A Physical Channel in a Digital World

Michael D. Deffenbaugh, Daryl Johnson, Bo Yuan, Peter Lutz
Rochester Institute of Technology
Rochester, NY
mddeff@zeroent.net,{daryl.johnson,bo.yuan,peter.lutz}@rit.edu

***Abstract* -- Gaming has always been a very prevalent part of our society. There is a long history of using various board games as methods of covert communication, from what we could call "differential" communication (i.e. the movement of pieces constitutes the "language") to the idea of "game state" communication (where the current state of all the manipulatable objects in the game world represents a message). For many years we have seen covert channels in games like chess, checkers and other simple board games and even evolution to more "complex" computer games like magnetron. Enter the 21st century and the advent of the high performance personal computer.**

Keywords – Covert Channel; Computer Game; Garry's Mod; Valve; Gaming

I. Introduction

In the 80's, 90's and early 2000's we see amazing advances in computer games both in the aesthetic/graphical aspects of computer games but also in the types and complexities of game play, including the associated physics engines [1]. One of the biggest advances we see is in the client/server model where more of the interaction is handled by the server, but the rendering is done client side. The development of this technology happens around the same time when we start to see more and more powerful personal workstations (likely not by coincidence). The gaming companies utilize this and migrate to a client-render model and this opens up a litany of opportunities for covert channels.

As Lampson defined back in 1972, a covert channel is one that is not intended for communication [2]. Covert channels are nothing new however there hasn't been a lot of development into game-based channels. This is not the first occurrence of using video games as a method for covert channels. Sebastian Zander, Grenville Armitage, and Philip Branch authored a paper [3] in 2008 titled "Covert Channels in Multiplayer First Person Shooter Online Games". In this paper, they detailed the use of the Quake 3 engine as a means of implementing a covert channel. Their channel relied on making minute modifications to player's movements (imperceptible to the "unwitting players" but detectable by client side programs).

II. Channel Overview

This covert channel utilizes a computer game called "Garry's Mod" (leveraging the Valve Source physics and graphics engine). Garry's mod is a "sandbox" game that allows users to spawn props and construct various inventions. An example is that a player can construct a car that they can then drive around the map. The Garry's Mod "paradigm" even allows users to create functioning electronics (using an add-on called "WireMod") that allows them to affect various in-game objects.

Covert channels span from physical channels such as leaving classified documents in a trash bag under a bridge to digital ones like embedding information in TCP checksums. This channel merges the two worlds. It is a digital covert channel in that it occurs in the digital world utilizing computer programs and network protocols however in the game it occurs as a "physical" channel (in that the player/sender creating the messages is manipulating physical game pieces).

This paper will discuss in detail the idea of using Garry's Mod in particular as a covert channel. We will provide an example language but it should be noted that this is one of many that can be used. Garry's Mod has a lot of different ways one could utilize various game mechanics to create a channel.

III. Garry's Mod In-depth

Before we delve into the nitty-gritty of how Garry's Mod is to be used as a covert channel, we first must understand how it is to be used as a game.

Each player has the game client installed on a local system and when they start up the game they can browse for servers hosting various instances of the game on the internet. Both companies as well as private individuals host servers for various purposes/game types. A large portion of the servers setup for the Garry's Mod community are "build" servers, meaning they are just there for people to come and build various contraptions, in the true spirit of Garry's Mod. There is a large variety of other servers created for specific purposes such as Role Playing Game servers (where players take on various rolls such as shop keeper, and have inventions, such as a shop, to augment the "reality" created) or space-build servers (where players must build contraptions that will provide life support, propulsion, etc. in a simulated space environment). For the purposes of this discussion, we will focus on general build servers which are

not only arguably the most prevalent type of server, but also the one that places the fewest restrictions on player actions.

Once a player enters a server they can start spawning props to construct whatever they want, whether it be some invention they thought of, or in our case, a covert message. They have the ability to spawn whatever prop or props they want and then are provided with a set of tools to either constrain props together (using methods such as "welding" two props together or binding them together with "rope") or attach forms of propulsion to them (adding motors and wheels, or adding vector-based thrusters). This leaves the player limited by almost nothing but their imagination.

While there is an entire ecosystem of modifications (mods) that can be added to Garry's Mod, the most prevalent is "WireMod" [4]. As described above, WireMod is a set of props that can have values or do simple (or complex) computations. The makers of WireMod have even developed a language called Expression2 [5]. A Perl/Gnu-C like language, Expression2 allows players to program some of the more complex chips to run their inventions.

An example of WireMod is where a player could build a car that, when it detects it's about to flip over, it fires a thruster to correct itself (via a programmed chip linked to a thruster on-board the car). This has even gone to the extent where we have people who have made their own operating system atop computers that interact in a client-server model with-in Garry's Mod (termed "ConOS" [6]) completely constructed out of WireMod components.

IV. The Covert Channel

The proof-of-concept was developed to demonstrate that this game could in fact be used as a means for a covert channel. The "example language" that was developed utilizes the "color" tool within Garry's Mod to alter the RBGA(red, blue, green, alpha) values of a specified prop; in this case, a 55-gallon drum. The encoding method is accomplished by encoding a character into its ASCII decimal value (0-255) as one of the RGB values of the prop. As such each prop can hold 3 characters. So for example if you had one prop with RGB values of 87, 105, and 110 respectively, this would be translated to "Win".

Ordering is handled simply by the order in which they were spawned (i.e. the first prop spawned is the first "packet", second prop is the second packet, and so on). The advantage to using this ordering method is that it is something that the message author doesn't have to worry about, as they spawn containers, they are in proper order. The main disadvantage is that if one needs to change part of the message, the message effectively needs to be re-entered/re-encoded.

The final part is channel noise, determining what is in the channel and what is not. The alpha value is used to determine if a prop is in the channel or not. The alpha value of the prop (for game purposes) dictates how translucent the object is (an alpha value of 0 is completely transparent; an alpha value of 255 is completely opaque). Essentially you specify a "key" to be used for a specific alpha value, and any prop with said value is in one's channel; however, this method comes with some inherent downsides.

If a random player (not participating in the covert communication) happens to set one of their props to the specific alpha value used as the key for the covert channel, their prop (and its RGB values) will appear as in the channel and as such be decoded by the client and added to the message. This is less likely to happen with sufficiently random values for alpha as alpha values tend to either be exactly or close to 0/255. Choosing something relatively random (and not on a round number, i.e. not ending in 0 or 5) should eliminate most if not all of the noise from that source.

The other source of noise is a little more difficult to work around. Through our testing we noticed an interesting behavior in the Source Engine. It seems that when you have a gib (a piece of scrap from a larger prop that was destroyed) that is on fire (say from a wooden crate or flammable barrel), as the client turns to face away from said gib, the client side renderer has the gibs alpha fade from 255 to 0, thus causing the gib to temporarily "enter the channel". As the client for this language is constantly evaluating the current game server and all of the props inside and displaying its results live to the client's screen, a reader of the channel would notice some artifacts in the message. It should be noted that when the client turns to face the gib that is on fire again (to bring it back into view), the effect is reversed and it fades (quickly) from 0 to 255. The flame effects themselves can also produce random alpha values (thus sometimes falling into the channel).

In the demo video, available at http://file.zeroent.net/mddeff/pub/gmcc/, we see that the user takes 2:18 seconds to create the message "Covert Channel Example", encode it into ASCII decimal values, and then affecting the properties of the 55-gallon drums. That roughly translates to 0.159 Bps or 1.275 bps. Garry's mod allows a player to make a contraption off-line then effectively "copy and paste" it into the current server. It would allow us to create a message off line and then paste it into the server. With preloading the message, we were able to load and paste a 53 byte stream in 2 seconds, this works out to be 26.5 Bps. The downside to this is that there is more preparation required off-line.

A. Advantages

As with every covert channel or covert channel, we must evaluate its ability to function effectively.

Noise in a covert channel is defined as the amount of data the falls into the parameters of the channel that was not intended to be part of the covert communication. In the case of the Garry's Mod covert channel, channel noise ends up being other players and other interactions in the game. However it all depends on what the language is defined as. For example, if your language is if a player spawns a certain prop at a certain time, that means a certain message; noise in that channel would be every time any player spawns any prop that wasn't intended to be part of the channel.

However with the proof-of-concept language discussed in this paper, the channel noise is any time a player modifies the default color properties (RBGA values) of a prop and happens to hit the specific alpha value, thus putting it in the channel.

Due to the sheer amount of Garry's Mod servers running at any time, it is not uncommon for a single player to be in a server "playing" by themselves. Normally it is the mark if an ineffective covert channel if it causes highly observable traffic, however in this case, even though the traffic is observable, it still appears normal.

Furthermore, the concern of having the channel be conspicuous to other non-covert channel users is to some extent a non-issue. Even if you have a player in the game who is doing nothing more than spawning barrels and changing the color (like the proof-of-concept language), other players or server administrators won't typically question what they are doing. They may not be viewed as terribly competent players, but no ulterior motives will be assumed. This allows us to operate almost any type of channel within Garry's Mod. As Harry Chriss (see acknowledgements) said, "Nothing is too weird for Garry's Mod."

While there are literally hundreds of Garry's mod Servers running at any time (see table 1), any average user has the ability to create a server at any time that other users can join. This allows the message sender to setup servers with specific settings that may aid in the particular language they have developed.

Table 1 - Survey of Garry's Mod Servers [7]

Date/Time	Servers	Players	Slots
5/22/2012 04:59	288	219	5283
5/23/2012 00:01	271	205	4978
5/24/2012 00:48	274	237	5068

Another property of this covert channel is that as long as the server stays up (and depending on the server configuration), any storage base channel can turn into a dead-drop style message. Some of the limitations on the dead-drop concept include the fact that many servers employ "prop cleanup" scripts where after a certain amount of time of a user disconnecting from the server, their props are automatically removed. That being said, if the message sender is running their own server, they can remove this limitation.

Depending on the language chosen, a Garry's Mod covert channel can either act as a timing based channel or a storage-based channel. For the purposes of this paper, we are utilizing the definitions as described by the Trusted Computer Security Evaluation Criteria (TCSEC). The proof-of-concept channel is a storage-based channel as it a form of communication by "modifying a stored object", in this case, a 55-gallon drum prop. An example of a timing based channel is where you have a player who sets off a series of explosions in the game to represent "S.O.S." in Morris code.

Observability is another advantage in using the Valve Source graphics/physics engine. The game client and server communicate in such a way that the client knows about every event that transpires on the server at any given instant. This means that if Player A spawns a barrel and colors it a certain color, all other players **in that server** know that just happened. Even though the actual user might not be able to see my barrel on their screen from across the map, each gaming client has the knowledge that it exists and in what state it exists. This is advantageous in that it doesn't require the receiver of the message to be in physical proximity (in the game) to the sender. This can even be more advantageous as when one does a "video capture", a client is essentially taking a recording of every event that is transpiring in the game (a player jumping, another one firing a gun, and yet another spawning a prop). When this "video" is played back, the client simply re-renders the set of instructions just as if it were getting them from the server.

B. Limitations

Unfortunately there are some significant limitations to the practicality of this channel. The biggest limitation of the channel is the large client-side requirements (for those sending and receiving the message). Given that both the sender and receiver of the message must have the Garry's Mod client installed. The requirements of which include a relatively high end graphics card and a decent CPU, this limits the types of systems that can be communicating in the covert channel [8].

One of the other limitations is that as of now we have not developed a way to programmatically create or transmit a message. This means that in order to send a message, a user must manually log into the game and create the message. This currently is the biggest limitation in the bitrate of most of the channels created for the GMCCF, in that the speed of the channel is based largely on the speed of the player's ability to manually encode the message. That being said, the reading/decoding of the message has been automated and example LUA script can be found at http://file.zeroent.net/mddeff/pub/gmcc/init.lua.

Another limitation that we run into is that there must be a hosted dedicated server for us to log into and transmit the message on. And if one doesn't exist that suits the needs of the particular language, then one must be created. Another note to mention about server requirements is that there is a server-side variable set called "sv_pure". This variable determines if clients are allowed to run client side scripts (in this case, our reader lua script), if set to 0, we can run client side scripts, if set to 1, we can't. Almost all of the Garry's Mod servers have sv_pure set to 0 however it technically isn't guaranteed and as such must be acknowledged as a limitation of the channel.

The largest limitation that should be noted is one that is more about the practicality of using this channel. This channel has the potential for transmitting a lot of data at once, that being said,

given the requirement of a custom gaming client using custom TCP ports, it requires that the transmitting party have the ability to either have access to a system with the Garry's Mod client already installed (not very likely in a secured environment where information would need to be exfiltrated from) or the sending party would need to have the ability to install it to one of the local computers (would be rather non-covert).

Furthermore, the transmitting party would need to be able to actually have their client's game traffic leave the network and hit the server being hosted on the internet. A lot of companies block common gaming ports inbound and outbound simply for loss of productivity reasons, and as such this presents another limitation. In summary, any covert channel leveraging Garry's Mod requires that both the sender and the receiver have some modicum of control of the systems as well as the network in between their client computers and the server hosting the game instance.

V. Additional language concepts

There are a few other ideas that are worth more research to determine the best method for communicating using Garry's Mod.

One method that was tested was having a time sensitive storage-channel where the message was time sensitive in that it was only viewable for a certain amount of time before it self-destructed, using the explosive barrels as the prop. It was insinuated that if the reader was too close to the message when it self-destructed (exploded), it would be considered a "volatile" message, as it would likely kill the player.

A variation of the proof-of-concept language was where each of the props "z-axis" value, or "elevation" determined which prop or "packet" came in which order. As the Garry's Mod physics engine allows the player to arbitrarily freeze a prop in any x, y, z, yaw, pitch, roll orientation, one could develop a language where the highest elevated prop is the first "packet" in the message, the second highest prop is the second packet, and so on.

Another covert channel could be stenography within the maps themselves. Some advantages include the fact that it does not require the actual existence of a server. The sender in this case would author a map that contains the message within the map, and then the reader would download it (either by joining a server running the map) or by downloading it out-of-band and then running it locally. The most prominent disadvantage includes the fact that it requires a different map to be created for each message that is sent. Map creation is no simple task and the time it would take to properly make a map and encode a message in it would be very lengthy, however could store upwards of gigabytes of data. It should be noted that computer game map stenography is not limited to Garry's Mod and could be accomplished with any reasonably current video game.

VI. Conclusion and Future work

The development of the Garry's Mod Covert Channel (GMCC) shows us that new types of covert channels are emerging very rapidly. We can expect to see a lot more research done not only in the traditional gaming arena but in anywhere where there is a content-rich user experience. Possibilities for future work include the creation of an API to programmatically manipulate in-game objects as well as the development of other languages and performing a differential analysis as to which language is the most effective as a covert channel within the GMCC.

VII. Acknowledgements

We would like to give special thanks to a personal friend, Harry Ethan Chriss. He was instrumental in implementing the proof-of-concept language discussed in this paper and was always there to bounce ideas off of.

References

[1] Valve, "Valve Source Engine," [Online]. Available: http://source.valvesoftware.com/. [Accessed 24 5 2012].

[2] B. W. Lampson, "A Note on the Confinement Problem," *Communications of the ACM*, vol. 16, no. 10, pp. 613-615, October 1973.

[3] S. Zander, G. Armitage and P. Branch, "Covert Channels in Multiplayer First Person," in *Local Computer Networks*, Montreal, Canada, 2008.

[4] Wiremod Community, "WireMod Wiki," [Online]. Available: http://wiki.wiremod.com/wiki/Main_Page. [Accessed 24 05 2012].

[5] "Syranide", "Expression2 - WireMod Wiki," WireMod Community, 5 4 2012. [Online]. Available: http://wiki.wiremod.com/wiki/Expression_2. [Accessed 24 5 2012].

[6] "LooperNor", "ConOS Wire OS," Youtube, 5 7 2010. [Online]. Available: http://youtu.be/9ET8jqOfIYM. [Accessed 24 5 2012].

[7] M. Deffenbaugh, "Garry's Mod Server Stats," 24 5 2012. [Online]. Available: http://file.zeroent.net/mddeff/pub/gmcc/server_stats.xlsx. [Accessed 24 5 2012].

[8] Valve, "Garry's Mod," [Online]. Available: http://store.steampowered.com/app/4000/ . [Accessed 24 5 2012].