhelm any hypergrid.

CASE STUDY #3 Lazy or Incompetent Administration

OpenSimulator grid administration is difficult and it is tempting to take shortcuts to get up and running quickly. To assist administrators there are many tools already in use to assist in user creation, region management, etc. Some of these are built into OpenSimulator. If they are activated, it can cause issues:

Problem 12: Allowing console commands to be called from scripts. By hiding a script deep inside of a linked set that is gifted to a user with appropriate permissions, a script may utilize the OpenSimulator console for many purposes. These could include overwriting all of the regions content by loading an oar file from a remote URL, overwriting local system files through a save oar call. You aren't running your regions as a user with administrative (operating system) rights are you, such as root?

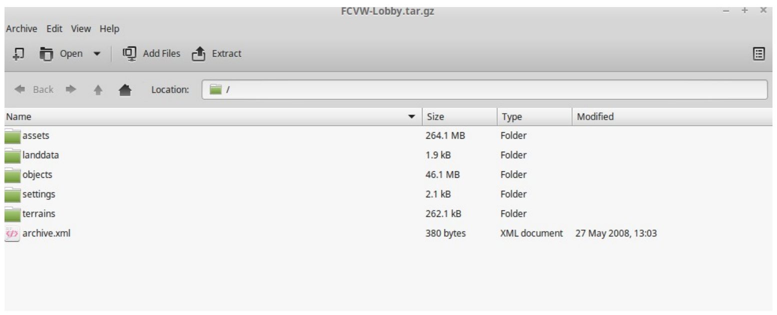
Problem 13: In lockstep with problem 11, the hidden script could easily kick or ban users.

Problem 14: A grid may also be configured to allow for commands to be called from a client. Anyone able to get the session ID of an administrative user could execute commands on the grid without stealing credentials or hacking in any other way.

Problem 15: Alternative scripting languages are powerful and fast. A quick Google search reveals how much faster a C# script is over an LSL one. However, C# and other languages are not API restricted. A C# script can read/write the local file system, open arbitrary network sockets and make arbitrary networking calls. Should an unknown user write objects to the file system and execute them – it is game over for your system's security.

CASE STUDY #4 IAR & OAR Content Laundering

One of the most popular features of OpenSimulator is that content can be removed and inserted at will using IAR and OAR files. While this allows for grid owners to archive, restore, and migrate content, it comes with its own problems:



Name	Size	Type	Modified
assets	264.1 MB	Folder	
landdata	1.9 kB	Folder	
objects	46.1 MB	Folder	
settings	2.1 kB	Folder	
terrains	262.1 kB	Folder	
archive.xml	380 bytes	XML document	27 May 2008, 13:03

Figure 4. Uncompressed and Exposed OAR File from the Federal Consortium for Virtual Worlds.

Problem 16: Assets can be included entirely, every texture, mesh, script, animation, sound, and note-card is included. The OAR and IAR file formats is simply a tar.gz compressed file. Contained in the file is a series of XML files containing object meta-data, as well as a folder containing all assets present on the region as separate files. These can be pulled out and used for other purposes.

Problem 17: User and Group accounts are not global, but only exist within each grid. When an OAR file is uploaded to a region, any user ID or group ID may not be found, and all ownership and permissions is assigned to the estate owner account. This is by design, but makes it trivial to not only move content between grids, but purge any permissions at the same time.

Problem 18: IAR functionality is protected by the users credentials, so a grid operator cannot normally save out a users inventory. However, it is a very small change in this open-source application to bypass that restriction, and allow a grid owner to literally download any users entire inventory. Hypergrid 2.0 attempts to prevent this by restricting hypergrid user inventories to their Suitcases, but it has to be configured to do so, and is only available in OpenSimulator 0.7.5 and later.

Conclusion

If after reading this article you still are not convinced the OpenSimulator needs a mature security model, there is one last bit of information we can share with you. Your IP addresses are exposed when you are logged into a grid and when you use the hypergrid. It is possible to to correlate your physical location with your IP address. You are not anonymous and it is easy to find you.

It is possible to create an open and distributed grid where users may interact with each-other freely, and trust that their intellectual property is protected. We cannot do that by ignoring what we actually need to protect. Each random grid is a walled garden and should not be trusted with the valuable intellectual property of another garden. We must create a larger garden that we can participate in together.

In our next article, we will propose a solution to these problems and explain how a formalized foundation could be the answer.