A Behavior-Based Covert Channel in a MMO

Brian Rowe and Daryl Johnson
Department of Computing Security
Rochester Institute of Technology
Rochester, New York USA
bxr9458@rit.edu, daryl.johnson@rit.edu

***Abstract*—This article proposes a behavior-based covert communication channel in World of Warcraft (WoW), a popular Massively Multiplayer Online Role Playing Game (MMORPG), to transfer data between 2 clients. This is done by modulating binary information onto the senders cast routine and capturing this routine on the receiver. This routine, while demonstrated in WoW, could also be applicable to any other MMORPG in existence. WoW has the advantage of a large user base, so common actions will draw very little attention.**

I. Introduction

Lampson first described covert channels in *A Note on the Confinement Problem* in 1973 [6]. He noted the difficulty of confining a program, so that it cannot communicate with any program besides its caller. Whether by shared resources, storage, or piggybacking on legitimate information, it is extremely difficult to prevent all inter-process communication. He called these illegitimate channels covert channels. Since then, these covert channels have been broken into two main groups with a proposed third: storage, timing, and behavioral. A storage channel is one in which a sending process communicates by directly or indirectly writing to a storage location where the receiving program can directly or indirectly read it, whereas a timing channel sends information by modulating system resource utilization so that the receiving program can observe this and derive information [5]. A third type, a behavioral channel, is one in which behavior patterns are modified to communicate a message between parties [4]. This last channel is more difficult to detect than a storage or timing one because an individual would have to know and understand a units normal behavior to be able to detect the irregularity, or modulation, resulting in a plethora of potential arenas in which to secretly and inconspicuously leak information. This article explores using an MMORPG as just such a carrier.

The idea of using a game as a covert channel is not a new one. In 2008, Zander, Armitage, and Branch proposed the idea of covert channels in first person shooters [9]. They described communication by encoding pitch, yaw, x, y, and z coordinates of a character to send encoded messages to a receiver which would then have to record and decode this information. An even simpler example, a state game called Magneton created in 2009, allowed players to send, save, and receive messages by passing a predetermined authentication pattern into the game [4]. This subsequently allowed the following moves to either send or receive a message between other clients. Later in this article, the authors brought up the concept of covert channels in an MMORPG, mentioning that it would be difficult to identify the clients and even harder to then examine them [4]. An MMORPG is a game environment in which large numbers of people can interact both with each other and with the game environment. These worlds create a different type of carrier to transfer information by using seemingly normal actions. The most popular MMORPG at this time is World of Warcraft. In an official press release from Activision Blizzards Earnings Call (a quarterly report detailing company finances) for the period ending December 31, 2012, this game had approximately 9.6 million paying subscriptions [1]. According to their website, WoW "is an online game where players from around the world assume the roles of heroic fantasy characters and explore a virtual world full of mystery, magic, and endless adventure [2]." This world containing millions of human players creates many normal environmental interactions that could be modulated to transfer information between clients. One example is explained below.

II. Method

World of Warcraft has many different ways of interacting with the environment that could be modulated to covertly relay information between a sender and receiver. This paper focuses on using the combat system and a targeting/training dummy as the communication channel within the legitimate channel of the game itself.

World of Warcraft has training dummies in every major in-game city (see figure 1). A training dummy allows a player to perform spells or melee attacks (depending on the class of the character as chosen by the user) against this opponent to test attack sequences or rotation for Player vs. Environment (PVE) encounters (also known as player versus computer). Within the game there are many kinds of spells that can be cast. Two of these, the Frostbolt and Ice Lance spells, have been chosen to represent the binary digits 0 and 1 to encode messages. They were chosen because they are "no cool down" spells which means they can be used repeatedly without a waiting period between the casts. This is the medium that will be modulated for the covert channel.

First, a Perl script called conv.pl, written by Ivo on cool-commands.com [3], is used to convert the input message into binary code. Then, AutoHotKey [7], a free open source automation, hotkey, and scripting language, is used to compile .ahk script files into .exe executable files. The script 1ahk.ahk (see figure 2) modulates a binary 0 to the press of the 1 key,

Fig. 1. World of Warcraft Training Dummy

the script 2ahk.ahk (see figure 3) modulates a binary 1 to the press of the 2 key. The WoW.ahk script (see figure 4) activates a running WoW process bringing its window to the forefront. Then a third spell, the 3 key, is used to modulate the beginning and the end of the sequence.

```
#NoEnv
; Recommended for performance and compat
; with future AutoHotkey releases.
; #Warn
; Enable warnings to assist with detecting
; common errors.
SendMode Input
; Recommended for new scripts due to its
; superior speed and reliability.
; Ensures a consistent starting directory.
SetWorkingDir %A_ScriptDir%
Send 1
```

Fig. 2. 1ahk.ahk AutoHotKey script to modulate a 0

```
#NoEnv
; Recommended for performance and compat
; with future AutoHotkey releases.
; #Warn
; Enable warnings to assist with detecting
; common errors.
SendMode Input
; Recommended for new scripts due to its
; superior speed and reliability.
; Ensures a consistent starting directory.
SetWorkingDir %A_ScriptDir%

Send 2
```

Fig. 3. 2ahk.ahk AutoHotKey script to modulate a 1

To begin the sender and receiver must be logged into their respective characters in the World of Warcraft game. The receiver begins by issuing the command /combatlog in the chat box to enable WoWs built-in combat log to begin recording the actions and events in the environment around him/her to Program Files\World of Warcraft\Logs\WoWCombatLog.txt (see figure 5). The environment that is recorded by the combat

```
#NoEnv
; Recommended for performance and compat
; with future AutoHotkey releases.
; #Warn
; Enable warnings to assist with detecting
; common errors.
SendMode Input
; Recommended for new scripts due to its
; superior speed and reliability.
; Ensures a consistent starting directory.
SetWorkingDir %A_ScriptDir%

IfWinExist World of Warcraft
{
    WinActivate
}
```

Fig. 4. WoW.ahk AutoHotKey script to activate a WoW process

log is limited to a 200 yard range surrounding the player. This is a hard coded limit and cannot be changed by the player. The receiver needs to be within the 200 yard proximity - at the same time and on the same server - as the sender for the log to capture the senders actions. The sender then invokes the sender.pl script (see figure 6) which uses the conv.pl script to convert the ASCII message into a binary string. Sender.pl then calls upon 1.ahk.exe and/or 2.ahk.exe to send the binary 0's and 1's in the converted string by generating the corresponding key presses. The receiver logs out of the game when the animation ceases indicating that the message is complete. The combat log file is save locally. The receiver then executes the receiver.pl script (see figure 7) against the combat log file. This converts the Frostbolt spell back into a binary 0 and the Ice Lance spell back into a binary 1. The receiver.pl then call upon conv.pl to convert the binary sequence back into an ASCII string.

III. Limitations

Limitations still exist in this system and the biggest in this example is bandwidth. A general formula for throughput is (60 secoconds) divided by (cast time) equals casts per minute (bits). Depending on whether a 0 (in-game Frostbolt) or 1 (in-game Ice Lance) is sent for this demonstration, the cast

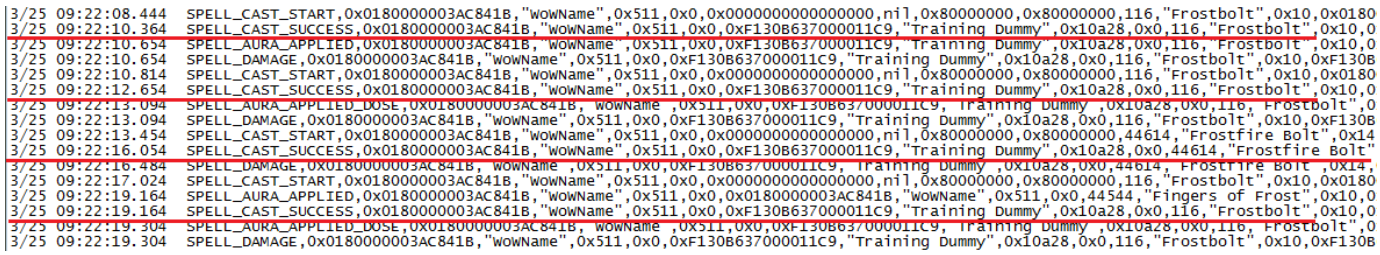

Fig. 5. World of Warcraft Combat Log

```
#!/usr/bin/perl

#getting input from user
#converting into binary string
print "Enter phrase to be transmitted\n";

my $ui = <STDIN>;
chomp($ui);
my $string = `perl conv.pl -b \"$ui\"`;
print "The binary string is " . $string;
chomp($string);
@array = split(//, $string);

#call wow.exe which brings WoW to the foreground
system("WoW.exe");

#foreach in array, if the binary translation is a 0 press 1
#                  and if a 1 press 2 waiting for in-game cooldown appropriately
foreach (@array)
{
        if($_ == 0) {
        print "A 0\n";
        system("1ahk.exe");
        select(undef,undef,undef,1.9);
        }
        else {
        print "A 1\n";
        system("2ahk.exe");
        select(undef,undef,undef,1.5);}
}
```

Fig. 6. sender.pl script

time for 1 bit of information is either 1.9 or 1.5 seconds, respectively. This limits the throughput to a best case of 40 bits per minute (60 seconds per minute divided by 1.5 seconds per cast), a worst case scenario of 31 bits per minute (60 seconds per minute divided by 1.9 seconds per cast), with a median of 35 bits per minute. The National Institute of Standards and Technology [5] labels this as a low bandwidth channel because it transmits less than 100 bits per second.

There are three classes of covert channel bandwidths: high, medium, and low. While there are no commonly agreed upon numbers for these ranges, their uses are clear. Document and image exfil/infiltration typically will require a high bandwidth channel. If the channel can be used over an extended period of time, a medium bandwidth channel would suffice. Messaging and botnet command and control work best with at least a medium bandwidth channel. Low bandwidth channels can be used for signaling if prolonged channel usage is possible. If the communication is highly encoded that would reduce the bandwidth required. For example, using a dictionary of 255 words would only use 8 bits per word. Low bandwidth channels modify the environment the least and consequently offer the potential for the greatest stealth.

While this channel is considered low bandwidth, it is still not insignificant as coordinates and encryption keys are still only a few bits to a few hundred bits in length. Logically, sensitive information could still be leaked very easily through this channel as important information can be worth the wait. This channel would also be a candidate for a botnet command and control channel.

A second limitation is that both clients have to pre-negotiate spell translations, character names, and a training dummy meet-up location to facilitate the communication. A scheduled meet-up time is also beneficial to prevent drawing unwanted attention, even though it isn't necessary for communication as the receiver can capture logs indefinitely until he runs out of local storage space.

```
#!/usr/bin/perl

#open input wow combat log
 $binstring='';
open ( INPUT, "<C:\\Program Files (x86)\\World of Warcraft\\Logs\\WoWCombatLog.txt" )
        or die "File does not exist!\n";
print "The printed text is:\n";

#change to match your sending character's name
$character="SecretSquirrel";

#foreach line of input from the combat log, if regex matches output the resulting 0 or 1
#then once the string size is 8, convert back to ascii

foreach (<INPUT>)
{
        chomp($_);

        if ($_ =~ m/SPELL_CAST_SUCCESS/ && $_ =~ m/$character/)
        {
                if ($_ =~ m/Frostbolt/) {       $binstring .='0';       }
                elsif ($_ =~ m/Ice/) {          $binstring .='1';       }
        }

        if(length($binstring) == 8)
        {
                $ascstring = `perl conv.pl -t \"$binstring\"`; chomp($ascstring);
                                $binstring=''; print $ascstring;
        }
}
print "\n";
```

Fig. 7. receiver.pl script

A final limitation is that both clients have to have WoW. Fortunately, WoW now allows players free play up until level 20 with no credit card necessary. It only requires basic information and an email address that can be made up readily, so this isnt much of a drawback.

IV. Detection and Prevention

As stated by multiple papers above [4], [9], it is extremely difficult to detect behavior-based covert channels. Monitoring the network traffic or the players actions via an automated system would not raise any flags as the actions are seemingly normal. Every action being carried out by sender and receiver is common in the game and would not draw any unwanted attention. There are two additional factors that can make this channel even more inconspicuous: the server population and the time of day. On a high population WoW server (having a large number of human players) at night or on the weekends, there are almost always players attacking these dummies. Monitoring a few high population servers at night (between 6pm and midnight server time) and on the weekends, there was on average less than one minute per hour during which it did not have at least one player attacking it. Therefore, these actions are extremely common and would go unnoticed.

To detect this communication channel the casting on every targeting dummy across all realms would have to be monitored. First, a baseline of the normal usage would have to be taken. Then, a person (or program) that understands casting rotations and mechanics would have to monitor the cast cycle and decipher the senders casts from all other combat events occurring simultaneously. During this examination the monitor would have to pick up on the irregularities of the casting routine, realizing that it is not normal and is potentially a modulated signal, and have to begin monitoring the sequence to figure out the encoding scheme and finally decode the transmission. This would be not only very difficult, but borderline impossible to actually implement as all preceding factors would have to fall perfectly into place.

Even if all of the above events were to happen, detection of the receiver would be even more difficult than the sender as the only requirement for the actions to be logged is that the sender and receiver be on the same server and within 200 yards. The player does not have to target, or even be within line of sight of the caster for these events to be logged. The only feasible way to track down a receiver would be to first identify the sender, then monitor all players within 200 yards and hope for a repetition of proximity. Like a One-Time Pass, this identification could be easily thrown off by simply creating new free accounts every time the individuals want to send information. They could also create a new character on a different server, thereby not making a repetitive pattern that could be followed and traced. Simply stated, if tracking down the sender is already almost impossible then tracking the receiver would be unthinkable.

V. Improvements

There are several improvements that could be made to improve upon this channel. For starters, the communication could be encrypted via a shared key between the sender and receiver which would provide an added layer of security in the event that transmission was properly intercepted.

A second improvement would be to find a class that has more than three no cool-down casts (not requiring a mandated wait time between casting the same spell again), like the Mage used in this example, to modulate more than 1 bit per cast. If a class could be found with 4, 6, or 8 moves, it could modulate 2, 3, or 4 bits per cast, respectively. This would drastically increase the bandwidth while still keeping the channel discrete.

Modulation of binary files rather than only ASCII characters onto the cast stream would be yet another enhancement. Then the only bandwidth limitation would be the server restart that happens once a week on Tuesday, inherently interrupting communication. In theory, 35 (bits per minute) times 10080 (minutes in a week), or 352,800 bits could be sent between server restarts.

Another idea, proposed by Benjamin Wollak [8], had a programming implementation that would allow players to communicate via emotes, a sound effect or movement used to express emotion, which are available in almost every MMORPG and have no cast time, making them almost instantaneous. Unfortunately, the game does not log these events and this idea was never completed as it would require programming a recording interface to interpret these visual events on the receiver.

Finally, fault tolerance and a checksum could also be introduced to verify the integrity of the transmission. This would provide the receiver with a guaranteed correct message rather than just a one-time chance. (WoW does perform network-based fault tolerance so this should not be a problem.)

VI. Conclusion

As the number of devices in our homes, cars, and on our persons grows, the challenge of securing these devices also grows. Firewalls are often employed to control and filter traffic protecting them from malicious activities. Covert channels like the one discussed in this paper can thwart typical measures employed for security. For example, this covert channel could be used by malware installed on a system inside a firewalled network to provide a command and control channel for the malware. It could also enable network scanning and reconnaissance behind the firewall. Both of these activities could be accomplished with a low bandwidth covert channel and would be invisible to typical protection mechanisms.

This covert channel illustrates both the simplicity of creating and the difficulty in detecting a behavior-based covert channel. To detect these channels, a fundamental idea of the normal behavior of the events would have to be understood and a baseline created. Only then might one be able to detect potentially modulated behavior sequences. The implications for covert communication are not restricted to this game.

The major attraction in an MMORPG is the ever evolving human and environmental interaction that will continue to create new items and sequences that can be modulated with information. Nobody knows what the next big game will be, but one can bet that people will look for new ways to use it as a medium for covert channels.

References

[1] Inc Activision Blizzard, *Activision blizzard announces better-than-expected fourth quarter and calendar year 2012 results*, "http://files.shareholder.com/downloads/ACTI/2310472046x0x634081/fbb6a75d-f965-442b-8d6a-bde335918118/Q4_2012_atvi_press_release.pdf".

[2] Inc Blizzard Entertainment, *World of warcraft game guide*, http://us.battle.net/wow/en/game/guide/.

[3] Inc. CoolCommands, *Coolcommands conv.pl script*, "http://coolcommands.com/".

[4] Bo Yuan Daryl Johnson and Peter Lutz, *Behavior-based covert channel in cyberspace*, 2009.

[5] *Nist dod trusted computer system evaluation criteria*, http://csrc.nist.gov/, CSC-STD-00l-83, dtd l5 Aug 83.

[6] Butler W. Lampson, *A note on the confinement problem*, Commun. ACM **16** (1973), no. 10, 613–615.

[7] Chris Mallet, *Autohotkey*, http://www.autohotkey.com/.

[8] Benjamin Wollak, *Behavior based covert channel in mmorpgs*, Unpublished.

[9] S. Zander, G. Armitage, and P. Branch, *Covert channels in multiplayer first person shooter online games*, Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on, 2008, pp. 215–222.