My experiences in the following presentation, and my status as a “subject matter expert”, are from my time working on product security during my employment at Rockstar Games, during the time of peak last-gen use. I’m prevented by NDA from revealing details about the titles I worked on, so to stay safe I’ve focused on non-Take Two game titles for the meat of this presentation, despite the fact that all of this info is considered “public knowledge”, due to my sources.

I will be using the terms “modding”, “mod”, and “modders”. These all are naturally are from the term modification, or modifying. When a game is manipulated to have its output changed for the user, this is referred to as a “mod”. The person applying the mod is a “modder”, and while running the mod, is in the act of “modding”. The terms exploit, hacker, attacker, etc. just don’t apply to this space all the time.

Just so it doesn't get lost in the references, my presentation materials: https://github.com/myfoostrong/busted-comp-preso

# #Busted Computing : Analyzing The Console Gaming Threat Model

The gaming industry is under attack, and has been almost since inception. The barbarians are forever at our gates, trying to steal our product, or trying to vandalize our infrastructure, or trying to harm our customers. The nature of the business is such that we invest the time and expertise and work-hours and money to create a product. And a profitable one at that[1]. We try to make our product secure. Then we discover the flaws in our product security only after those flaws have been breached by the bad guys. We create a patch, a fix after the fact. The bad guys are ever on the prowl for another weak spot. They breach. We fix. This dance goes on and on. In the gaming world that war is literally virtualized and the casualties often your users. This creates a bad user experience and can often kill a game if left unchecked. Console gaming took a unique approach with trusted computing platforms.Despite their promises as always no single control is a silver bullet fix.

So how do we defend ourselves from the spawn camping barbarians? Well, we must become them. This approach is far from revelatory, but the gaming industry often finds itself behind the times in terms of security. I plan to illuminate these commonly found weaknesses throughout the following talk, their root causes, as well as advice for game developers to mitigate said weaknesses.

## ##Gaming Threat Landscape

First off, I don’t think modding games is a bad things. Some great and beautiful things have been done by modder communities to games recent, and in the past. This following video exhibits a great example, and was before my time at R\* so I’m not breaking any rules[2]. Fairytale princesses knifing down pedestrians in the street. How could a publisher do such a thing without paying Disney millions of dollars for the right? Though I doubt Disney would ever let Elsa rock an MP5 like that…

While this artistic vision may be the evangelized mantra of the modder community most of the group sadly has more nefarious motivations in mind. They end up harming the user’s experience, and the publisher’s bottom line by disrupting gameplay. By looking at a video game from their eyes, we can figure out what they’re doing, and what motivates them. This better enables us to create patches that we know will work before they’re released, and not biting our nails waiting to see the effects.

### ###Threat Targets

In becoming the bad guys we must ask ourselves what it is they are after. Let us look at some of the common goals behind their actions:

\* Free Games: Who wants to pay $60 for a disk when I can just download it. We’ve all come across a pirated game before and I’ve always been for the open, free sharing of data and information. But after my time working in the industry, it's easy to see where all the money goes and why things cost so much. You can’t hire a talented team to build a beautiful game, if you don’t have the adequate resources to pay them in a competitive marketplace.

\* Scoreboard, Winning: The lazy competitors. Those who pay others to eat their cake for them. Or maybe they can’t cut it in the “big leagues”. Players can gain a competitive advantage over others, putting themselves artificially at the top of the list by hacking the list, or by using mods that enhance their abilities and gameplay, like wall hacks, aimbots, and no-clipping mods.

\* Notoriety: Less about playing the games, and more the mod development. The talented content producers in the community gain fame and respect from developing new mods used by the community. And the more popular, the better.

\* Profit: Some individuals will sell their mods and services to other community members. But this means a money trail, often being done over Paypal, which makes for easy tracking and litigation.

Griefing: These scourges of the gaming world seem to thrive on causing pain and misery. Disrupting and stealing from innocent players, they ruin a user’s experience. If all your players leave, who will want to play?

By realizing what the modders are targeting, we can focus on the key points to defend in a large attack surface.

### ###Threat Actors

The community targeting these games is made of groups of varying skill levels, all often referred to as "modders". “Hackers” implies too much skill or malice for most of them. Within this modder community, recognition and the respect of a participant’s peers are earned by the participant’s creation of mods. As in any large group, the “80-20” Rule is at work. Eighty per cent of the actual work is performed by twenty per cent of the people involved. In this case, the great majority of the mods are created by the minority of modder community participants. The remaining modder community participants, the majority of modders, contribute relatively little to the modder community's work output.

Let’s note that what we refer to as the modder community’s “work output” is sometimes, from the industry’s point of view, invasive, harmful and illegal. The great majority of modders are usually the one who create the harmful output. The great majority of modders are loud, ignorant, demanding leeches who often don’t understand what is really going on, But like skiddies they know how to use the tools but have little respect for the effects. Some of them attempt to profit from the productivity of the modder community’s minority.

The intent here is not to belittle the threat. While much of the population isn’t on average of an “advanced” skill set, many tend to be very active threats. I’ve privately coined the term SPT or “Stupid Persistent Threat” to describe some of the groups targeting gaming. The attacks may not be nation-state level, but they are damned persistent. The many-gigabit DDoS traffic we would see every week doubles or triples after a big ban wave. You won’t often find them breaking out of your web applications javascript sandbox, but if there’s a file on a web server protected by nothing more than an obscure naming convention, someone will find it. I’ve never worked in an industry as hostile as gaming. On one hand I rarely needed to justify the severity of vulnerabilities I identified. On the other, those bugs that slipped through, we found out about pretty quickly.

The modder community at large exists on web forums usually dedicated to gaming or specifically modding like Se7ensins, NextGenUpdate, PS3Hax and MPGH to name a few[3]. To understand what they are doing one needs to pour through the conversations being had. The first post detailing the mod in question is of import, but so is the community’s reaction and reported use. Most of the posts are drivel and trolling, but one must thoroughly sift through the haystack to find the few needles of valuable information from that 20 percent. Information about the mods being built, the custom built tools being used, information about the modders themselves, and more. You can’t just dedicate anyone to this, as an initmate knowledge of the mechanics of the game in question are needed to tell the difference between what’s an hypothesis of an ignoramos, and someone’s results of legitimate reverse engineering.

Their techniques don’t fall in line with those of many security practitioners either. For instance, Charles Proxy [4] was a tool I had never heard of before my time in gaming. I’ve always been a Burp Suite guy, but I see how some of the automated functionality of the tool would be beneficial to someone who doesn’t understand the HTTP protocol, plus its scripts being an easy read and portable.

### ###Network Session State, and Peer-based sessions

Games are just like any application, and essentially boil down to a complicated state machine. A network game must synchronize such a state between multiple peers, also known as the session state. Each user’s input must be consumed to calculate this state while concurrently keeping every user in the session updated about said calculated state. Each peer then must render their own version of the session state in real-time. All first person shooter players will know those moments when “there was no way I didn’t shoot that jackass in their fart huffing face!!11@#@!!!!!”. You may be pleased or perturbed to hear that you might actually have put a bullet in that flatulence breathing avatar’s head. It could also have been that the machine with session state authority disagreed with your system on the avatar’s location and disregarded your system’s claim of a valid hit. The way a game is architected determines where authority for this state lies, and that authority decides what actually happened.

Modern gaming consoles provide a UDP-based peer-to-peer networking system for games. The session state authority is one of the session peers, usually chosen based on their connection’s quality of service (QoS). With so many peers communicating over consumer (low) quality network infrastructure little time can be spared for validating this exchange of state. As such most or all input from peers is considered honest and acted upon without question. The situation can become even more dire when the malicious peer becomes the host, and now the source of truth when it comes to session state. Depending on the game’s architecture even an innocent client may gain additional perks or privileges. Halo 2 fans will remember the standy-by or lag switch attack that to this day allows the host to freeze their peers by unplugging their router[23]. And Gears of War players can’t forget the reaction speed advantages gained by the host player[24], I sure can’t forget their screaming about it.

### ###David v Goliath

Does anyone remember why 2011 is significant for Sony? How about issonydownnow.com? When GeoHot released the first jailbreak for the PS3, Sony tried to come down on him with the litigious hammer of civil violations of the DMCA, computer fraud, and copyright infringement[5]. And how did that turn out?

“We discovered that the intruders had planted a file on one of our Sony Online Entertainment servers named ‘Anonymous’ with the words ‘We are Legion.’ ”

Sony was the target of internet ire for some time, resulting in outages, breaches of user data, and more[6].

It never looks good being Goliath in any situation. Gamers are notoriously fickle and will abandon games for much less. Doesn’t help at all that most of the actors in question are minors. The end goal is trying to enforce a healthy user experience for your normal players and publishers must tread lightly to avoid being heavy handed. Segregate off modders instead of banning your user. Keep the bad guys away from the innocent but not away from your game.

### ###Defending against your customer

These modders may be referred to as threat actors, but most of these individuals are customers of the organization doing the defending. When you ban this player who’s been cheating, you might be sacrificing one for the good of the group, but you’re still removing a paying customer out of your concurrent user count. As a security engineer you need to design a way to block people from accessing a system that is designed to be accessed by people. Basic software security, but gaming provides its nuances.

The modders are using the application itself to initiate deep, complex business logic functions that cause intended effects within the game, effects that are expected to happen within the game naturally. Just not in the right context, and this context is what you need to quantify. Cars are expected to be spawned within a driving game. A player is expected to be crushed when located under a car. What’s not expected, is them to be spawning above a peer’s head, crushing them when they land. To detect the bad guys you must identify their footprint in your telemetry.

And what about false positives? A game is a complex application which naturally has complex (read: high volume) output, so a lot of legitimate noise to the signal we’re hunting. When acting on an event, you need to be sure to identify false positives, as punishing an innocent user could have drastic effects on the title’s anti-cheat program, coming under tons of public scrutiny. It’s not just a pop-up on the user’s screen, asking if the file is legitimate or not. Banning a paying customer who hasn’t violated your terms of service opens an organization up to potential lawsuit. Not to mention the lack of faith in the developer’s cheat prevention program, both internally and externally.

## ##Trusted Computing

I describe Trusted Computing (TC) as a hardware or software platform that enforces the separation between “user” and the “kernel” or “system” space, and can be used to perform sensitive and trustworthy computation. Modern day operating systems (OS) on a PC do have levels of separation from the kernel, but this is to improve user experience, and a knowledgeable user can bypass. If a hypothetical TC platform was a perfect control, then data on the platform would be 100% trustworthy. But our community of security professionals knows better than anyone how much we can trust “perfect” security. But I’m not here to talk about platform exploits. The problem is the developers who were promised a secure system are often caught unaware and undefended when the supposedly secure platform is eventually compromised. We’ll now go into some specific examples of TC, in mobile computing, and console gaming platforms.

### ###Mobile Computing

Modern mobile operating systems only run programs signed by a trusted authority, which are obtained from a trusted store that performs security testing on all the applications submitted to it[7]. Additionally, these application processes are usually run in a sandbox, or a private space in the system memory and resources. A great example of commoditized trusted computing. But vulnerabilities have been identified allowing users to “jailbreak” or “root” their phones. This enables system level access privileges, running of unsigned programs, and use of other app stores[8]. A simple look at the OWASP Mobile Top 10 will show a majority of the listed vulnerabilities relate to either unprotected interprocess communication ( M1, M2, M3, M4 ) or weaknesses exposed by permissive user access ( M5, M6, M8, M9) [9]. One success story of TC is until 2015, only five malware-infected apps had made it into the iOS App Store, out of 1.5 million total applications. The 39 applications identified in 2015 were due to use of a counterfeit version of XCode that installed a credential stealing trojan within every application developed[10].

### ###Console Gaming

Modern gaming consoles are only able to run signed programs, binaries and physical media, all solely obtained from trusted sources[11]. The "last generation" consoles produced by the leading manufacturers (Sony, Microsoft, Nintendo) have all been compromised in the past[12]. This led to rampant piracy, occasional disruption of gameplay and an overall hostile player experience when left unchecked, all of which directly affect a publisher's revenue streams, and therefore important to the business.

### ###Developer Consoles

It may not come as a surprise when I tell you developers don’t build games on the same systems their players do. These systems are often large, bulky and more powerful versions of the consumer console itemized and distributed by the manufacturers to contracted developers.

These machines give their users full admin capability within the system and provide connection to privatized online network environments segregated away from ousiders. Functionalities include file transfers, memory read/write, processor debugging capabilities, and much more. Then many orgs will create and maintain their own game engine and associated developer toolkit[13], or something based off popular framework like Unreal, Unity or Source. All of this allows a developer to build and debug their games. This same functionality can allow a modder to do just the same with their mods.

These consoles are numbered and tracked by the manufacturer, and signed for by the publisher. Yet as with most things in this life one of them falls off the back of a truck or disappears every once in a while. They are coveted by the modder development community, and will fetch a high price on many markets.

Disk Drives

Some of the simplest console mods entail replacing your console’s disk drive or drive firmware, to allow running disks without a manufacturer “signature”[14]. This allows for piracy, and modifying disk files. This technique known as ISO modding was named for the ISO disk imaging format often used to capture disk content.

A more advanced option called the xKey or 3key essentially plays man-in-the-middle with your disk drive, and sideloads ISO images off a USB connected hard drive[15]. The device pulls the required keys from the drive, and then loads the disk content from the hard drive. This speeds up mod loading and development while also saving money on copied disks.

### ###Hardware Exploits

The Xbox 360 models are compromised using different hardware exploit methods referred to as JTAG, and RGH[16]. In either situation a circuit board is connected to the motherboard, which allows for the loading of unofficial firmware.

JTAG stands for Joint Test Action Group, which is an industry standard for testing printed circuit boards. The JTAG exploit takes advantage of the JTAG debugging connections on the Xbox motherboard still being live. This exploit is only available for older generation consoles as Microsoft began removing the JTAG connections in later models.

RGH or ‘Reset Glitch Hack’ exploits a race condition during boot, constantly sending a reset pulse that changes the way the code runs, resulting in bootloader functions such as memcmp to always returning true, or “no differences”[17]. For instance, a comparison of two binary signatures will always report being equal during this time. Put simply it can’t tell that the custom bootloader looks any different than the legitimate one, and runs it.

The current PS3 jailbreak is a software one, but is only applicable to a certain firmware version of 3.55. When a console has been updated, its not permitted to rollback to an older version of software. To get around this, one can use products like an E3 Flasher to forcibly write the vulnerable firmware to the console memory’s contacts[18]. The system is then simply exploited by loading a software update exploit off a USB drive.

### ###Firmware

The end goal of hardware exploits is to be able to run custom firmware on your console. These can run pirated games, specialized bootloaders, game mods, media servers, FTP, Web... the list goes on. Since unsigned code is allowed anyone technically capable can develop firmware and software to run on these machines. The following is a non-exhaustive list of some of the popular tools modders have been using. Admittedly some of the nomenclature here may not fit the correct definition of “firmware”. Sue me.

Most if not all of the Xbox 360 exploit guides use a second-stage bootloader called Xell-Reloaded, a new release of the Xenon Linux Loader (Xell)[19]. A bootloader is simply a piece of software used to load an Operating System. With this modders load window managers that are used to load and run applications and games[49]. Certain console exploits can also run Xecuter Fusion, which claims to provide all the functionality a developer console would, but better[20].

For the PS3, the firmware is split up into three separate groups, original firmware from Sony (OFW), custom firmware (CFW/CEX), and developer firmware (DEX)[21]. The last of these is supposedly development firmware from Sony. This allows all the standard features from a developer kit, including using the Sony Target Manager API. These don’t play well online but are naturally very helpful for mod development. The CEX firmware is what most modders use, and is what most software is written for. There are three major players in this space, Rebug, Rogero and HABIB[22].

## ##Console Gaming Threat Model

As an application security engineer, I can't resist a chance to evangelize threat models. In my opinion they can be one of the most powerful tools for producing a secure environment if used correctly and early in the whiteboarding phase. Console video games are simply applications with weaknesses like any other. By focusing on where the bad guys are attacking from we can fix the root of the problem instead of just mitigating the current symptoms rendered by the victim’s clients. Console gaming with its restricted exposure, unique attack vectors and (modder) lack of industry support, creates a unique and narrow threat model worth studying.

### ###Attack Vectors

The games themselves can be modified just like any other computer application, but the delivery mechanisms are different from a PC. Below is a list of the standard attack vectors. I have documented all the standard attack vectors into a graphical threat model for this presentation[25]. Knowing where these attack points lie, allows us to target the same choke points with our defensive controls.

\* File Modification: Simple modifications may target files on disk, changing a graphic texture, or increasing the damage stat of a certain item’s integer value in a JSON or XML text file[26].

Ex: Changing an XML file that contains in the name, cost value pairs of items, lowering the in-game store costs. Swapping a graphics texture turning a sports car into the 60’s-era batmobile.

\* Binary Patching: Each console type has its own CPU architecture, so a specific instruction set is needed for disassembly. This is complicated, and usually only used to bypass simple checks[27], though for a skilled actor would be endless.

Ex: Disabling an OS version check that crashes game at launch.

\* Memory Manipulation: The community has built tools allowing for remote read/write access to the system’s memory by running a network API server. PS3 has a community-developed API called ControlConsole API (CCAPI)[28]. RGH and JTAG users are able to benefit from what seems to be a pirated version of Xbox Neighborhood, or XMDB[29]. Without deeper inspection this seems identical to the official 360 devkit SDK. These are often used to find string literals and integers in the game process’s memory, and writing values to known addresses and offsets.

Ex: Find memory offset of the player’s health value

\* Process Injection: Modders can have portable binaries sideloaded along with the game process and can interact with the running process like an injected DLL. PS3 uses a file extension called SPRX[30] and the Xbox 360 executables XEX[31]. These files are very portable with little installation time required.

\* Business Logic (Scripting): Many modern games utilize an internal logic or scripting engine that allows their developers to make changes to the game, without having to compile new binaries, for ease of development, if nothing else. Many game engines use commonly found scripting interpreters, like Python, Perl, and Lua[32]. Others cook their own[33].

Eventually someone reverse engineers this functionality. Often script files are in plaintext if not protected, and filled with logical syntax and semantics. The attackers now have a platform to conduct programmatic attacks against the game logic itself using internal functions within the game. Anything the developers allowed themselves to do with said engine, the modders now have access. These scripts often allow “hot-swapping” scripts, speeding up mod development time. In the PC Gaming world we see modders just bolt on their own scripting engine[34] .

Ex: Injected script that runs spawn\_vehicle(), with input of get\_avatar\_gps(target\_player\_id), crushing peers with cars out of thin air.

\* Server Attacks: Most online games today use some form of network services. Telemetry, matchmaking, recording scores, player profile, inventory management, and game logic engines as well. Many are web based, though some may still use proprietary protocols. These servers are subject to all the same pitfalls and weaknesses of standard server applications. Transport security, injection attacks, authentication, and of course tons of juicy business logic.

It may come now as no surprise that server attacks can even originate from a compromised client, where validation is often not performed, as the source is "trusted". These can make their way up to web servers displaying the game data to the public at large, bypassing the usual controls in place for Web-originated input.

Ex: SQL Injection, CLI Injection, Privilege escalation

\* Peer-to-Peer Attacks: Though usually piggybacked on one of the previous channels, the effects of attacks against game peers warrants its own section. Some games respect commands sent from peer consoles, as these usually should originate from the process context. As discussed earlier, many multiplayer games utilize peer-to-peer communication protocols for game state to avoid latency issues.

Both platforms provide a UDP based peer-to-peer channel for the consoles to communicate with each other[35]. Some modders have used this channel to identify the IP’s of their peers and sometimes flood them with traffic, effectively removing them from the session[36]. I haven't seen mention of anyone in the modder community discussing analyzing these comms, but I believe there's a treasure trove to be found for anyone willing to reverse the protocols.

Ex: Identifying a peer's IP, Targeting said peer with DoS attack.

### ###Presentation Demo

I've compiled a list of some of these different types of modifications to demonstrate for my presentation[37], all using Call of Duty: Black Ops 2 (COD, BO2) on a jailbroken PS3 running REBUG 4.82. The scripting language used by many COD games, referred to as GSC being named for the file extension used by the COD developers, has been researched and documented extensively by the community [38]. This often happens as developers, being good engineers, re-use and update the tools they built for previous titles, leading to modifications having some or even complete forwards-compatibility with newer titles by the same publisher[39].

I will first walk through how one might implement the mod we saw in the video earlier[37.a]. Then using a CCAPI simple memory scanning tool, we will search running memory for any SSL certificates in running memory, to attmept to bypass any SSL pinning that might exist. A sideloaded binary Error Reporter will help us debug our final demo, a GSC Script Menu that performs certain functions I found entertaining or interesting.

By understanding what exactly the mods are doing to the games, we can better understand our adversary. With this knowledge we can build an application better equipped to deal with whatever the modder community throws at it.

## ##Looking into the future

Both leading "current generation" consoles, the PS4 and XboxOne, currently don’t have a publicly known method of loading custom firmware. There have been partial exploits published for both systems, so new "jailbreaks" may be just around the corner.

### ###PS4

A researcher going by the handle CTurt had discovered that a vulnerable version of the Webkit library was being used by the PS4[40]. He released a detailed walkthrough of the development of a kernel exploit, allowing them to load a Linux OS from USB. This applies to a specific, older version of the PS4 firmware than is currently available on the market today. And there is also no known method of firmware downgrading. The OS is also known to be a variant of FreeBSD[41]. He consequently has released a “PS4-Playground” library for interacting with a compromised system like his[42].

The Sony developed PS4 Software Development Kit (SDK) was also released to the public by one who goes by the handle KromeMods[43]. Sony has been trying to keep it under wraps with DMCA violations[44], but once it's out there… it’s out. This may not necessarily lead to a system compromise, but it will allows the community to start working on tool sets.

### ###XboxOne

There has been at least one exploit published for the XboxOne, although it was patched some time ago[45]. It was based on the "Chakra" exploit and another MS Edge vulnerability (CVE-2016-7201, CVE-2016-7241). The exploit was supposedly not able to run executables, and no kernel level exploit was announced. The XboxOne runs a Hyper-V hypervisor OS, which runs both the XboxOS which runs the games, and another OS based on the "Metro" UI, which now runs on all Windows platforms[46]. Microsoft is even urging developers to move onto this new development environment[47]. Suffice to say, when you find an exploit in one product, it very well may apply everywhere its used.

It’s worth pointing out that both of these exploits actually had nothing to do with the hardware or the proprietary, game-playing OS. Both were vulnerabilities identified in the included web browsers identified previously by research completely unrelated to gaming consoles. I’m just reminded of the “Everything is better with (Bluetooth|Bacon)” principle. As we continue to add more and more functionality to devices, the more we expose them to potential points of compromise.

## ##What is my point?

Trusted computing is not a broken concept. I’d say it has been proven to be an extremely effective security control while in the hands of a “benign” user. What I am trying to point out is that no security control is iron-clad, especially not when in the adversary's hands, and that this fact should always be taken into consideration when developing on a trusted computing platform. So here's my advice to developers building any kind of application, especially those on such platforms.

### ###Security in Layers, Trust No One

Encrypt, obfuscate, validate and repudiate. What's already become an age-old adage for network security directly applies to applications, especially ones as complex as video games: Security in layers. The "trusted" platform will provide controls that should be utilized by the application. But any hardware in the hands of the public cannot be considered 100% trustworthy and data sent to and from such a client should always be validated at every trust boundary in the system. Assume the console will be compromised. Encrypt and obfuscate valuable data in memory. Validate incoming data sent by the clients, and have the clients repudiate the data coming from the servers.. Identify anomalous actions and data, and act on it.

### ###Bake In Security

I bring up threat models earlier because of the effectiveness they can have when applied at the beginning. If you build security into your application from the beginning you can save a lot of duct-tape headaches after you've launched. Threat model your application, and make sure to circle back to it at every stage of your software development lifecycle. The game should be designed to enforce the security of the system. Build your session state into integral parts of the game, to prevent tampering. What happens when a player is “kicked” from the session? Does it simply tell the player’s console to leave the session? What happens if the console ignores the command? Are the session peers aware of the player being booted? Do they act on this fact?

### ###Build a Player Profile

Your game should have some amount of data being sent to your backend. More often than not this is the only data you will have to act on when identifying a threat. You need to learn to work with the data you have on your players to try and detect when anomalous things happen. Build a profile and baseline for standard gameplay. Try and work with the development team responsible for the data. Once you understand enough of what is being collected already, you might be able to ask for more information to be funneled back to the home server.

### ###It’s Just a Game

Too often security practitioners fall into the expectation of perfect security. The most important fact is that the application being secured still needs to operate as intended. When we’re securing a video game, we need to make sure that it stays an enjoyable, playable game. It doesn’t matter how secure a game is if no one wants to play it. Video games are an artistic endeavor, and we must be sure not to become a roadblock in the creative flow. Too often as a professional have I seen organizations shooting themselves in the foot to get “perfect” while sacrificing “good”. And in the end all that is left is the current state of awful.

Security shouldn’t be about perfection, but about obtaining the most secure state possible with the resources available. Gaming companies are notoriously cost adverse and security is no exception. But the restriction isn’t always monetary. Games are one of (if not the most) resource intensive consumer computer applications on the market today. They often will exhaust the resource limits of consoles, forcing other functionality to be pruned out for performance. A good security practitioner needs to find ways to enforce security, while not draining system resources.

### ###Build a Modder Toolset

So you’ve implemented a bunch of controls and protections, but how does one know that they work? Building a tools suite that gives you the same view as those pulling your game apart. But no need to start figuring out how to build out a lab of modding machines, find the right mods, all without compromising the company. Take advantage of those powerful developer kits! Build memory scanners to look for specific values. Inject rogue scripts, manipulate network files and more. Target known data structures in memory by building easy to use tools. Use these tools to enable some of your QA and testing staff to help enumerate options. Games are complex programs with a wide attack surface to cover, leading to publishers employing armies of QA analysts. Figure out how to integrate some of your testing processes into their runbooks, and multiply your efforts.

I’ve written and shared a simple memory scanner for the PS4 SDK, that operates like a basic CLI version of the popular Windows-based tool Cheat Engine[37]. This falls short of tools like NetCheat[48], built to emulate those functions using the Target Manager and ControlConsole API's, the Sony and REBUG-based network API’s, respectively.

### ###Have Someone Else Take A Look

It’s a long established practice of making sure that the person inspecting a system for flaws, is not responsible for the design, construction, or management of said system. When we’ve become immersed or acclimated to a process, it can be hard to keep an open mind about things. A good, secure developer would fix any flaw they saw in their system by design, and often their intimate knowledge of the application would have them missing the forest for the trees, so to speak. It doesn’t have to be a security consultant like those at Lodestone. Not every publisher can afford to employ a full-time Security department, but many mature AAA developers employ their own internal dedicated team, who work alongside their developers.

# #References

**\*Any starred references should be considered integral sources to the paper**

1. GTA V Becomes Highest Grossing Media Title of all Time - https://wccftech.com/gta-v-highest-grossing-media-title/

2. **\*Youtube Frozen GTA - Let it Go - https://www.youtube.com/watch?v=DO1\_zzinQi0**

3. Online Web Modding Forums

a. Xbox 360 Modding & Tutorials - https://www.se7ensins.com/forums/forums/xbox-360-modding-tutorials.43/

b. PS3 Mods and Cheats - http://www.nextgenupdate.com/forums/ps3-mods-cheats/

c. PS3 Jailbreak & Custom Firmware - http://www.ps3hax.net/forumdisplay.php?f=9

d. Call of Duty 9 - Black Ops 2 (BO2) Hacks & Cheats - https://www.mpgh.net/forum/forumdisplay.php?f=616

4. Charles Web Proxy

a. Charles Web Debugging Proxy - https://www.charlesproxy.com/

b. Charles iFruit All Unlocks - https://www.se7ensins.com/forums/threads/ifruit-charles-all-unlocks-custom-plates-etc-no-device-needed.1073623/

5. Sony v Hotz - https://en.wikipedia.org/wiki/Sony\_Computer\_Entertainment\_America,\_Inc.\_v.\_Hotz

**6. \*Sony Response to House - https://blog.us.playstation.com/2011/05/04/sonys-response-to-the-u-s-house-of-representatives/**

7. Mobile Application Store Differences - https://www.testbytes.net/blog/11-differences-between-ios-and-android-mobile-app-testing/

8. 3rd party app stores

a. Cydia - https://en.wikipedia.org/wiki/Cydia

b. 10 Google Play Store Alternatives - https://fossbytes.com/10-google-play-store-alternatives/

9. OWASP Mobile Top 10 (2016) - https://www.owasp.org/index.php/Mobile\_Top\_10\_2016-Top\_10

10. More Details on the XcodeGhost Malware… - https://researchcenter.paloaltonetworks.com/2015/09/more-details-on-the-xcodeghost-malware-and-affected-ios-apps/

11. LastGen Console Stores

a. Xbox Marketplace - https://marketplace.xbox.com/en-US/

b. Playstation Store - https://store.playstation.com/en-us/home/games

12. \*Modern Game Console Exploitation - https://www2.cs.arizona.edu/~collberg/Teaching/466-566/2012/Resources/presentations/2012/topic1-final/report.pdf

13. Developer Tools

a. Xbox 360 Tools and Middleware - https://www.xbox.com/en-US/developers/xbox/tools

b. PS3 Dev Tools - http://www.psdevwiki.com/ps3/Dev\_Tools

14. Disk Drive Flashing Guide - https://www.se7ensins.com/forums/threads/%E2%96%88%E2%96%AC%E2%96%BA%E2%96%BAnooberts-flashing-guide-phat-slim-burner-max-payload-burn-games%E2%97%84%E2%97%84%E2%96%AC%E2%96%88.1150969/

15. xKey Manual - https://www.scribd.com/doc/74837835/Xk3y-Manual

**16. \*Xbox 360 Ultimate Exploit Guide - https://www.se7ensins.com/forums/threads/jtag-rgh-r-jtag-xbox-360-ultimate-exploit-guide.804054/ (Auth wall, I will get screenshots/local copy soon)**

**17. \*Reset Glitch Hack - http://free60.org/wiki/Reset\_Glitch\_Hack**

18. E3 Flasher Tutorial - http://www.ps3hax.net/2011/10/tutorial-install-e3-flasher-on-ps3/

19. Xell - http://free60.org/wiki/XeLL

20. Xecuter Fusion - http://team-xecuter.com/xecuter-fusion-v1-0-9-released/

21. PS3 Custom Firmware List - https://www.se7ensins.com/forums/threads/what-does-it-mean-ofw-cfw-mfw-dex-cex-sex-debug-rebug.961366/

**22. \*Installing PS3 CFW - http://www.ps3hax.net/showthread.php?t=56205**

23. Standby/Lag Switching

a. Halo Cheating - http://halo.wikia.com/wiki/Cheating

b. Lag Switching Returns - http://totalgamingnetwork.com/content.php?2374-Lag-Switching-Also-Returns-to-Halo-2-Anniversary

24. Cliff: GoW2 fixes annoying shotgun host advantage - https://www.engadget.com/2008/06/19/cliff-gow2-fixes-annoying-shotgun-host-advantage/

**25. \*Console Game Threat Model - https://github.com/myfoostrong/busted-comp-preso**

26. File Modifications

a. Save Game Modification - https://www.se7ensins.com/forums/threads/gta-v-save-editor-by-xb36hazard.1321805/

b. GTAV Vehicle Model Swaps - https://www.se7ensins.com/forums/threads/release-public-iso-mods-online-1-15-model-swaps-update-drive-a-train.1156607/

**27. \*Binary Patching - https://shrinefox.github.io/guides/p5/patch-eboot**

28. CCAPI - http://www.enstoneworld.com/articles/view/6/ControlConsole\_API\_Reference\_Windows/

29. Xbox Neighborhood

a. Xbox 360 SDK Setup - http://www.nextgenupdate.com/forums/xbox-360-achievements-jtag/646780-setting-up-xbox-360-sdk-neighborhood.html

b. XBDM Plugin - http://team-xecuter.com/forums/threads/70079-XBDM-plugin

**30. \*SPRX File Format - http://www.psdevwiki.com/ps3/SPRX\_File\_Format**

**31. \*XEX Files - http://www.noxa.org/blog/2011/08/13/building-an-xbox-360-emulator-part-5-xex-files/**

32. Game Engines (with Scripting Languages) - https://en.wikipedia.org/wiki/List\_of\_game\_engines

33. http://www.nextgenupdate.com/forums/black-ops-2-gsc-mods-scripts/916615-gsc-resources.html

34. LUA Plugin for Script Hook V - http://gtaforums.com/topic/789139-vrelhook-lua-plugin-for-script-hook-v

**35. \*Determination and Exploitation of Potential Security Vulnerabilities in Networked Game Devices - http://dtpr.lib.athabascau.ca/action/download.php?filename=scis-07/open/walterridgewellProject.pdf**

36. DoS and DDoS attacks on Xbox One - https://support.xbox.com/en-US/xbox-one/networking/dos-attacks-faq#cd1abe6d835f45f0bdd22f58d1866265

**37. \*Presentation Github Repo - https://github.com/myfoostrong/busted-comp-preso**

**38. \*Call of Duty - GSC Script/Function Index - http://www.zeroy.com/script/**

39. GSC Scripting in Call of Duty Titles

a. Black Ops III - http://phabricator.aviacreations.com/w/black\_ops\_3/guides/scripting\_guide/

b. Modern Warfare - https://github.com/Call-of-Duty-Scripts/CoD4MW

c. Modern Warfare 2 - https://www.mpgh.net/forum/showthread.php?t=181635

**40. \*CTurt: Hacking the PS4 - http://cturt.github.io/ps4.html**

41. Playstation 3 System Software - https://en.wikipedia.org/wiki/PlayStation\_3\_system\_software

42. CTurt PS4 Playground Github - https://github.com/CTurt/PS4-playground

43. PS4 SDK Release - http://www.maxconsole.com/threads/ps3-v4-75-and-ps4-v4-50-sdks-was-leaked-by-kromemods.45004/

44. Sony DMCA Squashing - https://arstechnica.com/gaming/2017/07/sony-using-copyright-requests-to-remove-leaked-ps4-sdk-from-the-web/

**45. \*XboxOne "Chakra" exploit - https://github.com/unknownv2/ms-xb1-edge-exp**

46. Xbox One Hardware and Software Specifics - http://www.extremetech.com/gaming/156467-xbox-one-hardware-and-software-specs-detailed-and-analyzed

47. Universal Windows Platform

a. UWP on Xbox One - https://docs.microsoft.com/en-us/windows/uwp/xbox-apps/index

b. Xbox Live UWP Sandboxes - https://docs.microsoft.com/en-us/windows/uwp/xbox-live/get-started-with-creators/xbox-live-sandboxes-creators

48. Netcheat - https://github.com/Dnawrkshp/NetCheatPS3

49. Setting up RGH - https://www.realmodscene.com/index.php?/topic/5060-setting-up-rghjtag-console-from-scratch/