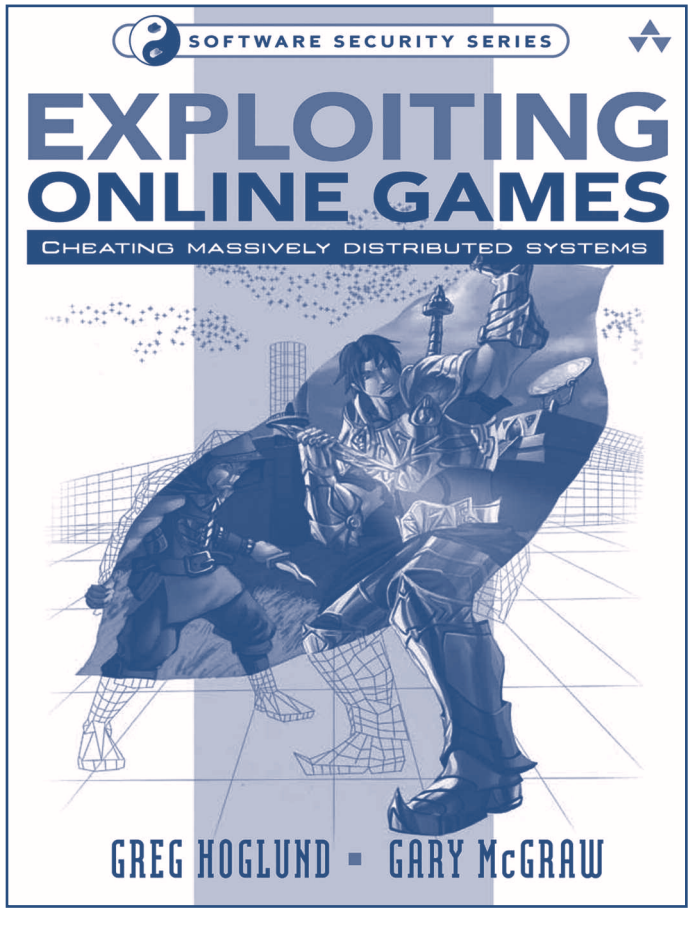
GARY MCGRAW WITH RIK FARROW

exploiting online games: an interview

Gary McGraw is the co-author of *Exploiting Online Games* and the author of *Software Security*. He is also the CTO of Cigital, Inc., a software security and quality consulting firm that has provided services to some of the world's best-known companies for a decade.



I HAD LUNCH WITH GARY DURING the 2007 USENIX Annual Technical Conference in Santa Clara, CA. We talked about his new book, *Exploiting Online Games* [1], written with Greg Hoglund, of Windows Rootkit fame [2]. Gary presented invited talks about the ideas in his book during USENIX Annual Tech and, more recently, USENIX Security. The summary of his USENIX Security IT appears in this issue, along with a summary of a talk by Greg about advanced topics in writing Windows rootkits.

I interviewed Gary via email as he traveled around the country promoting his book, with the goal of going beyond what I had already heard from him.

RIK: You mention the ability to teleport in both your book and your talk. If I understand you correctly, this simply involves poking in new values for your location, right?

GARY: The technique you're referring to is called "telehacking" by most game hackers. Note that some online games explicitly allow teleporting as one normal way to get around, but most don't. In our book, we use World of Warcraft (a game distributed by Blizzard Entertainment) as one of our key examples. We do this because WoW, as it is known to gamers, is the most popular massive multiplayer online role playing game (MMORPG) in the world with over 9 million users, some 400,000 of whom are usually playing together at any given time. In WoW, teleporting is not a normal movement option.

In this case, the mistake the game designers made was to allow the "state" representing character position to be controlled by the game client running on the gamer's PC. This is a fine design decision as long as gamers play by the rules. But if gamers want to cheat, all they need to do to telehack is change the memory values on their PC corresponding to position. You can do that by poking memory values. We have code for WoW telehacking in the book in Chapter 7: Building a Bot.

The real problem here is that the game designers made a critical mistake thinking about trust boundaries and where trust should and should not reside. Clearly any state that is allowed to be completely controlled by the game client on the potential attacker's PC should be considered with some skepticism when it arrives at the game server! If you trust everything the game client says and yet the game client is under the complete control of a cheater, big

problems ensue. This is the kind of error Greg and I expect to see more of as massively distributed software systems become more common.

I wrote an article about this trust boundary issue for IEEE's *Security & Privacy* magazine [3].

Of course, in the book we also discuss why people would want to cheat. Briefly, the why has to do with the value of virtual stuff, which can be sold on a booming middle market. Cheating pays off financially, and it is not at all clear that it is against the law.

RIK: Will monitor programs, like Blizzard's Warden, detect if you have suddenly changed your location (set of coordinates)? What about other changes, for example, changing your level, condition or health, or the amount of gold you have?

GARY: The Warden, which is a monitoring system used exclusively by Blizzard, watches your PC as you play a game. You give Blizzard permission to do this monitoring on your PC when you agree to the end user license agreement (EULA). Note, however, that this granting by the user of permission to install spyware xdoes not make it legal! My suspicion is that Blizzard's use of the Warden is not likely to be legal in California, where there are strict antispyware laws on the books.

The Warden as it is currently set up only monitors things outside the game client process itself. That is, it is concerned with things such as Window titles (of non-Blizzard Windows), other processes running on your PC, URLs in your browser, and Instant Messaging buddy lists. It also checks for DLL-replacement hacks, which we describe in Chapter 6: Hacking Game Clients. My co-author Greg was worried enough about what the Warden does that he developed a program called the Governor that keeps tabs on the Warden. Much of the code for the Governor is included in Chapter 2: Game Hacking 101.

So far, the Warden does not appear to keep track of state changes in the game client such as a change of position. It would be pretty silly to have the Warden do that if you think about it, because a cheater sophisticated enough to poke new values into memory can change the Warden's brain just as easily as the game client itself! (The Warden is simply another user mode process.) The real answer is not to monitor what the client is doing on the client, but, rather, to keep better track of things on the server so that when something crazy like a bazillion-mile hop occurs, a red flag is flown to alert the game minders.

I think one idea that might be useful is a "low-res" vector computed over all of the state that is controlled by the game client. This vector would be computed and stored on the server side at the time that the state crossed the trust boundary. Any such computation would need to be quick and dirty. Then when client state changed or maybe after so much time has elapsed, a new low-res vector could be very quickly computed and compared with the stored value. Too big a change in the vector might signal trouble and could lead to more scrutiny for the player.

The values that you described in the second part of your question would be ideal candidates for inclusion in the low-res vector. In fact, important inventory items are already tracked closely on the WoW server.

There is no perfect answer to the trust boundary/time and state issue, but there are a number of basic things that game companies could do that they are not doing now.

RIK: Earlier on, you mention a low-res vector as a reasonable solution, and that any checking of that low-res vector needs to be quick and dirty. You ap-

pear to be hinting at the enormous computational load that would be required to actually check that a user's state does not indicate that that user is cheating. But shouldn't game vendors, or SOA vendors, be expected to maintain the integrity of their offerings, instead of presuming that there isn't a problem? Perhaps vendors should provision their computing base to handle checking state.

GARY: Managing security risk always comes down to making tradeoffs. In this case, the hard question to ask is how much state checking is enough to thwart cheating? So far the market is answering that it is OK to have none, but I expect that answer to become less acceptable over time. The stakes in online gaming have changed, and the security tradeoffs must be reconsidered in this new light.

RIK: You mentioned manipulating the video card, so that when players using this hack play games, they can see through walls or even the earth in the game, or have their opponents show up in glowing orange. How is it that these hacks could avoid detection by gaming companies' countermeasures?

GARY: This is a kind of hack used against first-person shooter games such as Counter-Strike. The resulting approach is called an "aimbot." We describe aimbots in Chapter 2: Game Hacking 101. The attacker's insight is that the video card knows plenty more about the state of the game than the player actually sees in a traditional view on the screen. After all, the card has to render all of the graphics in a reasonable amount of time. It does this by precomputing and caching lots of things the gamer should not know. It may know, for example, the exact coordinates of an enemy player.

Early aimbots had no countermeasures. Current countermeasures are mostly statistical in nature. If a player is too good to be true, that player comes under scrutiny. Sometimes really good players are banned simply because they are "too good to be true."

RIK: Aren't monitor programs akin to rootkits? I know that EULAs that players agree to allow the gaming company to monitor their systems and even kill programs. But isn't this crossing the line, similar to what Sony [BMG] did with their CD rootkit for copy protection?

GARY: Monitor programs are not really akin to rootkits technically, but they are conceptually. Most monitoring programs that we're aware of run in user mode. Rootkits run in the kernel. A monitoring system in the kernel would be more effective since it could use stealthy rootkit technology to cloak itself. On the other hand, most gamers who play WoW are completely unaware of the Warden or the fact that they agreed to be monitored.

Make no mistake about it; the Warden does in fact work. I've been told by people who experienced it how very quickly the Warden detects the use of Bubba's Warcraft Hack and bans a player.

In my opinion, however, relying on monitoring software that users are not aware of does cross the line as a security mechanism. This won't stop such systems from being used, though. Think about some of the data security technologies currently on the market. They work by installing kernel-level monitors and basically spying on the user. Corporations are paying good money for these things.

I would prefer that game companies keep a handle on state at the server and not spy on the gamers' other processes running on their own PCs.

RIK: What is the coolest game hack you've come up with yet?

GARY: Greg has come up with some doozies, a few of which I covered in the talk. One involves the use of a kernel-level process to manipulate the game

program undetectably by living under it. Another involves freezing the computation, doing some calculations to help you cheat, and then injecting state before unfreezing the computation. The freezing can be accomplished with hardware breakpoints.

Basically what we have is a classic computer security arms race. In this case, because the cheaters have complete control over their PCs, they have a pretty large advantage given current game design. Better security design could help even the scales.

RIK: Did you ever try game hacks that got your account closed or banned?

GARY: When we were writing this book we wanted to start with basics and go from there. To do that, we wrote some pretty easily detectable code. During testing of the Hoglund_WoW_macro code in Chapter 2: Game Hacking 101, the character Xanier was detected and banned. Greg reports that he has had over 20 characters banned for various reasons, including characters that he bought using his wife's name.

I have never played WoW myself, so I have never clicked on the EULA or agreed to its terms, nor have I had characters banned from the game.

RIK: Do you see much of a difference between fat game clients and Web 2.0, where much of an application resides on the client side?

GARY: This is a critical point of the work that I alluded to before. I think the world is currently evolving toward software systems with lots of distributed fat clients, and I am really worried about the broken trust modeling that many of these systems are likely to suffer from. In my opinion, the kinds of time and state errors that are so pervasive in MMORPGs are the same kinds of attacks we can expect to see against SOA systems and Web 2.0 systems in the future.

If this interests you, you should read that IEEE *Security & Privacy* paper I mentioned before.

RIK: Can you (briefly) point out some lessons learned (or that should be learned) for future fat-client software programmers from the MMORPG world?

GARY: The basic lesson is that there is no substitute for considering the attacker's perspective while designing and implementing software. This is a very basic software security lesson that still needs emphasis. Expect attackers to do precisely what "nobody would do" and plan for it. Misuse and abuse cases at the requirements level (as I describe in my book *Software Security* [4]) are very useful for this.

Another critical lesson is that trust boundaries are important. Understanding who controls what information and what that means when you're trying to protect your system is essential. This gets particularly tricky when a distributed system is "massive" (with, say, over 250,000 simultaneous users) and big swaths of the system are outside the game's control. MMORPGs have thorny and interesting trust issues.

RIK: In my experience, programmers focus on the functional requirements of the software they have been asked to create. Just getting the software to come close to meeting the requirements is hard enough.

It appears that you are expecting programmers to think outside the box of the specifications for their software. Shouldn't there be a way that specifications could be written in a manner that forced programmers to expect aberrant behavior?

GARY: Yes. This is the main challenge faced by software security. We have had some early success among developers and architects building software

for financial institutions, and there is every reason to believe this can work for other kinds of developers as well.

In my book *Software Security* I spell out these kinds of best practices (called touchpoints in my parlance) in great detail. There is some brief coverage in the last chapter of *Exploiting Online Games* as well.

RIK: Can you imagine a game client that the gaming company could totally trust? Would doing so require hardware support?

GARY: I don't think we need to create completely trustworthy clients. What we do need to do is be much more skeptical about the data that untrustworthy clients send the server. If a client is being used to cheat, that should be made to stick out like a sore thumb at the server. That way, impossible goals like "create a completely trustworthy client" can be safely avoided.

Hardware support might help create a trustworthy platform, but these platforms would be trustworthy by the corporations that control them, not by the people who own them! In any case, approaches relying on hardware have also been successfully attacked (think satellite TV systems and the xBox).

RIK: You mentioned some advanced game-hacking techniques in your IT, for example, using multicore systems, and running the game-hacking thread on one core while the game runs on the other core. Have you actually succeeded in using this technique? How about running a game client from within a VM? Do gaming companies attempt to determine whether their software is running in a VM (or a debugging tool)?

GARY: We have a section in the book related to this question in Chapter 7: Building a Bot. Many of the techniques we describe there do work in the lab. We drew a line in the book and decided that it would not be helpful to the industry to release undetectable botting kits based on the techniques we outline. The fact is that we're not the only people thinking about these things, though. There are plenty of others writing and sharing game-hacking code all over the Internet.

An undetectable botting system is very valuable. All of the sweatshops in China (where people are paid a pittance to play the game and develop virtual wealth, which is then sold in a middle market) have created quite a demand for botting systems that are not yet banned (that is, that are not obviously detectable).

There are lots of things that remain to be explored, including rootkit-level bots, VM-based hacks, multiprocessor attacks of many kinds, and so on. It's important that the game companies understand just how dedicated to cheating some of their adversaries are.

My hope is that software security will progress nicely in the online game world and that will in turn teach us valuable lessons for building other software. Ultimately, I am a huge proponent of software security.

REFERENCES

[1] G. Hoglund and G. McGraw, *Exploiting Online Games: Cheating Massively Distributed Systems* (Reading, MA: Addison-Wesley, 2007).

[2] G. Hoglund and J. Butler, *Rootkits: Subverting the Windows Kernel* (Reading, MA: Addison-Wesley, 2005).

[3] Article on trust boundary issues: http://www.cigital.com/papers/download/attack-trends-EOG.pdf.

[4] G. McGraw, *Software Security: Building Security In* (Reading, MA: Addison-Wesley, 2006).