Identifying a Game Console Vulnerability and Potential Exploitation for eBusiness Attacks

Brian Cusack
Nickolas Robinson
School of Mathematics & Computer Science
Auckland University of Technology
Auckland, New Zealand
Email: brian.cusack@aut.ac.nz

Abstract

Game Information System (IS) consoles are used by millions of players at any one time to access the Internet and information resources. In this research we questioned the security of these systems and evaluated one identified vulnerability for the impacts on eBusiness IS. A number of systematic attacks were made in a controlled laboratory environment on a game system to evaluate the security. In general the system was robust and repelled the attacks. However one vulnerability was discovered and exploited to demonstrate how an Xbox or consoles with similar weaknesses can be used to access eBusiness web sites using false credentials. The demonstration emphasises the range of vectors and sources of vulnerability that may come to bear on an eBusiness IS on account of new and evolving IS applications. The paper demonstrates one vulnerability and the potential for exploitation. The steps to defend against such problems are outlined and suggestions made to improve current game console security.

Keywords

Game Console, Attack, Security, Spoofing, Vulnerabilities

INTRODUCTION

The stories surrounding the exploitation of game consoles and massive online gaming frauds have accelerated attempts to look for weaknesses in the scenarios of online gaming. In our research we found game consoles to be robust and resilient against many attacks but not all. There are a number of vulnerabilities that may be exploited but in general strong attempts have been made to protect game consoles from exploitation or illegal use (Langshaw, 2011). The motivation appears to be the protection of copyright and proprietary use rather than specific protection from a range of malicious exploits. The apparent motivation is understandable given the huge revenue market created by the consoles and the competition between the brands to differentiate in the market place. Game consoles have Internet connectivity and manage large quantities of data including personal identity data and data of monetary value. Most consoles provide opportunity to purchase product such as merchandise and software online. The content distribution method known as digital distribution is used where digital goods and services are offered to the consumer via various forms of digital storefront. The consequence is that the exchange of passwords, credit card details, and other valuables are regularly transacted. Such rich data can be attractive to both legitimate and illegitimate use and it can be expected illegal activities may also be prevalent (Xbox Live, 2012). In many respects game consoles are the new eBusiness platform that is sweeping the world (Tondel, Line, Jaatum, 2014).

In our research we attempted three attacks in a controlled laboratory condition on a console to assess risks associated with the console. These attacks were selected from the literature as being feasible within the proposed controlled environment. First we attempted to disrupt live game play, take play artefacts and to change login credentials. Secondly we looked at disrupting the game applications themselves and thirdly attempted spoofing session identities. Not surprisingly the heavy security prevented disruption to live game play. Also removing artefacts from play and changing login credentials for one player impersonating another were quickly shut down or prevented. Such security would be expected so that the integrity of the game and the integrity of the economic environment are maintained. However, we located one weakness in the web browser that came with the console that was open to spoofing attacks. A spoofing attack typically allows a user to take another user's session and to control the processes within a session (Sendi, Dugennais, 2014). Such activity allows the theft of private identity information, the transfer of money between bank accounts, reputation damage and the use of the console as an attack vector against other eBusiness web based entities. These are serious concerns but responsibility for the vulnerability may not be directed at the console as such. The vast majority of our attempts at exploits were rejected or defeated by the security features of the console. The web browser in the console used for Internet connectivity, game play and buying and selling was the identified weakness. Such a vulnerability can be treated by allowing regular upgrades and patches to the console.

The paper is structured to review a relevant background literature and to establish security definitions for consoles and the related applications. The focus of the literature review is to be on spoofing attacks because this is where the vulnerability was found and it is the risk that requires treatment. It is noted that the literature in the area of Game Console security tends to be proprietary with only general security concerns being addressed in the academic literature, such as network security, device security and so on. The security penetration tests are then described and the results reported. The results show a vulnerability that may be treated to prevent exploitation and the implications for eBusiness attacks is discussed. The paper concludes with suggestions for hardening game consoles from such vulnerabilities and recommendations for further research.

DEFINITIONS AND BACKGROUND LITERATURE

The literature for Game Console security, exploits and patches largely appears in web based information forums. These forums include Wikis, News releases, developer postings, and proprietary promotions and so on. Consequently our literature selection reflects the current state of communications in the area. A review shows Game Console IS security is strong. Game Console systems are removable and the media located upon them can be changed in a multiplicity of ways. Consequently strong security resources are used to protect copyrights and digital rights. Many games now feature downloadable content or can be entirely downloaded via the various marketplaces online. Hence, all of the current generation consoles use some form of unique identifier to prevent copying of code, and to prevent unauthorized programs from being run upon them (Beer, 2012). The exact mechanism varies depending on the system in question. The Nintendo Wii utilizes a One Time Programmable memory chip, consisting of thirty-two four-byte words. The Xbox 360 uses a method of signing all attached memory media via the generation of Console Security Certificates. The PS3 system uses a series of keys that are generated based upon a single key, located within the 'bootldr' portion of the Cell architecture, known within the system as per_console_root_key_0. This key is responsible for generation of sequentially numbered keys that are used at differing levels of the hardware and software. It is a direct descendant, key_1, being used for all subsequent key generations, while also being responsible for decrypting the unique id, or EID, of the system (Shrout, 2010).

There are three types of session hijacks: active, passive, and hybrid (Dittrich, 1999). In an active attack, the attacker replaces the victim in the line of communication between the victim and the target server/service. This typically involves preventing attempts at communication from leaving the victim, often via the use of distributed denial of service attacks. Passive is where the attacker allows communication between the victim and the service is allowed to continue, albeit monitored - this allows the attacker to gather information as it flows, essentially letting the victim do the work. Hybrid is where the attacker shifts from passive to active midway through the process, and possibly back to hide their tracks. The five steps that make up a session hijack attack are: locating a target, finding an active session, performing sequence number prediction, taking one of the party's offline, and then taking over the session fully and maintaining the connection (Vaughan, 2004). The tools used for this are packet sniffers – that are utilities or programs designed to analyze the flow of traffic through a network connection, be it wired or wireless. This is mainly done to find a target of interest, one with high traffic to exploit and to hide the additional malicious packets going in and out. The attacker sets up a direct connection between the attacker and either the target or the victim, by being on the same network. A wireless connection allows a roaming user to drop in and out, it also allows for a spoofer to sidle in and monitor the incoming/outgoing traffic for anything interesting (Reed, Taylor, Mackay, 2008).

The art of preventing spoofing involves attacking the limitations of the spoof itself. Key amongst these is the act of guessing the session sequence numbers. Unfortunately, the sequence numbers are generated via very simple algorithms, making guessing usually a simple process of monitoring and timing. This appears to be due to the lack of cryptographic requirements for TCP/IP connections. A recent standard development provides both an updated discussion on this very topic, as well as a solution, in RFC 6528. It improves upon the former algorithms via the use of a MD5 hash. However, this in turn suggests that sequence numbers can be avoided entirely, with the implementation of either IPsec as described under RFC 4301 or TCP-AO as per RFC 5925 (Ridgewell, 2011). The improvements are to replace the predictable sequence numbers with full cryptographic encapsulation of the data being sent, making traditional hijacking attempts far more difficult. Another option for prevention is the use of router stamping. This technique was originally developed to fight against distributed/denial of service attacks, using spoofed IP addresses in a similar fashion to the spoofed addresses used when transmitting the faked packets in a session hijack attack. It involves appending to the packets a fixed number of stamp slots, recording the routers and recording the ports used for the transmission. With deterministic stamping, this is done with the IP of every router encountered and produces large data (Xynos, Harries, Sutherland, Davies, Blyth, 2010). The suggested algorithm appends a certain probability of recording as the packet is sent, with a limited number of slots to be filled - the result is a slightly larger packet, but one whose path could be traced. The technique has a weakness with the possibility of one hijacker attempting a single assault, which means that they may be on the same 'path', router-wise, as the victim and hence have near identical router stamps.

Detecting Spoofing is extremely difficult. Despite much research, there is little to help the defender prepare for an attack in advance. The hijacker hides by simply allowing traffic to flow as normal. But this does not mean that it is impossible. In wireless networks the detection of unauthorized stations and access points can be made. For session hijacking in particular, the methods available are via detecting sudden jumps or randomness in the sequence numbering of the packets, or checking that the MAC of the hardware is valid. Also the Received Signal Strength (or RSS), and the round trip times of the Request To Send/Clear To Send (RTS-CTS) handshake can be monitored for variations. This has the potential to foil attacks before they happen (Xbox Live, 2014). The approach targets the spoof itself - an expected duration for a given handshake (the process of the two ends of communication recognizing each other) is calculated. As packets are sent, if they diverge from this predicted time and it is not a bad connection then it can be used as an indication that another source is then suddenly sending packets. Of note is that this method could be used similarly for smaller wired networks with few changes.

TEST SET UP

The test set up was kept simple and within a secure laboratory environment. The design consisted of an Xbox 360, a wireless router and an attacking PC (see figure 1). The action consisted of a game player using the console for play and services and an attacker who could monitor and catch the streaming packets going through the wireless network. In this way a wide range of attacks could be made on the console, the applications, and the actual games and on the communication streams. Session spoofing or hijacking could be achieved by catching a session ID and then assigning the ID to the attack vector. In some instances this worked well and in others it was picked up and defended against by the security provisions.

The monitoring PC was set up running Ubuntu Linux with the infiltration software suite Aircrack-ng. It was responsible for the packet capture as the 'victim' console went about regular use. Airmon-ng comes with eighteen related programs, all involving the infiltration of wireless networks. Of these, four were selected to infiltrate the wireless traffic and decrypt it into a format usable both by a potential attacker and in order to gather data on what traffic is coming from the game play. The four programs used were: Airmon, responsible for setting up monitoring of a target wireless access point; Airodump, which captures the packets and places them into several common packet formats, most importantly the common .cap format; Aircrack, which handles the actual cracking of a wireless network to provide its access key; and finally Airdecap, which handles the decrypting of the captured .cap format files to make them usable and readable. For the purposes of the test, the Aircrack program was provided with a word-list to be used in attempting to discover the access key. In this case, the word-list consisted solely of the access key, in order to speed up the process which may have otherwise taken many hours or even days. The resulting decrypted packet .cap file is still extremely hard to read using a common text editor. Consequently Wireshark was used so that packets could be viewed and examined. Using Wireshark, the decrypted packets were inspected for use and analysed to discern how the console transmitted data between itself and the servers and how these packets were identified as genuine, particularly during the initial login of the 'victim' to the services. In addition Wireshark was used to forge fake packets within the secure testbed to attempt to trick the console into accepting a login from the 'attacking' PC and to accept other fake interventions.

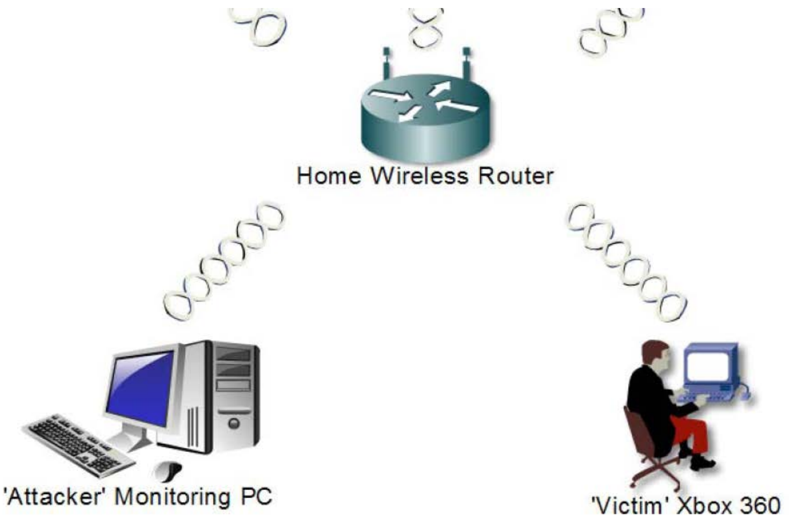


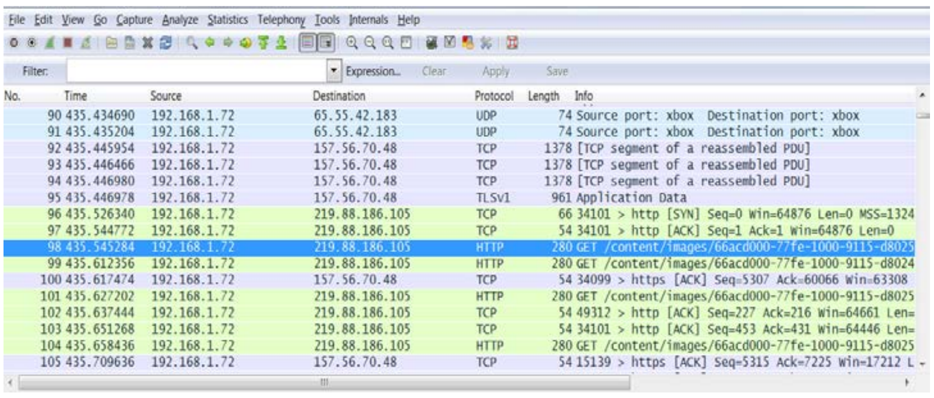
Figure 1. The Test Set Up

DATA ANALYSIS

There are three stages of data processing; firstly, a casual read through the decrypted packet file to see if any identifying features or words of interest can be identified. Following this, a keyword search was deployed, using Wireshark to attempt to filter packets containing information related to the act of logging into the Xbox Live service. Typically, these keywords are some variation on ‘login’, ‘pass’, ‘user’, or ‘session’; even more common words such as ‘hello’ would be of interest, as they are used as part of the typical process of two systems identifying each other over a network. Finally, there was an attempt to identify any patterns in the packets transmitted. All of this will be of use when it comes time to identify the weaknesses of the Xbox services, and what data it may leave open to a potential monitoring attacker. Once the attack has been completed, and the data from it has been collected, attention will then turn to analyzing the collected data for evidence.

The research environment presented a number of challenges that had to be resolved. HTTPS, SSL, and TLS secure protocols were the primary cause for change in the experiment’s design. SSL and TLS are both examples of asymmetric cryptography where the client and server exchange public keys to set up a session and are used to protect the game system. In this particular case, analysis of the captured packets indicates that the services rely mostly upon the TLSv1 protocol (the initial version of the TLS protocol that was released, and in turn superseded by the TLSv1.1 and TLSv1.2 protocols). Consequently, the TLS encryption meant that the data being exchanged could not be analyzed. It would have been possible to attempt TLS cracking or attempting to gain access to the consoles inbuilt private key but this ran beyond the scope of the research. The encryption issue affected spoofing attempts against the live processes and applications, which relied heavily on TLS for the exchange of non-vital data. As a result, the attention went onto the console’s built-in implementation of Internet Explorer. A simple monitoring of wireless traffic coming from the ‘victim’ (whose role was now relegated from game play to logging off and on) would be recorded and decrypted, with efforts focusing on unsecured/non-HTTPS traffic, in an attempt to obtain either a session ID or session cookie variable to be used by the monitoring PC, as per a more traditional PC-vs.-PC spoofing attempt. A change was also made to what portions of the captured packets were being inspected. Where previously all the traffic was inspected under the new approach only those that were of unsecured HTTP data and featured cookie information were of interest. This filtering process was made easy by the use of the Wireshark inbuilt packet filter that allows in this case, through the key phrase ‘http.cookie’. This automatically allows for http packets with cookie information contained within them to be identified, following which a manual search for words of interest (‘datr’, ‘session’, or some form of ID) can be done, or further refinement of the filtered list can be done by adding the term ‘contains’ followed by the words of interest.

Under the initial attack, the ‘victim’ console was repeatedly logged in and out through powering the system on and off, manually logging in and out of the services, and forcing the service to disconnect via de-authentication attacks. All three systems of attack produced similar packet numbers with the majority of these involving TLS encrypted communication. Of note, however, are the times where the Xbox 360 initiates HTTP GETs for static images, typically for ads or to display box art on the screen.


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression...
Clear
Apply
Save

```
No.   Time        Source        Destination     Protocol Length Info
   90 435.434690  192.168.1.72  65.55.42.183    UDP        74 Source port: xbox  Destination port: xbox
   91 435.435204  192.168.1.72  65.55.42.183    UDP        74 Source port: xbox  Destination port: xbox
   92 435.445954  192.168.1.72  157.56.70.48    TCP      1378 [TCP segment of a reassembled PDU]
   93 435.446466  192.168.1.72  157.56.70.48    TCP      1378 [TCP segment of a reassembled PDU]
   94 435.446980  192.168.1.72  157.56.70.48    TCP      1378 [TCP segment of a reassembled PDU]
   95 435.446978  192.168.1.72  157.56.70.48    TLSv1     961 Application Data
   96 435.526340  192.168.1.72  219.88.186.105  TCP        66 34101 > http [SYN] Seq=0 Win=64876 Len=0 MSS=1324
   97 435.544772  192.168.1.72  219.88.186.105  TCP        54 34101 > http [ACK] Seq=1 Ack=1 Win=64876 Len=0
   98 435.545284  192.168.1.72  219.88.186.105  HTTP      280 GET /content/images/66acd000-77fe-1000-9115-d8025
   99 435.612356  192.168.1.72  219.88.186.105  HTTP      280 GET /content/images/66acd000-77fe-1000-9115-d8024
  100 435.617474  192.168.1.72  157.56.70.48    TCP        54 34099 > https [ACK] Seq=5307 Ack=60066 Win=63308
  101 435.627202  192.168.1.72  219.88.186.105  HTTP      280 GET /content/images/66acd000-77fe-1000-9115-d8025
  102 435.637444  192.168.1.72  219.88.186.105  TCP        54 49312 > http [ACK] Seq=227 Ack=216 Win=64661 Len=
  103 435.651268  192.168.1.72  219.88.186.105  TCP        54 34101 > http [ACK] Seq=453 Ack=431 Win=64446 Len=
  104 435.658436  192.168.1.72  219.88.186.105  HTTP      280 GET /content/images/66acd000-77fe-1000-9115-d8025
  105 435.709636  192.168.1.72  157.56.70.48    TCP        54 15139 > https [ACK] Seq=5315 Ack=7225 Win=17212 L
```

Figure 2. Traffic During Xbox Testing

These HTTP GETs are, unlike the more secure HTTPS communication done with TLS, left unencrypted where all information is made plainly visible to both the receiver and sender. Typically, this sort of unencrypted traffic is used for obtaining similarly low-security items of importance; in this case, image files. While revealing the full address of where the images are coming from, they do not offer up anything in the way of allowing for session spoof attacks, especially as they were transmitted without any cookie information. Under the revised attack, the console was tested at three levels - firstly under the case of using the general features such as game downloads or account updates, secondly with the available applications that are made available through the marketplace, and finally through the Internet Explorer application that is made freely available to users. It is a simple web browser with some display and navigation limitations. The testing showed that the Internet Explorer application was transmitting everything in unencrypted HTTP traffic - including, crucially, cookie information. A particular packet was focused on soon after it was spotted, numbered 9626 in the decrypted packets. This particular packet was chosen for no specific reason, other than it appeared as part of the filtered Wireshark list of packets containing http.cookie information. The session ID was stored automatically within a cookie file. By using 'Edit This Cookie' the session identifier was assigned to the key name "session" and was able to be edited to match the session identifier given by packet 9626. The Browser was then refreshed and the saved cookie with the spoofed ID tricked the target into believing that the attacking PC was using the same login as the game console via the Internet Explorer. Hence a controlled spoofing attack was completed. The implications of such an attack is that the Xbox 360 can be compromised in the way described and used as an attack vector for any website accepting cookies and not defending the attack. The implications are for eBusiness sites that contain information of monetary and personal value that can be compromised by this attack.

DISCUSSION

Overall, the console services withstood the attacks quite well. The use of TLSv1 to encrypt traffic and the use of ports that were unique to the services ensured that the communications are quite secure, at the least when it comes to session spoofing attacks. While none of the applications used were observed to allow insecure, non-HTTPS traffic to be transmitted with the exception of the Internet Explorer, this does point to a lack of uniformity in the way the console application data packets are transmitted. However, two items of related interest were revealed during the investigation. Firstly, the use of unsecured HTTP GETs to obtain static image files for display on the console. While not exposing any information in and of themselves, they do show that the Xbox 360 has a potential for these packets to be intercepted and other images returned for display or other, more malicious objects returned such as steganographically encoded images or virally compromised files, and other malware. At the very least, it would mean that disturbing images could be transferred for display through the GUI, albeit in a very roundabout way. As the images appear to be cached for later display, however, this means that they could potentially be retrieved, allowing for the console to act as a go-between medium in transferring such images. In the latter case, that of a viral attack, the console would either already need to have been infected in some way with the 'base' virus in order for the malicious code within it to run, or the image would have to make use of buffer overflow errors to allow the arbitrary code to be run. Also of note was how the services reacted to de-authentication attacks done during the process of collecting the four-way wireless handshake in the initial experiment. While not useful in terms of spoofing, this sort of attack would have ramifications in the realm of e-sports; that is, in order to force the opposing team offline at a critical moment in order to secure one's own victory. This could be used to essentially rig competitions that rely on a wireless medium for connecting multiple users.

The Internet Explorer application however, shows that it is just as vulnerable as any other browser to the more simplistic session spoofing attacks. Beyond the spoofing results, however, the Internet Explorer application revealed one further item of interest; that of the meta-data containing the user-agent string. The user-agent metadata is used to help figure out if any special tricks are needed to be used to render a particular web page. While largely redundant due to the standardization of web code and websites over the years, there are some unique problems that still crop up for different browsers based on how they were coded. In this particular case, the user-agent of the Internet Explorer application reads as follows: User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0; Xbox).The key portion of interest here is how the browser identifies as Internet Explorer 9.0 - at the point of researching, the latest stable release of Internet Explorer was v11.0.9600.16438, two entire major releases later. This means that the application has missed out on years of patches and security additions that the PC version of Internet Explorer has received; and essentially, this gives an attacker a readymade list of exploits that could be used simply by referring to all the security bulletins and patch notes released over the intervening editions. The session spoof that was used is different than the usual and more technical method of sequence number prediction. The use of the cookie/session variable to trick the receiving server can only be done due to HTTP traffic transmitting what are essentially human readable values. The risk is that such vulnerabilities can be exploited to introduce backdoors into a target system and while not directly related to session spoofing, it would allow for an attacker to potentially introduce more direct means of accessing the console. A further risk is that the console can be used (knowingly or unknowingly) as a vector to attack websites that allow spoofed cookies.

The spoofing of cookies opens a range of potential IS and eBusiness attacks. In the first instance a player (and there are millions at any one time worldwide) could legitimately be playing an online game and they purchase an item online from within the game or they use the IS game console to purchase goods and/or services from any eBusiness store. Similarly they may only be bidding on an auction or simply surfing sites for information. The vulnerability of unencrypted GET functions from the web browser then allows an attacker to hijack any session for transferring information or if the player has used credentials to access a secure site the attacker can access the secure site content (such as bank accounts, user accounts, member accounts and so on). These matters are of concern for IS security and can be addressed in a number of ways discussed in the background literature. The outstanding matters in our experiment were the age of the browser supplied with the game console and the lack of updates applied to patch the vulnerability. In the first instance the manufacturer could enforce policy to assure all elements of the console are standardized and that they accept only encrypted traffic. In the second instance policies to enforce regular patching of the browser could be implemented at either the hardware or the software layers.

CONCLUSION

The detection of session spoofing is usually obvious and most sites protect against it. The cookie-based method applied here essentially dodges detection and is not influenced by the usual sequence number checks as it does not matter to the receiving server what packets have been received when, only that the session identifier matches what it is expecting. The cookie-based spoofing can be prevented by strongly encrypted traffic or if every session was being fully monitored, then the differing sources of the packets would be quickly identifiable. The second option is resource intensive and would put a lot more pressure on the servers in question as they track and compare the sources, sequence numbers, and travel times of the packets being received. Consoles provide a unique opportunity for deploying monitoring and encryption based hardware to defend against spoofing. Every console is manufactured identically, barring the rare hardware revision across a console's lifetime, applying hardware defenses could be applied straight to the console in the factory. The two defenses that would best benefit from this would be router-stamping and faux-login techniques. The former, router-stamping, could be ideal as the console's manufacturers would control both ends of communication; client and server. This would allow for end-to-end packet and routing tracing. Meanwhile, the faux-login approach could periodically attempt to send the fake login signals, possibly in reaction to suspicious activity in a console's traffic, seeing if anyone then takes the bait on offer. The research results point to weaknesses in the console tested but all of the vulnerabilities can be removed by a better thought out security strategy by the vendors and better end-user security for the whole system.

REFERENCES

Beer, J. (2012). "Rise of mobile gaming surprises big video-game developers", Canadian Business. Retrieved 4 April 2014 from http://www.canadianbusiness.com/article/75215--rise-of-mobile-gaming-surprises-big-video-game-developers

Dittrich, D. (1999). "Session Hijack Script". Retrieved 14 April 2014 from http://staff.washington.edu/dittrich/talks/qsm-sec/script.html

Langshaw, M. (2011). "Software Piracy: The Greatest Threat to the Gaming Industry?". Digital Spy. http://www.digitalspy.co.uk/gaming/news/a352906/software-piracy-the-greatest-threat-to-the-gaming-industry.html

Reed, J., Taylor, N., Mackay, J. (2008). "Advanced videogame consoles - a new and unrecognised threat to secure service provision?" *The British Journal of Forensic Practice*, 10(4), 15-18.

Ridgewell, W. W. (2011). "Determination and Exploitation of Potential Security Vulnerabilities in Networked Game Devices", Athabasca University. Retrieved 14 April 2014 from http://dtpr.lib.athabascau.ca/action/download.php?filename=scis-07/open/walterridgewellProject.pdf

Sendi, A, Dagenais, M. (2014). "Enhancing security of sessions in mobile networks using space caching". International Journal of Information Systems Security, 13(4), 355-366.

Shrout, R. (2010). "The New Xbox 360 S 'Slim' Teardown: Opened and Tested", PC Perspective. Retrieved from http://www.pcper.com/reviews/General-Tech/New-Xbox-360-S-Slim-Teardown-Opened-and-Tested?aid=940

Tindel, I., Line, M., Jaatun, M. (2014). "Information security incident management: Current practices as reported in the literature". Computers and Security, (45) 42-47.

Vaughan, C. (2004). "Xbox security issues and forensic recovery methodology (utilising Linux)". *Digital Investigation*, 1(3), 165-172.

Xbox Live (2014). Retrieved 14 April 2014 from http://koaku.com/5873604/is-microsofts-xbox-live-hacking-problem-worse-than-microsoft-realises

Xbox Live (2012). Retrieved 14 April 2014 from account login exploit - http://www.ign.com/articles/2012/01/13/update-microsoft-addresses-xboxcom-exploit

Xynos, K., Harries, S., Sutherland, I., Davies, G., Blyth, A. (2010). “Xbox 360: A digital forensic investigation of the hard disk drive”. *Digital Investigation*, 6(3-4), 104-111.

COPYRIGHT