

The Name of the Game

Dr. Igor Muttik, Senior Architect, McAfee® Avert® Labs
<mig@mcafee.com>

Abstract

Computing is fun for many people and a big share of this fun is online gaming. The number of online games, especially massively multiplayer online role-playing games (MMORPGs or MMOGs [¹]), has recently grown very rapidly.

Virtual online communities are sometimes referred to as “metaverses”, in which people fight, buy, sell, trade, study, travel, and do many other things that people do in real life. It is not surprising that online gaming is starting to get plagued with almost all the problems of the real world—identity theft, stealing of virtual assets, extortion, and even terrorist attacks! Metaverses grow their own economies and virtual currencies are converted into real money and back—it is only natural that virtual profits, too, get targeted by the bad guys. We present some statistics about data theft, both general and gaming-related.

This paper will focus on highlighting the design decisions that can make an online game either secure or insecure. We shall discuss issues surrounding plug-ins and scripts on both server and client sides. We shall touch on how an open-source approach may affect the security posture of online games.

We expect major gaming vendors to be mostly aware of the security issues related to online gaming, so the target audience for this paper is emerging software houses. But we hope even experienced professionals will be able to find some interesting points or unusual angles.

Online computer games are huge programs that require permanent Internet connections. Any exploitation of a vulnerability in an online game (either on a server or a client) can then be used to steal user data from both real and virtual environments. There is also the possibility of self-replicating worms.

There is a significant growth in inside-game advertising and shopping. We discuss MMOG-based spam, phishing, adware and spyware.

Finally, we shall devote a substantial part of the paper to predicting future trends by analyzing existing market and technological shifts.

Chapter 1. Growth of online gaming

The number of online games and their subscribers is growing very rapidly. Especially popular are MMOGs like “World of Warcraft” (a.k.a. WoW) and “Lineage”. There is a Web site specializing in counting the number of subscribers to these online games: <http://www.mmogchart.com>. This following graph (Fig. 1) covers 1997 to 2006 and shows the total number of subscribers and the distribution among major game vendors. (Take these numbers with a pinch of salt as many people register multiple times.) Nevertheless, the growing—even explosive—trend is undeniable.

¹ Massively Multiplayer Online Role-Playing Games and MMORPG/MMOG abbreviations are used interchangeably.

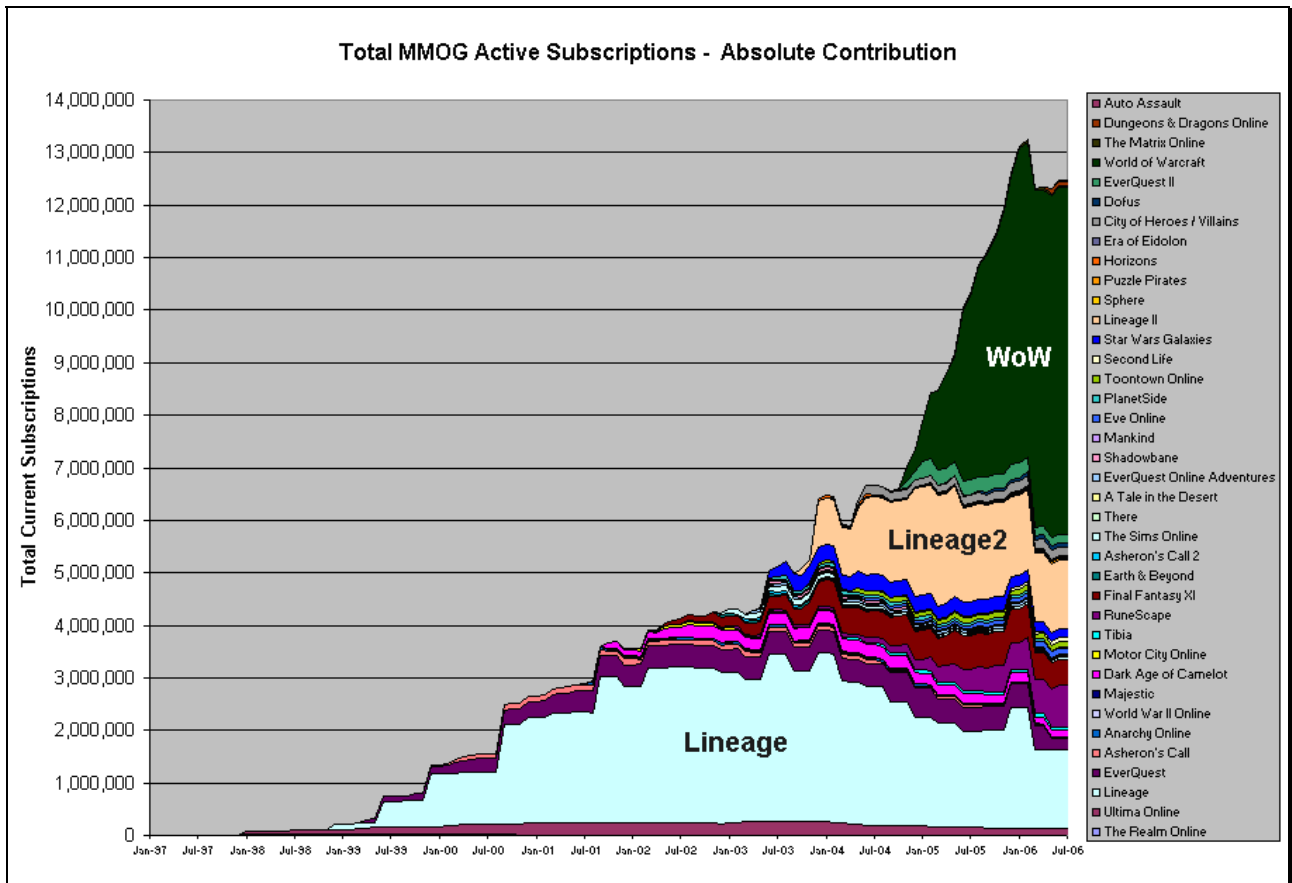


Figure 1. Total MMOG subscriptions. (Source: <http://www.mmogchart.com>)

Data based on counting subscriptions is supported by the market analysis. According to one study ^[2], the online gaming market grew 288% in 2002-2005. Worldwide revenues from online gaming exceeded \$1.1 billion in 2006 and by 2009 the revenues are to triple, according to the market research firm Parks Associates ^[3]. The biggest share is currently MMOGs and it is predicted they will stay this way beyond 2009.

Big companies are investing into gaming too—both Microsoft and Google, for example, acquired companies involved in in-game advertising. Intel bought a gaming video engine “Havok” ^[4] (Havok 1.8.3 is used in “Second Life”). And Sony starts its own virtual world called “Home” ^[5]. There are rumors that Google is developing a “Virtual World” of their own ^[6].

People spend a lot of time playing. More than 25% play for more than 30 hours every week ^[7]. In most games they collect and produce some sort of virtual commodities. They can be virtual objects (weapons, gear, clothes, property, furniture, music, etc.), money, relationships (one can be a lord of a castle with many subordinates, another can even get married virtually!). Even names of characters are valuable and could be resold at a profit—there is a virtual equivalent of cyber-squatting (registering many “good” names to resell in future).

Because players spend so much time and effort doing this, they value database records representing virtual assets as much as real objects. Unsurprisingly, people are ready to take short cuts and pay real money to get advanced virtual objects to avoid more boring routine work. So, naturally, we see secondary markets for

² http://www.researchandmarkets.com/reports/339096/on_line_and_mobile_gaming_in_the_united_states.htm

³ http://www.parksassociates.com/press/press_releases/2005/gaming-1.html

⁴ http://www.eurogamer.net/article.php?article_id=83422

⁵ <http://news.bbc.co.uk/2/hi/technology/6429039.stm>

⁶ <http://www.marketingvox.com/archives/2007/01/29/rumor-a-virtual-world-by-google/>

⁷ http://lindenlab.com/whitepapers/Escaping_Guided_Cage_Ondrejka.pdf

virtual commodities—for example, virtual currencies can be bought on eBay. This screenshot from eBay (Fig. 2) shows the rates for “Adena” (currency in “Lineage2”) for different game servers.

Starting bid: **US \$2.00** [Place Bid >](#)

End time: **Sep-14-07 21:51:19 PDT** (2 days 10 hours)

Shipping costs: **US \$1.00**
Standard Flat Rate Shipping Service
Service to [United States](#)
([more services](#))

Ships to: Worldwide

Item location: Miami, FL, United States

History: [0 bids](#)

You can also: [Watch This Item](#)
Get [mobile](#) or [IM](#) alerts | [Email to a friend](#)

Seller: [crooners097](#) (0)
Member: since May-23-07 in United States

- [Read feedback comments](#)
- [Ask seller a question](#)
- [Add to Favorite Sellers](#)
- [View seller's other items](#)

Buy safely

- Check the seller's reputation**
No feedback reviews at this time
- Check how you're protected**
PayPal Up to \$200 in buyer protection.
[eligibility.](#)

Description

Lineage 2 Adena On any Server

Server	Amount / Price	Server	Amount/Price
Bartz	10M/\$6.14	Hindemith	10M/\$5.26
Devianne	10M/\$4.56	Kain	10M/\$5.61
Erica	10M/\$4.09	Lionna	10M/\$5.08
Franz	10M/\$6.09	Sieghardt	10M/\$4.09
Gustin	10M/\$4.91	Teon	10M/\$6.66

The method for purchasin and delivery:

Payment:
We accept paypal and credit card.E-check via paypal.(E-check needs 3-4 business days to be cleared.)

Our Website:
www.worldgamebank.com

Figure 2. eBay auction for “adena” (Lineage currency)

Rates of virtual currencies are tracked almost as carefully as real money [8].

Virtual objects are traded in two connected markets—fully virtual and real. Intertwining of real and virtual markets is growing and there are now even real shops in virtual worlds (you can buy real goods for virtual money there) [9]. Both of these markets attract criminal elements.

Gaming is extremely popular in the countries belonging to the Asia-Pacific time zone. Consequently, we should expect statistics from this region to give us a hint about future trends for the rest of the world. According to a study [10] in Taiwan 37% of criminal offences are related to online gaming. We can see that the level of penetration of virtual offences into real life is alarmingly high. Many of the players are fairly young, which must have a bearing on the statistics that most offenders belong to 15-to-20-year-old bracket.

Multiple banks already announced their plans to open their virtual branches—eventually that would combine all the known risks of Internet banking with the risks of virtual identity and data theft [11]. But first, let us have a look at the statistics of traditional data theft, what drives it and we shall be able to better understand if virtual assets are at risk.

Chapter 2. Dynamics of data stealing

Growth of data-stealing Trojans

We have observed rapidly growing data theft since about 2000 (see Fig. 3). The typical scenario is that data-stealing programs record users' authentication data (ID+PWD) along with the IP address or the name of the

⁸ <http://www.gameusd.com> , <http://www.bankofwow.com/>

⁹ <http://www.vnunet.com/vnunet/news/2194117/iwoot-launches-second-life/>

¹⁰ <http://dev.hil.unb.ca/Texts/PST/pdf/chen.pdf>

¹¹ <http://www.nevillehobson.com/2006/12/03/abn-amro-bank-opens-in-second-life/> ,
<http://secondlife.reuters.com/stories/2007/03/02/danish-bank-moves-to-offer-trading-in-second-life/>

server they use, and transmit this to attackers (first stealers used anonymous email accounts (like hotmail) but contemporary ones hide outgoing traffic much better). Sometimes additional information (like the amount of money for a banking account; or the level of a player and contents of inventory—for a gaming one) might be transmitted. Later, the attacker can log into the compromised account and retrieve anything valuable.

Typically, when a gaming account is compromised, the attackers will convert obtained objects into virtual currency—then convert virtual currency into real money. Alternatively—if the motivation was more enjoyment than financial—they might use the equipment and money for their own gameplay (this latter scenario would apply only to gamers who play on the same server so such attacks are more likely to be targeted—possibly Trojan deployment would be organized via in-game messaging or via a dedicated forum). Most of these malicious programs are distributed using social engineering tricks that lure people into running them.

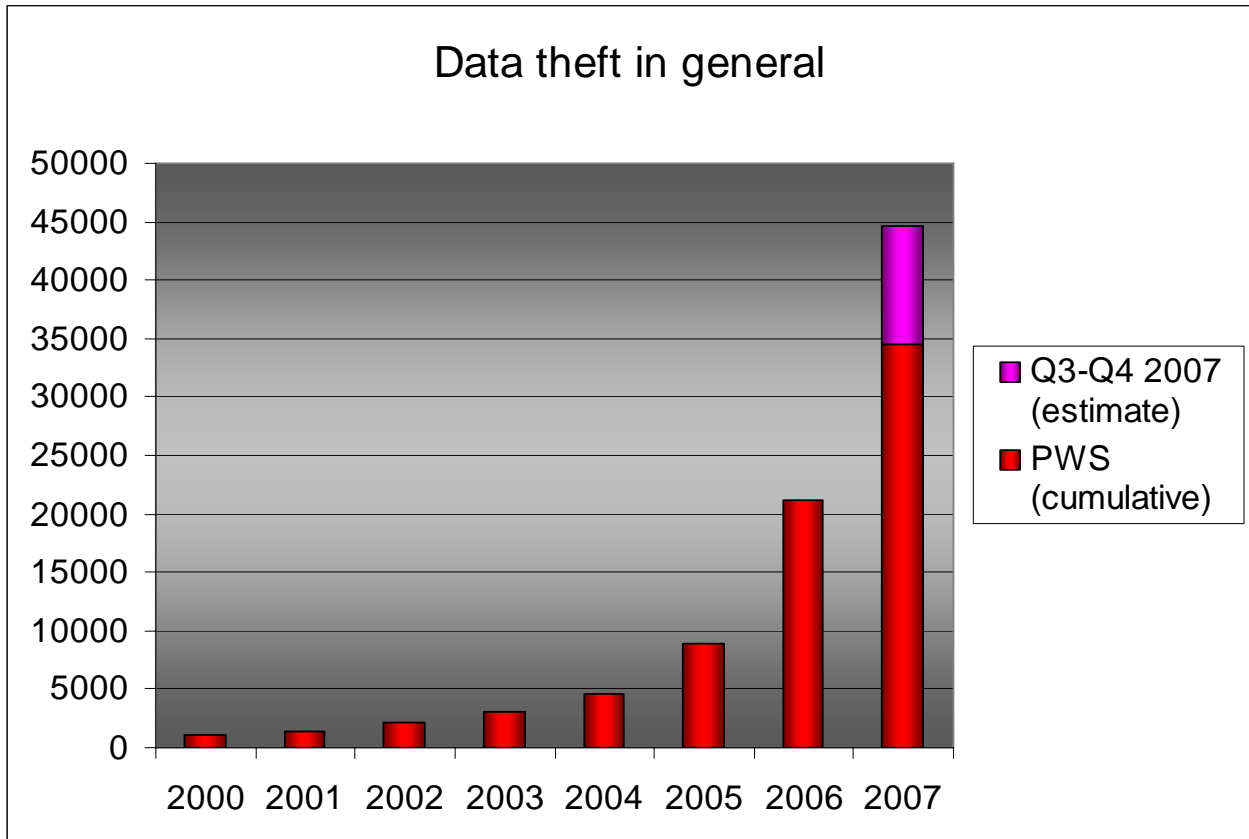


Figure 3. Data theft for 2000-2007 (total number for 2007 is an estimate)

The consensus of security experts is that financial motivation is the primary driving force behind the alarming growth of data theft. The distribution of targeted applications in 2007 is given in Figure 4.

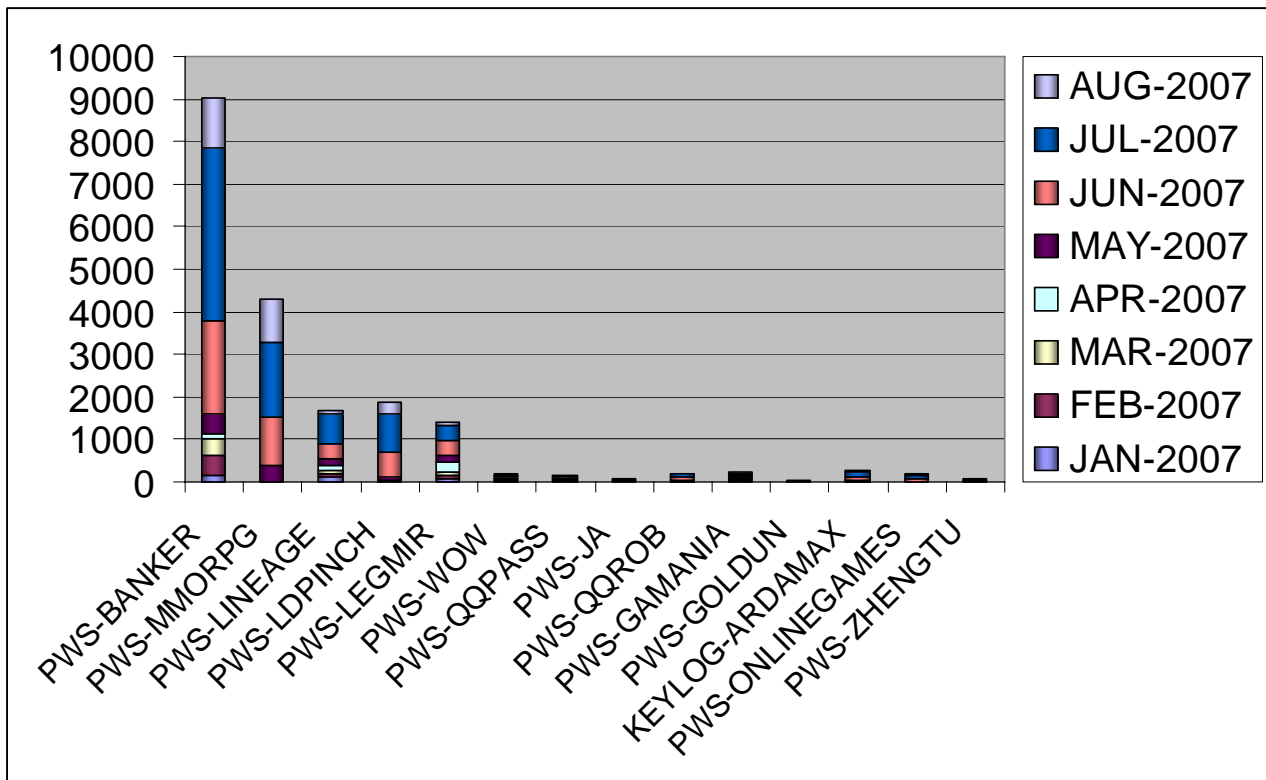


Figure 4. Major applications targeted by data theft for 2007

Trojan programs to steal banking data are leading the pack. Then follow stealers for online games; their quantity (in 2007) almost exceeds the number of banking Trojans [12].

We must stress here also that a huge raft of general-purpose malware—backdoors, password-stealers, key-loggers and form-grabbers can be successfully used to steal information related to online gaming. So, by counting only malware samples that explicitly refer to online games we get underestimated numbers. In fact many PWS-Bankers would successfully steal credentials for MMOGs as well as banking data.

Phishing

Phishing can also be considered data stealing even though normally there is no malicious code involved (other than programs that do the spamming of phish). It would be surprising if phishing was not used to rid gullible players of their virtual assets. Indeed, there are some examples confirming this [13].

It is curious that at the time of writing this paper (September 2007) perhaps the most intense spamming attacks so far, related to W32/Nuwar (a.k.a. Storm worm) were using gaming theme. The bad guys created a Web page offering “free games”, links to it were widely spammed but in reality any click on anything from this Web page would lead users to one and the same worm in ArcadeWorld.exe file [14].

Parasitic and polymorphic viruses targeting MMOGs

Trojan programs require a social engineering trick to work. Otherwise they would not have an opportunity to get installed. To overcome this serious limitation the bad guys attempted to use replicating code to deliver

¹² <http://www.avertlabs.com/research/blog/index.php/2007/07/16/password-stealers-targeting-games-are-growing-more-than-ever/>

¹³ http://news.netcraft.com/archives/2005/09/28/scams_targeting_online_games_old_phish_with_fresh_bait.html

¹⁴ <http://www.avertlabs.com/research/blog/>, <http://www.websense.com/securitylabs/blog/blog.php?BlogID=147>

the data-stealing payload. Obviously, when a virus spreads, it might propagate to the systems that are used for gaming. Then the virus can detect this and activate its data-stealing payload.

Employing parasitic technique would also make removal of malware a bit more complicated. There might be a slight delay in the release of an AV update capable of cleaning—that is an additional bonus for virus writers.

Let us have a look at just four notable virus families that have payloads related to online gaming (there are others). Figure 5 shows the number of variants in these four virus families over time:

1. W32/HLLP.Philis—a prepending virus. Appeared in early 2004, it was written in Delphi and downloaded malware that stole login details for “Lineage” and “Legend of Mir” games [15]
2. W32/Detnat—a polymorphic virus [16]
3. W32/Bacalid—another polymorphic virus [17]
4. In 2006 we saw a wave of W32/Fujacks that targeted “Lineage”, “Legend of Mir” games [18] and the popular Chinese MMOG game “Zhengtu” [19]. We have to note here that the members of W32/Fujacks family have significant code similarities with W32/HLLP.Philis. The change in classification is due to the modifications in the replication mechanisms—so much so that both families could, in principle, be merged for the purpose of counting. W32/Fujacks started using “Autorun.inf” and modifying HTM and ASP files.

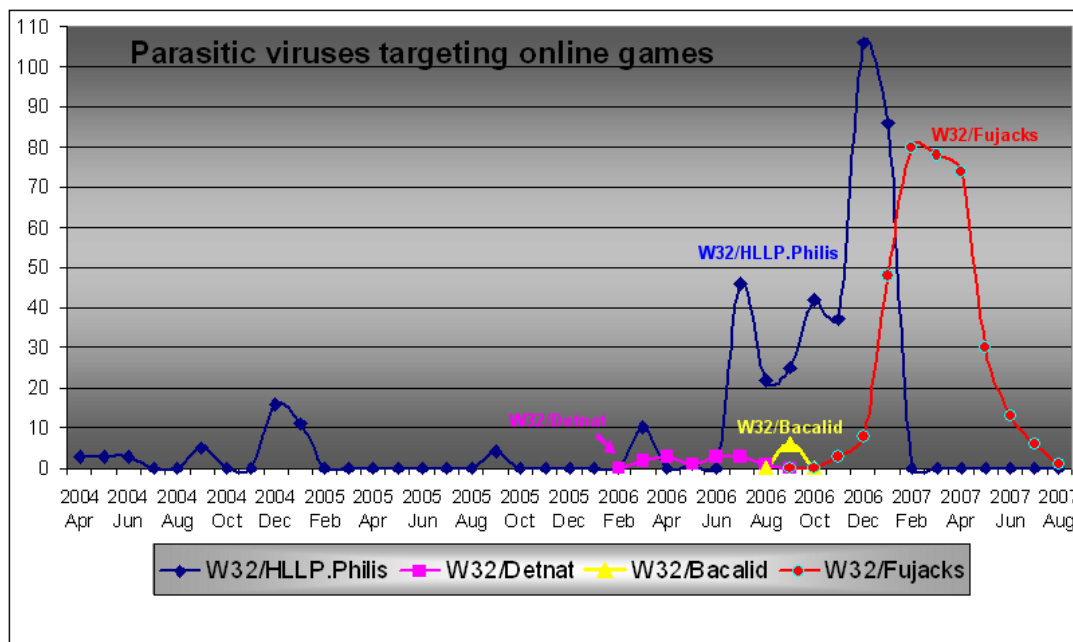


Figure 5. Four notable parasitic viruses targeting MMOGs

We can make the following conclusions from this graph:

- 2004-2005 were slow on the parasitic front, but in 2006 several infectors appeared
- Two attempts at creating polymorphic viruses in 2006 probably proved to produce too low a return on investment (a lot of work to implement polymorphic engine)

¹⁵ http://vil.nai.com/vil/content/v_140403.htm

¹⁶ http://vil.nai.com/vil/content/v_139344.htm

¹⁷ http://vil.nai.com/vil/content/v_140946.htm

¹⁸ http://vil.nai.com/vil/content/v_141877.htm

¹⁹ http://www.trendmicro.com/vinfo/secadvisories/default6.asp?VNAME=PE_FUJACKS%3A+Jacking+Up+to+the+Times&Page

- Perhaps a more successful strategy is to release lots of new non-polymorphic viruses
- New, more advanced families replaced old ones (new propagation methods in Fijacks vs Philis)
- There is a decline in 2007 that started around May (almost certainly due to the arrest in China of Li Jun, the author of W32/Fijacks—in September 2007 he was convicted to 4-years in prison)

Parasitic viruses written to steal gaming data, of course, affect not just people who play games. They are the bread-and-butter of AV business so, naturally, scanners quickly react to deal with them. The replicating nature of viruses draws attention to them, so their lifespan is significantly shorter. This could be an additional reason for the decline in 2007—in general it is a lot more likely that a static data-stealing program (especially used infrequently) will go under the radar of AV companies. On the other hand, self-propagation ensures that more computers get compromised and unprotected ones will keep pumping infections out even after AV protection is made available for most of the users.

We have discussed ordinary parasitic viruses. A scarier scenario would be if an auto-executing worm appeared. Such worms can utilize zero-day vulnerabilities in client or server software and propagate similar to, for example, W32/Codered [²⁰] or W32/SQLSlammer [²¹]. It is not likely a worm can utilize vulnerabilities in both client and server, so we can assume it will have to hop between only clients or only servers. Obviously, a server-only worm would have just a handful of machines to propagate to (more about P2P server clusters later) and it will be quickly detected and a fix developed. It would seem to make a lot more sense for the bad guys to covertly use any zero-day exploit for a period of time to steal or modify data rather than use it for building a worm (server or client based).

We have to conclude this chapter by stating the obvious—even though in most virtual environments trading virtual money for real money (RMT) is considered a violation of terms of service (TOS) we must assume that it will always be possible to find a way to convert virtual commodities into hard cash, into each other and vice versa. Even if it were to be illegal (by law or according to TOS)—there will always be a black market for virtual commodities. And automated ways of extracting money from online games (password stealing malware, using bots [²²] and similar cheats) would have advantage over manual methods. So, unless we find ways to control this, we should probably expect as much growth in online threats as we have seen with real malware. Let us have a look now at how we can make online games more resistant to malware.

Chapter 3. Client-server architecture and scripting

In this chapter we shall have a look at best practices in implementing scripting in general and we'll also focus on two scripting languages. First is "Lua", because it is common and because it is used in "World of Warcraft". Second is "LSL" because it is a very rich scripting language of "Second Life" and this environment offers enormous flexibility in supporting commerce, advertising and creativity. Therefore we should expect many standard attacks (phishing, spam, viruses, etc.) to materialize there first.

Dangers of scripting

The flexibility of online games comes from the fact that they are using "client-server" approach. On the server there is a database that ensures persistency of the virtual world. There are also rules that define events and transformations related to the objects held in the database. All essential information should be stored on the servers (positions, contents of inventory, status of characters, etc.) because anything stored on the client can potentially be tampered with (and not just data stored in files – memory contents or flying network packets can be modified just as easily). Therefore clients are usually designed to simply render the representation of the virtual world using pretty pictures after receiving all the necessary data from the server. Frequently the server-side rules are kept in form of scripts for additional flexibility [²³].

²⁰ http://vil.nai.com/vil/content/v_99142.htm

²¹ http://vil.nai.com/vil/content/v_99992.htm

²² Note: "A bot" in online gaming means a program that plays a game instead of a human. "A bot" in AV industry terminology is a malicious remotely-controlled program, which is a part of "a botnet" that is comprised of several controlled bots.

²³ Note: By "scripts" we mean any frequently modified code that can be dynamically changed, whether it is sandboxed, compiled, interpreted or executed in native code (e.g. via just-in-time compilation).

There could be four basic kinds of scripts (sometimes simple user scripts are called macros). Each of them has its own security issues:

- vendor's scripts on the server (isolated from users so the safest of all)
- user's scripts on the server (the most dangerous—they need full sandboxing and strict control of what they can access and do)
- vendor's scripts on the client (in essence they are the extension of the client so need the same level of checking as the game binaries to avoid users tampering with them; in any case having them in plaintext for the users to read would not be a good idea – some sort of p-code should be fine)
- user's script on the client (their access should be tightly controlled—either no access to server objects or following a strict policy)

The golden rules to ensure safe scripting are:

- scripts must not have access to other scripts (especially 'write' permissions) — otherwise viruses become possible)
- scripts should have as little persistency as possible—otherwise viruses can survive
- no auto-execution—otherwise viruses can activate easily (automatically or on a certain event)
- no access to dangerous objects (Email, IM, messaging, Internet access) or, if necessary, such accesses can be "throttled" (e.g. less than one Email in 30 seconds) or require recipient's consent (e.g. recipients can register on the server who can communicate with them) —otherwise viruses can propagate more easily and cause harm and disruption

Lua scripts

The very first recorded case when scripting for online games was used to implement viruses [24] was with scripts in "Lua" language [25]. These viruses were implemented for "Garry's mod" for "Half-Life2" (also known as "GMod"). The reason they were possible was that within "GMod" some scripts are copied from server to the client and executed there. Persistency was achieved as scripts were saved as files. Plus, it appeared that "Gmod" automatically executes scripts with ".res" with extension if they were in "maps\" folder. So the viruses, when they received control, copied themselves there and renamed from ".lua" to ".res".

"Lua" is widely used for many other games [26] because it is free. To name just a few—WoW uses it [27], so do Nintendo game consoles and "Warhammer Online" (apart from already mentioned "Half-Life2"). In WoW Lua scripts can be used to customize the interface of the game. The customizations (files with XML and LUA extensions) go into the following three folders— "Interface", "WTF", and "WDB".

We cannot resist to mention an incident in 2005 when bug in game scripting for WoW caused a serious epidemic ("Corrupted blood") in virtual world that killed scores of weak characters [28]. That would be the first case of a virtual virus and, hopefully, the last. In a way, it was a case of an "accidentally" created virus. There is an only single known case of anything remotely similar in real life so far—a naughty tool called "Already" [29].

LSL scripts and control of unsafe operations

Scripting in some online games is very developed and one of the most interesting examples is LSL (Linden Scripting Language [30]). LSL provides exceptional power—"Second Life" (SL) was developed to allow players create their own objects and define their behavior thus giving users the tools to create the scripts

²⁴ <http://www.avertlabs.com/research/blog/index.php/2006/08/>

²⁵ http://en.wikipedia.org/wiki/Lua_%28programming_language%29

²⁶ <http://www.lua.org/uses.html>

²⁷ http://www.wowwiki.com/UI_Beginners_Guide , http://www.wowwiki.com/Word_of_Warcraft_API

²⁸ <http://news.bbc.co.uk/1/hi/technology/4272418.stm> , http://en.wikipedia.org/wiki/World_of_warcraft

²⁹ http://vil.nai.com/vil/content/v_100205.htm

³⁰ http://en.wikipedia.org/wiki/Linden_Scripting_Language

that essentially define local game rules (normally only game developers would be able to do that). This exceptional flexibility makes LSL very interesting from security perspective.

LSL is an event-driven C-like language that gets compiled into byte-code and executed in a virtual machine on “Second Life” server (so, according to our classification, this is the most dangerous kind of scripts). There is no explicit persistency but scripts can be attached to in-game objects (to be precise, scripts are attached to so-called “prims” many of which can be linked into an object) which can be saved and reused. In LSL one can even send Emails (this function involves 20 second delay—no doubt to prevent spamming—thanks to Linden Labs!). LSL has advanced networking functionality and supports XML-RPC (with a 3 second delay—no doubt to prevent DoS attacks!). Plus, there are also *llHTTPRequest* and *llLoadURL* ^[31] (also throttled at 1 and 10 sec, respectively). HTTP requests generated by LSL scripts to URLs out of Linden Labs infrastructure get multiple “X-SecondLife-“ headers automatically inserted—this lets SL interact with the Internet and opens a lot of possibilities for building infrastructure that would interact with SL.

With LSL scripting one can create really complex objects and video simulations. It was even used to create a visual simulation of a terrorist attack in “Second Life” (Fig. 6).



Figure 6. Visual effect of a script simulating a big explosion in “Second Life”

The background of this event is that a long time “SL” player Marshal Cahill is campaigning for giving voting power to decide the future of “SL” to the players. He organized a so-called “Second Life Liberation Army” to assist in that goal. Finally, he and his followers “detonated a bomb” (a complex LSL script, really) near two in-game stores (“American Apparel” and “Reebok”) to draw attention of media and Linden Labs.

A curious fact is that LSL functions related to visualizing explosions (*llMakeFire*, *llMakeExplosion*, *llMakeSmoke*) suddenly became deprecated (Figure 7 shows these functions crossed out from the list). But do not hold your breath just yet—I am just joking! These functions simply got replaced with a better and more generic alternative—*llParticleSystem*.

³¹ Note: In LSL scripts all standard functions start with double “ll” that stand for “Linden Library”.

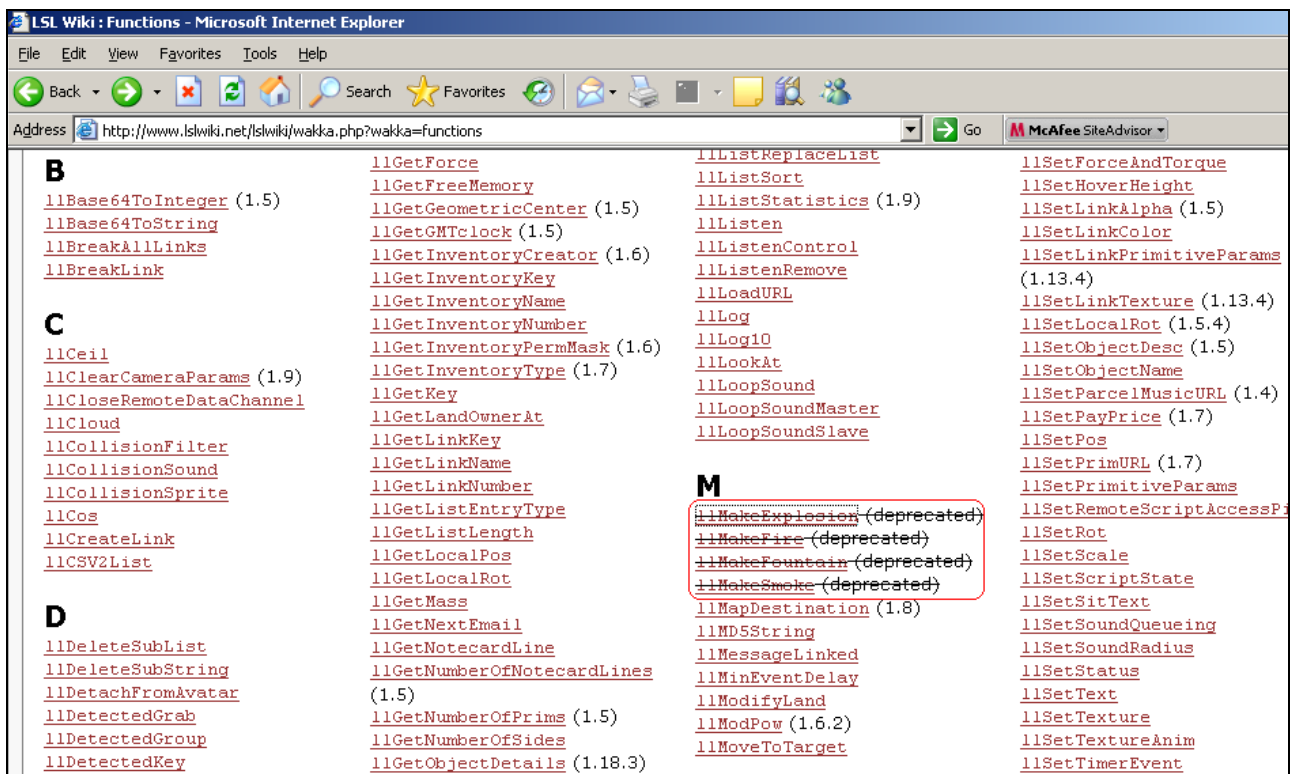


Figure 7. List of LSL functions indicating deprecated entries—explosions, fire and smoke.

The point is, though—once scripting power is given to the users it must be tightly controlled. The most dangerous functions (Email, IM, networking, messaging) involve a delay in scripting which is the main mechanism Linden Labs use to reduce potential problems. A table [32] demonstrates what “throttling” limits were imposed in LSL on to some dangerous functions.

If one uses `llRezObject` and `llGiveInventory` a new object can be created (a script attached to one of the prims will replicate the object). To control creation of too many objects Linden Labs used more complex logic than trivial “throttling”—both the frequency and quantity of created objects are measured and a 12 second delay is introduced if a threshold is exceeded [33]. The idea is to allow functions that support replication but throttle the yield of new objects. For Linden Labs such replicating scripts pose a denial-of-service problem because a server with a quickly growing number of objects is likely to become very slow. Another interesting method employed to control how LSL scripts interact with objects is a concept of “an object’s energy” [34]. Massive objects acquire energy slowly while tiny objects—quickly. At the same time only when the energy level is 100% scripts can effectively affect an object—objects with the level of energy close to zero are almost unaffected by scripts. It seems obvious that the “energy” was invented for security and stability reasons.

To make scripts benign one only needs to remove persistency. If the persistency property is missing or throttled from the user scripts we can very seriously reduce security risks (this is the case with many game consoles that do not store programs on writeable media). Vendor’s scripts (scripts downloaded from the server for local execution on the client) can be verified using digital signatures and executed only if they are authentic. This checking would put a high barrier for tampering with such scripts, especially if verification code in the game’s binary is protected with mechanisms like GameGuard or Warden [35].

If we look at the history of security risks associated with scripting in general we can see what makes scripting less prone to abuse—the lack of file/media/storage access support (no persistency in files or other

32 <http://www.lslwiki.net/lslwiki/wakka.php?wakka=communications>

33 <http://www.lslwiki.net/lslwiki/wakka.php?wakka=GreyGooFence>

34 <http://www.lslwiki.net/lslwiki/wakka.php?wakka=energy>

35 <http://www.rootkit.com/blog.php?newsid=358> , <http://news.bbc.co.uk/1/hi/technology/4385050.stm>

media), access to dangerous objects (e.g. the address book, raw memory, TCP/IP stack, other scripts or VBA modules). Most of these problems can be addressed by proper sandboxing (WoW does that for Lua just as SL does this for LSL).

Open-source and proprietary scripting in MMOGs

There is a visible trend to use open-source scripting in gaming more frequently. This is supported by open source efforts like PyGame which is Python-based game library that is capable of talking directly to DirectX [36]. Many other game-related scripting solutions are listed in [37].

Mentioned already “Lua” is also an open-source project. “Linden Labs” plan to switch “Second Life” scripting to use “Mono” [38]—an open-source incarnation of .NET.

A shift to use open-source scripting engines is clearly happening—it reduces the development costs and paves the way for easier integration of different games in future. The drawback, on the other hand, is that common programming language might, at some point in future, assist in creating cross-game threats.

Meanwhile, though, an adequate balance between convenience and security can be achieved by utilizing custom object models. Incompatible object models will ensure that the interoperability of such scripts is reduced. For example, in one game a script can refer to the script attached to the 4th item in the inventory as “Inventory[4].Script” but in another game same scripting language may have to select an object and then use “CurrentObject.RootElement.Script”.

For a lot more security implications of open source please read McAfee Avert Labs “Sage” Vol.1 Issue 1 “Paying a price for the open-source advantage” (July 2006) which is available in PDF from [39].

Chapter 4. Three pillars of security

We shall discuss technological solutions, economical measures and human factor—all of these affect security and neither can fully succeed if others are ignored.

Technology

Proper technological measures are very important but they would never be able to solve all security issues. It was demonstrated by Yee, Korba, Song and Chen that most of threats to online games (12 out of 17 that they identified) can be significantly reduced by properly designing a secure MMOG system [40]. One simple example of what cannot be covered by technology is, for example, a physical threat to a player or a server administrator.

³⁶ <http://www.pygame.org/wiki/about>

³⁷ http://www.igda.org/wiki/index.php/Casual_Games_SIG/Whitepaper/Technology

³⁸ http://en.wikipedia.org/wiki/Mono_%28software%29

³⁹ http://www.mcafee.com/us/threat_center/white_paper.html

⁴⁰ <http://iit-iti.nrc.gc.ca/iit-publications-iti/docs/NRC-48457.pdf>

We would recommend the following measures:

- scripting with limited or no persistency, auto-execution and access rights (e.g. via sandboxing)
- better authentication of a user to the client and client to the server:
 - tie user accounts to specific IP ranges
 - use a separate authentication channel (like SMS messaging) in addition to standard ID+PWD authentication
 - use one-time keys
 - employ physical authentication devices like RSA tokens or similar (e.g. biometrics)
 - use public keys infrastructure (PKI) to verify users' identity
 - securing user input (via keyboard and mouse) during login (a curious obfuscation approach was demonstrated by [⁴¹])
- encrypted and authenticated transmissions (so that packets in transit between clients and servers cannot be tampered with)
- code reviews, code inspections (for core binaries and all scripts) and penetration testing in order to improve the quality of software (minimize the number of exploitable scenarios known as "cheats", number of bugs in general and remote exploits specifically)
- proper segregation of what kind of data is stored on servers and what—on clients
- not transmitting any information to the clients that is not necessary (this may be tricky as it is generally very hard to predict exactly what local rendering will require; but certainly no debug information and sensitive data should be transmitted)
- in-game communication methods (instant messaging, bulletin boards, VoIP, discussion forums, etc.) should not allow any active content (executable code, scripts) or links to such objects (clickable URLs that activate browsers capable of executing such objects)
- QA, adequate alpha and beta testing
- backups (and convenient "per-transaction" roll-back functionality in server DB)
- secure logging of events (up to a level when all events can be "replayed" and viewed via, say, standard game client), logging of virtual transactions and DB mods
- take measures against distributed denial-of-service (DDoS) attacks—Internet-based DDoS attacks are well known and some defenses do exist [⁴²] but consider also in-game attacks (e.g. deliberately too many characters/objects/scripts in one area/sim)

We have to say that enforcing security for MMOGs is easier than in real networks because the whole environment is fully controlled by game vendors. Updating of game client and server code is in the hands of vendor—so explosive threats (worms, viruses, widespread cheats), critical bugs and zero-day exploits will be very short-lived (can be rapidly and fully exterminated). Thus, if any security issue emerges the corrective action can be taken as soon as the solution is developed (except maybe for some free-shards and emulated environments like L2J [⁴³]).

Still, of course, it is best to minimize the number of security issues by building software securely from day one.

Economical measures

Probably the most important security paradigm is that cost of a security breach should exceed potential benefit (for example, the cost of equipment to cut a safe multiplied by the risk of getting caught while robbing a bank should be, ideally, higher than the amount of money in this safe). Breaking this rule creates problems which can be seen, for example, from the phenomenon of spam. Emails are free so spammers can afford to send millions of Emails and still benefit from the business even though very few people actually buy advertised goods or services. If Emails cost even a penny each—the ROI (return on investment) would likely

⁴¹ W. Allen, R. Ford, A. Saugere "A spyware resistant virtual keyboard". Proceedings of the 17th Virus Bulletin Conference, 2007, Vienna, pp.94-98

⁴² http://www.sans.org/reading_room/whitepapers/intrusion/1212.php

⁴³ <http://www.l2jserver.com>

become too low for spamming to survive. Unfortunately, it is too late to change the rules because neither the Internet nor SMTP protocol was designed with security in mind.

Fortunately, MMOG situation is very different and charging, for example, for in-game messages could be easily made part of a game (and even nicely presented as “royal mail stamp duty” or something like that). At the same time, understanding of security implications is also growing. Our recommendation would be to introduce floating charges for literally all in-game services so that laws of virtual economy would make most antisocial and dangerous behaviors non-profitable.

Apart from spam (we include here phishing and IM spam into it)—another class of security issues that can be solved by introducing small charges would be denial-of-service attacks that originate from in-game scripts. For example, in LSL one can initiate access to remote (out-of-game) URLs every 3 seconds. If many enough objects have scripts accessing one URL they can create a DoS attack when scripts start. The throttling would not help if there are thousands of such scripts activated at the same time (for 1000 scripts there will still be 333 TCP/IP connections per second—or more if more such objects were created). If there were a small charge introduced (e.g. 1 Linden\$ per request) for external XML/RPC accesses the problem will likely go away. Even more drastic measure would be to charge some minimal amount for any “public” script. Or allow access to only registered sites/IPs and charge L\$ for such registrations.

One has to have in mind, though, that economical pressure cannot prevent attacks mounted to prove a point or out of curiosity. That brings us to a question how else we can improve human behavior in the metaverse.

Human factor

It has been known for thousands of years that the inevitability of punishment is the greatest deterrent from committing any crime. MMOGs are in a unique situation here because surveillance over entire population is actually technically possible and such measures would potentially allow reviewing any event that took place in virtual world. Most likely the problem here would lay in the lack of investment by game vendors into the area of logging in-game events (movements, actions, transactions, etc.) and analyzing logs.

Proper logging might even help with cases like granting extra privileges to a player in exchange for a bribe or under a physical threat. Unfair methods of playing (e.g. outsourcing and so-called “power-leveling”) can be detected by IP analysis and tracking of virtual currency inflow into accounts. Countermeasures for these would also be based on logging.

The biggest loophole of total logging would be when virtual commodities are converted into real money (RMT is common place even for games where such trading is against TOS) and then need to be tracked in reality. Throttling idea may again help here—for example if there were a little delay between the exchange of the goods and money (the game vendor would play a role of safe escrow for financial transactions). Waiting for a transaction to go through would be a significant risk for the bad guys because during the delay alarms can be raised. Thus, such a delay may enormously help in preventing virtual fraud plus it will assist in analyzing the logs and would control RMT. It would seem natural to impose limitations only when a certain currency limit is exceeded over a short period of time—allowing a few minor transactions.

If real money is converted into virtual currency (e.g. for money laundering) logging and delaying such transactions (perhaps, exceeding certain amount) may help law enforcement to track down the criminals. Modifications to a server’s databases after exploiting vulnerability or successfully deploying a piece of malware on the server can be tracked down and undone if there is sufficient logging and backups.

Some malicious acts, unfortunately, may not be possible to record in logs—for example, a buffer overflow that comes in a TCP/IP transmission is likely to either crash the server or execute malicious code but it would not be in the server logs. Protecting the servers with special in-line network devices (from exploitations, DoS, and to provide some level of logging) that shield the server from the outside attacks may be a good idea. Logging at least some incoming traffic may reveal the nature and sources of attacks—this may be extremely important if, for example, a distributed denial-of-service (DDoS) is launched to achieve an unfair advantage in a game. One scenario could be to initiate a DDoS attack when virtual competitors (e.g. opposing team) are in a vulnerable situation—this is likely to create unacceptable “lags” that might lead to virtual deaths or losses. Alternative is to use a zero-day vulnerability in the server software to cause a crash at a convenient moment or to run malicious code on the server. Such attacks can be financially motivated. If

it were possible to log these violations and analyze the sources then the frequency of such attacks would undoubtedly go down.

All in all we urge game developers to invest time into proper logging and telemetry systems (by telemetry we mean a set of scripted rules—perhaps even heuristic ones— that examine the logs and raise alarms) from the very beginning. For P2P server grids it is not trivial to build a system for centralized collection and analysis of logs. So building distributed logging systems from day one is essential as bolting it on afterwards may be problematic.

We should expect that special teams will have to be involved in reviewing the logs on an ongoing basis to respond to complaints (from users) and to review specific situations (from users and/or telemetry). Usability of logging systems (e.g. an ability to “replay” events in 3D) is important to minimize the cost of this work.

Chapter 5. Predictions

When I spoke with Dr. Alan Solomon [⁴⁴] in 1997 he made a prediction that the Internet in a few years would become approximately what the current AOL client was offering (lots of pictures, graphics, IM, user-friendly, etc.). At the time the Internet was used mostly by geeks (scientists and programmers alike) so this prediction was difficult to believe. There was no Internet commerce then, no Internet banking, no Web sites for most companies. But Alan was right—what was attractive and convenient to the computing public won.

Convergence of “Metaverses” and Web

With the increasing capabilities of the Internet and advances in connectivity we can expect that pure Web-based entertainment communities will converge with traditional specialized client-server MMOG software. The signs of this are already present—for instance “Second Life” allows you to use the “secondlife://” protocol to create HTML-style links to any location within the “Second Life” world. Double-clicking on such a link launches the game at a specified location. That is a safe operation but if, for example, when you arrive at a destination some script would run automatically, then it may pose a security risk.

The behavior of users in “Second Life” seems to be very close to Internet browsing. They visit location after location, stay where it seems more interesting (read texts, watch pictures/movies, listen to music, chat to other visitors) and then move on to another place. The interface of “Second Life” has clickable objects, including Web sites and links (Fig. 8).

⁴⁴ A. Solomon was the father of “Dr. Solomon’s Software” that produced one of the first AV products in the 1990s.



Figure 8. Clickable links in “Second Life”—a paradigm resembling browsers (in upper right corner—a warning about redirection to an external Web site)

Some may say that SL is a 3D equivalent of the Internet, where owning a piece of land (a.k.a. “a sim” or “island”) is equivalent to owning a domain. And building on your land (LSL scripting and all) is equivalent to Web design. This analogy suggests that in future we could see malicious sims and hacking into existing ones. Tools that index a virtual world (à la Google) might get manipulated by the bad guys^[45], and exploits and sophisticated social engineering (the combined power of scripting and human involvement can be used) might be unleashed on visitors.

We know that malicious links are now widely used to disseminate real malware and exploits. So when many online games start supporting clickable URLs and compatible scripting, then we would have a dangerous cocktail on our hands (clicking a link in a game could open a browser and cause trouble). We must make sure that appropriate measures are taken to prevent, for example, spam runs advertising links to scripted virtual resources, and we should expect a rise in this kind of spam in near future. We should also see the same spam to occur via in-game communication mechanisms—“virtual spam”, if you will. It is quite possible that future Web browsers would support “links” into many different games—this will have security implications.

Now let us look at online gaming from the point of view of Web browsing. One cannot help noticing that there is an ongoing trend to create a richer browsing experience—the Internet provides increasing amounts of movies, sounds and clips. Standard browsers respond by handling more objects and plug-ins like “Flash” are already capable of creating MMOG environments. Have a look, for example, at offers from Neopets.com and ClubPenguin.com (the latter is now owned by Disney^[46]). These popular sites create virtual worlds using contemporary browsers as clients.

Here is how a browser window looks for the ClubPenguin site (compare Figure 9 with Figure 8 above—a significant difference is that picture is 2D instead of 3D, otherwise they are very similar environments):

⁴⁵ http://www.mcafee.com/us/local_content/white_papers/threat_center/wp_imuttik_vb2005_manipulating_the_internet.pdf

⁴⁶ <http://www.hemscott.com/news/latest-news/item.do?newsId=48082158898237>



Figure 9. ClubPenguin.com—2D gaming environment

If you click any of the three yellow tablets located in this room you would activate a “Find Four” game. So we can play “in-game” games. We should watch for the level of virtual nesting! Fortunately, security issues on all virtual levels should not really be different.

Web technology is also developing in a way that clearly paves the way to richer content. Recently, Microsoft released “SilverLight”, which is an alternative to Adobe’s “Flash” player. SilverLight is based on .NET (so it is cross-platform). 3D support in browser plug-ins has been announced in August 2007 for Adobe “Flash” by using the third-party “Swift 3D® v5.0”, produced by Colorado-based “Electric Rain” [47]. Very soon 3D functionality will be available for all browsers, and advances in broadband connectivity will allow 3D content to be routinely used. Some attempts in this direction already exist – 3D browsers [48] although they are currently more of a standard 2D browser presented via a 3D environment (e.g., on the sides of a cube that can be manipulated in 3D) but this is still an indication of a trend.

Ajax technology can use more atomic client-server interactions, which are very useful for gaming. Ajax is based on XMLHttpRequest—supported by both “Internet Explorer” and “Firefox”. At the same time LSL supports XML-RPC [49]—so there is some support of Web 2.0 in LSL.

As we can see, Web and MMOG technologies are converging. Giving that the browsers were being plagued by severe security issues all along—there is a significant risk that such convergence will make the attack surface on MMOGs much wider than it currently is. It would seem natural to assume that in future more MMOGs will be based on browsers (no need to develop a client, so it is much cheaper), thus making malicious attacks on the online games easier. Attacks can utilize vulnerabilities in browsers and steal, for instance, virtual identities and commodities (apart from the real ones, or addition to).

⁴⁷ <http://news.thomasnet.com/fullstory/529168/rss/2585>

⁴⁸ <http://www.browse3d.com/>

⁴⁹ <http://www.lslwiki.net/lslwiki/wakka.php?wakka=xmlrpc>

Integration of metaverses

Online gaming is booming—new players (we mean corporations here!) appear all the time, small and big—everybody wants a piece of the pie. It would not be surprising to see a rather low level of coordination and standardization during this phase. But the users' demand for communication to work across different games would inevitably lead to vendors providing interfaces to existing communication media (browsers, email, IM, VoIP, SMS, etc.) and between each other. People tend to play for long periods of time—they would prefer to have all communication methods integrated and “at hand”. Because of that we would expect access to existing communication vehicles to be organized first via in-game mechanisms (by utilizing appropriate APIs to respective protocol providers).

There will also be a strong user demand for increasing interoperability of different virtual games, and we should expect the appearance of virtual “gateways” (à la checkpoints at the countries boundaries—quite possible even with customs and taxes when you want to pass!). Such gateways will likely initially be able to transmit messages. Later, at some point, it may become possible to send parcels with virtual commodities between virtual worlds and transmit virtual currencies via these bridges.

It is important to make sure that no active content (executable code or scripts) can be transmitted. If necessary, they should be inspected (automatically and/or manually), charged for and logged. Transmitting messages with links (even with standard HTTP links) without any vetting or control would assist the bad guys because traditional real-life spam would turn virtual.

P2P shift

When the number of users for a single MMOG server exceeds 10,000 to 20,000, new subscriptions are usually redirected to a new server (frequently called “a shard”). This has to be done in order to let the servers cope with the load. New players are isolated from the old crowd and populate a replica of the existing world. The drawback, of course, is that new and old users do not share the same “metaverse” and cannot enjoy simultaneous playing. Economies are also separate, which is reflected even in slightly different exchange rates as we have seen in the eBay screenshot in Figure 2. This is how things were. Modern technologies, though, allow for server clustering and can be infinitely expandable.

The biggest of the games running off one server cluster is “EVE Online” [⁵⁰] and the record was achieved on 3rd Aug 2007 with 35,313 simultaneous users.

“Second Life” uses a grid of servers. Each supports one sim (island) with all objects and avatars on it. When objects or avatars move to another sim, their database records are passed to another server. We expect similar P2P approaches to become more common as the drive is to support more players and a common “metaverse”. Perhaps at some point in future P2P technology would also use P2P-connected clients to process data. If DRM technology is utilized, it might even become possible to store parts of the main database on the client, which can make a push towards using clients' computing capacity to support wider and more powerful P2P environment. This, of course, may have security implications, as (despite DRM) data tampering (or DoS - even if by just manipulating the presence in P2P network) might become an issue.

Within massive P2P server grids a risk and implications of an auto-executing worm could also become very significant. For this reason, even if a server cluster communication is done on a separate network or via VPN it would make sense to use encryption and authentication for interserver communication. That would also provide some protection from internal compromises.

Windows Vista has built-in support for P2P networking, so this should either be disabled to prevent propagation of P2P worms or P2P communications should be strictly controlled.

While P2P clusters are more resistant to DDoS attacks, measures should be taken to allow the filtering of unwanted traffic—hardware in-line network appliances would probably be a good option.

⁵⁰ http://en.wikipedia.org/wiki/Eve_online

Traditional AV, threats to online gaming and cooperation

As we know there is a super-linear growth in quantity of malware. There may be a paradox here because this growth is likely related to the effectiveness of AV in blocking known threats. As a result of scanners detecting existing threats, new malware is constantly being written to replace previous versions that were rendered ineffective by AV. The bottom line is—traditional security software finds it hard to cope with the current inflow of threats.

Security research requires detailed knowledge of inner workings of the targeted environment. To cover threats developed for different games security researchers need to have:

- higher than average knowledge of the environment
- access to the environment (preferably with debug capabilities; an isolated server-client pair may be needed, for example, to examine how an exploit works)
- clearance from the employer (you do not expect AV companies to pay malware researchers for playing games, do you? ;-)
- enough demand from the customers to provide such security solutions

These conditions are hard to meet so existing knowledge is currently simply not adequate in coverage and in depth. For example, the first viral Lua scripts were discovered by GMod players and only then brought to the attention of AV companies—simply because this was a new propagation vector and it was not monitored by AV.

With the number of incompatible virtual worlds expanding (as the market is growing—many new games will be created) it is natural to assume that initially we shall be facing a multitude of problems specific to this or that virtual environment. At this stage AV companies will have as much chance of covering security issues for all the multitude of games as a snowball has in hell...

When traditional AV programs provide inadequate protection (frequently because they do not have adequate visibility, like with Lua viruses) measures will have to be taken by the game vendors. It is very likely that in the short term the security problems in MMOGs (apart from data stealing Trojans) will be predominantly covered by security/policy departments of respective game developers. Cooperation between security departments of AV companies and game vendors is certainly welcome and, in my opinion, would be invaluable and mutually-beneficial.

Game vendor's protection systems are quite common because tampering with the client would otherwise flourish. Major concern is using bots or other cheating additions to game clients. For example "The Warden" for WoW is a scanning tool to detect unwanted programs (bots, cheats and similar).

It should not take a long time before we start seeing polymorphic or metamorphic bots ("bots" in MMOG sense!) as well as bots that make use of stealth/rootkit technologies and/or virtualization. These technologies are already widely used in malware so we should expect them to find their way into unwanted programs that plague online gaming very soon. Detecting these threats is not always a simple task and proper cleaning, to be honest, is frequently a real pain in the backside. Another area where AV companies have a lot of knowledge is anti-spam solutions—many games already have serious problems with internal messaging systems being abused. The knowledge of AV companies about how to deal with these security issues should help a lot. For example, cooperation can take form of licensing antivirus/antispam engines for game security or providing APIs to AV/AS engines to interrogate in-game scripts and other objects.

Follow the money and more cooperation

We know that money was the main factor driving up production of real malware and the one fuelling many new attacks since at least 2003. That means, of course, that commercialization of the virtual environment should lead to a similar wave of virtual malicious activity. And when virtual worlds start merging into a virtual universe this will be the time for global security companies to step in—because everybody will expect them to! We are bound to see virtual spam, abundant and annoying advertisements, phishing scams, virtual identity theft and so on **unless**, by design, MMOGs do not make the environment technologically, economically and socially resistant to these attacks.

Currently, to track down computer criminals we use the “Follow the Money” principle. Cooperation of different bodies in different countries is frequently needed for that to work. Occasionally, this turns into an impossible task due to differences in legislation, language barriers and even time zones! It would be a very good thing if major game developers could reach agreements about preserving the logs and cross-checking intergame transactions (when they become a reality). It would seem beneficial to encourage such intergame exchanges and discourage conversions into real money, because the latter would make transactions significantly more difficult to trace. We can even envisage an appearance of virtual Interpol (or, perhaps, real Interpol would invest in a “virtual” department?) and sooner or later cooperation between real law enforcement and virtual police will have to take place—most likely on an international scale.

It seems essential that cooperation among major online gaming companies will be required to achieve a goal of merging different games into a single “metaverse”. From the security point of view, cooperation is, perhaps, even more important to be able to thwart similar kinds of attacks elsewhere once they have been identified for the first time.

I would like to describe how this worked within the AV industry because, in my opinion, it was extremely lucky to have been built around a group of security-conscious and mutually-trusting people. Back in the 1990s virus research was a hobby for them and protecting users was in their blood. Now, when they do it for a living and work for different firms, the initial ties still allow the free flow of essential security information among competing AV companies. Essentially, the AV industry grew from a circle of several dozens of professionals who knew each other very well.

Because of that we have many mailing lists to discuss technical matters, share samples and ideas—participants come from many AV companies. We regularly meet at security conferences (like VB, AVAR and Eicar); informal meetings and contacts are common place too. When our technical people get together we look more like a group of friends—it is easy to forget that our marketing and sales forces constantly battle each other!

For example, we report bugs and false positives/negatives to each other, and exchange sample collections and statistics. We regularly discuss “nightmare scenarios” together. Error and bug reports that we send each other are always done in a way that is even friendlier than the “responsible disclosure” method.

This cooperation of security professionals in the AV industry seems an exceptional situation, but it is very beneficial for security.

Conclusions

During the explosive growth of online gaming and the struggle of vendors to reduce their time to market, we should expect that security issues will get a bit sidelined, at least for a while. But we urge game developers to build the basic security foundation from the very beginning. As we very well know bolting on security later, to an already existing product, is a far from perfect approach.

Most of the attacks that we have witnessed in real life will surface in virtual worlds unless the environment is built with security in mind. We need to leverage our knowledge and work together—the AV folks and gaming vendors—to avoid falling into the same trap again. It is possible to make most attacks in virtual life impossible or uneconomical. There are no good reasons why virtual characters should suffer from the same troubles that currently plague our day-to-day life—spam, phishing, adware, spyware, Trojans, viruses, worms, etc.

If we succeed, then the population of merging “metaverses” would grow—users would flock to the virtual worlds because they will feel more secure and protected. If we fail, well ... it is a virtual world so we can erase and start from scratch, can't we? I am afraid not—we really have to get it right the first time!

Acknowledgements

I am very grateful to my colleague Francois Paget for statistical data on data-stealing malware.