L3DGEWorld 2.2 Input & Output Specifications

Lucas Parry
Centre for Advanced Internet Architectures, Technical Report 071218B
Swinburne University of Technology
Melbourne, Australia
lparry@swin.edu.au

***Abstract*—This technical report briefly describes L3DGEWorld 2.2, a product of the L3DGE Project [1]. L3DGEWorld is a data visualisation tool based on the OpenArena derivative of the Quake III Arena game engine, being used in the monitoring and control of networks. The report describes the input interface specification for conveying information to the L3DGEWorld server for real-time visualisation and representation as a number of different visual characteristics, and the output abstraction layer through which actions are conveyed from within the virtual environment and made available to external output daemons to interpret and perform real world actions based upon.**

***Index Terms*—L3DGEWorld, Input, Specification**

I. Introduction

Network monitoring and control leveraging 3D game engines as a development platform, has previously been described by Harrop and Armitage [2] [3] [4]. In [2] a set of mappings were defined between in-game characteristics and network metrics to allow live network events to be displayed in a 3D world. A major goal of this work was to create mappings that allow in-game events to intuitively represent underlying network anomalies in real-time. [3] first used a 3D game engine to implement the characteristics defined in [2] and detailed technical descriptions of the prototype implementation and 3D game engine modifications appear in [4].

L3DGEWorld 2.2 is a stand-alone tool based on the GPL'd OpenArena 0.7.0 [5], a free open source game that uses the ioquake3 engine [6], which is in turn based on the GPL'd Quake III Arena (Q3A) game engine [7]. With the use of OpenArena's GPL'd textures and resources, we are able to freely re-distribute everything needed to use L3DGEWorld.

The input daemon provided to demonstrate L3DGEWorld 2.2 is a 'greynet' [8] (or in other terms a 'distributed sparse enterprise based darknet'). A greynet is a set of 'dark' passive listener hosts dispersed amongst 'lit' (normal) network hosts on an

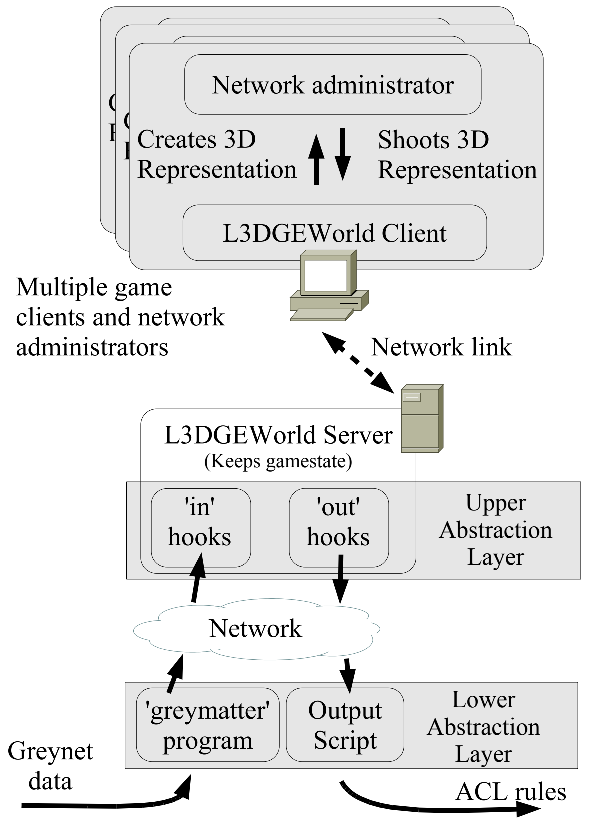

Fig. 1. L3DGEWorld abstraction layers to allow external input to influence world start and world actions to influence outside systems

enterprise network. When malware scans across the network attempting to detect vulnerable hosts, it not only scans real network hosts, but also alerts network administrators to its presence as greynet hosts report their incoming packets.

In the evolution of L3DGEWorld 2.2, we have continued to use a set of abstractions between the 3D game engine server and the underlying network devices as developed for L3DGEWorld 1.0, but as of version

2.2 we are using a UDP based protocol for both the input and output layers. Doing so allows us to "push" updates from input daemons to the L3DGEWorld server, and makes it simpler to have the output daemon run on a separate machine to that of the L3DGEWorld server.

Using this set of abstractions has allowed us to to develop the various components in a modular fashion, enabling the ability to write new input/output modules without additional modification to the 3D game server. Proof of this can be demonstrated in that L3DGEWorld 2.1 was adapted to display information about Swinburne's Supercomputer [9]. This shows the flexibility of our design, in that the L3DGEWorld server can easily be adapted for other uses, through feeding it alternate input data.

In this document we will describe the interface for communications between the upper and lower abstraction layers as previously defined in [4] and [10] and seen in Figure 1.

II. L3DGEWORLD 2.2

The L3DGEWorld 2.2 comprises of an input daemon, a L3DGEWorld client & server, and an output daemon. In our publicly released tarball we provide an example input daemon known as 'greymatter', a tool for monitoring greynets [8], and an output daemon known as 'LRCD' (L3DGE Router Control Daemon) for controlling Cisco routers.

The greymatter daemon generates statistics about traffic entering greynet hosts and sends these statistics to the L3DGEWorld server. The server adds this information to the world state, which is then is then automatically transferred (through core server/client functionality inherited from Q3A) from the server to all connected clients. Conversely, actions by clients are transferred to the server, and indication of the type of action and information known about the entity are sent to the output daemon. The provided output daemon listens for these messages, and used them to create ACLs on the router defined in the configuration file. These ACLs can be placed either via telnet or more securely over a direct serial link.

L3DGEWorld 2.2 has continued the modular design of previous versions, which allows new input and output daemons to be easily be written that are able to interface with other hardware and software.

Users of our L3DGEWorld 1.0 tarball saw pyramids representing the hosts of the associated greynet under scrutiny as in Figure 2. This was an improved version of an earlier prototype based on the 'Cube' game engine [4].

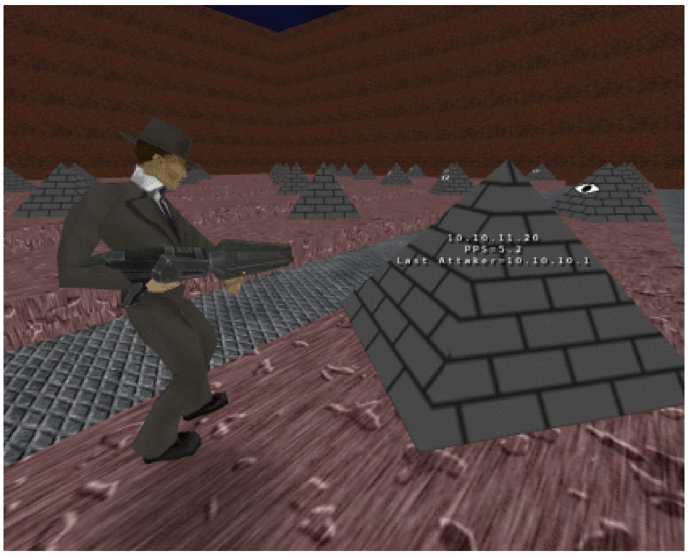

Fig. 2. Moving closer to a greynet entity reveals additional information in the form of a textual overlay.

In [4] an entity's spin rate was boolean, there was no indication of magnitude. In L3DGEWorld, using greymatter as the input daemon, each entity's spin is based on the packets per second that are destined for the IP they represent.

L3DGEWorld 2.1 went further by introducing 8 visual characteristics which could be applied to an entity in many different combinations to indicate all manner of meanings. Examples of some of the visual characteristics include 'scale', which causes entities to grow, 'bounce height', which causes entities to move up and down to varying heights and 'colour', which tints the entities textures with the specified colour. A full listing of the characteristics can be found in Table I.

Users of L3DGEWorld are provided with several tools with which they can administer their network, each of which has a different effect on the entity it is used to interact upon.

The interaction takes place in the form of "shooting" the entity with the selected tool. The tools can be used from a great distance away and are very accurate.

The first tool is the "ACL placer", which allows users to attempt to place ACLs by shooting a greynet host. The ACL will be block traffic from the host's last attacker, dependant on the host and users administrative weights (discussed below). The ACL action is depended on a setup where LRCD is the output daemon, other output

Characteristic No.	**Name**	**Description**	**Sensible Input Rates**
1	Spin Rate	Rotate the entity (rate/5) degrees each frame.	-200.0 to 200.0
2	Bounce Height	Linearly mapped from the rate, with smooth transitions between values. Rate of bounce is constant.	0.0 to 50.0
3	Scale	Entity is ((rate/10) times the normal entity size. Minimum size 1 times normal size, maximum size 4 times normal size.	10.0 to 40.0. Rates outside this range will give the same result as the corresponding limit.
4	Roll	When rate is 0, Entity is upright. When rate is 1, Entity is rolled 90 degrees. (Sideways) When rate is 2 or more, Entity is rolled 180 degrees. (Upside-down)	0, 1 or 2.
5	Colour	RGB value input as a 6 digit hex code. Note: The colour characteristic requires that entities are textured using shaders to have any effect. See 'baselw/scripts/l3dge.shader' for examples.	Six hexadecimal digits RRGGBB (without a leading 0x). Red, green and blue levels are separate 8-bit values, each contributing two hex digits from 00 to FF.
6	Bounce Rate	Oscillates with a period determined by the formula: $cos((time + 1000) * 0.00025 * rate)$ with smooth transitions between different inputs. Overrides bounce height (Characteristic 2). Height of bounce is constant when using this characteristic.	0.0 to 100.0.
7	Sound	Plays the entity's specified sound every 1 second when rate is not zero.	0 or 1.
8	Alternate Skin	Applies the entities alternate skin when rate is not zero. Overrides colour (Characteristic 5).	0 or 1.
9	Alternate Model	Applies the entities alternate model when rate is not zero. Overrides alternate skin and colour (Characteristics 5 and 8).	0 or 1.
0	Info Window	Not a visual characteristic. Name and value are used to populate the detailed info window. Name is displayed across the top of the window in blue and value fills the rest of the window.	No rate should ever be passed for characteristic 0.

TABLE I
TABLE OF CHARACTERISTICS, A DESCRIPTION OF WHAT EACH ONE DOES AND SOME SENSIBLE RATES

daemons can have completely different effects.

The second tool is the "Inspector". By shooting an entity with this tool a window will be overlaid on the users screen with more detail about the entity. In the case of greymatter this window will show the most commonly hit ports on the greynet host in the past 30 seconds.

The last tool offered in L3DGEWorld 2.2 is the "Host mover". When an entity is shot with this tool it disappears from the playing field and is then carried by the user. The user can take the entity to another location and drop it by pressing the use key.

L3DGEWorld also has a flexible multiple administrator permissions system. In version 1.0 the default setting required two administrators to agree that traffic is anomalous and shoot the same entity within a configurable period of time, for an action to be turned into a 'live' ACL on a router. In more recent versions administrators are assigned a configurable amount of 'administrative weight', and entities can be configured to require a particular amount of 'administrative weight' be used on them before actions are carried out. This allows a powerful administrator to carry out actions alone, or a number of less powerful administrators to work together to carry out the same action.

Users are able to pick up and move entities (using the 'host mover' to pick up, and pressing a key to drop), in order to arrange the representation of their network in a manner that is intuitive to them. These changes are persistent in the 3D world, meaning that the L3DGEWorld server may be shut down or restarted and the objects will retain their new positions.

Users are also provided with more detailed data as they require it. When a user comes within an arbitrary range of entities, additional information is shown in textual form overlaid above the entity. This information includes the configured IP or hostname, and the 'name'/'value' pairs for each characteristic.

Users are able to get additional information in the form of a detailed entity window. This can be accessed by shooting an entity with the 'host inspector' or by touching the entity (configurable via the console). This window shows the contents of characteristic 0's 'name'/'value' pair and is described in greater detail in section IV

III. Files used by L3DGEWorld

A. Directory Structure

All files related to configuration are stored within 'baselw' modification subdirectory of the main program directory. Files containing custom entity positions are stored within the 'hosts' subdirectory, which is within the 'baselw' directory.

The 'hosts' directory contains one directory per map, named the same as the map name it corresponds to. eg. 'hosts/l3dge' for the included map named 'l3dge'. Each of the map directories may contain files where the file name is the host's id number and the contents of the file are the coordinates for that host.

As of L3DGEWorld 2.2 all files that were previously used for communication with the input and output daemons have been replaced with a UDP based protocol (described in sections IV and V of this document).

B. 'l3dgehosts.conf' contents

The 'l3dgehosts.conf' file is used by L3DGEWorld 2.2 to assign names to in-game entities and define how much "administrative weight" is required to cause actions to be carried out on those entities. Entities without a specified name in this file will not be displayed in-game.

The file is also used by the provided input daemon, greymatter, which expects the name to be an IP address. This restriction only exists when using greymatter as the input daemon, and can be avoided by using two separate version of the file, one with names and the other with IP address.

Lines beginning with a '#' are treated as comments and ignored.

Lines beginning with a '[' are interpreted by greymatter as definitions of VLAN membership for the proceeding hosts. Such lines are in the format '[vlan 1]'. All hosts following such a line belong to the specified VLAN until another VLAN is defined. VLAN membership definitions are ignored by the L3DGEWorld game server.

All other lines assign an name/IP address with an entity ID number, and assign how much administrative weight is required to act on the entity. These are in the format '<Name> <IDNumber> <Weight>'. eg: '10.11.12.13 6 4' which configures entity 6 to refer to the greynet IP 10.11.12.13 and to require 4 units of "administrative weight". Weights should be whole integers.

An alternate hosts configuration file can be specified at runtime specifying "+set g_confFile <name>.conf" when starting L3DGEWorld.

C. 'l3dgeclients.conf' contents

The 'l3dgeclients.conf' file is used to allocate 'administrative weight' to the various administrators who are identified by IP address.
Lines beginning with a '#' are treated as comments and ignored.
All other lines should be in the format '<IP address>,<weight>'. It is also possible to specify a default weight to assign to administrators who do not appear in the file with a line in the format 'default,<weight>'. Weights should be integers.

D. 'allowedinput.conf' contents

The 'allowedinput.conf' file is used to list the IP addresses of hosts that are allowed to send status updates and feedback to the L3DGEWorld 2.2 server.
Lines beginning with a '#' are treated as comments and ignored.
Allowed IP addresses should be listed one per line.

When using 127.0.0.1 in the 'allowedinput.conf' file, it is important that the input daemon actually sends updates to the loopback address, rather than to the machines hostname or public IP.

E. 'outputip.conf' contents

The 'outputip.conf' file is used to specify the IP address and port for the listening output daemon. The IP should be in the form "127.0.0.1:12345"

F. 'branding.txt' contents

The 'branding.txt' file can be used to change the text contained in the on-screen branding towards to top of the screen. The file may contain up to 3 lines of text. If it is not present, the default text is used.

G. 'vm/tools' contents

The 'vm/tools' file can be used to enable extra tools for administrators (or conversely, remove access to certain ones). It should contain a number in hexadecimal of which each binary bit corresponds to a weapon. This number is used in game/g_client.c, and can be derived by summing together the hex numbers from Table II correlating to the tools you want enabled.

Tool No.	Hex Representation
1	0x002
2	0x004
3	0x008
4	0x010
5	0x020
6	0x040
7	0x080
8	0x100
9	0x200
10	0x400

TABLE II
TABLE OF TOOL HEX REPRESENTATIONS

H. Entity position files

The position files are used to store the coordinates of entities in the 3D game world, allowing persistent placement of objects. These files are not intended to be modified by a general user directly, but an advanced user may use to to reorganise a map using an external tool. They are created when an entity is moved and read by the L3DGEWorld 2.2 game engine on loading of a map. The files are created in "hosts/<map_name>" and are named 001, 002, ... with the filename correlating to the entity ID number. The format of these files is simply three floats separated by commas. eg. 'X.x,Y.y,Z.z'.

I. Models for L3DGEWorld Entities

There are 10 unique entities defined that map makers can place in their maps that the visualisation can be applied to. These are named in the generic pattern 'model01', 'model02', etc.

All files relating to an entity (except textures) are stored in corresponding folders, eg. 'models/l3dge/model01/', within the 'baselw' modification sub-directory. Each of these folders should contain the following files in order for all visualisations to work correctly.

- alt1.skin - Alternate skin that will be applied to indicate an entity has been shot.
- alt2.skin - Alternate skin that is applied for visual characteristic 8.
- alt.md3 - Alternate model that is used for visual characteristic 9.
- icon.tga - Icon used on the loading screen, and when carrying entities.
- model.md3 - The default model for the entity.
- sound.wav - The sound that will be played for visual characteristic 7.

By using a standard set of filenames for the files relating to each model, it becomes trivial to swap one model for another by simply renaming folders, or alternatively using symbolic links on platforms that support them.

Models textures should not be stored in the model folders, as this limits the ability to easily "swap out" one model's folder for another.

We provide one full example entity in the L3DGEWorld 2.2 tarball. More example entities may be released at a later date.

IV. INPUT LAYER PROTOCOL DESCRIPTION

As of L3DGEWorld 2.1, the input abstraction layer has been implemented as a simple UDP based protocol. Update messages are authenticated using a very simple handshake to attempt to verify their source, while feedback messages are trusted based only on the packets source IP address. To begin the handshake, senders request a token, the server checks if the sender is in it's list of allowed IPs and if so sends back a token to be prepended on all update messages. The complete input protocol is detailed below.

A. Requesting a token (server -> daemon)

A L3DGEWorld 2.2 server will only accept update messages from IP addresses listed in it's *allowedinput.conf* file. To protect against sending UDP packets with forged source address we require update messages to contain a unique 16 byte token that is allocated by the server.

To start the process of getting a token, the client sends a *gettoken* request to the L3DGEWorld server. This is a UDP packet, with payload beginning in `0xFFFFFFFF`, then followed by the ASCII text "`gettoken`". (Figure 3)

B. Receiving a token

When the server receives a *gettoken* message, it compares the senders source address with the ones listed in *allowedinput.conf*. If the IP address is matched, the server sends back a *tokenreply* containing the unique token for that IP address. This is a UDP packet, with payload beginning in `0xFFFFFFFF`, then followed by the ASCII text "`tokenreply:`", followed by the 16 byte token. (Figure 4) The sender must use this token in status updates to authenticate themselves.

C. Invalid Token Notification

If for any reason, the server receives a status update with an token that does not match the one stored for the source IP address, it ignores the update and sends

```
0000  00 00 03 04 00 00 00 14  f1 02 78 00 00 00 08 00   ........ ..x.....
0010  45 00 00 29 00 00 40 00  40 11 5e 4e 88 ba e5 80   E..)..@. @.^N....
0020  88 ba e5 80 80 57 6d 38  00 15 dc 9c ff ff ff ff   .....Wm8 ........
0030  67 65 74 74 6f 6b 65 6e  00                        gettoken .
```

Fig. 3. A hex and ASCII dump of an Ethernet frame containing a L3DGEWorld data source's *gettoken* request

```
0000  00 00 03 04 00 00 00 14  f1 02 78 00 00 00 08 00   ........ ..x.....
0010  45 00 00 3b 00 00 40 00  40 11 5e 3c 88 ba e5 80   E..;..@. @.^<....
0020  88 ba e5 80 6d 38 80 57  00 27 dc ae ff ff ff ff   ....m8.W .'......
0030  74 6f 6b 65 6e 72 65 70  6c 79 3a 7a 57 61 6f 4e   tokenrep ly:zWaoN
0040  6e 45 73 45 72 65 36 79  71 7a 30                  nEsEre6y qz0
```

Fig. 4. A hex and ASCII dump of an Ethernet frame containing a L3DGEWorld server's *tokenreply* response

an *invalidtoken* notification. This is a UDP packet, with payload beginning in `0xFFFFFFFF`, then followed by the ASCII text "`invalidtoken`". (Figure 5) On receiving such a notification, senders should cease sending and attempt to get a new token.

D. Update Messages

After the sender has obtained a valid token, it is able to send status update messages, prefixed with the token. These are UDP packets, with payload beginning in `0xffffffff`, then followed by the ASCII text "`l3dge` " (with a trailing space), then the 16 byte token followed by another space, then the update message. (Figure 6)

1) Update Message Format: Update messages are delimited by the tilde character '∼' and are in the format:

`∼<entity id>∼<characteristic number>∼<field>∼<value>∼`

In the Figure 6 we see an update message about entity 1, setting the name of characteristic 1 to the word spin. The three fields defined are *name* ('`n`'), *value* ('`v`') and *rate* ('`r`'). The *name* and *value* fields are used to construct labels which are overlaid on the entity when an administrator is in close proximity to the entity as seen in Figure 2. These labels are in the form: "<name>=<value>". The *rate* field is used to determine how the visual characteristic is applied to the entity. Some characteristics will have a finite number of specified states, while others will accept ranges of floating point numbers. A list of these can be seen in Figure I. There is nothing strictly tying the *name/value* pair of a characteristic to the rate of that characteristic, so a pair of labels set for a characteristic may bear no relationship to the *rate* field of that characteristic.

2) Default Characteristic Names: Setting the *name* field of a characteristic for the imaginary entity "0" will set the default name for that characteristic. If a entity does not have a name specified for the characteristic in question, it will fall back to the default name. This greatly reduces the required number of updates to set the name for every characteristic for every entity, and minimizes the amount of redundant data stored in the world state. If no default name exists then no label for that characteristic will be printed.

3) Detailed Info Window: Characteristic 0 does not correlate to any visual manipulations, therefore rate information should not be set for characteristic 0. The *name* and *value* fields of characteristic 0 are used to populate the detailed info window of the entity. The *name* field is displayed in blue below the hostname, and the *value* field is displayed below it in white.

The *value* text will be automatically line wrapped to fit the window, although no scroll function is available in the current version so it is up to the sender to keep the amount of text down to a size that fit in the window

To explicitly specify line breaks, place the '@' character where you want the line to break. This is used instead of the '\n' character to work around certain characters getting "lost" in L3DGEWorld's string tokenising routines.

An example of this can be seen in the packet captured in Figure 7, which will be displayed in the detailed info window as below:

Port: 25568 Hits: 8
Port: 18219 Hits: 18
Port: 12609 Hits: 2
Port: 56670 Hits: 9
Port: 33581 Hits: 15
Port: 42671 Hits: 7

```
0000  00 00 03 04 00 00 00 18  f3 29 60 6b 00 00 08 00   ........ .)`k....
0010  45 00 00 2c 00 00 40 00  40 11 5e 4b 88 ba e5 80   E..,..@. @.^K....
0020  88 ba e5 80 6d 38 80 58  00 18 dc 9f ff ff ff ff   ....m8.X ........
0030  69 6e 76 61 6c 69 64 74  6f 6b 65 6e               invalidt oken
```

Fig. 5. A hex and ASCII dump of an Ethernet frame containing a L3DGEWorld server's *invalidtoken* notification

```
0000  00 00 03 04 00 00 00 14  f1 02 78 00 00 00 08 00   ........ ..x.....
0010  45 00 00 46 00 00 40 00  40 11 5e 31 88 ba e5 80   E..F..@. @.^1....
0020  88 ba e5 80 80 57 6d 38  00 32 dc b9 ff ff ff ff   .....Wm8 .2......
0030  6c 33 64 67 65 20 7a 57  61 6f 4e 6e 45 73 45 72   l3dge zW aoNnEsEr
0040  65 36 79 71 7a 30 20 7e  30 30 31 7e 31 7e 6e 7e   e6yqz0 ˜ 001˜1˜n˜
0050  73 70 69 6e 7e 00                                  spin˜.
```

Fig. 6. A hex and ASCII dump of an Ethernet frame containing a L3DGEWorld data source's update message

```
0000  00 00 03 04 00 00 00 18  f3 05 fc a1 00 00 08 00   ........ ........
0010  45 00 00 bc 00 00 40 00  40 11 5d bb 88 ba e5 80   E.....@. @.].....
0020  88 ba e5 80 80 5d 6d 38  00 a8 dd 2f ff ff ff ff   .....]m8 .../....
0030  6c 33 64 67 65 20 74 62  69 59 66 39 61 54 74 6a   l3dge tb iYf9aTtj
0040  53 63 72 53 51 38 20 7e  30 30 38 7e 30 7e 76 7e   ScrSQ8 ˜ 008˜0˜v˜
0050  50 6f 72 74 3a 20 32 35  35 36 38 20 48 69 74 73   Port: 25 568 Hits
0060  3a 20 38 40 50 6f 72 74  3a 20 31 38 32 31 39 20   : 8@Port : 18219
0070  48 69 74 73 3a 20 31 38  40 50 6f 72 74 3a 20 31   Hits: 18 @Port: 1
0080  32 36 30 39 20 48 69 74  73 3a 20 32 40 50 6f 72   2609 Hit s: 2@Por
0090  74 3a 20 35 36 36 37 30  20 48 69 74 73 3a 20 39   t: 56670  Hits: 9
00a0  40 50 6f 72 74 3a 20 33  33 35 38 31 20 48 69 74   @Port: 3 3581 Hit
00b0  73 3a 20 31 35 40 50 6f  72 74 3a 20 34 32 36 37   s: 15@Po rt: 4267
00c0  31 20 48 69 74 73 3a 20  37 40 7e 00               1 Hits:  7@˜.
```

Fig. 7. A hex and ASCII dump of an Ethernet frame containing a L3DGEWorld data source's update message for the detailed info characteristic

V. Output Layer Protocol Description

As of L3DGEWorld 2.2, the output abstraction layer is now also implemented as a simple UDP based protocol. Output messages are authenticated using a simple handshake to attempt to verify their source. When LRCD is started, it begins by requesting a token from the L3DGEWorld server in the same manner as input daemons (token a). When the L3DGEWorld server has a action queued to send to the configured output daemon it checks if it has a token from the daemon already (token b). If not, it sends a token request to the daemon, which then replies with the token for the server (token b) authenticated with it's previously fetched token (token a).

A. Requesting a token (server -> daemon)

The LRCD daemon will only accept output messages from the IP address listed in it's *lrcd.conf* file. To protect against the sending of UDP packets with forged source address we require output messages to contain a unique 16 byte token that is allocated by the daemon.

This token request is the same as previously described for the input protocol, for both directions, server to daemon, and daemon to server. (Figure 3)

B. Returning a token

When LRCD receives a *gettoken* message, it compares the senders source address with the one listed in *lrcd.conf*. If the IP address matches, the server sends back a *outputtoken* message containing the unique token for that IP address, authenticated with a token previously fetched from the L3DGEWorld server. This is a

UDP packet, with payload beginning in `0xFFFFFFFF`, then followed by the ASCII text "`outputtoken `", followed by the 16 byte token FROM the L3DGEWorld server, a space, and then the 16 byte token FOR the L3DGEWorld server. (Figure 8) The server will then use this token in output messages to authenticate itself.

C. Action Messages

When an administrator instantiates an in-game action, the L3DGEWorld server communicates this to the output daemon using an *action* message. This is a UDP packet, with payload beginning in `0xFFFFFFFF`, then followed by the ASCII text "`action ~`", followed by the 16 byte authorisation token, another delimiter (the tilde character '~'),a sequential message number followed by another delimiter, the host id of the host being acted upon followed by another delimiter, the weapon id used to instantiate the action followed by yet another delimiter, then a string containing all of the name/value pairs that are set for the host, each seperated with a '\n', terminated with a final delimiter.

D. Action Acknowledgement

When an output daemon recieves an Action Message, it needs to send an acknowledgement message, otherwise the L3DGEWorld server will continue to retransmit the action. This is a UDP packet, with payload beginning in `0xFFFFFFFF`, then followed by the ASCII text "`ack `", followed by the 16 byte authorisation token followed by a space then the message number we are acknowledging.

E. Feedback

It is often desirable that the output daemon be able to send feedback back into the virtual world to indicate to users that an action has taken place, or alternatively to notify them that the action failed. For this we have defined simple feedback messages. These messages are only allowed from IP addresses listed in 'allowedinput.conf', but no handshaking is done to verify senders. When the server receives a feedback message is displays it "as is" at the top of the screen for all connected users. Feedback messages are UDP packets, with payload beginning in `0xffffffff`, then followed by the ASCII text "`feedback `" (note the trailing space) followed by the feedback message as ASCII text. (Figure 11)

VI. L3DGEComms Library

To make developing different input and output daemons simpler, we have created a C library that abstracts the communications channel, and takes care of token requests/re-requests for the developer. To set up the communications channel a call to *setupComms()* is made, passing the IP of the remote server to establish a channel with, the remote port number, the local IP address and local port number. Using NULL for local IP/port will use the default IP and a random port number. To send update messages a call to the function *sendMessage()* is made, passing the host ID, characteristic number, a letter representing field type and the value for the field. *closeComms()* takes care of tearing down the channel.

LRCD, greymatter and fakematter all make use of this library, and can be inspected to provide further detail on how it can be used to write other input and output daemons.

VII. Example Attack Scenarios

The following is an example attack, detected and acted upon using L3DGEWorld 2.2, greymatter and monitorhosts.sh, running on our testbed described below.

A. Testbed Configuration

The L3DGEWorld testbed is shown in Figure 12 and consists of a Cisco 7140 router (running IOS 12.3) and 3 standard PCs. One interface on the router has two sub-interfaces created in VLANs 10 and 11. The PC representing the attacker is placed on VLAN 10, the other two PCs, representing the Greynet listener and the L3DGEWorld 2.2 server are placed on VLAN 11.

The second interface on the 7140 is placed on Swinburne's 136.186.229.0/24 network, representing the Internet. PCs on the "Internet" running the L3DGEWorld client (and with appropriate routes in their table) are able to connect to the server and monitor and control traffic routed between the two VLANs.

B. In-game Perspective

We first outline the attack from the perspective of the in-game network administrators.

1) Two administrators 'in-game' monitor the unmoving greynet entities (as in Figure 2).
2) Greynet entities with logically close IP addresses to each other within the map begin to spin.
3) The network administrators move closer to one of the entities and receive textual information explaining precisely the packets per second and the attacking host the greynet host is detecting.
4) The first network administrator decides that the event they are seeing is indeed a malicious network anomaly that needs to be prevented.

```
0000  00 00 03 04 00 00 00 14  f1 4a 7c 00 00 00 08 00  ........ .J|.....
0010  45 00 00 4e 00 00 40 00  40 11 3c 9d 7f 00 00 01  E..N..@. @.<.....
0020  7f 00 00 01 1b 39 6d 38  00 3a fe 4d ff ff ff ff  .....9m8 .:.M....
0030  6f 75 74 70 75 74 74 6f  6b 65 6e 20 63 54 56 34  outputto ken cTV4
0040  76 4c 4c 31 4b 66 79 75  7a 42 6e 44 20 6b 4f 77  vLL1Kfyu zBnD kOw
0050  6e 77 67 68 42 53 49 4d  50 77 58 6f 62 00        nwghBSIM PwXob.
```

Fig. 8. A hex and ASCII dump of an Ethernet frame containing an output daemon's *outputtoken* response

```
0000  00 00 03 04 00 00 00 40  63 cb 56 e5 00 00 08 00  .......@ c.V.....
0010  45 00 00 ca 00 00 40 00  40 11 3c 21 7f 00 00 01  E.....@. @.<!....
0020  7f 00 00 01 6d 38 30 39  00 b6 fe c9 ff ff ff ff  ....m809 ........
0030  61 63 74 69 6f 6e 20 7e  72 58 67 41 50 4e 69 37  action ~ rXgAPNi7
0040  53 44 6c 44 44 6d 32 4d  7e 36 7e 31 7e 32 7e 70  SDlDDm2M ~6~1~2~p
0050  70 73 3d 35 30 2e 36 0a  6e 75 6d 62 65 72 20 6f  ps=50.6. number o
0060  66 20 61 74 74 61 63 6b  65 72 73 3d 31 0a 6e 75  f attack ers=1.nu
0070  6d 62 65 72 20 6f 66 20  75 6e 69 71 75 65 20 70  mber of  unique p
0080  6f 72 74 73 20 68 69 74  3d 33 32 33 0a 70 72 6f  orts hit =323.pro
0090  74 6f 63 6f 6c 20 70 65  72 63 65 6e 74 61 67 65  tocol pe rcentage
00a0  20 62 72 65 61 6b 64 6f  77 6e 3d 69 63 6d 70 3a   breakdo wn=icmp:
00b0  30 20 74 63 70 3a 39 39  20 75 64 70 3a 30 0a 6c  0 tcp:99  udp:0.l
00c0  61 73 74 20 61 74 74 61  63 6b 65 72 3d 31 30 2e  ast atta cker=10.
00d0  31 30 2e 31 30 2e 31 30  0a 7e                    10.10.10 .~
```

Fig. 9. A hex and ASCII dump of an Ethernet frame containing a L3DGEWorld server's *action* message

```
0000  00 00 03 04 00 00 00 00  0c 07 ac e5 00 00 08 00  ........ ........
0010  45 00 00 37 00 00 40 00  40 11 3c b4 7f 00 00 01  E..7..@. @.<.....
0020  7f 00 00 01 30 39 6d 38  00 23 fe 36 ff ff ff ff  ....09m8 .#.6....
0030  61 63 6b 20 71 77 6b 35  4f 59 33 54 79 6f 51 54  ack qwk5 OY3TyoQT
0040  4c 7a 70 6e 20 36 00                             Lzpn 6.
```

Fig. 10. A hex and ASCII dump of an Ethernet frame containing an output daemon's *ack* response

5) The first network administrator shoots one of the spinning greynet entities. It turns yellow to indicate it has been acted upon (as in Figure 13) and stays yellow to indicate more administrative weight is required.
6) The second network administrator agrees on the course of action and shoots the same entity.
7) A message is displayed to both users informing them of a successful block being placed against the attacking host and the shot entity returns to a grey colour.
8) All greynet entities slow and then stop rotating, as further malicious scans on the network from that source have been prevented.

C. Technical Perspective

We now outline the same attack from a technical perspective of L3DGEWorld's underlying systems:

1) L3DGEWorld 2.2 and its support programs greymatter are launched.
2) greymatter registers with the L3DGEWorld server and receives a token.
3) In the L3DGEWorld all greynet entities are stationary.
4) An 'attack' (port scan) is launched from the attacking host at IP address 10.10.10.10 directed at the entire 10.10.0.0/16 network.
5) The greymatter program, listening passively to a number of IP addresses in the 10.10.10.0/24 space, detects the scan.

```
0000  00 00 03 04 00 00 00 16  47 4a e2 a3 00 00 08 00   ........ GJ......
0010  45 00 00 5f 00 00 40 00  40 11 3c 8c 7f 00 00 01   E.._..@. @.<.....
0020  7f 00 00 01 80 19 6d 38  00 4b fe 5e ff ff ff ff   ......m8 .K.^....
0030  66 65 65 64 62 61 63 6b  20 41 43 4c 20 62 6c 6f   feedback  ACL blo
0040  63 6b 69 6e 67 20 74 72  61 66 66 69 63 20 66 72   cking tr affic fr
0050  6f 6d 20 31 30 2e 31 30  2e 31 30 2e 31 30 20 68   om 10.10 .10.10 h
0060  61 73 20 62 65 65 6e 20  70 6c 61 63 65 64 00      as been  placed.
```

Fig. 11. A hex and ASCII dump of an Ethernet frame containing a L3DGEWorld output daemon's feedback message "ACL blocking traffic from 10.10.10.10 has been placed."

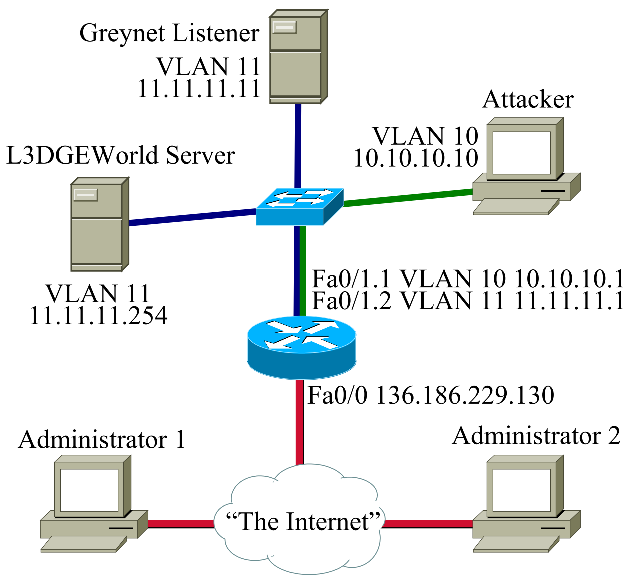

Fig. 12. Testbed Configuration

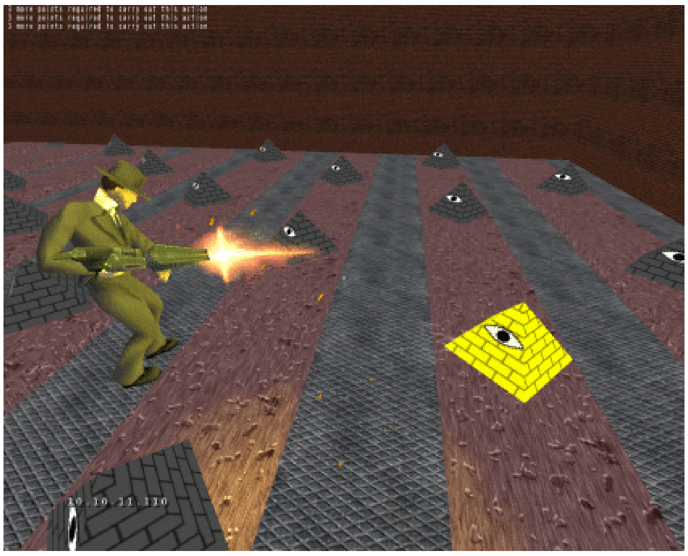

Fig. 13. A collaborating network administrator marks a greynet for ACL placement.

6) greymatter sends UDP updates setting the pps and last attacker for each greynet entity.
7) The L3DGEWorld 2.2 server modification receives these updates and sets them in the world state.
8) The server sends out these changes to all connected clients in the next world update.
9) L3DGEWorld clients receive the changes, and manipulate the corresponding entities according to the characteristic set (ie. Spin).
10) Two network administrators 'in-game' each shoot a spinning greynet entity within a short period of time.
11) The server tracks this, and writes out 'action' and 'infodump' files for the entity.
12) The output daemon finds the action file and sets an ACL on the configured Cisco router.
13) The attacking machine can no longer send packets onto the network, so greymatters pps for each host falls to zero.
14) greymatter sends UDP updates zeroing the pps for each entity.

VIII. Conclusion

This technical report has described L3DGEWorld 2.2, a standalone data visualisation tool based on the OpenArena game engine. Used in combination with the sample input and output daemons, it allows users to perform network monitoring and control on a live network from within the 3D virtual world.

We have detailed our updated interface specifications for conveying generic information from an input daemon into the game engine for real-time visualisation and representation, and we have described how the output abstraction layer can be used to perform real world actions based on in-game events.

We have provided examples of how L3DGEWorld 2.2 may be used as a tool for monitoring greynets and placing ACLs, but this only scratches the surface of what is possible.

IX. Acknowledgements

L3DGEWorld is being developed is under the direction of Grenville Armitage and Warren Harrop. This project has been made possible in part by a grant from the Cisco University Research Program Fund at Community Foundation Silicon Valley

References

[1] "The L3DGE Project," August 2007, http://caia.swin.edu.au/urp/l3dge.

[2] W.Harrop and G.Armitage, "Intuitive Real-Time Network Monitoring Using Visually Orthogonal 3D Metaphors," in *Australian Telecommunications Networks & Applications Conference 2004 (ATNAC2004)*, Sydney, Australia, December 2004.

[3] ——, "Real-Time Collaborative Network Monitoring and Control Using 3D Game Engines for Representation and Interaction," in *VizSEC'06 Workshop on Visualization for Computer Security*, Virginia, USA, October-November 2006.

[4] ——, "Modifying first person shooter games to perform real time network monitoring and control tasks," in *5th Workshop on Network System Support for Games 2006 (Netgames 2006)*, Singapore, October 2006.

[5] "OpenArena - A Completely Free Game for the FOSS Quake 3 Engine," August 2007, http://openarena.ws/.

[6] "ioquake3," August 2007, http://www.ioquake3.org/.

[7] id Software, "Doom 1, 2, Quake 1, 2 and III," July 2006, http://www.idsoftware.com/.

[8] W.Harrop and G.Armitage, "Defining and Evaluating Greynets (Sparse Darknets)," in *IEEE 30th Conference on Local Computer Networks (LCN 2005)*, Sydney, Australia, November 2005.

[9] Carl Javier, "LCMON," August 2007, http://caia.swin.edu.au/urp/l3dge/tools/lcmon/.

[10] L. Parry, "L3DGEWorld 2.1 Input and Output Specifications," CAIA Technical Report 070808A, August 2007, http://caia.swin.edu.au/reports/070808A/CAIA-TR-070808A.pdf.