ATHABASCA UNIVERSITY

Determination and Exploitation of Potential Security Vulnerabilities in Networked Game Devices

BY

Walter W. Ridgewell

A project submitted in partial fulfillment

Of the requirements for the degree of

MASTER OF SCIENCE in INFORMATION SYSTEMS

Athabasca, Alberta

September, 2011

DEDICATION

I dedicate this work to my family, those people who have been patient and supportive throughout this endeavour. Specifically, my wife Nancy whose encouragement and weekly coffee outings helped keep me rejuvenated and focused. My son and fellow online virtual world explorer Christopher with which much quality de-stress time was spent gaming and watching B-rated sci-fi movies. My darling daughter Carly, for her assistance as my in world beta tester of objects and supplier of sour gummies that important online exploration foodstuff. Lastly my faithful canine lab companion Taz, who never failed to remind me that the real world was good for exercise and fresh air and his 'its 1 a.m. Time for sleep now' non verbal admonitions.

ABSTRACT

An examination of the security aspects of networked game devices in relation to the current threat environment presented to networked wireless devices, and their associated connectivity to the Internet. Historical data regarding security threats and exploitation of vulnerabilities in personal computing devices and recently mobile devices indicate that once a technology reaches a maturity level in the realm of network connectivity and user adoption, that attacks and exploitation of such devices is typically inevitable. As growing numbers of consumer entertainment devices such as game consoles and portable game devices become part of a home computing network, having increased utilization as an Internet Appliance for surfing the World Wide Web, they will face the same online threat potentials as experienced by other devices. In this instance however, unlike personal computing devices and similar to many mobile devices today such as cell phones which implementing embedded processor operating systems and technologies, little if no software exists to provide an indication of attack or compromise by malicious software or users.

Utilizing the same analysis and penetration testing methodologies as for networked personal computing devices, this being the closest analogous hardware implementation and networked environment in which the devices are configured operate, the potential for exploitation will be examined. Collective results of this research begin to determine the state of vulnerabilities related to current console and portable game devices in their network active state. This will provide useful baseline information which can give an indication of the networked game devices security/vulnerability profile with regards to other online networked devices. In a broader perspective it outlines the possibility of exploitation of non personal computing network active devices and the threats they may present to a network infrastructure. Information from this research will form the framework of a 3D immersive game designed to educate the networked game device user to potential security risks, utilizing the appeal of such environments to engage the learner.

ACKNOWLEDGEMENTS

Thank you to Dr. Vive Kumar, my supervisor on this project for all his encouragement, support and helpful insights during my work on this research project. My gratitude as well to Dr. Kinshuk and the NSERC/iCore research group members, both faculty and fellow graduate researchers, for the opportunity to share work in progress and reap the benefits of the feedback from such a talented group. Thanks as well to the Athabasca University Research Centre, Graduate Students Research Fund for providing equipment resources and the Athabasca University Student Awards department for additional funding towards my research endeavours.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

There is in use today, a large and growing number of console and portable networked game devices being utilized for personal Internet access with the recent addition of web browsers to their operating systems capability. This new form of utilization, in conjunction with the incorporation of automated operating system maintenance functions, such as software updates or the ability to download and run new user applications, potentially exposes these devices to a number of new security vulnerabilities. As these networked game devices are provided with applications and capabilities that one would see in a typical personal computing device, so too do they become vulnerable to the same threats personal computer users experience.

While there is much data and ongoing research into the threats and vulnerabilities of networked personal computers (Richardson, 2008) , (Pouget, Dacier, & Pham, 2008) including recommendations of course content and structure related to the study of the subject (Holland-Minkley, 2006), there is little if any data related specifically to networked game devices. More recently, research on mobile personal devices vulnerability and security issues in devices such as cell phones and Personal Digital Assistants (PDAs) (Friedman & Hoffman, 2008) has begun to emerge. As with the personal computer systems, these devices are becoming more prevalent and adopting the usage characteristics of personal computing devices has exposed them to targeting by malicious individuals.

With the utilization of networked game devices both fixed and mobile increasing, the potential for attacks on these types of systems either by directed means or as unintentional collateral damage, as has been seen with other networked computing devices such as medical equipment (Ackerman, 2009) adds a new dimension to the types of devices that need to be taken into consideration when talking about networked computing security issues. If these game devices experience the same threat evolution characteristics as personal mobile devices and the personal computers that came before them, it may be only a matter of time before such an incident occurs, prompting this researcher to investigate this possibility.

Research Focus

The intent of this research is to examine the security aspects of networked game devices both console and mobile, in relation to the current threat environment that these systems would be exposed to in today's online computing environment. As is characteristic of today's home computing networked devices, associated connectivity to the Internet will be assumed. These devices will be assessed in regards to their operating charactersistics as a node connected to typical home computing network. The course of research will be examining their vulnerability to directed security threats, such as malicous content or applications, eavesdropping, spoofing or traffic redirection and unathorized access attempts and inducing device malfunction. The posibility of exposure to indirect threats such as aquired malware or subversive programs, exposure which may occur as the user accesses websites sites via browsing will also be examined. When completed, a clearer picture will emerge as to the vulnerabilty of these devices with

respect to the threat environment a typical networked consumer personal computing device would experience. As such this research will provide a basis for evaluation of these devices in light of current cyber security concerns while raising awareness of potential vulnerabilities introduced into a home or other network environment. Focusing on the end user of such devices, a proof of concept game developed in a three dimensional (3D) immersive simulation environment is developed to educate users of such devices to potential threats. The choice of a 3D world is similar to popular game environments experienced by the users of these devices and as such would be an engaging form of media for the educational exercise at hand.

Significance of the Problem - The Numbers Game.

Growing numbers of consumer entertainment devices such as game consoles (116 million worldwide)[1] and portable game devices (169 million worldwide)[2] are currently implemented in a home computing network. As additional network nodes seeing utilization as an Internet appliance, they will face the same threats and vulnerability potentials as seen with predecessor network capable devices. In this role however we also see characteristics in the devices similar to many mobile communications/computing devices today.

Implementing embedded processor operating systems and utilizing closed architecture technologies, running task specific software (games) and offering only a constrained

[1,2] From the earnings reports of Microsoft, Nintendo and Sony

interface to the user, little if any software exists to provide an indication of these threats or to protect the devices from compromise by malicious software or users. This is unlike the case of personal computing devices, which operate in the same networked environment and yet necessity has shown us that various forms of security software are necessary to protect the computing devices integrity and individual's privacy.

Historical data trends regarding security threats and exploitation of vulnerabilities in personal computing devices and more recently mobile devices (Chen & Peikari, 2008) indicate that once a technology reaches a maturity level in the area of network connectivity and widespread user adoption (large numbers of devices in operation), that attacks and exploitation of such devices is inevitable. Indeed, often the attacks increase in sophistication and numbers as time goes on, requiring a less knowledge on the part of the intruder to develop due to maturation and availability of information and attack tools as illustrated in Fig 1. (Carnegie Mellon CERT Coordination Center, 2003). Such developments broaden the number of attackers from those with a motive and technical skills, to those just curious to see what can be done by the tools provided having no specialized technical skills or potentail motive in mind.

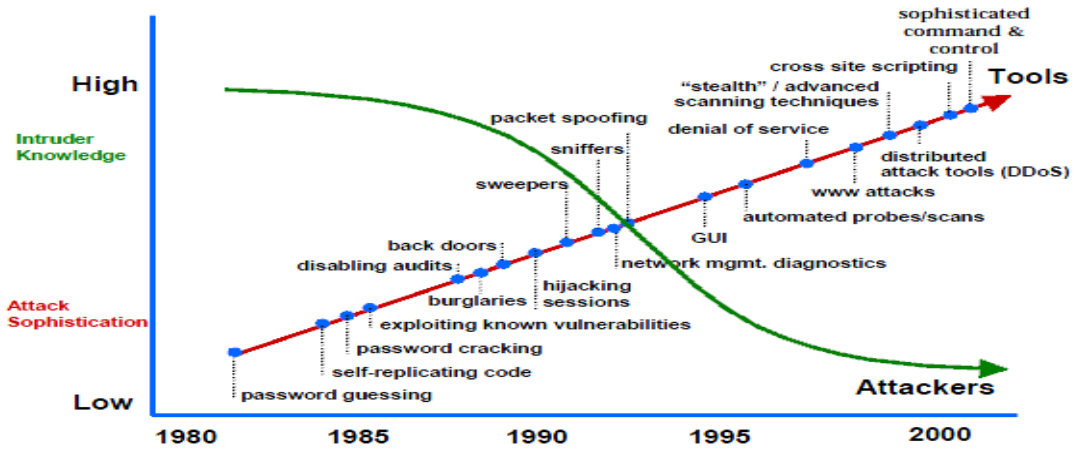


Figure 1. Sophistication Versus Attack Complexity Over Time (Lipson, 2002)

As seen with popular operating systems and applications, motivated attackers will look to exploit weaknesses in the original design or recent upgrades of such systems so as to utilize them for their own purposes. The preponderence to this particular method of attack is due to the devlopment nature of modern operating systems which seeks to add or improve functionality, this having the unfortunate side affect creating new opportunities for vulnerability exploitation (Lee & Davis, 2003). Additionally as operating systems designers attempt to repair vulnerabilities targeted by exploits, they can potentially open up new vulnerabilities, creating a cycle of exploitation/patching which then repeats. These same sofware deveolpment conditions currently exist for the networked game devices operating systems as well, lending weight to the speculation that similar vulnerability issues will be experienced. In fact many exploits to exisiting networked game devices in the regards to the aspect of running pirated software packages via exploit have been a result of such software upgrades or changes.

Another vulnerability that may be exploited is the operating system routines that access the Internet to do upgrades. These are often implemented in devices so as to provide the ability for manufacturers to add new features designed to augment or upgrade existing device functionality. These capabilities include tasks such as:

- Automatically downloading and updating or patching the devices core operating system.
- Updating installed applications to add capabilities or fix problems.
- Updating software drivers for existing peripheral devices or new devices.

This expansion of functionality creates the possibility of being able to introduce applications or operating system changes into these devices for functions other than those for which they are designed without the knowledge of the user. Tsow et al. (Tsow, Jakobsson, Yang, & Susanne, 2006) demonstrated this type of vulnerability with upgradeable wireless routers. By initiating and controlling the upgrade process, it was possible to install modified versions of the devices operating system giving control of the device to the attacker. Target systems for this attack where identified by the Organizational Unique Identifier (OUI) which utilizes the first three bytes of the Media Access Control (MAC) Address assigned to the network device. This hexadecimal number allows for the identification of a target device by its' built-in network card and by such means facilitated the ability to perform a device specific type of attack. Identifying target networked game devices systems with vulnerabilities in the operating system may be exploited through the same techniques, resulting in similarly compromised devices.

Other Factors Contributing to the Potential for Exploitation

With the advent of the home network and Internet access, recent generations of networked game devices have started to incorporated many features normally found in personal computers. The ability to play many forms of media, from videos to MP3 music files led to the inclusion of applications to facilitate the sharing of such media from personal computers on home networks. Thus, the concept of utilizing a game device as a media center or home entertainment system led to an evolution of usage beyond the standard game playing platform. The ability to access and utilize removable storage media, as well as portable MP3 players created new possibilities for the storage and transfer of data among these devices. Manufacturers progressively upgraded capabilities to include the ability to allow users access to data from the Internet for these devices, first in a controlled manner allowing downloads of content from specifically provided sites and later to an open format allowing individuals the freedom to 'surf the web' for content with the use of a browser. As such networked game devices are currently being utilized in the quite the same way as personal computing systems or personal mobile device with Internet access.

In contrast to the current defensive cyber security environment affecting personal computing however, sophisticated software is not available to attempt to prevent and/or provide indication when these systems are exposed to such threats or attacks from malicious sources. This lack of anti-virus/malware protection software opens up the possibility for exploitation of these devices without detection, a very desirable trait for an

attacker. The removal of the possibility of detection or discovery increases the amount of time available for any threats such as a virus and worms to spread should one be developed, thus increasing the number of infected systems. The prospect of a low detection rate could result in the design of simpler malicious software systems that do not need the layers of obscuration code and stealth processes designed to hide malicious software. This more complex code is a common design aspect of modern malicious personal computing software as it derives from the need to hide from the more sophisticated security software and operating systems safeguards. No such extra effort would be required for game device variants as the defensive level of the systems is much lower. In this respect, malicious software threats would conceivably be of a more simplistic design, mirroring the development of first generation virus or worms seen in PC's or cell phones.

This state of affairs in regards to defensive and detection software is similar to the current emerging threat situation with many embedded processing mobile devices (cell phones and PDA's). Illustrating an environment where the existing application or operating system software on the device provides little or no indication of such an attack or compromise. As such in recent years these types of devices are more frequently being targeted directly or indirectly as a new ground for exploitation.

The role of Game Device Location in Network Topology and Exposure to Vulnerabilities

In some cases these networked game devices may be found offering connectivity outside a firewall device or deployed in the bastion host zone of a home network

configuration, if the system in question requires specific TCP or UDP communications ports to be open for application functionality such as multiplayer games. The exposure of these devices in relation to the home network topology location, along with the element of wireless access, provides reason to believe that in some cases networked game devices could be easily accessed. Without Firewall protection, the device is susceptible not only to detection and direct access, facilitating scanning and vulnerability probing but also potentially to directed attacks like Denial of Service. Depending on the utilization of wireless communications technologies (Wi-Fi and Bluetooth) mode, the possibility exists for direct detection and potential communications with the device without the requirement of an Internet connectivity path. If located in the network topology as a bastion host, detection and vulnerability will be a product of the network routing device offering that mode. Some routers place the device virtually on the public side of the firewall with no protection or NAT filtering, creating the scenario for most exposure while other modes may only open up access to specific ports and protocols as configured by the user. In this latter case, the detection and vulnerabilities are limited to the ports presented on the public side of the network interface.

Mobile networked game devices travel with the user and need to connect to a host network for Internet access. As such they will be exposed to a variety of foreign network topologies which could exposed them to insecure network configurations as the user accesses public Wi-Fi locations. This is the same threat environment to which Laptops, Notebooks and Netbook computers are currently exposed. Utilization of public access locations exposes the user to such threats as attackers setting up spoofed access points

creating Trojan ‘hot spot’ deployments. This opens these devices up for eavesdropping, credential stealing or redirected traffic attacks. These redirected traffic attacks, the man-in-the-middle attack (MITM), results in the traffic from the victim’s device intercepted, possibly modified and then relayed to its original destination. Here the attacker may modify or collect the data being sent without the victims knowledge. The extent to which the devices can be affected by various attacks is related to the devices network communications ability. Devices that have Internet capability by default are more exposed to threats than those that only support ad-hoc or peer to peer access modes.

While devices that have only peer to peer or adhoc capabilities have a greatly reduced threat risk, it is not non existent. Such devices may be exposed to previously contaminated devices in the establishment of peer to peer connections with other compatible systems. Should those devices have malicious software payloads that can be shared or transferred to network partner, the possibility of attack exists.

Mobile Game Devices as Potential Threat Carriers

The mobile networked game device presents a new interesting possibility of acting as a carrier for malicious code, or as a potential Trojan attack vector as they have the ability to piconet (peer to peer) with other networked game devices and interact with their respective console equivalents. Such a scenario here would be the ability of these devices to carry on multi-user game play. In this configuration, only one device may need to have the payload application and it transfers that code to the satellite devices for participation in the game.

As such, potentially non Internet connected devices can still be exposed to malicious software when interconnected with compromised or contaminated mobile networked game devices via this piconet. Alternatively, there exists the possibility that the compatible console device may be compromised by its mobile portable counterpart in such a device pairing. This attack vector is similar to the threat of malicious material housed on USB flash drives or portable MP3 players infecting a computer system upon connection of the portable device for the purpose of data transfer. In the game device context the malicious payload is transferred to the console device, from the mobile device its purpose to be executed on said device, or spread to other mobile devices that may be connected. These types of portable device malware infection occurrences utilizing storage type devices need not employing the same operating systems or processor architecture as has been seen in cases involving Apple Ipods in 2006 and digital picture frames devices in 2008 (Lemos, 2008). In both these cases the code contained on these non-PC devices was designed to target host PC systems upon device connection. In this respect, a peer console device may be compromised by its mobile portable counterpart even if it is located safely behind a firewall, should it contain some form of payload designed to target the host device.

Peripheral Device Communications Vulnerabilities

The inclusion of Wi-Fi and Bluetooth technologies in these devices provides an environment of detection and interception exclusive of a connectivity path through the Internet. The usage of wireless connectivity for their associated peripherals such as game

controllers and cameras creates opportunities to exploit weakness in those devices. Technologies for monitoring Bluetooth and Wi-Fi traffic could be utilized in much the same way as they are applied to personal computers and mobile devices, eavesdropping on and injecting information into the communications stream. These associated peripheral devices, containing Bluetooth or Wi-Fi transmitters provide an opportunity to access audio, positional (on screen keyboard) or activation sequence streams possibly resulting in malicious usage. Again drawing on the state of the current cyber security situation, this is analogous to the interception of wireless keyboard and mouse signals for personal computers or the eavesdropping of Bluetooth headsets and hands free automobile devices.

CHAPTER II

REVIEW OF RELATED LITERATURE

There exists a large and growing number of networked games devices both console and portable being utilized for personal Internet access. This fact along with the incorporation of automated system maintenance functions (software updates and downloads) and the ability to download content via user direction exposes these devices to the same potential cyber security threats as seen in precursor devices. As personal computers gained popularity and network connectivity became commonplace they became targets for online attacks, a situation which continues today. More recently this same trend was repeated initially with PDA and now mobile phone devices. The mass deployment of wireless network technologies and their integration into new computing devices for networking has created new avenues for system attacks and exploitation. The question at hand is will this cycle of connectivity and exposure to online threats repeat itself with networked game devices.

Attack Trends in Computing Devices and Utilization

Leavitt (Leavitt N. , 2000) notes in an article in 2000 that in a pattern following the same adoption levels and utilization as personal computers, the development of Personal Data Assistants (PDAs) would result in the devices being targeted by malicious code. This code could be spread by the PC to the PDA during the process used to synchronize data with the device or conversely the PDA could act as a payload carrier for malicious

code specific to the PC. He notes that the Infra-red and Wireless Bluetooth communications channels may also be utilized as a transmission medium invalidating a need for a direct connection and that weak security protection exists for the devices, as they are a new technology. Today mobile game devices present the same possibility of acting as a target or carrier for malicious code. Potential attack vectors in this case could involve their ability to piconet (peer to peer data sharing) with other similar devices via Wi-Fi or Bluetooth as seen with earlier PDA and cell phone type viruses, through interactions with their respective console game device equivalents or personal computer devices as a removable drive/device.

In 2005, Leavitt (Leavitt N. , 2005) notes that the increased adoption of cell phones incorporating network access was again resulting in same emerging threat trend with regards to malicious code. Here he notes the possibility that utilization of a common language for multiple device architectures, such as Java could result in non architecture specific attacks. While none of the current game devices utilize Java as an application language and therefore have no standard Java Virtual Machine, unlike many cell phone and mobile devices incorporating J2ME, it is found providing support for Blu-ray Disc Java (BD-J) in the firmware system function of those devices having Blu-ray drives. BD-J having a security model based on standard Java platforms will allow signed .jar files access to local and remote storage as well as permit network access. This greater access to device resources could result in vulnerabilities and possible attack vectors utilizing BD-J for such an approach.

Embedded Device Applications and Threat Vectors

Embedded applications supplied with many devices, such as web browsers and media players also rely on the concept of cross platform functionality to display and manipulate data through the use of JavaScript, Extensible Mark-up Language (XML) or Hyper Text Mark-up Language (HTML). It is this cross platform compatibility that creates the environment in which non target specific malicious code can be crafted. On the Internet these notational and programming languages were implemented to work across multiple platforms utilizing web browsing and media playing applications, as such they interact accordingly being designed to utilize such commonality. Networked game devices with the inclusion of a web browser and the subject to the browsing habits of the individual accessing the Internet will similarly be exposed to the same threats that a personal computer user would experience due to this interoperability. In this regard websites containing specially crafted code malicious code utilizing application vulnerabilities may affect a broad range of devices. Therefore, by utilizing these same Internet applications (browsers and media players) networked game systems are now also potentially at risk from malicious software embedded in compromised websites, the number of which grows daily.

In the study by Provos et al. (Provos, McNamee, Mavrommatis, Wang, & Modadugu, 2008) done in 2006 it was determined that of 4.5 million sampled websites, 450,000 were of malicious intent, while another 700,000 were of a lower confidence level of malicious activity. Web based activity has become the main avenue for attempts to download and install malicious software with the highest activity being the deployment of Trojan

software on users systems unknowingly, compromising 300,000 sites of their data set. The utilization of Proxy systems or Network Address Translation (NAT) firewalls found in many home network configurations offered no protection against these threats as the communications traffic for this activity utilizes the standard ports for HTTP. The utilization of JavaScript played a major role in all the activities noted, determining the types of applications software available to the browser, media types supported and redirection and obscuration of malicious URLs. Hence any browser platform supporting JavaScript opens itself up to the possibility of exploitation and the ability of the attacker to determine other candidate application software with inherent vulnerabilities.

A study by Adams (Adams, 2009) related to self directed user actions, found that users were highly vulnerable to both the threats provided by compromised websites as well as compromised removable storage media. Devices used for web surfing would be exposed to the possibility Cross Site Scripting attacks, JavaScript exploits or other types of redirection/infection attack vectors that may occur in what are referred to as 'drive by download' attacks. This refers to a tactic utilizing social engineering motivators to target potential websites to host malicious code. It is accomplished by systematically exploiting sites related to high profile events where the associated websites are targeted in the context that they will have a high chance of being accessed by large numbers of users. Systems visiting these compromised web sites are exposed to attacks via web browser vulnerabilities and its associated applications support (Flash animations for example). In this attack context no distinction would be made that the user is on a game device as

opposed to a computer system, only that they were part of the target demographic for whom the malicious site(s) where intended to attract.

Wireless Connectivity Threat Vectors - WiFi

All recent networked game devices incorporate the wireless connectivity as a means of providing ease of network connectivity or for ease of peripheral usage. Thus eliminates the need to run wires from the family room, living room or bedroom of an individual utilizing the device, to the network routing device thus creating an ease of use, low technology literacy requirement. Within that context of minimal user technical literacy, little thought is given to the idea that directed attacks accessing the wireless communications channels of the device might be utilized. Dagon et al. (Dagon, Martin, & Starner, 2004) note that the inclusion of wireless technologies such as Bluetooth and Wi-Fi in a device, does not limit them to falling victim to short range attacks within the specifications of distance familiar to the consumer. This factor contributes highly for example to the perception of a false sense of security exists that in regards to mobile phones and other portable consumer electronics. These are seen as personal, social devices with a limited or isolated communications framework and as such fall outside of the scope of what one would consider as a target worthy of interest by hackers or other malicious attacks. This presumption leads then to the exclusion of such devices when assessing these technologies with regard to potential attacks or vulnerabilities. They are however not immune from the possibility of being targeted.

An examination threats and vulnerabilities related to wireless mobile devices can be found in Friedman and Hoffman's work (Friedman & Hoffman, 2008). Vulnerabilities of portable devices expressed in their analysis are:

- Utilization of network infrastructure outside of network perimeter defences, not protected by a firewall.
- Utilizing vulnerable communications methods with the implementation of wireless technologies.
- Such devices that can be easily lost or stolen due to size and mobility.
- Lack of attention to security issues due 'personal device perception' as noted by Dagon et al.

As such, these factors expose the devices to a wide variety of online threats, the taxonomy of which can be broken down as follows:

- Malware Attacks through the acquisition of malicious software.
- Phishing Attacks through the use of redirection or impersonation of credentials.
- Direct Attacks on the device in question, resulting in device malfunction or unauthorized access to data.
- Data Communications attacks on the wireless link itself such as Interception and Spoofing.
- Loss or theft of the portable device, resulting in the exposure of personal or confidential data.

Portable networked game devices can also be perceived in the same mindset as these mobile devices as personal social devices used purely for entertainment and/or communications. As such they could also be considered a target not worthy of pursuing by hackers or malicious software developers. Console game devices are perceived safe, having a wireless signal range which is limited to the home environment. Users of mobile networked devices have their security concerns subdued by an 'implicit trust relationship' related to the validity and security of a provided wireless hotspot, such as in a coffee shop. Yet the operation of these wireless networked game devices and associated signals make them part of the greater pervasive wireless environment which exists today.

It is in this regard that new threat concerns arise due to the prevalence and proximity of wireless devices. Situations may arise where the networked game device is the target of an indirect form of attack simply by being an active part of this pervasive wireless environment. Work done by Akriditis et al. (Akritidis, Chin, Lam, Sidiroglou, & Anagnostakis, 2007) supports this possibility noting the trend that any technology which reaches critical mass will attract the attention of attackers, wireless devices being one of those technologies. They determine that the density and proximity of wireless devices today creates an environment where the potential exists for worm type infections, phishing or other malicious activity that can spread without Internet access. As per their proposed scenario and utilizing a proof of concept implementation, they created a worm Wildfire/A. The worm travels from Access Points to Access Points and Access Points to Hosts (and the reverse) directly, utilizing the overlapping detection and connection of adjacent wireless signals (targets) and exploiting a chunked encoding vulnerability found

in Apache Web server 1.22. Mobile devices have a role to play in this sans-Internet scenario, providing a means of transporting malicious payloads between areas of sparse connectivity. These types of self propagating malicious payloads may be constructed to utilize a specific vulnerability in the hardware utilized in the wireless adapter, the operating systems of the device and weaknesses in the wireless protocol stacks utilized or applications present on the device. They note that while accessing/traversing open (unencrypted) devices is the easiest method, it would not be inconceivable for such a payload to carry code to break encryption, exposing those nodes to attack as well.

While most devices using wireless will employ some form of encryption, a large percentage still do not, public access points specifically. In either case, wireless systems will be targets for detection, eavesdropping and possible attack. Berghel and Uecker's (Berghel & & Uecker, 2005) provide a good foundation for a summary of attack vectors that the various modes of encryption utilized by Wi-Fi are susceptible to, noting "Wi-Fi will continue to be more vulnerable to attack than hardwired LANs as long as electromagnetic radiation fails to obey property lines". In reviewing attack vectors they note that even with the application of stronger Wireless Privacy Access (WPA) encryption variants commonly used today, vulnerabilities will continue to exist. This weakness in the encryption methods implemented results from the lack of a dedicated security/authentication device such as a RADIUS server implementation in a typical home network. Lacking such a dedicated authentication server, current Wireless Privacy Access-Pre-Shared Keys (WPA-PSK) implementation designs attempt to provide a software surrogate adding strength to a secure home network configuration. While the

additional design changes provided greater security than the previous Wireless Equivalent Privacy (WEP) encryption, it was only a matter of months before vulnerabilities where found, allowing it to be compromised. Work done by Moen et al. (Moen, 2004) showed that the Temporal Key Integrity Protocol mode of WPA (WPA-TKIP) had scheduling weaknesses allowing for the determination of the Temporal Key (TK) and the Message Integrity Check (MIC) key. This discovery allowed for tools to be developed which determined the pass phrase used in WPA-PSK by comparing dictionary attack generated MIC values to actual captured MIC's.

This review of threat considerations shows that all variants of wireless encryption utilized in Wi-Fi network configurations to date, without a separate security server have been actively exploited. The typical scenario illustrating that it is only a matter of months between the adoption of new encryption methods and tools to compromise that method. With the existence of such tools and the utilization of wireless technologies for the large population of networked game, it places them in the well in category of possible attack targets.

Wireless Connectivity Threat Vectors - Bluetooth

The inclusion of Bluetooth wireless communications in networked game devices likewise produces opportunities for attack or manipulation. Although utilizing encryption by default, work done by Shaked and Wool (Shaked & Wool, 2005) showed that it was possible to determine the Personal Identification Number (PIN) utilized during the Bluetooth pairing and authentication process, resulting in the ability to decode the

encrypted data stream between paired devices. This facilitated the development of tools purposed for performing eavesdropping, replay and device impersonation attacks on Bluetooth protocol devices. Solon et al. (Solon, MJ, Harkin, & McGinnity, 2006) reviews some of the more common vulnerabilities in Bluetooth capable devices and the attacks they are susceptible to. Bluesnarfing attacks provide access to restricted areas of a user's device without their knowledge. Bluebugging provides access to command capabilities in a device via remote access and backdoor attacks which establish a hidden trusted pairing of devices, this granting the ability to access any resource on a device unknown to the user. BlueBump attack tools, force the re-pairing and re-authentication process for targeted devices, creating the opportunity for an attackers system to establish communications with a target device.

All of these aforementioned attacks are crafted utilizing the widely available information of the Bluetooth protocol standards and design vulnerabilities in the Bluetooth devices themselves. Su et al. (Su, Chan, Miklas, Po, Akhavan, & Saroiu, 2006) analyzed the possibility of such an issues being utilized to develop a worm like malicious payload for Bluetooth devices. They surmise that due to the complexity and size of the code in the Bluetooth stack implementation that exploitable vulnerabilities to accomplish this type of software probably exist, citing the pre-existence of other malicious Bluetooth code development. They suggest that as Bluetooth capable devices are forecast to exceed the number of Wi-Fi enabled devices and therefore create a larger base of potential targets that the likely hood of some type of viral like infection code or large scale attack will occur at some point. In their examination of the premise, they

found that although restricted by limited range of communications (ten meters on average) compared to Wi-Fi and primarily being a mobile device communications technology that the possibility for the development of a worm type of infection still exists. The limited connectivity time of moving devices was not a deterrent to such potential infection and transmission, while the scanning, detection range and speed of the Bluetooth protocol more than adequate for target detection and infection. This makes the potential for continued Bluetooth threats an ongoing possibility.

Previous Works Related to the Vulnerability of Game Devices

Large amounts of work have been done and continue to be done in regards to computing devices vulnerabilities and defensive measures. Historically, there is much data and ongoing research into the threats and vulnerabilities of networked personal computers which will not be reviewed here in depth. More recently increased research and data on portable embedded processor devices like cell phones and Personal Digital Assistants (PDAs) is currently available, but as yet there is little research currently directly related to networked game devices. Device specific research can be found in Myrmo (Myrmo, 2007) which examines if game consoles create new vulnerabilities in the home and whether extra security precautions necessary for the home network. He explores if a modified game console containing a 'modchip" , an additional hardware device which changes the operation and circumvent manufacturers design networked game devices creates vulnerabilities. He also examines the question of whether the users of game devices and virtual worlds see their privacy protection the same as with other computing devices. The latter question provided in a survey component, focused on

gathering information in regards to the latter item, in which he examines the concept of a user's security perspective with relation to game devices from a parental/child's viewpoint.

Myrmo's work presents an examination of several game platforms in a non detailed technical level in regards to vulnerability and as such makes no dedicated study of one particular manufactures device. For related work he selects areas of interest and perspectives that may provide useful information to the topic at hand. Specifically the topics of the Modification chips (Modchips) community, security lessons from the computer world, the functioning of root kits, distributed computing, cheating in games and child grooming. Examination of the usage of networked game devices as Internet appliances is not the focus of his research, but rather the context is that of the device utilized in its primary function mode, that is for that playing games and the usage of associated vendor facilities associated with that purpose. In the context of general exposure to vulnerabilities, he speculates that console game devices could be utilized in a distributed botnet type environment since they can participate in grid computing and the possibility that firmware updates to the devices may be modifiable to contain malicious code.

Potential Reasons to Exploit Game Devices

Computing devices with unique architectures have been utilized successfully for brute force password or encryption breaking, through the use of the on board enhanced graphics accelerators and the parallel processing architecture. The advanced processor

technologies used in the current generation of console game devices approaches the abilities of small supercomputers (Fowler, 2005), (Nagle, 2008). With the availability of software to develop applications, these devices become very attractive for computationally intensive processes. The usage of the Sony PlayStation 3 (PS3) as a development platform for such intensive computational work is indicative of the availability of the tools and reference material available to individuals for such development. MIT (MIT Masschusetts Institue of Technology, 2007) recently offered a course on programming utilizing the PlayStation 3 system and the Linux development environment Yellow Dog.

These elements would factor highly in the choice of this platform for users wishing to develop and explore parallel processing architecture applications in regards to computer security (Marechal, 2007). The Sony PlayStation 3 has been utilized to brute force passwords, showing the usefulness of its processors for cryptographic applications (Hedquist, 2007). Klienhans et al. (Kleinhans, Butts, & S, 2009) found that the use of a PlayStation 3 system for such purposes outperformed equivalent capacity AMD and Intel processor systems with a cost efficiency price/performance index factor of 3.7 to 10 times respectively. A dozen PS3 systems with an approximate cost of $5000 can provide equivalent computing power to a cluster of conventional workstations costing $200,000. This is important in the context that the security of many encryption algorithms is based on pretext that the time required to brute force keys is significantly long and that the cost of acquiring the required amount of computing power to reduce that time would be cost

prohibitive. The availability of low cost, high computational capability devices creates potential vulnerabilities in that scenario.

The researchers note that such a cluster of PS3 systems utilizing their parallel processing architecture and network capabilities lends itself to the development of a distributed password cracking framework. In this process the key space is broken into chunks and distributed to multiple PS3's or multiple processors in a PS3. Computational work is preformed on the pieces of the key simultaneously, and results evaluated until the correct key is found. With grid processing applications running as background tasks on networked game devices, like the numerous @Home applications developed by Berkeley University on many desktops (Martin, 2007). It is certainly possible that similar applications could be developed and misused to create a networked game device distributed password cracker. At the very least, the aforementioned availability of reference materials and development software make it feasible that malicious code could be developed for the Sony game console devices.

Examination of Game Device Vulnerabilities Utilizing Computing Systems Approaches

Contextually for this research we can liken networked game console devices in the same category as computer systems in a fixed local area network (LAN) environment and portable game devices similar to mobile computing systems. It is from this perspective that we can examine the threats associated with these utilizing the established security frameworks of networked computing devices as applied to the context of networked game devices, utilizing analogous device functions and technologies as a starting point in

the exploration for potential vulnerabilities. This is an acceptable approach as work by Jiwnani and Zelkowitz (Jiwnani & M, 2004), shows that is reasonable to assume that similar vulnerabilities can exist in differing systems which have similar functionality.

Work by Igure and Williams (Igure & Williams, 2008) provides a summary of attack taxonomies related to computer systems, with a breakdown these of attack and vulnerability taxonomy types. They point out that searching for known security weaknesses in a system is an objective process, whereas looking for unknown security weaknesses in a system is a subjective process. Questions such as what to look for, how do we quantify what we find and by what metrics, create challenges in the subjective process. Indicative to solving this problem in the examination of vulnerabilities of networked game devices, where we are looking for unknowns, is the fact that we are using the vulnerabilities of analogous devices such as personal computers, mobile phones and PDA' and their respective applications, as a context framework for research at hand. Doing so provides us with a starting point of known security weaknesses and a comparative evaluation processes and expected results, most beneficial as we have no previous game device specific framework to go by.

Utilizing Virtual Worlds for Security Education Related to Game Device Users

We can presume that from the evidence of prior research into emerging technology trends and attacks, software design complexity issues resulting in the introduction of exploitive flaws and the need for compatibility with Internet content that we have a high probability of finding vulnerabilities in the current generation of networked game devices.

With that in mind, the question of how to mitigate the risk of such vulnerabilities comes to the forefront. As indicated earlier, the initial line of defence is the education and increased situational awareness of the game user, lacking protective software capabilities in the game devices themselves.

In this regard the most successful educational tools would be those that are patterned after the game environments themselves, having the greatest chance of engaging the users, the majority of whom will be millennial or 'digital natives'. This generation of users is comfortable and familiar with networked 3D game worlds and easily transitions experiences and ideas between the virtual and real world (Macedonia, 2007). To this end an immersive three dimensional simulated world, a Multi User Virtual Environment (MUVE) in which physical elements represent their digital counterparts would be effective (Dede, 2005). In this world the learner can be an active participant, interacting with objects and scenarios with the goal of educating the user to the potential online vulnerabilities, experiencing the associated effects of such interactions and offering mitigation strategies.

One could consider this construct a Virtual Leaning Module, allowing the subject to interact with the simulated environment via an in world avatar (Boulos, Hetherington, & Wheeler, 2007). Placing the game user in this type of immersive environment where the subject interactions with objects result in differing outcomes, provides an experiential learning model as defined by Kolb and Fry. The concepts involved in the actual mechanics of online vulnerabilities are abstract actions in regards to the real world. One

cannot physically see things like port scanning, traffic sniffing or Cross Site Scripting manipulations; they exist in the context of the digital word and are observable only through appropriate tools and with a technical background to understand the processes. Using a three dimensional virtual environment allows one to construct a world a 'thought space' which these invisible concepts can be created and visualized and their actions observed, and interacted with (Hollins & Robbins, 2008). Such simulated environments have been utilized to teach students how to create insulin proteins by sequencing DNA codons, creating amino acids in a virtual world at a molecular scale (Krange & Sten, 2008). Second Life and other OpenSim based worlds are mature virtual world platforms familiar to millennial game device users and therefore a good choice for immersive learning environments. This factored in the development of the simulation developed by Shi et al. to teach business strategies to millennial students in an engaging interactive learner centered game experience. Here they create an ancient Chinese village to illustrate traditional Chinese Military strategy, the ancient 36 stratagems as an environment to teach the business concepts of strategic management principles. In doing so, they take what is dry conceptual material and ideas and present it in a more exciting and interactive format for the learner (Shi, Lee, Hinchley, Corriveau, Kapralos, & Hogue, 2010).

Additionally utilizing such established multi user virtual environments (MUVE) such as Second Life or OpenSim enables the researcher to draw upon a broad range of resources. Access to work done with regard to previous learning models and collaborative communities of practice as well as open source tools for the integration of in world objects with Learning Management Systems (LMS). Sloodle is one such tool,

allowing the resources of the MUVE and LMS to be integrated, creating an ability to utilize standard teaching tools such as quizzes, polls, feedback and web based content in the 3D learning environment (Kemp & Livingstone, 2007). This combination facilitates the development of a mixed mode teaching environment with structured content. This can ease the burden of having to creating standalone interactive objects for presentation or learner assessment of educational materials presented in the virtual environment.

In a MUVE environment, it's possible to design guided, goal oriented or combination scenarios for the learning objective. This allows for exploratory or 'navigation assisted' movement through the simulated environment. The decision of which method(s) to employ can be utilized to address various levels of user familiarity with the environment and its associated interaction mechanisms. The same facilities can also simplify the design of the virtual environment by compartmentalizing lesson objects in specifically designated and constructed locales. These design factors will influence the learner's retention and overall experience in the virtual environment (Barker, Haik, & Bennet, 2008).

CHAPTER III

METHODOLOGY

Overview

The research goal is to analyze these devices in the context of the potential for security vulnerabilities and exploitation with regards to current and possible future environments. Work will be done utilizing the same analysis and penetration testing methodologies as utilized for personal computing devices, this being the closest analogous technology implementation and the environment in which the devices are configured operate. As such elements of the Information Systems Security Assessment Framework (ISSSAF) will be utilized and referenced where applicable. The devices will be deployed in a lab environment analogous to a typical home network in both wired and wireless configurations, firewalled and non firewalled deployments. An example of the lab topology is illustrated below in Fig. 2 showing the test setup utilized for a Sony PlayStation3 hard wired and PlayStation Portable devices making wireless connections to the open access point. Other game devices undergoing testing will utilize the same topology locations, dependent on their network connectivity being wired or wireless. The area in grey represents the devices home network, devices outside the lab area play the role of external systems. All firewalls in the configuration have the capability of virtually placing any device 'outside' the firewall perimeter allowing unfiltered access.

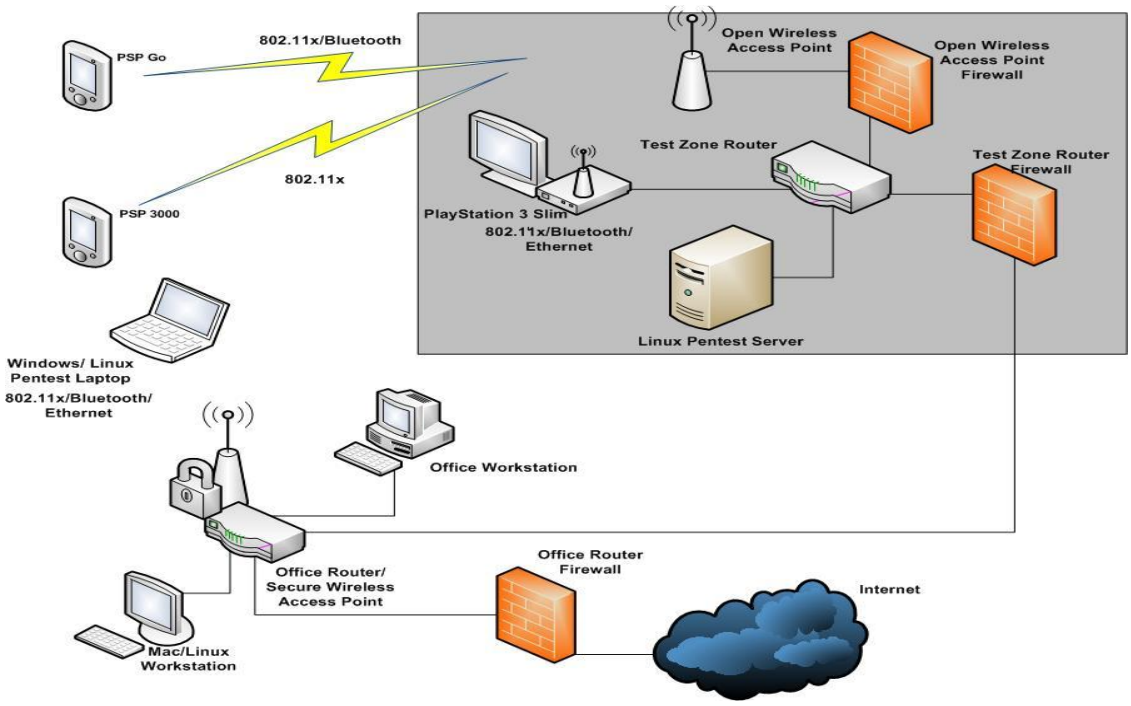


Figure 2. Laboratory Test Configuration

The lab environment facilitates the creation of simulated threat environments such as fake wireless access points or malicious web sites which will be utilized to enable attack scenarios to be created for the tests in question. This environment also allows exposure to threats based on information from computing security sites related to real world locations of online threats. This will allow for the assessment of devices when exposed to compromised or infected websites to in a controlled and monitored lab environment.

Looking to the analogous security related items in personal and mobile computing, provides a springboard to the examination of potential weaknesses a networked game system may have and as such illustrates which potential attack vectors should be examined. An assessment of the technologies utilized by both the networked game devices being examined and standard network computing devices will result in the

determination of possible attack vectors via utilized technologies. The selection of an appropriate taxonomy also assists in the analysis of devices with regards to security vulnerabilities. Utilizing attack characteristics for example, allows for a number of vulnerabilities to be placed in a well defined and understood category. Thus all attack techniques that resulting in "denial of service" have the associated vulnerabilities listed under that heading. This process then clarifies the framework for performing assessments of a system by establishing common terms for attack classification that are independent of system architecture. Benefiting the analysis process we avoiding the creation of lists of attacks, which while differing in technique, result in the same effect. This selection of dimensions of classification for the taxonomy, appropriate to the purpose of analysis at hand will help the security researcher focus on the functional components of the device being analyzed. This tailoring results in a more objective vulnerability assessment process.

This classification strategy is applied to the analysis of networked game devices bringing the search for vulnerabilities from the subjective to the objective context. In our case we define a device vulnerability matrix with which to categorize the areas of investigation with relation to the system being examined. This will help to focus research into the areas that most likely will generate results. We utilize the dimension of attack classifications and the technologies or applications they affect and associate that possibility of exploitation to those devices matching those categories.

This device evaluation, correlated with an exploration of known attacks and vulnerabilities applicable to the technologies and applications that affect networked computer systems, provides data for the development of a device vulnerability matrix. The device vulnerability matrix is comparable to others that have been developed first for personal computers and most recently for mobile devices; as such it is an extension of such works related to other technologies. It utilizes the taxonomy of possible attack vectors that may be applied against devices based on the capabilities or features of that specific device.

Beginning this process the information gathering phase looks to assess the target and determine in what areas we can begin to proceed to look for security weaknesses. Creating a vulnerability matrix helps to define the probable areas for investigation, step one in this process is the determination of device characteristics.

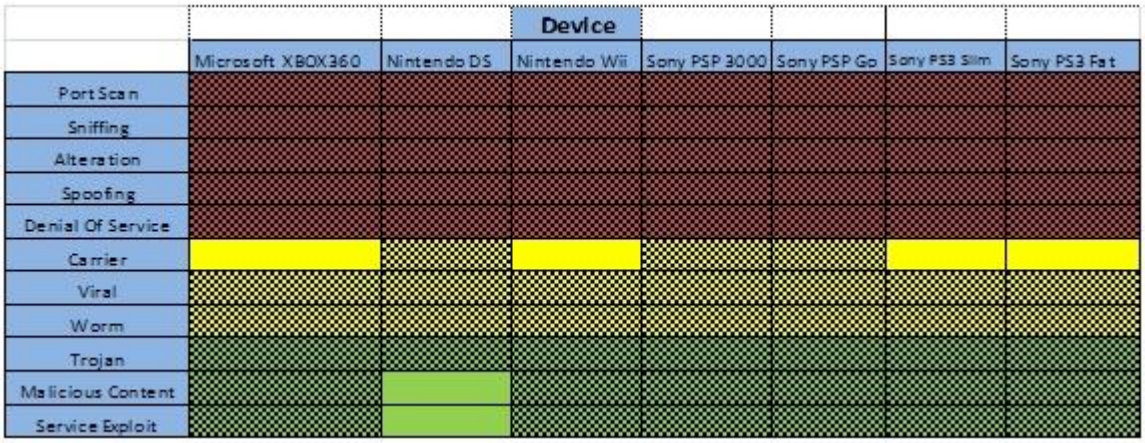
Table 1. Devices and Built-in Capabilities

			Device				
Native Capability	Microsoft XBOX360	Nintendo DS	Nintendo Wii	Sony PSP 3000	Sony PSP Go	Sony PS3 Slim	Sony PS3 Fat
Wired Network Client	Yes	No	No	No	No	Yes	Yes
WiFi Client	No	Yes	Yes	Yes	Yes	Yes	Yes
WiFi Access Point	No	No	No	No	No	Yes	Yes
Peer To Peer	No	Yes	No	Yes	Yes	No	No
Bluetooth	No	No	Yes	No	Yes	Yes	Yes
Removable Storage	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Web Browser	No	No	Yes	Yes	Yes	Yes	Yes
Media Player	Yes	No	Yes	Yes	Yes	Yes	Yes
Email Client	No	No	No	No	No	No	No
IM Client	Yes	No	Yes	No	No	Yes	Yes
Portable Device 'Syncing'	No	No	No	Yes	Yes	No	No
OS Updates	Yes	No	Yes	Yes	Yes	Yes	Yes
Remote Desktop	No	No	No	Client	Client	Yes/Server	Yes/Server

The known device characteristics are then broken down into system components and system features as displayed in Table 1. Components have physical hardware related aspects, features being more software oriented. This step segments the device into equivalent technologies as related to personal computing devices. These can then be correlated against the taxonomy of vulnerabilities associate with those components as illustrated in Table 2. Data from this matrix is then mapped back to the devices having those associated characteristics, creating Table 3. a vulnerability matrix illustrating potential areas of risk for each device based on their specific characteristics. The shaded areas in the tables indicate which vulnerabilities effect the system component or system feature listed. Column colours are for vulnerability type segregation (Network, Infection and Applications) and do not reflect any measure of severity.

Table 2. Vulnerabilities Associated with Components and Features

Table 3. Potential Game Device Vulnerability Matrix



Tools and Techniques

Tools used for this work consist of utilizing 'Off the Shelf' (OTS) software packages used in penetration testing and security evaluation of networks and computer systems to carry out this task. There are many open source frameworks and tools for penetration testing providing wide variety choices for this purpose. The primary collection of tools utilized will be from the BackTrack security suite. Other open source tools not in the BackTrack collection may be utilized when required for specific tasks and will be identified as such in the documentation. The experimental testing procedure will follow the elements of the Open Information Systems Security Group (OISSG) Information Systems Security Assessment Framework (ISSAF). Fig. 3 provides an example of guided documentation framework which can be applied to the examination of the devices in question. For our purposes, we are primarily interested in the first four elements of the ISSAF methodology. Beginning with information gathering we proceed to detection of the devices (Network Mapping) then vulnerability identification of the device (Host related items, Service Discovery, Firewall testing and Operating System determination)

and finally Penetration and Gaining Access. The following diagram provides an indication of the process flow for testing:

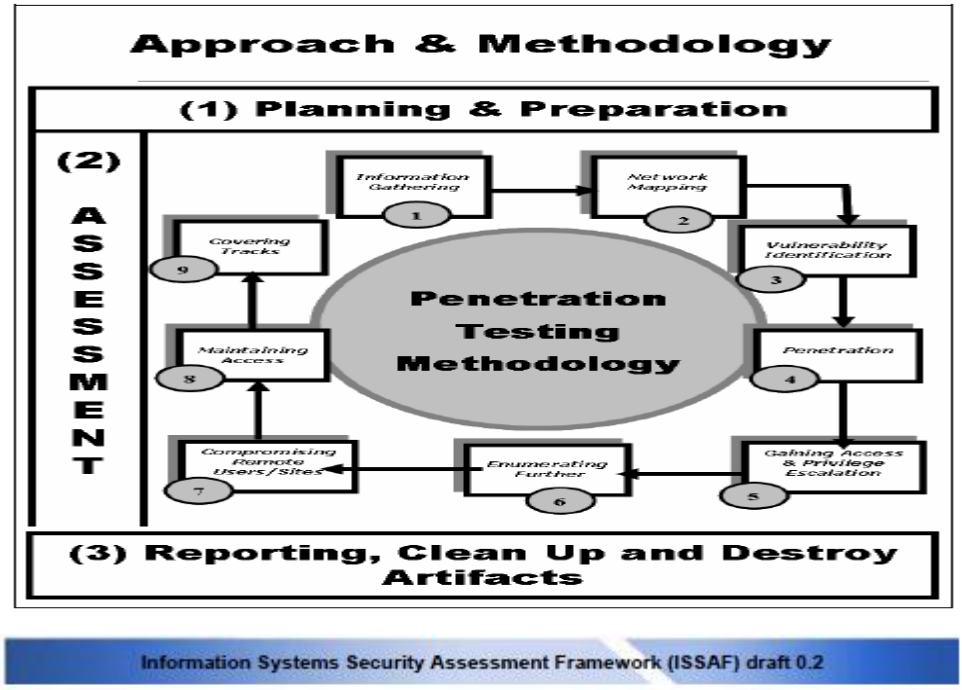


Figure 3. Testing Approach and Methodology

Device Evaluation and Experiment Determination

The first step of the process (Information Gathering) is examinations of the devices to be tested to document the characteristics that will later help refine what experimental tests should be run to look for vulnerabilities. This activity is common to security audits in the sense of gathering as much information as possible about the target in question so as to formulate directed methods of attack. For reference this is analogous to section B.1 in the ISSAF document. For this work we are looking to determine such characteristics as:

- Device hardware specifications
- Device Usage profiles
- Device Software applications

- Device Location in Network Topology
- Known Device Vulnerabilities

Data can be collected via literature searches, manufacturer supplied information and direct device examination creating a profile of the devices functional characteristics. This information provides a correlation for the applicable threats potential from the device vulnerability matrix, based on the devices capabilities.

The second step of the process begins the direct examination phase, giving an indication of how the devices in question may be detected and recognized in a network configuration. These will be the results from device fingerprinting either via active or passive network scans. In a directed attack scenario, the existence of a device must be determined before said attack may occur, and this element fulfills that step. The devices fingerprint characteristics will also provide indications of the devices uniqueness in a network structure or its appearance as just another personal computer, to which an indirect attack scenario may be applied.

Steps three and four begin the experimental phase of the device evaluation; this is a combination of penetration testing and an evaluation of the devices operating characteristics and configuration. It is positive results from here that may be applied to develop proof of concept applications demonstrating or specifically detecting potential vulnerabilities found in a game device. This information is also applicable to the determination of what threat scenarios should be utilized in a virtual educational gaming

environment, as they may be experienced by game device owners and therefore applicable.

These experimental attack attempts are then made with the premise to exploit possible weaknesses, whose purpose for an actual attacker may be the intent of

- gathering private information
- modify existing functionality for unintended purposes
- implementing new functionality
- using the device as a stepping stone to compromise other devices

Possible attack vectors here could include

- Target operation or manipulation via forged association through redirection and replay.
- Target operation or manipulation via direct intervention
- Target manipulation through indirect intervention
- Target operation disruption via direct intervention
- Target operation disruption via indirect intervention
- Direct access to the device via Wi-Fi access.
- Direct access to the device or peripherals via Bluetooth
- Unauthorized access to target device or associated local network devices via information from target device.

The proposed usage of existing exploitation tools during testing hopes to show susceptibility to the same attacks as personal computing devices. This would lead to the conclusion that these devices represent a target of potential indirect attack and exploitation therefore they should be protected accordingly in the network topology or with additional software tools report or other documents as required. While all of the possible attack vectors mentioned here are not explored in this research project, the proposed methodologies and suggested potential tools that may be utilized for exploring those vectors on the game devices are still discussed.

Analysis of the gathered research data will determine the state of vulnerabilities of the current console and portable game devices in their network active state. The success or failure of these research endeavours related to vulnerability detection and exploitation, combined with the background review of the devices in question provides a greater understanding of the level of vulnerabilities present in such devices and potential mitigation techniques. This data will provide baseline information which gives an indication of the devices security status with regards to other types of networked computing devices.

Device Scans TCP/UDP Ports

Initially the networked game devices will be scanned utilizing the Nmap and Nessus software packages from a system on the internal network. The presumption here is that the device has been positioned outside the firewall, or that the attackers have associated the attacking device with the host network. This is a focused application of ISSAF

section B.2 procedures, as we know of the target devices existence. In the default power on mode devices will be scanned five times to eliminate false positives or to attempt to detect random port assignments of services on power-up. Target systems will be shutdown and then a second set of five scans performed to identify this. Notation must be made of the devices menu status and operational mode, this is necessary as the one of the systems has a remote device access mode, opening specific ports for that service which needs to be documented. In this case a second set of detailed TCP/UDP scans will be performed and shutdown/restart sequence will also be required to determine if any ports associated with such remote data connections may change.

Port scans will entail utilizing the various inquiry modes available in the Nmap and Nessus packages, starting with the default methods of TCP protocol connections commonly utilized to determine the existence a firewall or other port blocking software. The system will then be scanned comprehensively for the existence of all possible TCP/UDP ports. Since the purpose is to identify ports primarily and not hide our intention of doing so, options for disguising the scan in progress (stealth mode options) are not required facilitating a speed up of the scan process. However, packet timings may require changes depending on the medium utilized for network connection, as such the parallel scanning function T3, may need to be changed to slower timeout/response parameter over the wireless links.

Initial scans utilize the following Nmap parameters:

Initial Default Scan: **nmap -sC –sT –T3 –O –v –n -PN <target ip>**

Here we specify run the scripting scans to determine more details of the device if possible

(-sC), do a standard TCP connect scan (-sT), normal parallel scanning mode (-T3) ask for the possible operating system identity (-O) , verbose output (-v) , no reverse DNS lookup (-n) and no Ping response detection (-PN).

Comprehensive TCP Scan: **nmap -sC –sT –p 1-65535 –T3 -O –v –n –PN <target ip>**

The intent here is to determine any TCP ports that may be opened by services or applications on the device being examined. We add the command to scan all 65535 ports (-p 1-65535) to the previous default scanning parameters.

Comprehensive UDP Scan: **nmap -sC –sU –p 1-65535 –T3 -O –v –n –PN <target ip>**

For UDP port detection and analysis we substitute the appropriate scan type parameter switch changing –sT to –sU. Packet time outs with regard to individual device characteristics may require additional timing changes to ensure completion of the UDP scan process.

For the Nessus scans, the package version is 4.2.2 home, parameters for the scanning policy utilized for the game consoles are as listed in Appendix B. The information from the scans of the Nmap and Nessus packages should provide fingerprinting information on the Operating Systems in use on the devices as per ISSAF B.2.1.9 and B.2.1.10

Follow up port examination – Service / Operating System Type Detected

Moving on to ISSAF section B.3, Port scan information that returns verifiable and identifiable services on specific ports or Operating System type needs to be investigated further. Research on these using vulnerability databases such as SecurityFocus.com, PacketStorm.org and the Open Source Vulnerability Database (osvdb.org) will determine if there are any applicable exploits for that specific result that may be tested. The type of exploit used and observed results of exposure can then be recorded providing an indication of its effect comparable to the originally intended platform or operating system environment. This is in line with performing tests of vulnerabilities that are related to computer systems on the Internet and their effect on the game console device.

Further examination of potential exploitation may be performed automated software packages such as Metasploit or the Social Engineering Toolkit (SET) incorporated in the Backtrack 4 (BT4) software suite. These tools may then be applied against any found port or service vulnerabilities generating results as to the effectiveness of these tools with regards attack scenarios. It is noted however that to utilize any of the exploits that deliver a ‘payload’ an application specifically crafted to perform a specific function, such a creating a root shell or other backdoor, device specific binary executables would be

required. Testing of any exploits requiring websites can be facilitated through the use of any standard web server for or utilizing the web services available in the SET software package. This would be adequate for the replication of real world web services and the test scenarios that may need them.

Wireless Target Determination

Wireless detection scans are performed utilizing Kismet with Global Positioning Satellite (GPS) data coordinates enabled. Kismet is also utilized as a wireless packet capture application acquiring data for analysis in Wireshark related to communications of applications on associated devices. The collection method will utilize an OTS Acer Aspire Netbook and its internal Wi-Fi adapter which utilizes an Atheros chipset, GPS information is provided by a Pharos USB module. Data sets collected here allow for the determination of typical wireless fingerprint activity which will assist an attacker in the detection and identification of targets. Wi-Fi signals will be detectable at long ranges and therefore would allow for the determination of a targets existence remotely. In this scenario, consideration will be given to the ability to detect a target as part of a wireless network (node), as a wireless access point (AP) or an Adhoc network wireless device. Scan datasets from the Kismet sessions are plotted on Google Earth maps utilizing the KNSGEM software, resulting in graphical plots of signal range, channel selection and locations.

Note of the wireless MAC address will indicate if the devices utilize a unique manufacturer's chipset and an examination of the possible vulnerabilities that may exist

associated with the chipset. This unique device information will be tabulated deriving the ability to determine system type. An examination of geographically collected wireless information correlated these known device characteristics will determine existence of the devices in question in operation in a typical neighbourhood environment potentially giving some idea as to the density of such devices.

Wireless Access Security Testing

It will be assumed that encryption is utilized with the devices in the network node role and as such an examination of the resiliency of the devices to attack by wireless attack tools needs to be considered. The BackTrack suite of tools will be utilized in the examination of this process. Attempts will be made to examine weaknesses in the encryption or parameters utilized in remote access operations and peer to peer adhoc access modes. Device encryption possibilities include available standards such as WEP, WPA and WPA2. Default encryption settings will be utilized and examined for any of the device modes used. User selected password scenarios assume the use of the onscreen keyboard and the preclusion of the usage of a password generating program or website for setting the passwords.

Wireless Remote Access

Examination of wireless traffic between the PlayStation portable (client) device and the PlayStation 3 (server) is of primary interest in regards to the possibility of interception or data modification. Tools utilized are Kismet for wireless captures, Ettercap as the in situ network data capture tool and Wireshark for data packet examination. In respect to

ports utilized and associated potential running services during the remote screen functions, this information was collected during the device scan phase, as a device configuration option possibility.

Connection from the PlayStation Portable (PSP) to the PlayStation 3 (PS3) system is made via the following modes:

1. PSP to PS3 acting as a Wireless Access Point
2. PSP to PS3 via Test Lab Network Wireless Access Point
3. PSP to PS3 via External (to Test Lab) Wireless Network Access Point

In mode 1, all device communications are via wireless mode and as such Kismet is utilized as a data capture tool, in modes 2 and 3 we have both wireless and wired traffic. Kismet is utilized for the wireless traffic capture from the PSP to the access points with Ettercap functioning in Man in the Middle mode, via ARP poisoning to capture the wired segment of data. The PlayStation 3 and external network APs utilize WPA encryption; the Test Lab AP is in open (non-encrypted) mode.

For the wireless attacks, we presume the attacker has derived knowledge of the encryption mode and MAC addresses in use through passive monitoring and has knowledge of said parameters usage required for the peer to peer or remote access functions. For active monitoring and examination of this mode, an Acer Aspire laptop running the Backtrack 4 software suite is configured so that the MAC address represents

that of a PlayStation portable device. The Aircrack suite of tools is used to test the ability to determine the strength of encryption key via the collection of handshake data generated by repetitive disassociation of the client systems (game devices) from the network access point. Airodump-ng is utilized for the capture of association and handshake data between the client device and the encrypted access points in modes 1 and 2.

Bluetooth Device Communications

Bluetooth tests utilize an external USB dongle containing the Broadcomm chipset and the test tools available in the BT4 installation. The examination of the game devices peripherals is undertaken to document their operational characteristics, through the use of the Hciconfig UNIX command and Bluetooth security software applications. The game device and their associated peripherals are scanned individually and monitored during the pairing process if possible. As with the examination of the Wi-Fi security, attempts may then made to intercept communications between the devices. This will derive an indication of the level of security provided through the encryption methods utilized and the currently vulnerability to available software applications.

Applications and Services Compromise

Examination of the individual devices also requires looking into the possible vulnerabilities of the application or services software that may be in use. This requires utilizing the respective media of the application in question. Tests may utilize online

media repositories setup for security tests, or the establishment of the required source material for the acquisition of compromised test media.

Web Browsers

Initial testing of the browsers on the PS3, PSP and Wii begins with running the online tests available to determine if any security vulnerabilities are present. Tests are run from Scanit Security's online website at http://bcheck.scanit.be following the online tests, recording observed results. These tests relate to known exploits of specific browsers. Utilizing the software code from iexploder - (http://code.google.com/p/iexploder) and Zalewski's MangleMe – (http://lcamtuf.coredump.cx/mangleme/mangle.cgi) local fuzzing tests are run on the lab network. These tests expose the browsers to malformed web pages containing HTML, JavaScript, CSS and DOM code. Respective device browsers are connected to the test system running an Apache web server and left to run loading the pre-generated web pages, any apparent system lockups or browser freezing is documented and the tests resumed at the next applicable page reference until test set completion.

Media Applications – Sound, Video and Images

Data types such as photographs or music are presented to the devices via web page download or from a media server device. Web distribution of files will be via the Apache software on the lab server and the media server software will be ps3mediaserver (http://code.google.com/p/ps3mediaserver/) an open source Java based system. Modified image and sound files (media fuzzing) are accessed in an attempt to adversely affect

system operation. This is a technique which has be utilized in the past to generate stack overruns on other devices , allowing for the injection of specifically crafted code to circumvent or modify system operation. In this case we are looking at disrupting normal device functionality and generating some kind of operational error indication affecting device or application function.

Shared Application Access

Utilizing a PlayStation Portable unit, a game application with a game sharing option is utilized containing a modified content file. The selected file is modified is such a way that it will provide some identifiable characteristic on the client device. Game modules in the PlayStation Portable are in an encrypted compressed archive format, containing game code, sounds and image files. The modified file would be contained in the compressed encrypted archive file which is downloaded to the satellite system, when a request is made to play the shared game. The display of the identifiable characteristic resident in modified file in question on the satellite system will indicate a successful transfer of a manipulated shared access.

CHAPTER IV

DISCUSSION

Tests utilizing OTS security penetration tools designed for personal computers do indeed produce valid results from networked game devices as well. The web applications, networking systems and file sharing are susceptible in the same respects as with previous computing and mobile devices, despite differing design and architecture. A more detailed breakdown now follows.

Device Fingerprinting/Port Scans Results

Utilizing Nmap and Nessus as port scanning tools, each of the games devices was scanned in an effort to identify the type of TCP and UDP ports in use. This provided an inventory of running services of the game devices. Information from the scans allows for a determination of the potential Operating Systems (or its family derivative) in use on the systems. Identifiable services with a corresponding Operating Systems family could then be further explored in the next step, looking for specific documented vulnerabilities. The Nintendo DS system does not have a standing TCP/IP network connection, and so cannot be scanned in this manner. Its Wi-Fi / network capabilities are utilized in a peer to peer/adhoc network arrangement with the Wii, and so will be examined in that area. Nmap scan output screens can be found in Appendix A.

In some cases, the port scans alone did not accurately identify a system; other information may be need to be utilized such as the devices network MAC address or banners from running services. In this regard it's not uncommon to see a list of possible

Operating System/device types for some systems. This will be seen in the following summary Table 4. , where multiple OS possibilities are shown but the device OS is identified with a high confidence, utilizing this other information.

Table 4. Nmap Host Scan Results - Operating Systems/Device Assessment

		Host Scan - Nmap			
	Operating System /Device Identity Assement	Proposed Certainty	TCP Ports	UDP Ports	Possible OS Derivatives
Device					
Microsoft XBOX360	OpenVms 7.2	88%	1	0	Tru64, ReliantUnix
Nintendo DS	N/A - No Standalaone Network connectivity				
Nintendo Wii	Radware Linkproof load balancer	100%	1	1	Dell Switch, Game Console Nintendo,
Sony PSP 3000	Sony PSP game console (modified, running Custom	100%	0	1	NetBSD 1.X,BSD 4.X,SecurOS 6.X,SmartEdge 5.0.3.2
Sony PSP Go	Sony PSP Game Console	100%	0	1	NetBSD 1.X,BSD 4.X,SecurOS 6.X,SmartEdge 5.0.3.2
Sony PS3 Slim	Sony Playstation 3 Game Console Test Kit	100%	0	4	NetBSD 3.X, 5.X
Sony PS3 Fat	Sony Playstation 3 Game Console Test Kit	100%	0	5	NetBSD 3.X, 5.X

It's interesting to note that the Xbox360, Microsoft's product is assumed to be running an operating system or its derivatives, manufactured by Hewlett Packard (HP). To follow the next bit of detective work, one needs to know that HP acquired the company Compaq who themselves had previously acquire Digital Equipment Corporation (DEC). The Xbox360 is not an HP product, so having OpenVMS for an operating system is obviously not the case; it does however offer some clue as to the systems actual OS origins. Looking at the Microsoft product line at the time of the original Xbox production, their 32 bit OS available was Windows2000. This leads to the assumption that a game device may utilize that for its OS. Microsoft indicates however that the Xbox360 operating system is not Windows NT/2000 based as some may assume, but is a unique OS that utilizes the Windows 32bit API's. These base API's came from the 32 bit libraries originally developed for Windows NT, Microsoft's early 32 bit OS (later Windows2000). This early Microsoft OS was in fact developed by a team of

programmers from Digital Equipment Corp (DEC) who developed the RSX-11 PDP and VMS OS. There are historical notes documenting that the NT operating system contained elements from and is similarly functional to VMS, DEC's operating system at the time. The identification of OpenVMS as the perceived operating system on the Xbox360, is then an indication of a carryover of a developer's signature to this device. Hence the operating system of the Xbox line of devices based on a Windows2000/NT type base, utilizing 32 bit Windows API's but not having a clear indicator of Windows lineage shows up as the next probable OS type match in the Nmap signature database, that being OpenVMS.

The Nintendo Wii, is also interesting in the perspective that the assumed system / OS is an embedded device (Radware Linkproof Load Balancer) and yet the derivative OS list has the actual game console device listed. This is an unusual result as the Wii device's MAC Organizationally Unique Identifier (OUI) is clearly identified as being manufactured by the Nintendo Corporation, not Radware. As a comparative example, in the case of the PS3, the OUI and the OS assessment are used to make high confidence and correct device identification, listing the other potential devices as derivatives.

The Nmap port scan summary in Table 5. identifies the TCP and UDP ports detected on the devices. In the case of the PS3 systems, some ports such as the web server and dhcps (dynamic TCP/IP address allocation for the remote client) are only active while in remote desktop mode. The high number UDP ports changed on every system boot up and from data monitoring are related to the devices outgoing web connectivity back to Sony's

online systems. This is in the context of getting information for system updates, new product information, user message notifications and other related items.

Table 5. Nmap Port Scan Results Summary

	Port Scan - Nmap						
	Microsoft XBOX360	Nintendo DS	Nintendo Wii	Sony PSP 3000	Sony PSP Go	Sony PS3 Slim	Sony PS3 Fat
TCP Ports							
	1029 - Ms-lsa - Open		1 - ? - Closed				
UDP Ports							
			30518 - ? - Closed	1 - ? - Closed	1 - ? - Closed	67 - dhcps - open	67 - dhcps - open
						1024 - ? - open	68 - dhcpc - open
						9293 - remoteplay web - open	1024 - ? - open
						58076 - random bootup - open	9293 - remoteplay web - open
						59950 - random bootup - open	58045 - random bootup - open
							58587 - random bootup - open

Utilizing Nessus as a scanning tool produces identification results which are more conservative and less revealing. In Table 6. , we see the software identifying system type possibilities at a lower confidence or not classifying system types at all. Unlike Nmap, it does attempt to determine more accurate systems identification via other acquired information, even when the network MAC OUI is specific to a manufacturer's device. As a security tool, it is however better at detecting some types of services running on the devices, such as indicating systems that respond to UPnP (Universal Plug N Play) queries and providing details on running web based services indicated in Table 7.

Table 6. Nessus Scan Results - Operating Systems/Device Assessment

		Host Scan - Nessus			
	Operating System /Device Identity Assement	Proposed Certainty	TCP Ports	UDP Ports	Possible OS Derivatives
Device					
Microsoft XBOX360	NetBSD 1.6	54%	1	0	RICOH Printer
Nintendo DS	N/A - No Standalaone Network connectivity	No Data	No Data	No Data	No Data
Nintendo Wii	No Data	No Data	No Data	No Data	No Data
Sony PSP 3000	No Data	No Data	No Data	No Data	No Data
Sony PSP Go	No Data	No Data	No Data	No Data	No Data
Sony PS3 Slim	NetBSD 3.0	65%	1	1	
Sony PS3 Fat	NetBSD 3.0	65%	1	1	

Table 7. Nessus Port Scan Results

			Port Scan - Nessus				
	Microsoft XBOX360	Nintendo DS	Nintendo Wii	Sony PSP 3000	Sony PSP Go	Sony PS3 Slim	Sony PS3 Fat
TCP Ports	1026 - www - open					9293 - remoteplay web - open	9293 - remoteplay web - open
UDP Ports	1900 - udp - open					1900 - udp - open	1900 - udp - open

Port Scan Associated Vulnerabilities

The strength of the Nessus tool is that it performs an automated vulnerability assessment based on discovered data. This information can be related to hardware specifics, services running or known vulnerabilities based on the identified devices. Its assessment ranks detected vulnerabilities as High, Medium or Low in concern which are displayed in a summary report table. The table indicates any ports seen, the protocol type, any identifiable service names and the vulnerability counts of those. In regards to counts, the tcp count is actually one less than displayed, as the software counts the scan information summary in this field. In actuality the qualification regarding ranking the 'level of threat' of vulnerability is related to the security environment level required. As an example the ability to 'ping' a system and verify its existence (target detection) is rated

as low category vulnerability. Whereas in other software packages like Nmap, the detection of a system via a ping is considered to be inconsequential and thus not considered vulnerability at all, but yet it does indicate the presence of a target. But if part of your systems defensive strategy is security by obscurity, then one would consider detection a valid vulnerability.

Starting with an aggressive Nessus scan of the Xbox360, Table 8. displays the vulnerability results:

Table 8. Nessus Scan Vulnerability Summary Xbox 360

Xbox360 Agressive | 192.168.0.116 | 4 results

Port ▲	Protocol	SVC Name	Total	High	Medium	Low	Open Port
0	tcp	general	5	0	1	4	0
0	udp	general	1	0	0	1	0
1026	tcp	www	6	0	0	4	2
1900	udp	ssdp	1	0	0	1	0

Here the device responds to UPnP queries (port 1900) and has a web application running at port 1026. Both of these items are noted as low vulnerability. The UPnP query response data shown below clearly identifies this system as an Xbox360, something the port scan data could not.

```
HTTP/1.1 200 OK
LOCATION: http://192.168.0.116:1026/
EXT:
USN: uuid:04763537-2807-2000-0000-0017fa688920::upnp:rootdevice
SERVER: Xbox/2.0.20158.0 UPnP/1.0 Xbox/2.0.20158.0
CACHE-CONTROL: max-age=1800
ST: upnp:rootdevice
```

The response from the web server application has less information that may be useful to an attacker, providing only minimal data to work with.

Protocol version : HTTP/1.1
SSL : no
Keep-Alive : no
Headers :

CONTENT-TYPE: text/xml
Content-Length: 1601

This limited response is due to the presence of built in defensive software detected and reported as:

Synopsis
The remote web server appears to be using anti-DoS countermeasures.

Description
The remote web server shuts down temporarily or blacklists us when it receives several GET HTTP/1.0 requests in a row.

So to garner further information, less aggressive probing measures are necessary. Additional scan data indicates that this port to answers to UPnP query and may have services reachable by SOAP (*Simple Object Access Protocol)* requests. Simply querying the system with a web browser, it responds with an XML output page providing more information on the device such as the devices serial number, a Unique Universal Identifier (UUID) and information related to its capabilities as a media player/server. Detailed output listings of the responses are found in Appendix B.

The scan report also indicates a medium level vulnerability which can be utilized to perform a Denial of Service (DoS) attack or for stealth (hidden) port scans of the host network, Fig 4. The type of DoS is identified as a spank attack, wherein the target system responds with acknowledgement (ACK) packets from a multicast source address. The large volume of generated traffic from this type of attack clogs the network and effectively shuts down the devices networking ability. If the system is not located behind a firewall or router that filters multicast packets, this could be an issue.

Figure 4 Nessus Scan Denial of Service Attack Warning Xbox360

Table 9. Summarizes the results generated for the Sony PlayStation 3 (Fat model), with remote play inactive in the top half and activate in the lower half. These scan also contain indications of active services for UPnP and associated web ports. Here the service serves a similar function those found on the Xbox360 indicating to other devices on the network its required port requests and its available services, but provides much less information when queried as it is functioning in a UPnP client mode only. The web services on the PlayStation system unlike the Xbox360 , are utilized only during the remote play mode utilized in conjunction with a PSP portable device.

Table 9. Nessus Scan Vulnerability Summary PlayStation 3 (Fat Model)

PS3Fat Aggressive | 192.168.0.104 | 4 results

Port	Protocol	SVC Name	Total	High	Medium	Low	Open Port
0	udp	general	1	0	0	1	0
0	tcp	general	4	0	0	4	0
0	icmp	general	3	0	0	3	0
1900	udp	upnp-client	1	0	0	1	0

PS3Fat Agressive Remote | 192.168.0.104 | 4 results

Port	Protocol	SVC Name	Total	High	Medium	Low	Open Port
0	udp	general	1	0	0	1	0
0	tcp	general	9	0	0	9	0
0	icmp	general	2	0	0	2	0
9293	tcp	www	4	0	0	2	2

Information queried from the PlayStation web service in this mode provides little information at first glance, but inferences can be made from the following output.

Plugin Output
Protocol version : HTTP/1.1
SSL : no
Keep-Alive : no
Options allowed : (Not implemented)
Headers :

Connection: close
Pragma: no-cache
Content-Length: 0
PREMO-Version: 0.3
PREMO-Application-Reason: 80029804

The application running is PREMO perhaps a reference to PlayStation REMOte , this is the mode the device is in when this web server is active. The error code 80029804 is a standard eight digit proprietary format used by the PS3 and PSP units as seen from the error message example below:

The PS3 system that this system is registered with was not found. Check that you have selected [Network] > [Remote Play] on the PS3 system. (80029843)

More details related to the investigation of this application are in the covered in the wireless remote assessment section.

The PlayStation 3 slim models vulnerability assessment is seen in Table 10. It is similar to the PlayStation 3 Fat model, with the exception of indicating an additional vulnerability in standard and remote play modes when totalling ICMP counts.

Table 10. Nessus Scan Vulnerability Summary PlayStation 3 (Slim Model)

PS3Slim Agressive - Wireless | 192.168.0.107 | 4 results

Port ▲	Protocol	SVC Name	Total	High	Medium	Low	Open Port
0	udp	general	1	0	0	1	0
0	tcp	general	4	0	0	4	0
0	icmp	general	4	0	0	4	0
1900	udp	upnp-client	1	0	0	1	0

PS3Slim Wireless - Remote | 192.168.0.107 | 4 results

Port ▲	Protocol	SVC Name	Total	High	Medium	Low	Open Port
0	udp	general	1	0	0	1	0
0	tcp	general	9	0	0	9	0
0	icmp	general	3	0	0	3	0
9293	tcp	www	3	0	0	2	1

This increase is related to a common vulnerability with regards to network communications which exists in the PlayStation 3 slim and the PlayStation Portable 3000, whose vulnerability summary is shown in Table 11.

Table 11. Nessus Scan Vulnerability Summary PlayStation Portable 3000

PSP3000 Agressive Wireless | 192.168.0.105 | 3 results

Port ▲	Protocol	SVC Name	Total	High	Medium	Low	Open Port
0	icmp	general	4	0	0	4	0
0	tcp	general	4	0	0	4	0
0	udp	general	1	0	0	1	0

Both devices exhibit ‘Etherleak’ vulnerability in regards to TCP/IP communications. With the Etherleak vulnerability, short received Ethernet frames are padded to their proper size, utilizing data from the devices local communications buffer, system memory

or some other copied data. As such, an attacker may get pieces of user information from ongoing communications sessions. A far more secure method of padding the short frames would be to randomly generate fill data, as the problem with using copied data is the possible disclosure of data.

The leak is triggered by sending short length ICMP ping echo packets, which are then padded with contents from the target hosts data buffers to proper size and returned by the target system to the inquisitor. This can been seen in the samples from a set of captured data, illustrated below in Fig. 5 displaying contents of a broadcast UPnP beacon packet, and a shortened ping sent to the target device.

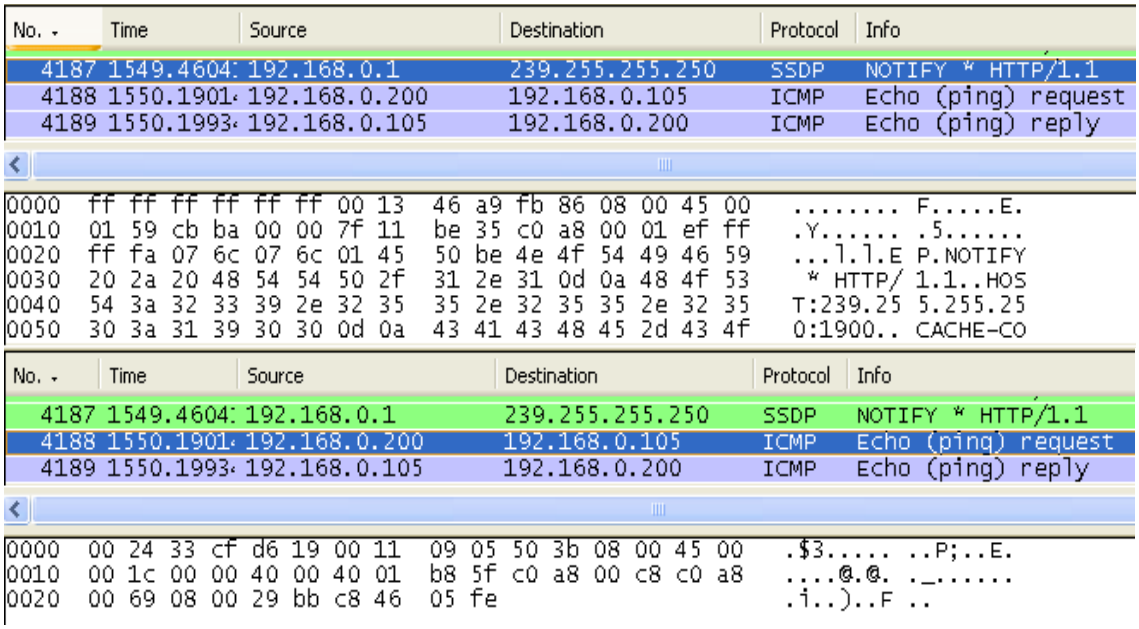


Figure 5 PnP Beacon Broadcast Followed by ICMP Ping to Target Device.

The reply in Fig.6 illustrates the ping response containing a fragment of the broadcast UPnP beacon it has heard.

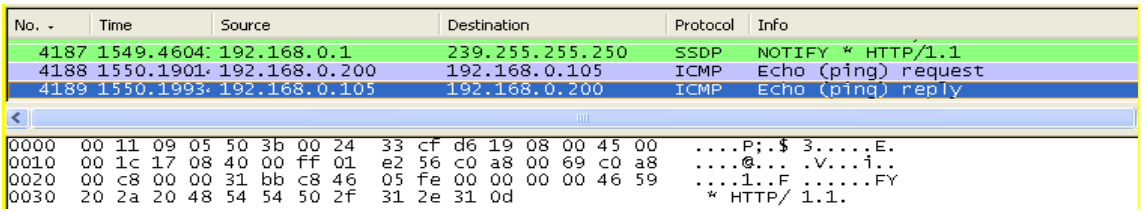


Figure 6 Ping Reply Containing Buffer Response with UPnP Beacon Fragment.

In Fig. 7, the stream of pings is directed to the PSP system while the user is entering account information on the Facebook login page using the portable device web browser, a fragment containing the ‘login’ field text is returned.

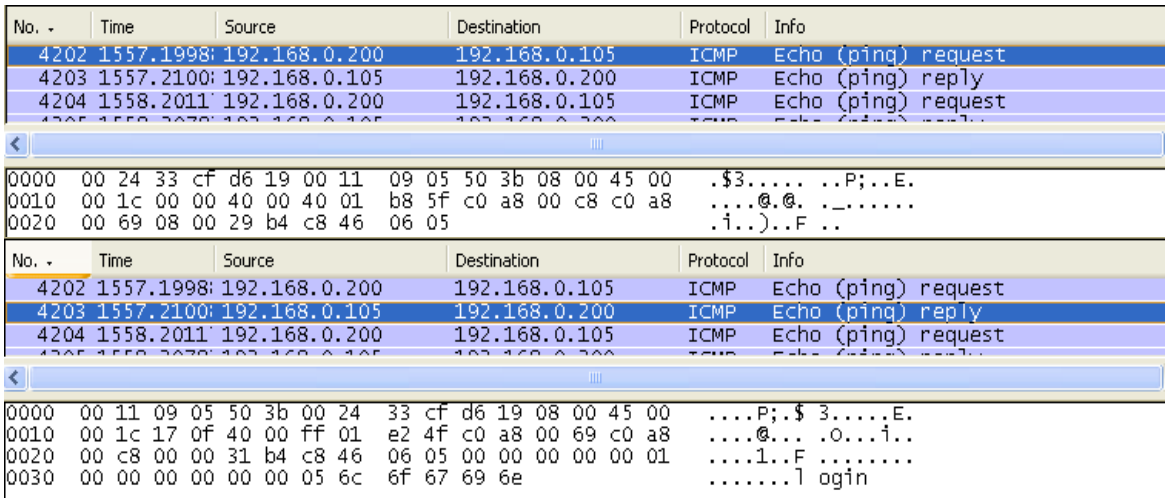


Figure 7 Etherleak Response from Target Containing Facebook Login Fragment

The Sony PSPGo portable device summary Table 12. indicates only low level vulnerabilities, these being the same as the PSP3000 with the exception of the icmp Etherleak which is not found in this device.

Table 12. Nessus Scan Vulnerability Summary PlayStation Portable Go

PSPGo Agressive Wirlesss | 192.168.0.106 | 3 results

Port ▲	Protocol	SVC Name	Total	High	Medium	Low	Open Port
0	icmp	general	3	0	0	3	0
0	tcp	general	4	0	0	4	0
0	udp	general	1	0	0	1	0

The Nintendo Wii has the least number of low vulnerability detections, five in total as seen in Table13. All of the detected vulnerabilities are classified as general in the service name category. The general service name is utilized to encompass a wide range of issues, not associated with a specific port number definition and characterized as port zero.

Table 13. Nessus Scan Vulnerability Summary Nintendo Wii

Wii Agressive - wirless | 192.168.0.117 | 3 results

Port ▲	Protocol	SVC Name	Total	High	Medium	Low	Open Port
0	icmp	general	1	0	0	1	0
0	tcp	general	3	0	0	3	0
0	udp	general	1	0	0	1	0

In Table 14. a matrix summary is presented of the names of these general service types and the devices that responded to them. Colours of the cells represent the protocol utilized (TCP/IP, UDP and ICMP) and the cell shading indicates the presence of the detected vulnerability relative to the devices tested.

Table 14. Nessus Detected Vulnerability Description by Device

Nessus Vulnerability Plugin Activated/Detected	Device						Protocol
	Microsoft XBOX360	Nintendo Wii	Sony PSP 3000	Sony PSP Go	Sony PS3 Slim	Sony PS3 Fat	
Web Server Response							TCP
Detect Ethernet Card Brand							
OS Identification							
Common Platform Enumeration							
Respond To Multicast Packets (Spank Attack)							
Universal Plug and Play							
Respond to Ping							
IP Protocols Scan Response							
Open Port Re-check (Possible Active Firewall)							
AppSocket & socketAPI Printer Response							
TCP/IP Timestamps Supported							
Respond To Traceroute							UDP
Scan For UPnP Hosts (Multicast)							
Timestamp Request Remote Date Disclsure							ICMP
Mutiltple Ethernet Driver Frame Padding Dsiclosure (Etherleak)							
Source Routed Packet Weakness							
Record Route							

Wireless Test Results

Wireless connectivity is the most common method of connecting game devices to a home network and in mobile (portable) environments. The ability to determine the existence of such devices allows for the possibility of directed attacks or the exposure to indirect attack as a node of a wireless network. Characteristics that typically compromise the online security of wireless computing devices are a combination of detectability, encryption and access control. Networked game devices are no exception to these. If the attacker can successfully master these three elements, compromise of the intended target is possible. Using a combination of active and passive scanning tools like Kismet and Wireshark, it is possible to determine the type of device in use and other operating characteristics, such as the mode of encryption utilized and the distance at which the target can be identified, determined through GPS mapping coordinates.

Examined here are only the units that contained onboard wireless connectivity, devices requiring an external transceiver as such the Microsoft Xbox360 excluded. While all the game devices operate in a standard wireless network client mode, the PSP and Nintendo Wii can also operate in a peer to peer (P2P) adhoc network mode. The PSP and Nintendo DS portable units utilize P2P to provide game sharing services, allowing for multiple units to intercommunicate. In the case of the PlayStation 3, it can act as a wireless access point and remote host device, providing PSP clients with remote desktop functionality.

Wireless Device Identification

The determination of device type can be from a variety of wireless fingerprint characteristics including the MAC address of the wireless transceiver, any broadcast beacons and/ or analysis of captured traffic between devices. The Nintendo Wii and the Sony PlayStation 3 both have MAC addresses registered to their respective manufacturers identifying those devices associations, while the PSP portable devices utilize more generic wireless chipsets. The PSPGo device has an Ani Communications chipset and the PSP3000 using a chipset from Alps Electric. Thus while one can distinguish the type of game console devices on a wireless network using only the MAC address as it is directly related to the manufacturer, the portable devices would prove to be more challenging.

Beacon and wireless traffic wise the devices offer more identity data in the context of the portable devices and provide additional identity verification for the console systems,

including encryption used. The PlayStation3 broadcasts a ‘PS3-xxxxxxx’ beacon in remote mode as seen in Fig. 8.

Figure 8 PlayStation 3 Remote Mode AP Beacon Info

The Wii, when in Peer to Peer (P2P) mode offering downloadable game demos to a Nintendo DS portable device beacons as ‘Nintendo Ch’ and the name of the game download offered, displayed in Fig. 9, the Wireshark capture below.

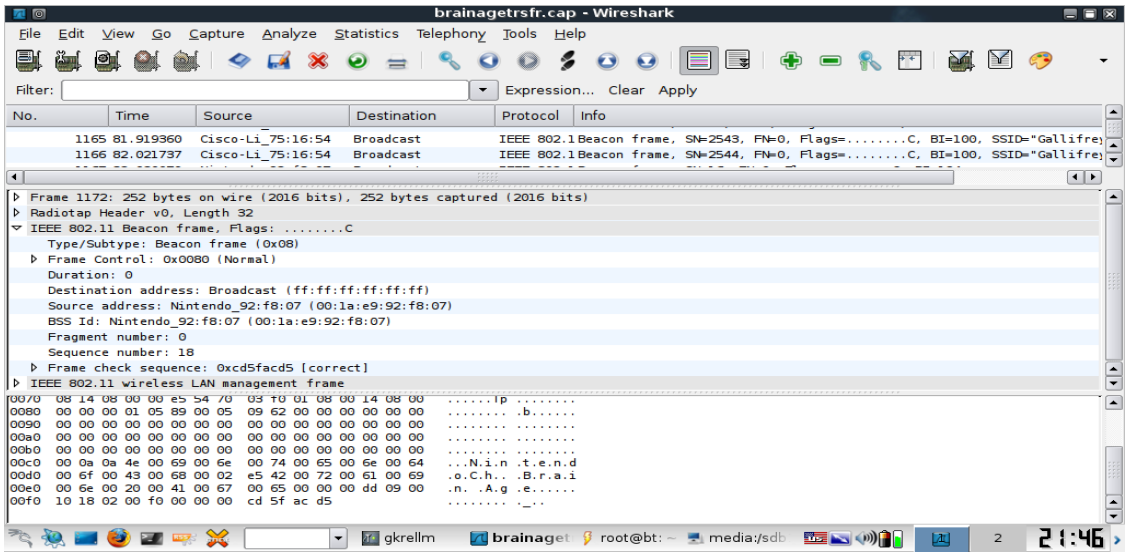


Figure 9 Wii DS Download Beacon Remote Info

This self identifying beacon characteristic related to portable devices also avails itself in the case of the PSP devices in their P2P game sharing mode, broadcasting a beacon of

'PSP_...*download game id*... GameShar, as illustrated in Fig. 10. Information from these beacons allow for positive identification of the devices in question when they are utilizing such game sharing modes should an attacker be looking for a specific portable target. While this data is collected by passive device sniffing, active data can also be collected if the attacker associates with the wireless network the devices are on. Again tools such as Nmap and Nessus are utilized to probe them; as seen in the device fingerprint data shown earlier Nintendo Wii, PS3 and PSP device information was collected in this manner.

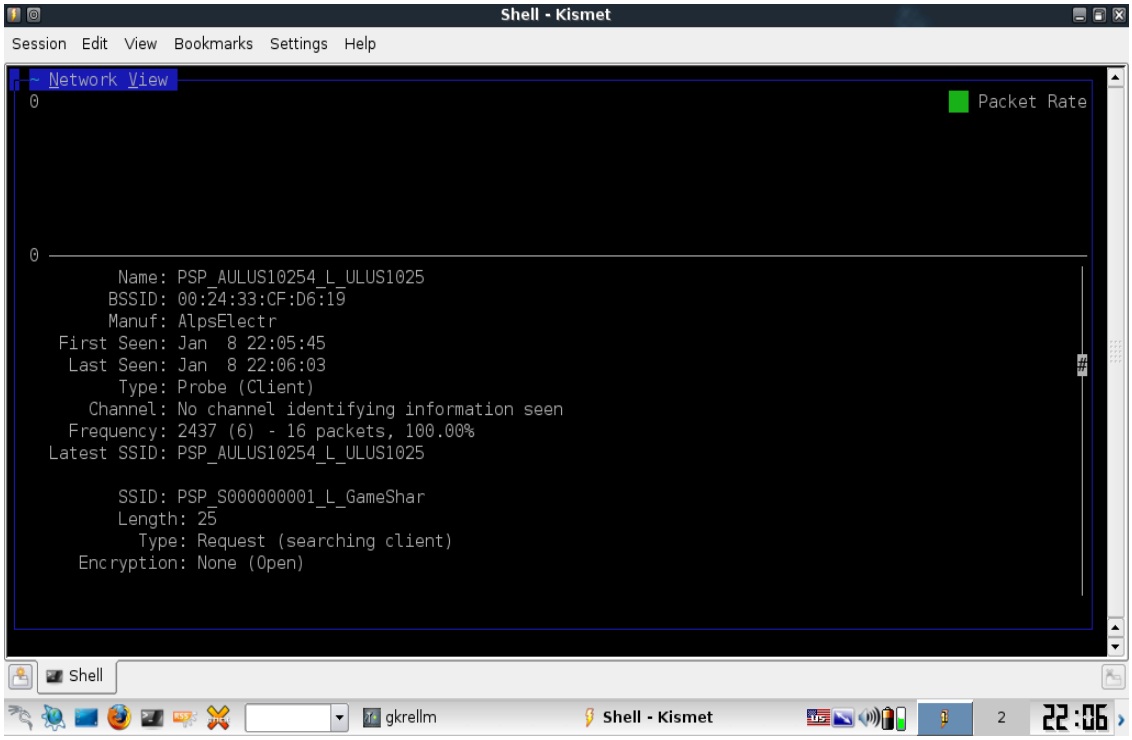


Figure 10 PSP Game Share Beacon Remote Info

Wireless detection range - consoles

Console devices are the most likely candidate for detection at long range due to the greater power output of the transmitter unit as well as the propensity to leave the devices powered on for long periods to act as hosts for the portable units. Knowing these devices can be detected and identified, we can use passive scanning combined with GPS coordinates to determine the range of such signals. The range of these wireless devices under certain circumstances can be quite large. Examining signal range of a PS3 system in remote play access point mode located on a ground floor as illustrated in Fig. 10 versus a second floor operation in a typical house shows a possible detection distance of 200m with the unit on the second floor. This information is depicted in following diagrams, where collected signal strength is mapped with GPS coordinates then overlaid on an aerial map.

Figure 11 PlayStation 3 AP Beacon Range - Ground Floor

Figure 12 PlayStation 3 AP Beacon Range - Second Floor

The large distance indicated in Fig. 12 is due to the line of sight propagation characteristics of the 2.4 GHz band signal used by Wi-Fi. The target location is always visible along the long axis. Utilizing that observation, any elevated location within the circular zone indicated in Fig. 13 should allow for interception and interaction with the game console.

Figure 13 PlayStation 3 AP Potential Beacon Reception Area - Second Floor

Wireless Security Assessment

Connecting to a wireless network as clients, all the examined systems have the capacity to utilize the standard Wi-Fi encryption modes and as such the possibility exists that their traffic may compromised utilizing the same techniques that are applied to wireless personal computer networks. The encryption formats available for the devices include the older Wireless Equivalency Privacy method and the newer Wi-Fi Protected Access techniques. Tests against the devices indeed show that the same software applications utilized to break these encryptions for personal computer networks, work on the game devices as well

Additional Wi-Fi configurations of the portable game devices such as the PSP and Nintendo DS rely on an Ad-Hoc network mode which has no encryption present in the communications channel, but does utilize device specific encryption formats for the data contents being transmitted. This content encryption is an extension of the Digital Rights Management technology utilized to protect the content, typically games from being pirated. As such each device contains the necessary keys utilized to decrypt the game binaries, thus requiring no exchange of key data between the devices for decryption. In the test scenario that follow looking at data traffic, Kismet software is utilized determining the type of encryption protocol in use during a remote access session between a Playstation3 and PlayStation portable. Once determined the appropriate packages for compromise are then utilized in an to attempt to get the devices wireless encryption key (password or pass phrase).

Remote Access

While most of the devices operate only in a network client capacity, the PS3 additionally has the capability to operate as an access point in the remote host mode. This mode of operation includes remaining active when the PS3 is powered down and the ability of the portable unit to wake the system. Reviewing the beacon data of the PlayStation in remote access mode Fig.14, we see the following information, indicating the encryption in use is WPA PSK AESCCM or WPA2 as it is commonly known.

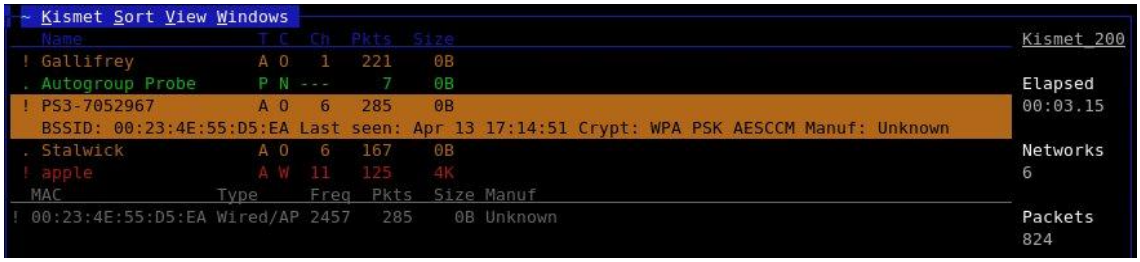


Figure 14 PlayStation 3 AP Beacon in Remote Play Mode

Before being able to utilizing a PSP device for the remote access connection, it is required to 'tether' the portable unit to the PS3 via a USB cable. During this process the PS3 transfers its wireless password to the PSP and then records the potables MAC address. This then creates an automatic remote access profile for the game player (no need to remember or key in the WPA password). The PS3 also supports remote connections from other devices that are specifically of Sony or Sony related manufacture, Sony VAIO PCs and Sony-Ericsson Aino Cell phones. When selecting a remote connection for these devices, there no request to tether the device via USB. Rather the user is asked to connect to a specifically generated access point name for the PS3 and enter a system provided eight digit access code, all within a five minute time limit.

Trying to break the encryption key for the traffic between the remote access devices requires first determining if there are any modifications WPA2 protocol encryption utilized. As there appears to be recognition as a standard 802.11x AP via Kismet, we can capture the traffic between a previously tethered PSP and the PS3 and see if there is a recognized encryption protocol we can attempt to attack. Capturing a connection request between the two devices as seen in Fig. 15, we see Wireshark it recognizes the handshake protocol being utilized as Extensible Access Protocol over LAN (EAPOL) and its two-way authentication traffic. Hence it should be susceptible to a standard WPA decryption attack.

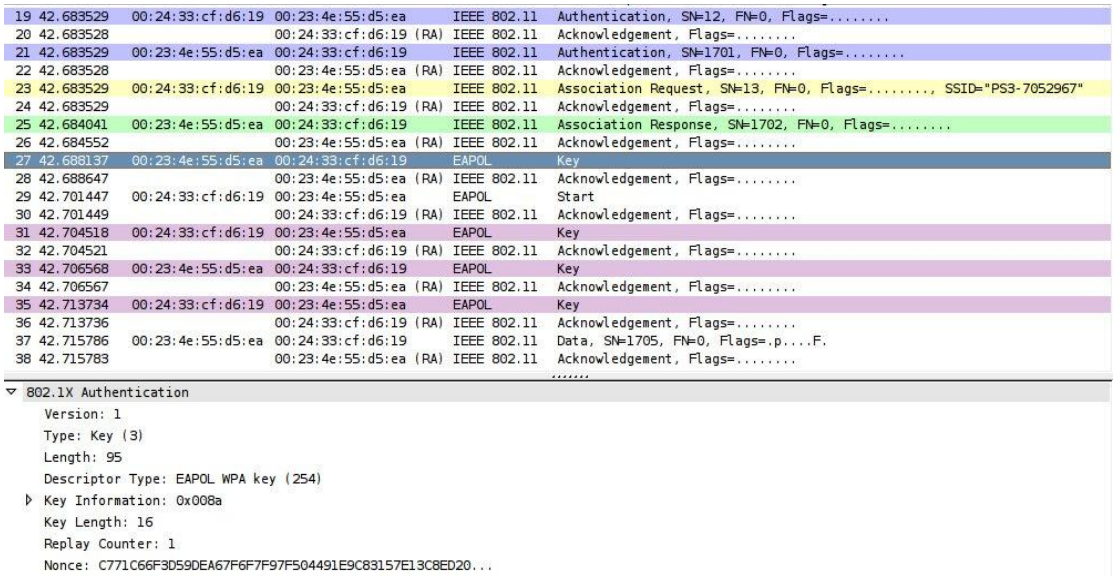


Figure 15 PlayStation 3 to PSP Handshaking

The process to test this utilizes two steps; we capture the WPA authentication sequence and then attempt to decipher the key via dictionary attack (presuming the algorithms are standard). Capturing the traffic between the PSP and PS3 with Airmon, an application designed to detect and capture the data needed to extract a WPA key. As we

initiate remote play on the PSP, we should be able to capture the necessary data. The data is then processed through Aircrack which utilizes a dictionary attack method to determine the key in use. From the Wireshark data we see a current key size of 16 bytes, either system generated or predefined. As this would take a long time to brute force or with Rainbow tables, be equally as long to determine the randomly generated key, we will create a simpler key test scenario to validate some of our assumptions. For the purpose of our second step, we will set the PS3 Remote Play/WPA password to **AthabascaU** and include that word in our key search dictionary for our password cracking software. We will then re-register our PSP with the PS3 for the remote play session, our chosen password should now be the key utilized for association having been transferred from the PS3, verifying that this is indeed what happens.

Running Airmon during the connection process in exchanges between the PS3 MAC Address (00:23:4E:55:D5:EA) and the PSP MAC Address (00:24:33:4E:D5:EA) we do indeed get handshake data of 2289 packets as indicated in Fig. 16

Figure 16 Airmon captured WPA Handshake data

Taking that data and feeding it into Aircrack, we successfully retrieve our chosen password shown in Fig.17. This verifies that the wireless password information is transferred to the portable device, that the Wi-Fi encryption format utilized is standard and that OTS wireless attack tools will work against these game devices.

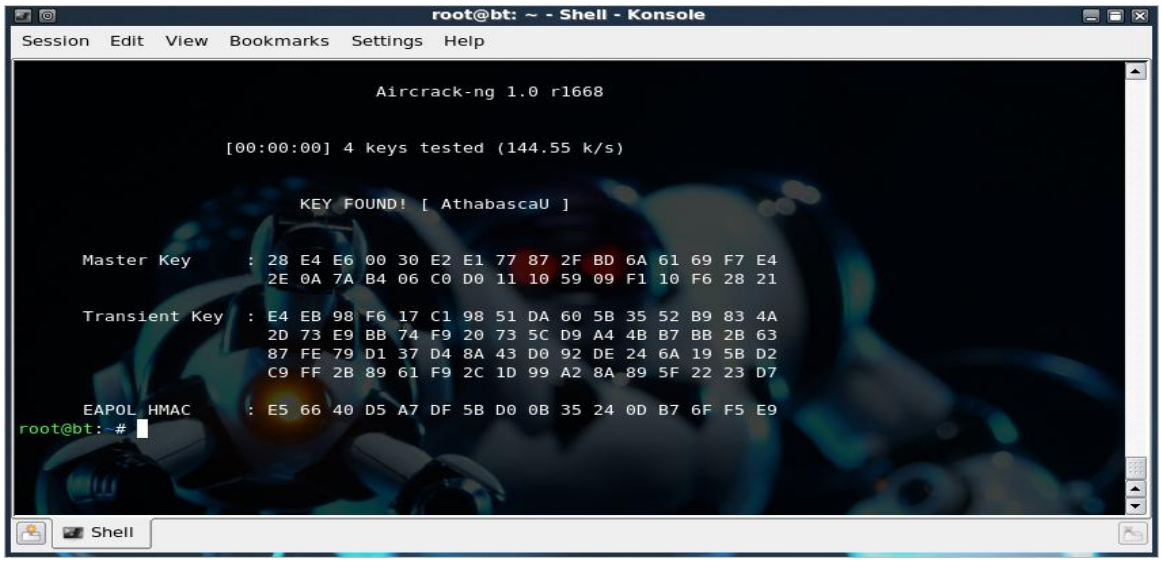


Figure 17 Aircrack Discovered WPA Handshake Password

Game Devices - Application Vulnerabilities

Here the focus is on the applications available on the devices (or within an application – e.g. game sharing) which may be exploited through the use of modified content containing malicious payloads. A typical vector for these attacks is on websites, planted by attackers who would rely on the content payload to be activated via pages viewed, media displayed or games downloaded to carry out the attack. Additionally a peer to peer vector may be utilized through the game sharing function.

Web Browsers

The web browser application tests utilize the Sony Playstation3, the PSP portable units and the Nintendo Wii. Here the purpose is to see first, if exploits or vulnerabilities that are previously known will affect these browsers, secondly to see if browser testing tools available can elicit those errors or malfunctions. While the Nintendo Wii utilizes an Opera based browser, the Sony systems use a non standard embedded browser. The PSP systems report to be Netscape 4.0, a pre Firefox Mozilla based browser, while in reality it is NetFront, a mobile browser found in many phones produced by the Access Company of Japan. The PlayStation 3 system browsers identify itself as a Netscape 5.0 browser line.

The site at http://bcheck.scanit.be is utilized first and a preconfigured test set of 40 items is run while observing the systems for any signs of malfunction. These tests examine the ability of the applications to handle flaws in JavaScript, PDF, Content Style Sheet (CSS) errors and malformed HTML pages. The primary target of the exploits on this page are directed towards Microsoft, Firefox and Opera based browsers. The Nintendo Wii system displayed no signs of any identified errors when run, the PSP portable devices while not experiencing applications errors in the context of malfunctions, did have problems with memory limitations in tests that required the opening of multiple windows. Such an occurrence in a mobile application can be classified as a resource attack if deliberately employed on a website targeting the PSP. Similar attack concepts have been utilized against embedded portable devices (PDA, cellphone). Those types of attacks utilize items that make the mobile devices use excessive memory or cpu, slowing the device or expending battery power at a higher than normal rate.

The Playstation3 browser(s) (the FAT and slim model) both experienced errors when running the browser tests from the website. Lockups were observed to occur with two tests as indicated by the halting of an animated ribbon display on the browser background screen. These occurrences resulted in the requirement to perform a hard shutdown and restart the of the PS3 systems. During system these shutdowns, the PlayStation issues three beeps, indicating an internal error condition exists at the time of the shutdown. Upon restart, the system prompts the user to permit the system send an error report to Sony, an indication typically of a serious error having occurred (similar to application crash messages in Microsoft and other software packages). This error dump is transferred via an encrypted SSL session to Sony.

When a typical program/application execution error occurs in most modern device operating systems it will result in the shutdown of the application only, not the operating system. Typically it takes a severe error condition to not only affect the application but the Operating System (OS) as well. Web browsers on personal computers are capable of experiencing severe errors as well, but typically only the application is affected being shutdown and the user is then returned to the operating system level. However the Sony PlayStation utilizes a Hypervisor to restrict and control access to the operating system and the hardware therein. This is done to isolate applications and the user from the operating system and its associated hardware components. As such, one would expect that a greater level of "safety netting" would be experienced with the game console and resulting severe error conditions causing it to lock up be a rare occurrence.

The Hypervisor construct has three levels referred to as Lv0, Lv1 and Lv2 represented in Fig.18:

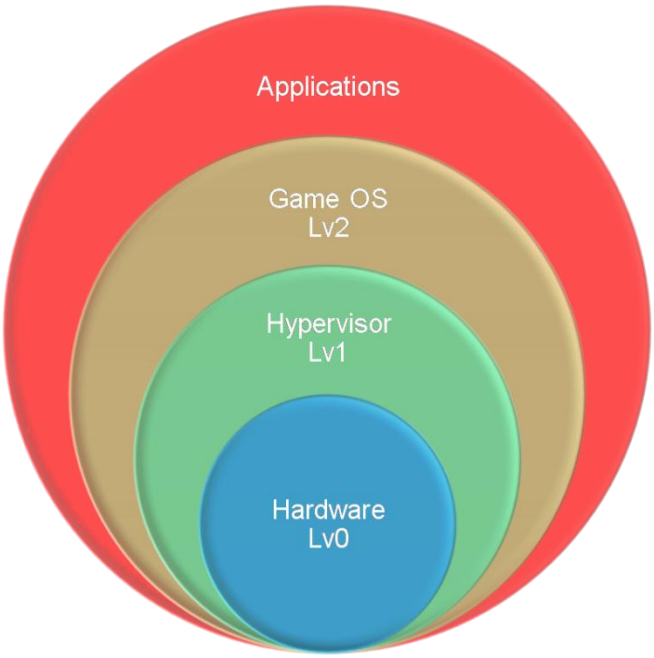


Figure 18 Hypervisor / Virtualizer Levels

From this figure, we can see that to stop the GameOS (the background ribbon representing the OS screen) we are affecting Lv2 of the Hypervisor. Any secondary program running in the background on the PlayStation, such as an audio stream player also stops when the ribbon animation quits during the tests, suggesting we are affecting more than just the applications level of operation (i.e individual processes). Scanning a halted system with Nmap, before resetting it shows that the same ports that were active during normal operation are still shown as open, suggesting that Lv0 and perhaps Lv1 are still active. This would account for the ability to shutdown the system with the touch activated on / off switch (Lv0 Hardware) and the ability for the system to write out a

dump/log file containing the error information to send to the manufacturer (Lv1 Hypervisor).

The web errors that trigger this malfunction on the device are two old exploits from 2004 and 2002 related to Microsoft's Internet Explorer. Looking them up on the Securityfocus.com site via their Bugtraq ID numbers (bid) we find they are:

- Microsoft Internet Explorer JavaScript Method Assignment Cross-Domain Scripting Vulnerability (bid10689)

Here originally the condition to worry about was that two opened browser windows from two sites are executing JavaScript code with different trust (security) levels. This made it possible to create/execute method calls in the higher level security window, from the lower level window and not have the trust relationship validated (which would normally not permit such an action).

- Microsoft Internet Explorer Document Reference Zone Bypass Vulnerability (bid5841)

This error was related to Microsoft's Internet Explorer concept of Security Zones (trust levels) related to websites. Here an attacker could execute code via a document object which was accessed via a saved reference. The system did not perform access control checks on the code, executed in this fashion. Again the problem is having two windows (websites/domains) one having a trusted security status the other un trusted and being able

to execute code from the untrusted domain in the context of the trusted domain. The PS3 browser does not utilize any trust checking concepts such as security zone settings, so the question here is what aspect of the JavaScript called for in the code execution in the second opened browser window leads to a system lock up?

Next various software scripts and test packages are utilized locally to generate and present malformed content pages to test the browser a process known as 'fuzzing'. These test are run utilizing Iexploder , originally a Quality Assurance test kit created for Firefox 1.0 testing and based upon Michal Zalewski's mangleme.cgi code. Iexploder code is now available at the Google Projects site for use as an open source public test tool. Additionally, from the BackTrack 4 security suite, Browser Fuzzer 2 (Bf2) from Krakow Labs Development is used. A Perl based package that generates 82,400 test web pages containing HTML, Content Style Sheets (CSS), Document Object Model (DOM) and JavaScript content, totalling 76GB worth of data. Both of these packages run iterative tests on the browsers (each page executed successfully calling the subsequent page), with experimenter intervention required to document system anomalies and restart the tests for continued iterations. Web services are provided for Iexploder by a ruby webserver, included with the application while a typical Apache web server is utilized to run the Bf2 tests.

As with the previous set of browser tests, the PSP portable units had a hard time running the test page configurations. Typically the run times for pages being extremely slow and running out of system memory quickly, as such test runs were not completed utilizing these tools due to the large data sample sizes that are supplied. For the

Iexploder tests, the Nintendo Wii completed the entire test run with no indicated errors. The PlayStation 3 systems both the slim and the fat models appear 'hang' on test #44. Attempts to restart test page #44 by accessing the browser menu and entering test page URL causes the systems to lockup (no ribbon movement on the OS background) requiring a restart, as does simply trying to exit the browser from the menu after this point. Unlike the previous tests however, there is no memory dump/upload request on system restart. Thus while causing the system to hang; it appears this flaw either does not affect the Operating Systems level to the extent that it is necessary to send Sony an exception report or it impairs that function. Restarting the test pages at #45 allows the PS3 to run the remaining tests to till completion, as seen with the Wii, however exiting the browser at the end of the completed test run causes the system to lockup.

Results for the Browser Fuzzer 2 (bf2) package yielded a series of 'memory out error' messages on the Nintendo Wii for pages that where in excess of 5 Mb in size. This was consistent for all file types CSS, DOM, HTML and JS. While this slowed the browser and required user intervention to acknowledge the error, it did not cause the browser to fail or close. The PlayStation 3 systems, the slim and fat models showed differing results in the processing of these test files. While both PS3 systems have the same hardware processor and memory amount, the slim runs the latest version of the PlayStation OS whereas the fat remains at an older version that supports a second operating system.

Both PS3 systems exhibit a loss of controller input processing and current page display by the browser after executing the css261.html file. The css261.html web page is

a five megabyte file, containing a small header section and a body consisting of a repetitive string sequence of 'A/A/A…' This page results in what appears as an endless load attempt; as indicated by the onscreen clock animation, the symbol representative of the browser loading a page. While the browser URL title display indicates it is 'stuck' on css261.html, monitoring the ongoing connection utilizing the Ettercap sniffer shows the browser will continue to load the sequence of test pages thru to the final page. Upon exiting from the browser back to the |OS screen, there is no ability to utilize the onscreen XMB interface thru the controller. The ability to scroll via the joystick inputs is frozen and the control buttons inactive. The only controller command response is to the system shutdown button, the PS button also utilized to start the system from a power down state. Upon restart the system reboot process presents a request to submit a file dump to Sony on power-up, as with earlier critical operating errors.

The DOM page test sequence executes without problem on both PlayStation platforms, as does the Nintendo Wii, not exhibiting the 'memory out' error messages that it displayed with the CSS test pages. Moving onto the HTML pages we do see both PS3 devices displaying an 'out of memory' error for a subset of pages and the ability to lockup a system (background ribbon stall) executing other subsets. Running the pages from html1 to html708 causes a system lockup, with the subsequent need to do a shutdown/restart and resulting in a system dump. Narrowing down to a subset of those, from html700 to html708 the URL display title indicates progress to page html708 then generates one or two memory out error display messages. After user intervention (answering onscreen message prompts) the URL display title indicates the browser was

actually halted at html706. Since the page loading was stopped (no indication of download activity) doing a refresh from that point resumes the page loading sequence which advances till page html708 where the system locks up. For these system restart scenarios, the error reporting function is not consistent as with previous crashes. Here we see a 45% occurrence of the system dump send request upon power-up on the PS3 fat model and a 100% occurrence on the PS3 slim model.

Moving onto the fuzzed JavaScript pages, they also caused out of memory errors and system lockups on the PlayStation units, no errors were seen with the Nintendo Wii. The page causing problems js704.html is relatively small in size 500 kilobytes, unlike the previous test pages causing memory errors which were 2 megabytes in size or larger. This page contains a single variable allocation and an evaluation 'eval()' function on a large string comprised solely of "AAAAA…" Executing this page causes a lockup of the system immediately (no ribbon movement), and requires a system restart resulting in a 30% error dump reporting screen on restart for the PS3 Slim and 45% occurrence for the PS3 fat.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

It has been found that the game devices having implemented characteristics of personal computing devices in regard to network communications, applications software and common content access, such as web and media files find themselves susceptible to the tools and attack methods implemented against personal computing devices. This is consistent with Jiwnani and Zelkowitz (Jiwnani & M, 2004), in that similar vulnerabilities can exist in differing systems which have similar functionality, here represented as game devices and personal computers. As such one can assume that these game devices are in the potential attack/vulnerability chronological target 'queue' when considered with respect to Leavitts work (Leavitt N. , 2005), (Leavitt N. , 2000) on devices with networking and personal use capabilities and work done by Akriditis et al. (Akritidis, Chin, Lam, Sidiroglou, & Anagnostakis, 2007) related to the critical mass of user adopted technologies.

While currently the devices respond to attacks or vulnerabilities associated with personal computing devices, it is conceivable that in future directed attacks on these systems will occur. The potential exists to exploit the vulnerabilities currently present in game device applications, while recent data released by hackers regarding the security and encryption of executable binaries for the PlayStation 3 and PlayStation Portable make a scenario of malware infection possible.

Device Characteristics Tests Potential Exploitation

The two currently existent vulnerabilities are related to characteristics related to the network functionality of the devices, the UPnP protocol and network communications buffering. When the Xbox360 UPnP service is queried with a web browser it responds with an XML output page providing a large amount of information on the device such as the devices serial number, a Unique Universal Identifier (UUID) and information related to its capabilities as a media player/server. This is a prime example of probing for device information which can be further explored to determine if there are any potential vulnerabilities in this regard to this acquired information. For example does the device serial number and UUID give an attacker some relevant information edge to access the owner's online information in Microsoft's Xbox Live environment? Such divulging of information is of concern in a world where attackers have the ability to piece together disparate pieces of information, which when combined provide identity or authentication credentials.

The Etherleak discovered in the Sony units is a design vulnerability that could have been avoided. The premise of using this as an attack vector on a PlayStation console would take effort on the part of an attacker if the unit is not located outside the firewall perimeter. In this regard it would be needed to penetrate the home network and get behind the routers firewall, which if accomplished one could sniff the traffic on the network through Man in the middle (MITM) redirection methods. However this vulnerability may be appealing from the portable device perspective. Such devices may be using public

access points, as in a coffee shop or school where the attacker can also easily establish a connection to the same network, making this attack trivial.

Wireless Mode Tests

The security concern with the remote play mode of the Playstation3 is heightened in that the console activates its own wireless access point capability, even though the game system may be hardwired into the network, or utilizing the home network access point. As such another wireless network connectivity point (Access Point) is activated behind the established presumed secure firewall as illustrated in Fig. 19.

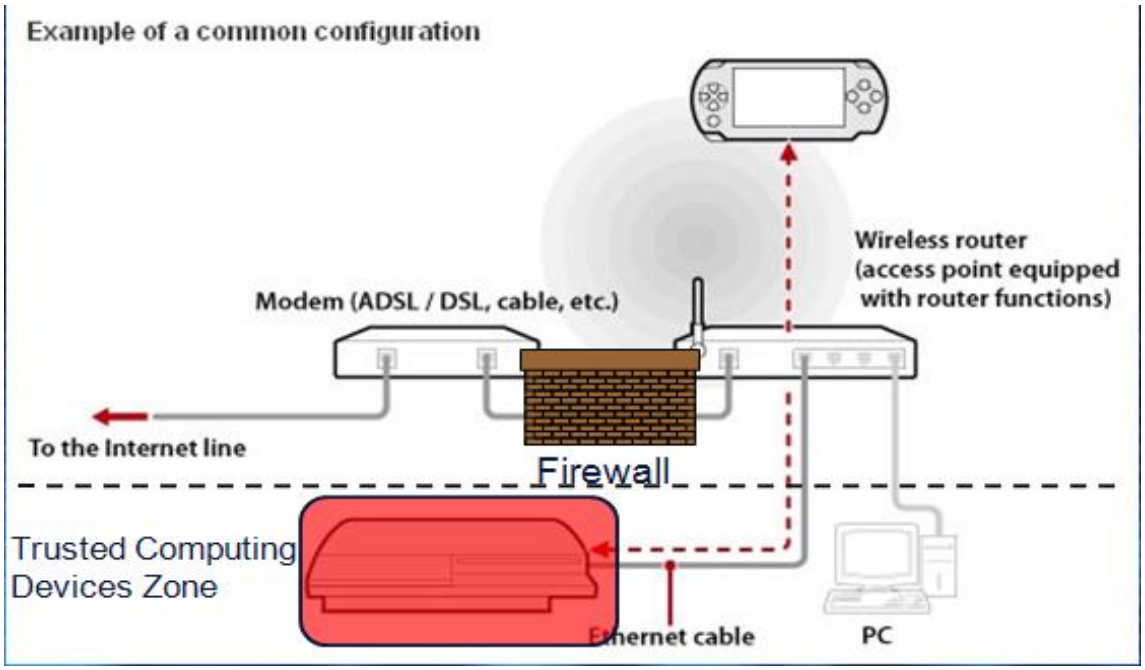


Figure 19 PlayStation 3 AP Inside Secure Home Network

If a connection can be made to the game device via the attacker impersonating the registered remote PSP user, this then presents the possibility of accessing resources of the host network system from the inside the presumed safe perimeter, behind the users

firewall. Another potential problem here is the remote play function can be set to allow the PSP unit to 'wake' the PS3 remotely and start a remote session. While this mode is a convenience for the game player, having it activated on keeps the wireless Access Point mode functional even while the game system is powered down.

The recording of the MAC address as displayed on the authorized remote devices screen of the PS3 implies that some form of MAC address filtering could be utilized. If so, software can be utilized to easily spoof the PSP MAC address (captured via monitoring with Kismet for example) of the attacking computer, making it look like a registered client bypassing the MAC address filtering protection. It has been shown that the password for the remote device connection to a PS3 is be determinable if a user enters a weak custom password, rather than the utilizing PS3 generated default. By applying enough processing power, cloud computing based perhaps or utilizing new software and algorithms, it may also be able to determine the system generated password. This is vulnerability exists due to the fact that standard encryption protocols (WPA2) are utilized and the negotiation and connectivity to the access point relies on standard 802.11 protocols, hence the existing tools for cracking such codes can be utilized.

In regards to the remote access via Sony manufactured devices, such as cell phone or computer, from an attacker's standpoint, cracking the eight digit access password is in reality only determining four digits. The numbers broadcast in the AP beacon constitute the first four digits of this code. The format for the beacon in PC remote play access mode is PS3REGPC_XXXX, where XXXX constitutes the first four numbers of the

required eight digit key code, while the phone mode beacon is PS3REGIST_XXXX. One may assume, since the PlayStation system has a pre existing wireless encryption key for remote access (or one selected by the user) , that the 8 digit code given to the user to enter is either:

i. A temporary WPA2 encryption key, valid only for that device session.
ii. A temporary encryption key for a protocol which allows for the secure exchange of the existing remote access WPA2 key.

If one can create the ability to generate multiple connection attempts within the allotted five minute timeframe, it would be possible to brute forcing the last four digits of the registration key. If the pass code usage is for a secure key exchange, this may lead to a method of acquiring the PS3 default remote access WPA key. An examination of the remote connection process using a modified version of the Sony Vaio client software, Open Remote Play (Open-rp) http://code.google.com/p/open-rp/ made available by the hacking community may provide answers to this question, through an examination of captured wireless session data. An attacker may also need to consider if MAC address filtering is still in use somehow dynamically here, utilizing the client devices MAC address of Sony manufacture when an authorized connection is made but this would again be trivial to spoof.

Applications Vulnerability Implications

Of interest here is the discovery that vulnerabilities related to web browsers that are six to eight years old and for a different vendor's product can affect the game console web browsers. This is an excellent example of an indirect attack vector that can affect game consoles, where the intended target system is not the type accessing the material but is still adversely affected by the code. It also exemplifies that the web browser can be utilized as a direct attack vector influencing device operation. The ability to generate an error in a device by a modified web page and then transfer and activate a piece of malicious software (payload) is commonly utilized by Cross Site Scripting (CSS) attack or social engineering phishing or redirect attacks. This web based avenue of automated attack, along with the possibility downloading malicious software can be considered a high probability as per the findings of Provos et al (Provos, McNamee, Mavrommatis, Wang, & Modadugu, 2008) and Adams (Adams, 2009).

For a web based attack, the required elements would need to consist of:

- Causing a malfunction in the web browser which affects device operation.
- The utilization of that resulting error in the execution of some piece of transferred code via a buffer overruns condition for example.

Or

- Making available for download of a piece of software which has been modified to perform functions other than originally intended.

The defence against such modified binaries is the necessity of having valid 'signed' code, verified by the device before execution. Executables for the PS3 and PSP are of a standard Executable and Linkable Format (ELF) which has a security header appended to it, creating 'Secure' ELF file or SELF. This header contains a encrypted metadata key which when decoded using the game device keys provides the AES Keys required to decode the ELF binary and validate its SHA-1 signatures. The metadata key also decrypts the Elliptic Curve derived signature, used to sign and authenticate the secure header section, preventing it from being tampered with. These types of security features were intended to ensure that only validly approved and developed code could be run on the devices. As long as the chain of trust is maintained with the encryption process the devices secure status would be maintained.

However in December of 2010 the processes for retrieving the various crypto keys for the Sony systems were divulged through the efforts of a group of hackers known as fail0verflow. In a presentation at the 27th Chaos Communications Congress, an annual Hacker information sharing conference, it was revealed a number of vulnerabilities existed in the security architecture design that systematically enabled the compromise of the device. One flaw contributing to this was that the Hypervisor in boot loader mode did not check for signed (authorized) code, allowing for custom code to be executed, without additional security checks. As such an environment was available were the typically isolated cryptographic functions in the systems firmware and hardware were now

accessible via modified operating system code being introduced. In conjunction with an exploited USB service loader process utilized to 'jailbreak' a PlayStation 3 system (the process of modifying firmware to run non standard code application) it was possible to load an encrypted binary into the systems memory space. Once that was accomplished, it was possible to have the PS3 system itself extract the encrypted files AES keys. Additionally, Sony neglected to use a random value in the use of its Elliptic Curve cryptographic algorithm ECSD. By utilizing two ECDSA signatures extracted from encrypted binary applications, it was possible to work out the private key originally used to sign the applications (Borza, 2011).

These factors make it now possible to create and sign binary executables, including boot loaders for the PlayStation 3 and thereby modify it functionality. These encryption exploits also allowed for the discovery of the information related to PSP encryption keys as well, as the PS3 system can play mini games, originally designed for the mobile devices and therefore needs o be able to load those. As the potential to create 'signed' software that will execute on both devices exists, the ability to create malicious software payload packages now becomes available.

Application Exploitation via Game Sharing

Game sharing is a mode for the portable devices where by an ad-hoc network is activated between devices allowing multiplayer game scenarios or the transfer of a demo copy of the host's game software to a client system. The Nintendo DS and the PSP units both have this capability; additionally the DS units have an inbuilt messaging function as

well as the capability to download demo games from a Nintendo Wii system. Potential vulnerabilities affecting these devices can be privacy and identity related, as in the potential to eavesdrop on chatting sessions or the possibility of running a shared demo program that has been tampered with maliciously. The ad-hoc networks employed have no wireless encryption protocol active but do utilize other methods to obfuscate the data being exchanged.

The Nintendo DS utilizes RSA encryption to protect data, while Sony's PSP and PS3 employ an AES128 encryption format. As noted previously, this encryption is the basis for a security trust model which should protect the person receiving the shared or downloaded game from malicious tampering. This trust model presents problems however if the encryption utilized is compromised, which many parts of it have been to date. With access to the crypto keys, the possibility to exploit shared application programs also now exists as an attacker can decrypt, modify, encrypt and sign software making it executable on the devices. As such, a properly signed, a modified version of a game demo could be created which would be run able and shared, just as with any as a trusted game application.

The PSP's offers the greatest possibility of distributed software tampering as game files acquired for the device are from online sources and are installed via USB connection to a PC. This process presents the opportunity for examination and modification of those files while they are on the personal computer system and before uploading to the game device. A PSP software developer kit (SDK) has existed for some time to create

'homebrew' (non digitally signed) software on PSP systems that have custom firmware (cfw) installed. Custom firmware was required previously to allow the execution of non-digitally signed applications. The proof of concept test for this hypothesis would be to create a modified file on the host system which would then be transferred to the client and run without problem. This would validate the hypothesis as to the feasibility of malicious shared game software as device vulnerability.

Executable files on the PSP are packaged in the EBOOT.PBP encapsulating file format, containing multiple files. Contents of EBOOT include graphical files in .png format, audio, and other data files. One of these the embedded elements in the EBOOT file, the PlayStation Archive file (.psar) contains the files pertaining to application execution. It is this encrypted psar archive file that contains the transferable game demo files/code that could be modified by a malicious attacker.

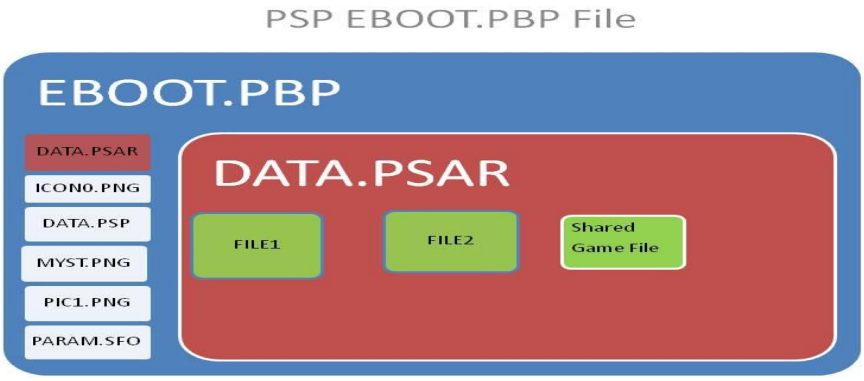


Figure 20 EBOOT.PBP Archive Contents

EBOOT.PBP is not only a common filename utilized for all game applications, it is also used for firmware and system updates for the portable device. This file format is the portable device equivalent of the EBOOT.BIN loader files, recently compromised on the PlayStation 3 game console via the acquired encryption keys. As such the potential also exists to modify the device operation at a firmware level via this method of attack, placing such a modified file in the proper directory for auto execution.

Utilization of Results

As is the case with all security research, there can be positive and negative aspects to the application of the derived results depending on its usage. Clearly determining the existence of any security vulnerability in a device which may be utilized for malicious purposes is a positive outcome and the first step in mitigating that occurrence. Secondly, the conducting of and dissemination of such research is raising the level of public awareness with regards to the need to be diligent in deploying and utilizing devices in a manner which protects individual privacy and reduces the potential for malicious use.

The potential negative aspects of vulnerability discovery, e.g. providing information necessary to exploit that vulnerability or perhaps even bringing to light new vulnerabilities is always a concern. For the discovery of what may be a serious vulnerability, delaying public disclosure pending resolution of the problem by involving or notifying manufacturers of the discovered vulnerabilities, may be an appropriate course of action. This consideration provides the opportunity for the manufacturers to possibly

produce a software patch, or issue recommendations to users in regards to device configuration which may mitigate the vulnerability found.

Potential solutions to any security related vulnerability problems need to encompass both technical and behaviour based solutions, as you cannot take the human element out of the security equation. The devices themselves do nothing without human interaction; therefore the first and most effective line of defence is always educating the user to the potential threats and ways in which they can mitigate those. This is particularly applicable for both the mobile and console games as while technical solutions will work, their effectiveness will be restricted by the amount of resources available, such as CPU, memory or wireless usage characteristics. Conceivably software vendors could produce software applications for anti-viral, firewall and anti-spy ware or the device manufacturers could modify the embedded firmware elements for firewall functionality. However, as long as these devices are considered primarily used for entertainment purposes, that scenario is not likely to occur, and as such user education comes to the forefront as the most applicable current solution. In this respect, user education through the use of some form of interactive media web content or games based material can be utilized to address aspects of the vulnerability problem, as these mediums would be the most attractive to the intended audience.

Application of Defensive Measures

Applicable solutions to vulnerabilities found in game devices can draw upon work done in both the realm of mobile devices and personal computing systems. Characteristics

related to the employment of both of these devices, will bring to bear elements that may be utilized. Many examples of defensive measures for the console game devices may be derived from previous work related to networked computer systems. Utilizing common best practices, defences related to network computing systems often involves a layered approach. This characteristic is due to the numerous components involved in a network, the respective applications on component devices and their associated functions. So while the game console may not have a firewall function, a properly implemented secure home network will have such a component, therefore we can consider that some of the vulnerabilities will be mitigated in this way. Mobile devices however operate in networking environments of varied configuration, so one cannot presume that such addition external protective measures would exist.

We are however are looking to directly address and examine the vulnerabilities that exist on the game devices. For this we will look at work that is chronological in relation to the adoption of PC like functionality in non computer devices, as such mobile phone devices. Previous work by Friedman et al. (Friedman & Hoffman, 2008) shown in the table below Fig. 21, provides a summary of defensive measures applicable to mobile devices. This is a good starting point for identifying remedial security measures, with consideration given to the limited resources the game devices have and their restrictive device configuration settings. Indeed it is these characteristics that make the game devices even more similar to the mobile devices in functionality, when one considers system resources in comparison to a standard personal computer. As such we can see what may be the applicable solutions to vulnerabilities that are found in such devices. Whether or

not it is possible or practical to implement such solutions will depend on the availability of the defensive measure and the willingness of the user to adopt that measure.

Summary of threats to mobile devices and applicable defenses

Threat Category	Description	Defenses
Malware	Worms, viruses, Trojans, bots, spyware, logic bombs and other malicious software programs.	– Personal firewalls – Anti-virus – Anti-spyware – Intrusion prevention systems – Zero-day malware protection
Phishing and social engineering	Methods of duping computer users into sending confidential information to third parties or downloading malware.	– Education – Content and spam filtering
Direct attack by hackers	Hackers identify, probe and launch exploits against a specific device.	– Personal firewalls – Vulnerability and configuration management
Data communications interception and spoofing	Interception of wireless data streams and attracting users to fraudulent hot spots and access points.	– Disabling unneeded communications – Inbuilt encryption – Password encryption – Virtual Private Networks – Restricted access to hot spots
Loss and theft of devices	Loss and theft of laptops, handhelds, cell phones, USB thumb drives, and removable storage media.	– Data encryption and time bombs – Backup and recovery – Device control
Malicious insider actions	Theft of data or deliberate creation of vulnerabilities by employees and other insiders for revenge or profit.	– Identity and access management – Data leak prevention and content management – Auditing, logs and traffic monitoring
User policy violations	Creation of security vulnerabilities by users through ignorance or indifference to corporate security policies.	– Data encryption – Device control – Data leak prevention – Education and training

Figure 21 Mobile Devices Summary of Threats and Defences (Friedman & Hoffman, 2008)

In the context of this research, the first five threat category items listed in the table are applicable to game devices, as experimentation has found them to be vulnerable the same threats, or to have the potential to fall vulnerable to such threats. Reviewing the threat categories in question, the following points can be stated or surmised related to game devices.

- Malware – Tests against the game device web browsers have shown them to be susceptible to coding that creates error conditions and operational malfunctions, one of the first steps required to introduce malicious code payload from the web onto a system. With the exposure of the encryption keys to utilized sign binaries executables, as in the case of the PlayStation 3 this makes the potential for creating such a malicious payload a viable possibility. This combination of factors increases the likelihood that Malware or Viral type malicious software may be developed.

- Phishing and Social Engineering – By implementing web browsers on the devices with which users may surf the web and utilize the systems in a personal computer like fashion, this vulnerability is a given. This type of vulnerability makes no distinction with regard to the devices original purpose as it is vector of attack is the capability of a device to access web or email based content.

- Direct Attack by Hackers – Tests have determined the devices respond to the same off the shelf tools utilized by attackers to detect and identify target computing devices, in both wired and wireless capacity. In some cases these tools have identified vulnerabilities that may be exploited such as Etherleak. With the responses of the devices being similar to standard computing systems, attackers will classify the device as a valid target. As such the risk of direct and indirect attacks exists, where these systems may fall prey to attack vectors originally designed for personal computers or mobile devices.

- Data Communications, Intercepting and Spoofing - Wireless traffic and beacons have been seen to identify the devices and in some cases the mode of operation. When acting as wireless access points, attacking computers can associated themselves with the devices upon acquisition of proper authorization coded and spoofing of game device MAC address identities. Data communications encryption formats are compromisable with the same tools utilized against personal computer wireless networks, making this a viable attack vector. The fact that the game device operates as a wireless access node behind the users firewall defence creates the potential for a backdoor into an otherwise secure network, with the distinct beacon identifier highlighting such a device.

- Loss and Theft of Devices – The portable devices may contain information in the caches of the web browsers that could lead to privacy issues. The remote play options utilized on the devices or the activation of game sharing options rely on stored wireless network access authentication information as an ease of use function. This information can be utilized to gain access to a home network or access point associated with the game device; any stored account information may likewise be exploited.

Looking at the applicable technologies for the mitigation of these threats, parameters such as network configuration decisions, firewall usage and settings, wireless encryption settings, device specific settings and individual applications settings all come into play

when implementing defensive measures. However even with the variety of similar technologies and applications utilized by these game devices, there are as yet no available applications such as those indicated in the previous table (anti-virus or anti-malware for example), nor the ability to perform some of the suggested functions (encryption, firewall, intrusion detection systems) in firmware. As such the most readily available and effective protective measures for these devices must come from the users themselves, therefore the solution of education plays the significant role in mitigating the security risks and vulnerabilities of game devices.

Education of the user could be facilitated either in the traditional form of a security application implemented on the devices or a virtual world type of educational experience retaining the appeal of a 3D game type environment. An educational security application would be an advisory type which can access an updatable database to relating to the potential vulnerabilities that the specific devices operations may expose the user to. It should have a simple interface as user input controls are constrained, its advice presented in concise and simple manner with a graphical indication of a risk factor and information on how to mitigate the vulnerability. A 3D virtual environment approach would illustrate the potential vulnerabilities in an interactive and engaging immersive context, giving shape and action to the online world and the game device. Here the user learns about the threats, and the actions that may be taken to avoid them in game playing fashion. The appeal of this approach being its commonality to the devices intended primary recreational purpose.

User Education via Application - Risk Assessment/Mitigation Tool

It is foreseeable that with access to data regarding the device type in use, combined with a previously constructed database of vulnerability information related to that device a Risk/Mitigation education application could be utilized to:

- Assess device vulnerabilities.
- Assess vulnerabilities regarding the usage of the applications available on the device.
- Assess vulnerabilities based on usage characteristics.
- Provide information regarding best practices with regard to wireless security.
- Present the information in an engaging format for easy comprehension.

User education can then be realized through the implementation of a risk assessment type application whose output is guided by a combination of user interaction (usage query) and device self assessment (lookup) providing platform enumeration information. These informational elements can be utilized to construct a query which can then access an information source (database) to provide advice regarding the vulnerability in question. The information provided would display an indication of the vulnerability/risk existence and a sense of the severity level. For example, a list of applications available on the device can be displayed with text colours indicating the existence of and severity of a vulnerability, green indicating no threats, yellow indicating low risk threats and red indicating a high risk threat. Similar assessment indicators can be provided for proposed device usage scenarios.

Mitigation strategies could then be presented; keyed to the recognized risks and derived from an existing solutions or best practices base. Information would be presented in a format keeping in mind the non technical literacy level of the user. The desired objective is to produce information relevant to the risks at hand, guided by either the users proposed usage situation or the device characteristics in such a way to make the user more aware of risks or vulnerabilities and offer suggestions to mitigate those.

Risk/Mitigation Application Design Elements

The first challenge of designing an application to educate users in the risks associated with the use of networked gaming devices is to create a simple yet engaging interface to guide the user through the experience. Gaming devices as with other personal device technologies are designed with the concepts of ease of use and limited application of technical skill. Therefore any application destined to run on them should follow the same tenets. Device input screens must to accommodate a basic interface device like a game controller and as such a scroll type menu selector (joystick) with minimal keyboard input (controller buttons) is required. Ideally a cross platform package could be developed to consolidate the information required such as the risks and mitigation databases or if a web based type application is provided. In keeping with these design constraints, a prototype application is developed using JavaMe and run on a minimalist mobile device emulator, such as the one providing the screenshots used here for illustration. Currently none of the game devices examined in this paper utilize a Java Virtual machine for applications, but the PlayStation 3 containing a Blu-ray drive does support Blu-ray Disc Java (BD-J)

The phases of a risk mitigation application can be considered as follows.

1. Device Enumeration
    a. Identify the device type.
    b. Identify the applications on the device.
    c. Identify the network environment of the device.
2. Risk Identification/Selection of Category
    a. Display risks associated with the device
    b. Display risks associated with the applications and their usage
    c. Display the risks associated with the devices role in a networked environment.
3. Display Mitigation Advice
    a. Explain how to mitigate associated with usage of device type.
    b. Explain how to mitigate risks associated with applications usage.
    c. Explain how to mitigate risks associated with the network connectivity choices.

A flowchart diagram is provided in Fig 22. Illustrating the processes steps utilized in the application.

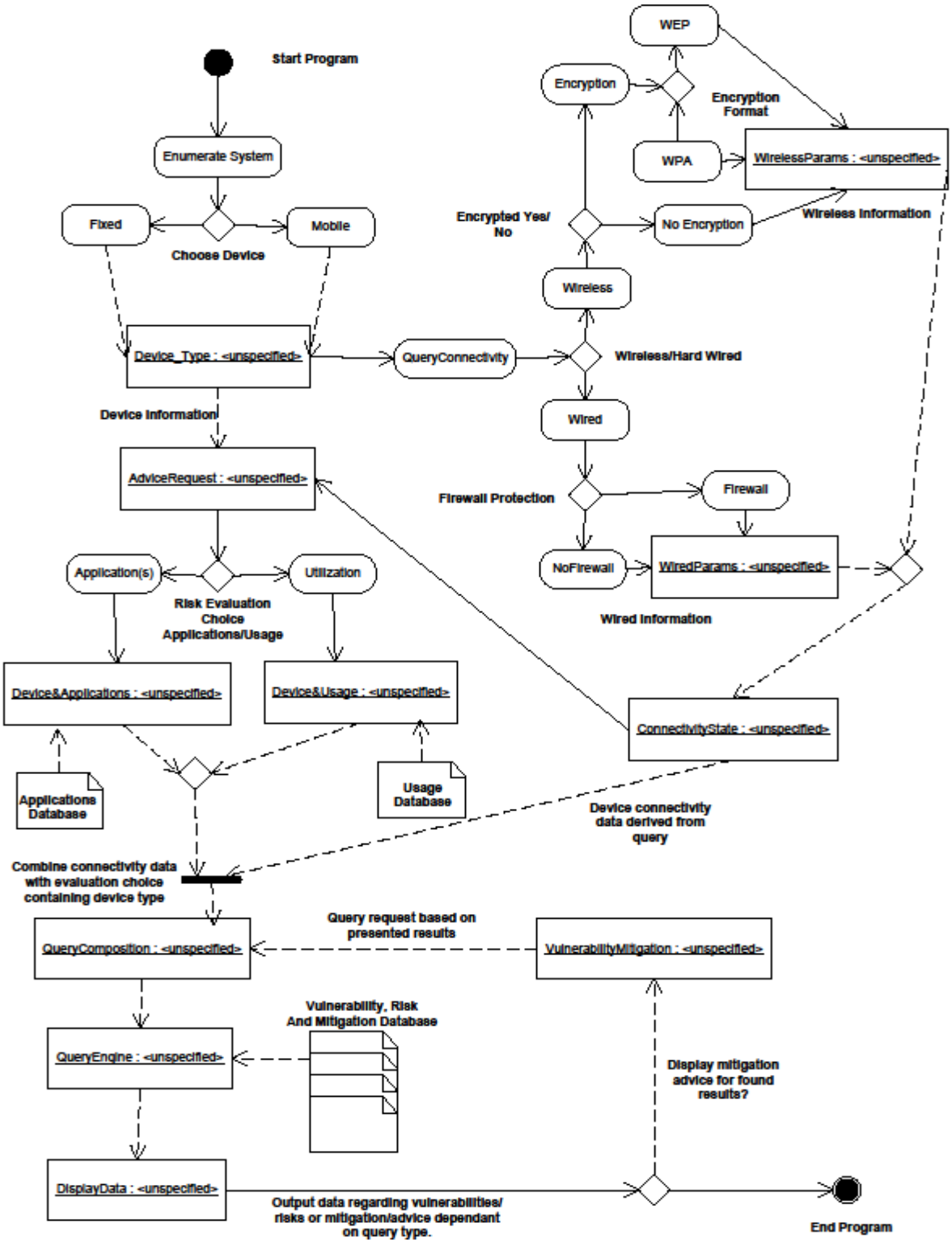


Figure 22. Risk/Mitigation Application Flowchart

Design considerations should entail attempts by the software to predetermine as much relevant information as possible about the device type as in the prototype screens shown in Fig 23. This frees the user from the necessity of providing input information, except in situations where it is a necessity and where the user may have provided such input previously. One example of this would be in the choice of wireless encryption formats, knowledge which the user would be required to supply in order to connect to a wireless network.

Figure 23. Device Type Identification and Information Verification

Where information is automatically provided by the risk application, the user would be required to verify the information, the validity of which would then determine the information utilized in the next phase as illustrated in Fig 23. Here the option to verify is provided as a menu selection. The affirmation of the device type is utilized to verify the

correctness of the pre-selection of default applications installed on the device. This approach then frees the user from having to micromanage the list of applications resident on the device to be assessed for vulnerabilities, as one would be presented as a result of the device choice verification.

Where the self enumeration process fails or additional inquiries are sought, the example screens in Fig. 24 shows potentially how the available device configuration options could be presented for review and selection. In the same way it would be possible for the user to customize the applications list if required, but ideally the selection process of displayed applications should from the user's perspective be of an exploratory nature based on presented choices derived from the self enumeration process of the software. The user initiated exploration process would also allow for the selection and review of risk and mitigation information that the user feels is relevant to them at the time, their favourite application perhaps or one that they have seen in the media recently having issues.

Figure 24. Examples of User Supplied Configuration Choice Screens

Risk/Mitigation Application Initial Design Decisions

The creation of a cross platform risk application requires some initial design decisions as to the best way to provide maximum flexibility for differing device platforms and the timeliness of advice feedback. Device and network security is a moving target, and while the ability exists to provide blanket best practices advice in most scenarios, better defence is provided in exact detailed information. Herein lies the dilemma, a pre-packaged application could have self contained default databases, containing information on a variety of devices and therefore likely to have generic best practices information along with a potential subset of device /application specific data. The concern here is that the information related to the default applications may be outdated and would not contain any information on any new installed applications or new device capabilities. The benefit of this configuration is it would not require an active network connection to be utilized and so could be utilized immediately. This approach then provides the user with some

guidance in advance of establishing external network connectivity, increasing awareness of potential threats and reducing risks. In contrast a network active application would have the ability to access current risk information on the installed base of application and possibly provide that in much greater detail. However in the process of having establishing that online connectivity, the device has then been exposed to the hostile environment it prepares to provide advice on.

With respect to the two aforementioned scenarios the first one described tasks the application in the role of gate guard, the user needing to progress through portions of its risk information selections providing a baseline level of security literacy. The application then would provide the choice to go online and access more detailed online information. This approach reinforces the concept of making the user a knowledgeable participant in regards to what applications on the device intend to do. Thus increases the awareness of using the device in a network environment, potentially decreasing risks. An implementation variation of this software in the more restrictive sense of an access control application could see it acting as an educational key, unlocking usage of other networked applications on the devices. Once a user has completed the review of the possible risks and mitigations of using a particular application, one is granted usage of the application, perhaps outright or after the completion of a small quiz on the subject. This usage may be an appropriate configuration in the case of young users or in a corporate/educational environment where there are concerns regarding user privacy, protection of data on the device or maintaining optimal functionality through best usage practices.

Depending on the portable game device type available, system resources will vary and large (respective to the host device resources) local databases may not be feasible if the storage capacity is limited. Following this reasoning, designing the initial configuration of the application as a client with a small initial database only providing generic best practices data to begin with and then allowing for access to more in depth information via network access is a prudent choice. As such the ability to expand query capabilities and provide current information related to the risks associated with the applications on the device in question and their mitigation strategies is provided as a server based request function. This design of such an application client results in a small size and would not tax mobile device resources. Additionally applying this type of approach the default application can be more generic in nature and therefore utilized on a wider variety of device types.

Where one is building an application for a targeted device platform with sufficient resources for data storage such as a console based game system, a version which encapsulates all current data locally can be developed. In this configuration the network access function is more likely to be used to update the locally held information stores. It is also conceivable that the data provided may be augmented to utilize a wider variety of multimedia based resources based on the capabilities of the host device. This type of application with the ability to provide more information as a result of access to more data can enter the realm of an educational resource which could be utilized in a broader context of course related material. Distribution of a console based application would

likely be via removable storage media such as a DVD format disk. The inclusion of Java (BD-J) in modern Blu-Ray DVD players as a support function offers some interesting possibilities here.

Risk/Mitigation Application Interface Design Characteristics

While the deployed device platform itself will ultimately determine the look and feel of the interface, the application should strive for simplicity in regards to navigation, flow and display of information. User provided choices or input requests where provided should be clear and unambiguous. Menus and option buttons should be utilized so not as to require the user to input large amounts of information as game devices do not have full keyboards. There should be indications on the screens offering some kind of guidance as to the function of the choices presented or the indicators displayed. Choices and responses should automatically trigger appropriate flow through the application, guiding the user's experience.

Animations, sounds and colors choices can be used to reflect the existence of and severity of risks found. The use of red, yellow and green indicators can be utilized to signify threat levels of high, medium and none. In this respect the status colour can be applied to the background of the text presented, an associated icon or an animation used to list the names of applications and potential uses. Associated metrics in the context of rating a threat level of vulnerability and determining indicator colour can be derived from appropriately authoritative information sources.

The prototype screens in Fig. 25 illustrate this concept, the top left screen displaying the group indicator (applications or device network usage) showing the highest threat level related to that area. The top right screen shows upon drilling down to the next level after selecting Applications, the particular application at highest risk. In the bottom left image, we see a similar itemized display upon selecting 'Device Usage'. Displaying information as such provides both quick assessment of a vulnerability category such as Applications or Device Usage, and a more detailed breakdown of the items within those.

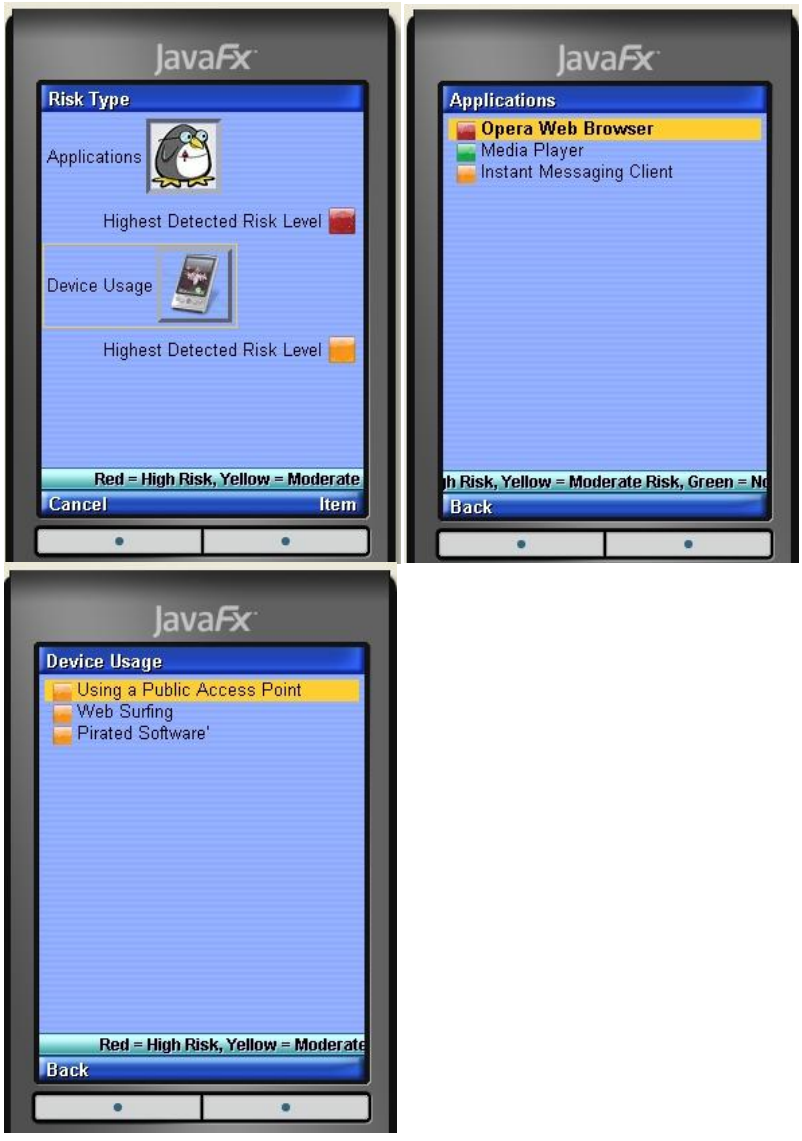


Figure 25. Vulnerability Threat Levels Indicated by Coloured Icon.

In the same context the colour of the mitigation advice text or associated graphic might be utilized to indicate the resolution level in reducing the threat when applied. This then assists the user in determining what the result of the applied advice will be in respect to the presented vulnerability. Fig. 26 illustrates the prototype code utilizing this concept through the use of indicator icons. The left screen illustrating state of the issue with the Opera Web Browser application, here advising that a new updated version of the browser will be necessary to fix the problem, but is not yet available hence the still red status. Conversely on the right advice on device usage with Public WiFi access ports provides the necessary information to mitigate the risk to a green (safe) level.

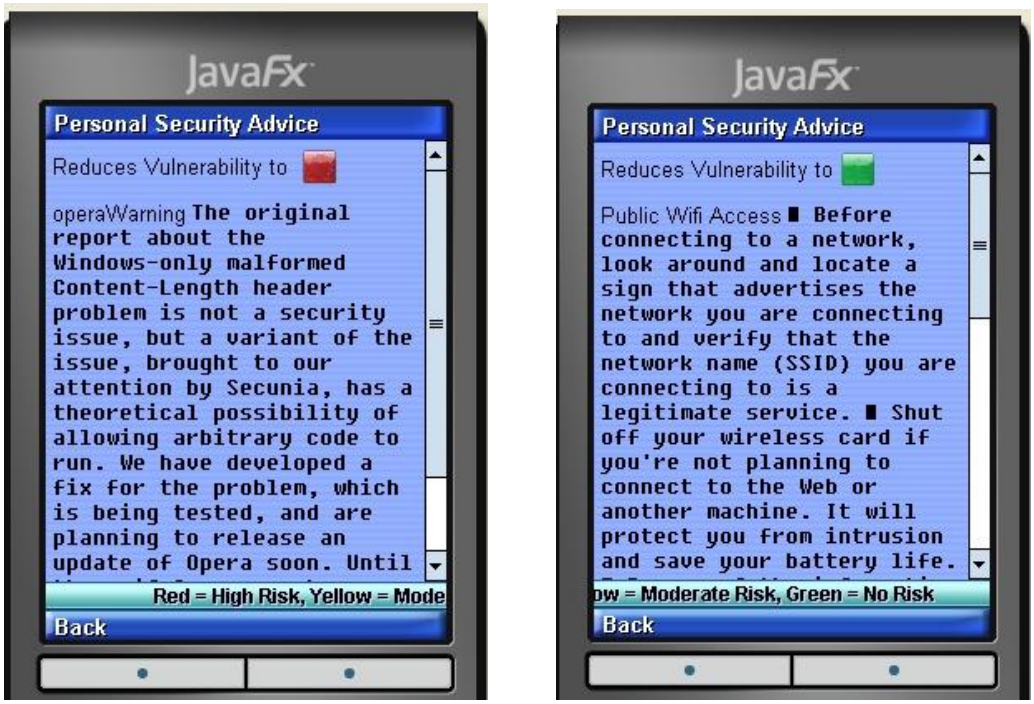


Figure 26. Vulnerability Advice Screens and Anticipated Threat Reduction Indicators.

Vulnerability resolution advice screens may also include dynamic information, such as hyperlinks to facilitate the location of updated software applications or patches. In addition to advice given by the application, dynamic links could be embedded to online sites providing a drill down to more in depth information regarding the situation at hand and best practices involved. Such an addition to the code would expand the applications ability to advise and reduce the amount of data that would need to be stored specifically for the device application in its server based or local databases.

User Education Via Immersive 3D Game Environment

As the context of game devices is social/recreational, their design and utilization requires no specific technical skills, as such users will typically not have any background in this area to assist them in making appropriate security choices. It is here that the educational aspect of vulnerability problems can be addressed through the development of an immersive three dimensional game world, containing interactive objects and information related to usage situations. Presenting this information in a game like context, would be engaging to the user of such network game devices, and would facilitate the comprehension and assist in the retention of safe computing practices.

The utilization of an immersive 3D game like world interfaced to a Learning Management System for online course content would facilitate this. A virtual 3D world helps the learner visualize and think through the scenarios that may take place in an otherwise invisible medium. The game environment is immersive and as such, engaging

to millennial and neo-millennial students. It also provides a model for how such technologies combined can be utilized by other disciplines of study that may benefit from visualization of conceptual models.

Game Elements Derived from Potential and Discovered Vulnerabilities

From the results of the research on the vulnerabilities present in these network game devices and the potential ones from the derived vulnerability matrix, concepts that a game simulation should cover are:

- Online Hacker Threats
- Wireless Security Threats
- Web Browsing Threats
- Media Downloads / Game Sharing
- Firewall or Antiviral Defences
- Application Upgrades
- Best Practices in Password Selection

The game scenario should endeavour to do the following:

- Place the user of the game device in a virtual construct of the Internet with objects related to activities they might engage in online related to the aforementioned concepts.

- Create an environment to allow exposure to these concepts through a task oriented scenario outlining the goals while allowing the user to explore freeform in achieving the outcome if possible.
- Simulate device vulnerabilities/threats involved in those activities, regarding the usage of the applications available on the device and the resultant effect through the interaction or non interaction with objects in the simulation.
- Provide informative scenarios and access to objects regarding best practices and acquisition of objects related to the mitigation of the vulnerabilities/threats and choices for the user regarding applying them.
- Utilize some form of in game performance rating or goal achievement recognition for motivation and some kind of review metric as to the concept currently being explored.
- Upon simulation completion provide some test of retention of the overall vulnerability/security concepts to assess the experience.

The challenge of designing such a game environment to educate users in the risks associated with the use of networked gaming devices is to create engaging scenarios. Users of networked game devices have available a large number of games that take place in 3D immersive worlds with storylines utilizing goals or quests create the motivation for in world exploration and activities. Likewise in a well thought out game construct/simulation related to the vulnerabilities of networked game devices, goals or tasks should provide this same sense of purpose and direction while enhancing the immersion perspective of the game. The setting in which the simulation environment

takes place should relate to the learning outcomes of interactions with in world objects, resulting in the loss or acquisition of skills and/or materials required to complete tasks and goals. All these aspects influence the game design as an there is a linkage between the objects in the game, the game environment and the educational goals.

For our purpose, the endeavour is to create a 3D game environment to help the learner catalyze the concepts of online security vulnerabilities thru exploration and actions in world, utilizing proxy objects and locations related to the use of game devices on the Internet. In this simulation, the game user's environment is relevant to the learning experience (a simulation of Cyberspace/Internet); they are gathering skills (situational awareness) and knowledge (threat/vulnerability identification and best practices) that they can apply through exploratory actions by interacting with items and locations (data caches, VPNs, web pages, media downloads, network access points) representative of concepts or non physical entities.

It is not necessary to create an entire 3D environment from scratch; an existing virtual world simulation framework can be utilized. In this category, platforms like Second Life, or alternately its Open Sourced equivalent OpenSim may be utilized. OpenSim is a multi-user virtual environment (MUVE) which can be integrated with the Moodle learning management system. This software package combination is the basis for a number of online virtual worlds it is utilized by a number of universities as an enhanced learning platform, Purdue, Stanford, The University of Auckland, Leeds and the Open University (UK) to name a few. This sandbox like environment allows for the researcher to

concentrate on content development and creating objects to be utilized in the educational experience at hand, without the need to delve into the mechanics of creating a 3D world to host them in.

In the case of computer security for example we can give form and function to the concept of Virtual Private Networks (VPN) and their protected ‘tunnel’s thru which user’s information passes safely. In the image below Fig. 27, they take the form of physical interactive objects, protective tunnels which providing he has the access password, will shield him from harmful hacker objects.

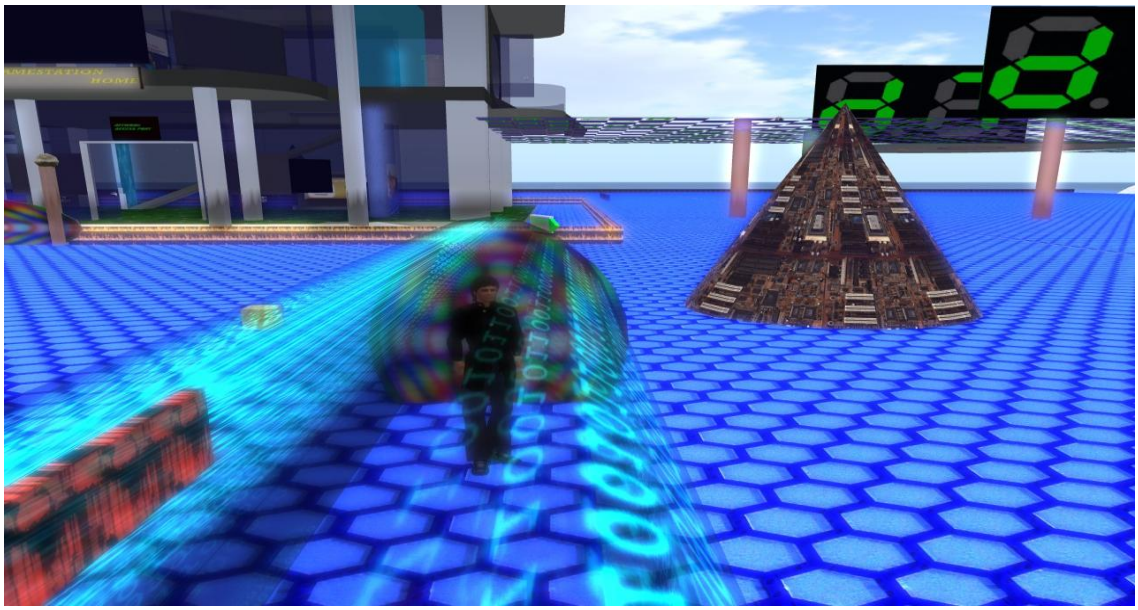


Figure 27 Users Avatar in VPN Tunnel

Instructional elements in this environment have the ability to contain many forms of content including multimedia and can link to or pull information from the Internet or external systems. Objects in world can have embedded programs or scripting languages

controlling their actions or interactions with other objects or avatars, creating dynamic virtual environments. OpenSim supports a scripting language OSSL which is similar to C# allowing for the creation of objects or agents that can interact with the user's avatar or other objects. In the context of a security game simulation these scripts would be utilized for the programming the A.I. of the Hacker objects, security related concept objects, communications with external objects (web server access) and animations. Some of these characteristics have been implemented in the ExoSpace CyberSecurity Simulation, examples and details of which are available in Appendix C.

In world game information regarding the educational experience can be provided from standalone objects, as well as information presenters linked to a Moodle hosted security course. Moodle course material is accessed via Sloodle (Simulated Linked Object Oriented Dynamic Learning Environment) module objects which integrate the LMS system and OpenSim. This combination of technologies allows for the use of Moodle course objects, normally accessed via the web, to be used in a virtual world environment. This approach then meshes the standard online course content into the virtual world environment. Students can enrol in courses; view media displays accessing course materials; take quizzes, and upload coursework to the instructor through various representative Sloodle objects in the 3D game. Using such an approach, standard courseware elements such as lecture info and tests take on an interactive and engaging format which can be integrated into the context of the environment at hand.

One example of the usage of these types of objects is the ‘quiz chair’, illustrated in Fig. 28. Here with each correct answer the chair rises upwards or conversely moves down for an incorrect answer. In our virtual world learning context of engagement, feedback and goals this chair is placed beneath an object that needs to be acquired by the student for completion of an in game task. Elevated above ground, the only way a student may reach the game object is by correctly answering a required minimum number of questions (set by the instructor, equating a passing mark) from a quiz activated when the user sits, thus raising the chair to the necessary level. Using this approach, game objects necessary to facilitate access to further information or to progress to other learning space areas (course content) are provided as rewards for demonstrated progress by testing in the learning process at hand. This technique then allows for the simulation learning experience to be evaluated via conventional means currently utilized in other online classes, it also is an indication of the possibilities of utilizing other Moodle based material in a 3D environment.

Figure 28 Game Player in Quiz Chair Elevated to Access VPN Key Provider

The OpenSim environment is also be compatible with the developing Metaverse accessible on the Internet; the Metaverse is a reference to a collection of online virtual worlds (simulations). These worlds are linked online to other systems via what is referred to as the Hypergrid, the virtual world equivalent of the World Wide Web. Rather than links to websites, each location on the Hypergrid is a computer system, housing one or more virtual world simulations accessible by the travelling the grid. This growing collective of systems increases the social aspect of using these technologies as well as the ability to share content and ideas, just as with the World Wide Web. Working on the appeal of this combination of technologies, allowing for the sharing and open access to material the opportunity exists to address current and future online security education in a manner that appeals to and hopefully will benefit to users of game devices and the online population in general.

Of the state of vulnerabilities in game devices and their exploitation, it is as with all computer security a moving target. There are areas yet to be examined and work to be done on finding adequate solutions, new devices will appear adding more possibilities to the list. It is undoubtedly not a matter of if but of when may these devices be utilized in some form of botnet, viral or malware related attack. As determined by the Sony hacking incident, one can not be assured that proprietary information and closed device systems are a defence against the exposure of data or system modification. As long as there is a curiosity to examine, explore and create with available technology there will be the purposing of that technology for better or worse, as it is the way of human nature.

REFERENCES

Works Cited

Ackerman, E. (2009, april). *Conficker worm hits hospital devices*. Retrieved from Physorg.com: http://www.physorg.com/news160331005.html

Adams, I. (2009). *Chance and Deception: Attacks That Rely On User Actions.* Research Paper, University of California, Santa Cruz, Santa Cruz.

Akritidis, P., Chin, W., Lam, V., Sidiroglou, S., & Anagnostakis, K. (2007). Proximity breeds danger: emerging threats in metro-area wireless networks. *Proceedings of the 16th USENIX Security Symposium*, (pp. 323--338).

Barker, T., Haik, E., & Bennet, S. (2008). Factors that hinder and assist learning in virtual environments: An empirical study. *Relive 08* (pp. 22-38). Milton Keynes: The Open University.

Berghel, H., & & Uecker, J. (2005). WiFi attack vectors. *Communications of the ACM* , 21-28.

Borza, M. (2011, May 5). *The Sony PlayStation 3 hack deciphered: what consumer-electronics designers can learn from the failure to protect a billion-dollar product ecosystem.* Retrieved June 13, 2011, from Electronics Design, Strategy, News: http://www.edn.com/article/518212-The_Sony_PlayStation_3_hack_deciphered_what_consumer_electronics_designers_can_learn_from_the_failure_to_protect_a_billion.php

Boulos, M. N., Hetherington, L., & Wheeler, S. (2007). Second Life: an overview of the potential of 3-D virtual worlds in medical and health education. *Health Information & Libraries Journal, 24* , 233-245.

Carnegie Mellon CERT Coordination Center. (2003). *Incident and Vulnerability Trends, 2003.* Annual Report, Carnegie Mellon.

Chen, T., & Peikari, C. (2008). Malicious Software in Mobile Devices. In Y. Zhang, J. Zheng, & M. Ma, *Handbook of Research on Mobile Security* (pp. 1-10). IGI Global.

Dagon, D., Martin, T., & Starner, T. (2004, October-December). Mobile Phones as Computing Devices:The Viruses are Coming! *IEEE - Pervasive Computing* , pp. 11-15.

Dede, C. (2005). Planning for "Neomillennial" Learning Styles:Implications for Investments in Technology and Faculty. *Educause Quarterly* , 7-12.

Fowler, D. (2005). A Supercomputer in Every Home? *netWorker , 9* (3), 5-8.

Friedman, J., & Hoffman, D. (2008). Protecting data on mobile devices: A taxonomy of security threats to mobile computing and review of applicable defenses . *Information, Knowledge, Systems Management* , 159-180.

Grove, P., & Steventon, J. (2008). Exploring community safety in a virtual community: Using Second Life to enhance structured creative learning. *Relive 08* (pp. 154-170). The Open University.

Hansman, S., & Hunt, R. (2005, February). A taxonomy of network and computer attacks. *Computers & Security , 24* (1), pp. 31-43.

Hedquist, U. (2007, November 28). *'Crackstation' Uses Game Console for Hacking.* Retrieved July 15, 2009, from PCWorld:

http://www.pcworld.com/article/140037/crackstation_uses_game_console_for_hacking.html

Holland-Minkley, A. M. (2006). Cyberattacks: a lab-based introduction to computer security. *SIGITE '06: Proceedings of the 7th conference on Information technology education*, (pp. 39-46). Minneapolis, Minnesota, USA.

Hollins, P., & Robbins, S. (2008). The Educational affordances of Multi User Virtual Environments. *Relive 08* (pp. 173-180). Milton Keynes: The Open Univesity.

Igure, V., & Williams, R. (2008). Taxonomies of attacks and vulnerabilities in computer systems. *Communications Surveys & Tutorials, IEEE , 10* (1), 6-19.

Jiwnani, K., & M, Z. (2004). Susceptibility Matrix: A New Aid to Software Auditing. *IEEE Security and Privaccy , 2* (2), 16-21.

Kemp, J., & Livingstone, D. (2007). PUTTING A SECOND LIFE "METAVERSE" SKIN ON LEARNING MANAGEMENT SYSTEMS. *Second Life Education Workshop at the Second Life Community Convention* (pp. 13-18). San Francisco: University of Paisley, UK.

Kleinhans, H., Butts, J., & S, S. (2009). Password Cracking Using Sony Playstations. In I. A. Technology, *Advances in Digital Forensics V* (pp. 215-227). Spinger Boston.

Krange, I., & Sten, L. (2008). Students' procedural and conceptual problem solving using a computer–based 3D models within the field of science education. *Relive 08* (pp. 189-199). The Open University.

Leavitt, N. (2000). Malicious Code Moves to Mobile Devices. *IEEE Computer , 33* (12), 16-19.

Leavitt, N. (2005, April). Mobile phones: the next frontier for hackers? *IEEE Computer , 38* (4), pp. 20-23.

Lee, S., & Davis, L. (2003, Jan). Learning from experience:operating systems vulnerability trends. *IT Professional , 5* (1), pp. 17-24.

Lemos, R. (2008, January). *Malware hitches a ride on digital devices*. Retrieved from SecurityFocus.com - Malware hitches a ride on digital devices: http://www.securityfocus.com/news/11499

Lipson, H. F. (2002). *Tracking and Tracing Cyber-Attacks.* Challenges and Global Policy Issues, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, Pittsburgh .

Macedonia, M. (2007). Generation 3D: Living in Virtual Worlds . *IEEE Computer* , 99-101.

Marechal, S. (2007). Advances in password cracking. *Journal in Computer Virology , 4* (1), 73-81.

Martin, M. (2007, September 5). *Xbox 360 'Folding@home' project a possibility, says Moore*. Retrieved July 12, 2009, from gameindustry.biz: http://www.gamesindustry.biz/articles/xbox-360-foldinghome-project-a-possibility-says-moore

MIT Masschusetts Institue of Technology. (2007, June). Retrieved from 6.189 Multicore Programming Primer: Programming the PLAYSTATION®3 Cell Processor.: http://groups.csail.mit.edu/cag/ps3/index.shtml

Moen, V. R. (2004). Weaknesses in the temporal key hash of WPA. *SIGMOBILE Mob. Comput. Commun. Rev* , 76-83.

Myrmo, H. (2007). *Game Consoles - Are They Secure?* Gjøvik: Gjøvik University College.

Nagle, M. (2008, February 16). *Games consoles reveal the supercomputer within* . Retrieved January 16, 2009, from New Scientist Magazine: http://www.newscientist.com/article/mg19726436.200-games-consoles-reveal-the-supercomputer-within.html

Peter, P. (2005, May 1). *Car Viruses and Other Future Computer Related Threats.* Retrieved July 14, 2009, from Security Management: http://www.securitymanagement.com/article/car-viruses-and-other-future-computer-related-threats

Piercy, M. (2004). Embedded devices next on the virus target list. *IEEE Electronics Systems and Software , 2* (6), 42-43.

Pouget, F., Dacier, M., & Pham, V. (2008). 'Understanding threats: a prerequisite to enhance survivability of computing systems. *Int. J. Critical Infrastructures , 4* (1/2), 153-171.

Provos, N., McNamee, D., Mavrommatis, P., Wang, K., & Modadugu, N. (2008, April). The ghost in the browser analysis of web-based malware. *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets* , 1-15.

Richardson, R. (2008). *CSI Computer Crime & Security.* Survey Results, Computer Security Institute, San Francisco, CA.

Shaked, Y., & Wool, A. (2005). Cracking the Bluetooth PIN. *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, (pp. 39-50). Seattle, Washington.

Shi, W., Lee, G., Hinchley, J., Corriveau, J.-P., Kapralos, B., & Hogue, A. (2010). Using a Virtual Learning Environment with Highly Interactive Elements in Second Life to Engage Millennial Students . *International Conference on e-Education, e-Business, e-Management, and e-Learning, 2010. IC4E '10.* (pp. 255-259). IEEE.

Solon, A., MJ, C., Harkin, J., & McGinnity, T. (2006). Case Study on the Bluetooth Vulnerabilities in Mobile Devices. *International Journal of Computer Science and Network Security , 6* (4), 125-130.

Su, J., Chan, K. K., Miklas, A. G., Po, K., Akhavan, A., & Saroiu, S. (2006). A preliminary investigation of worm infections. *WORM '06: Proceedings of the 4th ACM workshop on Recurring malcode,* (pp. 9-16). Newy York.

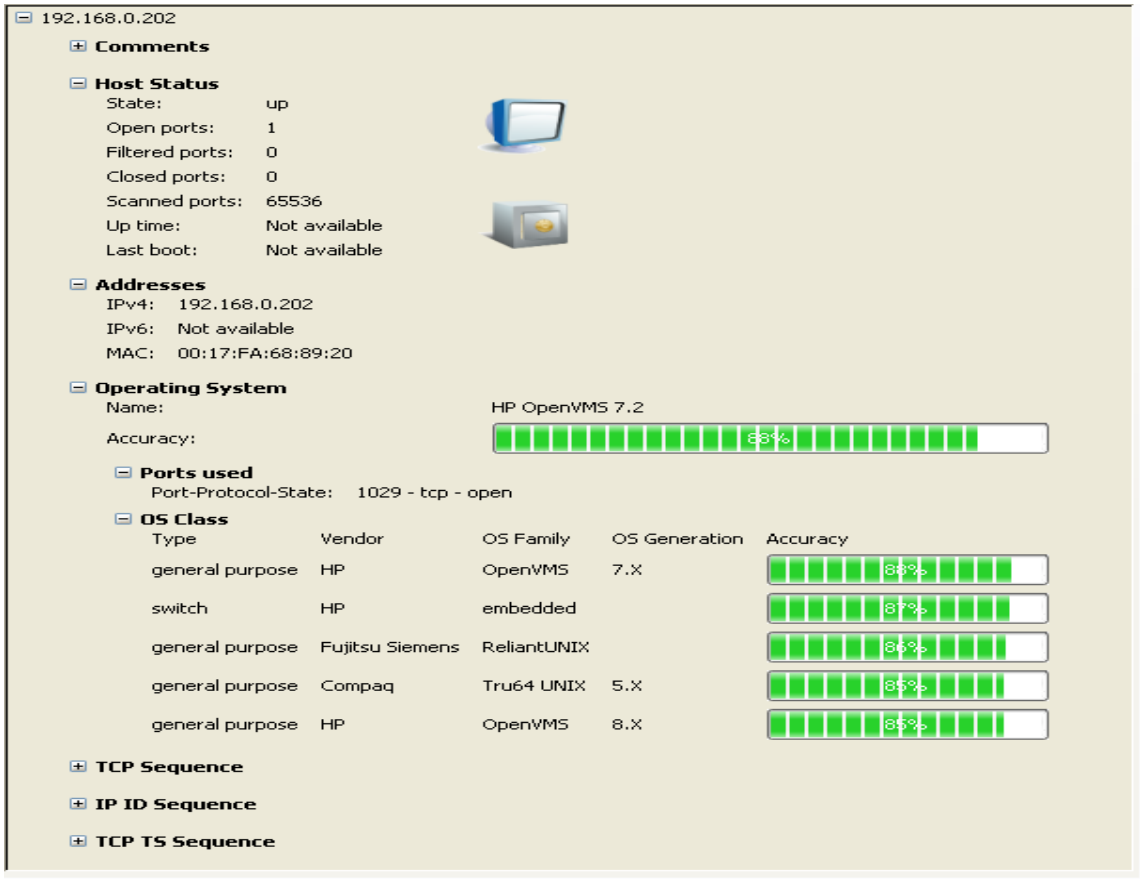
Tsow, A., Jakobsson, M., Yang, l., & Susanne, W. (2006). Warkitting: The Drive-by Subversion of Wireless Home Routers . *Journal of Digital Forensic Practice* , 179-192.

Wikipedia. (2009, July 13). *History of Video Game Consoles*. Retrieved July 13, 2009, from Wikipedia: http://en.wikipedia.org/wiki/History_of_video_game_consoles_(seventh_generation)#cite_note-Nearn-115

APPENDIX A - Host Scan Details

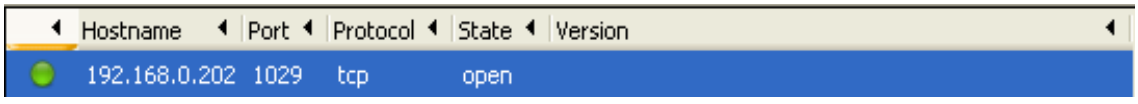
Microsoft XBOX360

Nmap Scans – Host Details



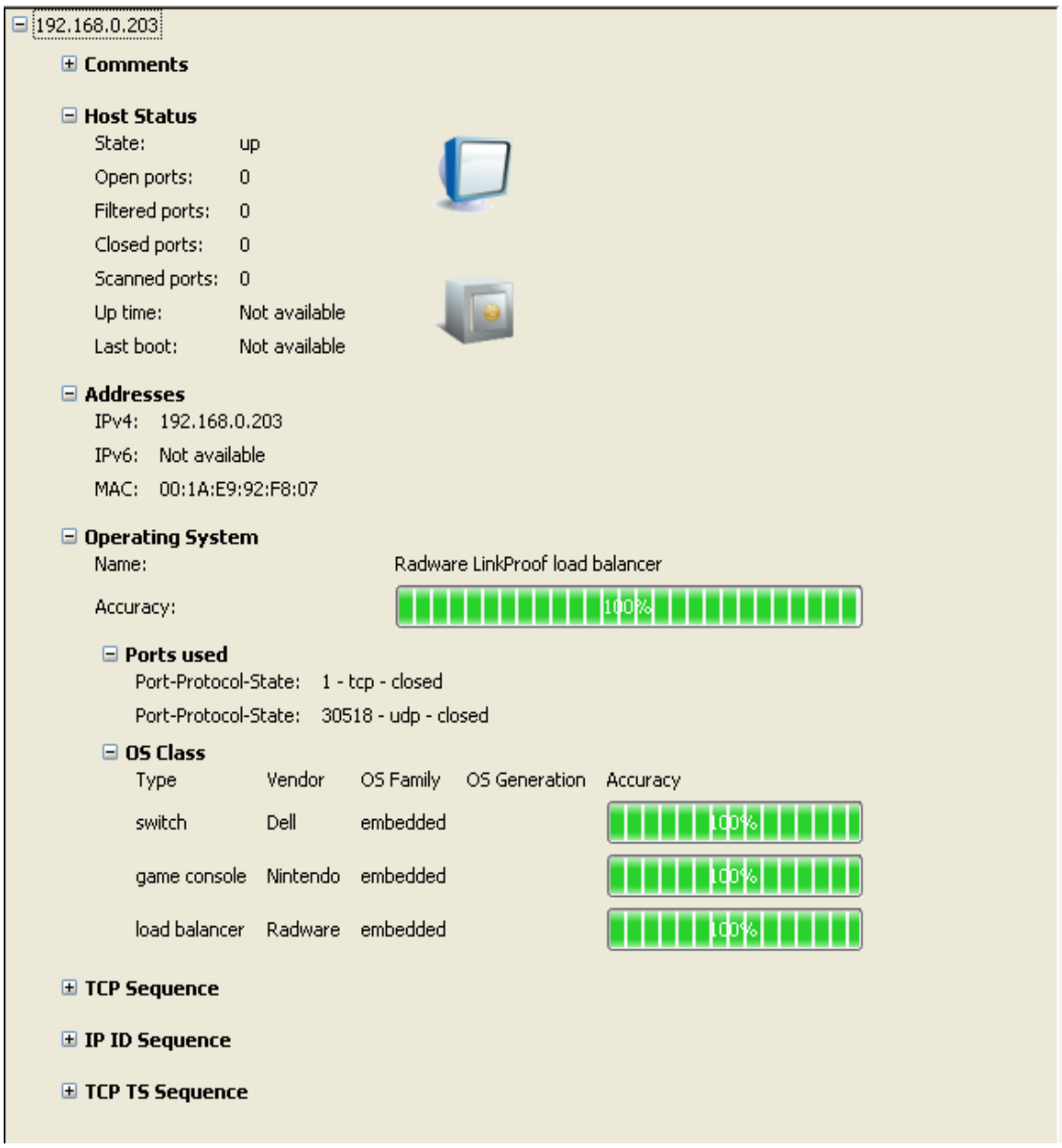
Microsoft XBOX360

Nmap Scans – Port Details



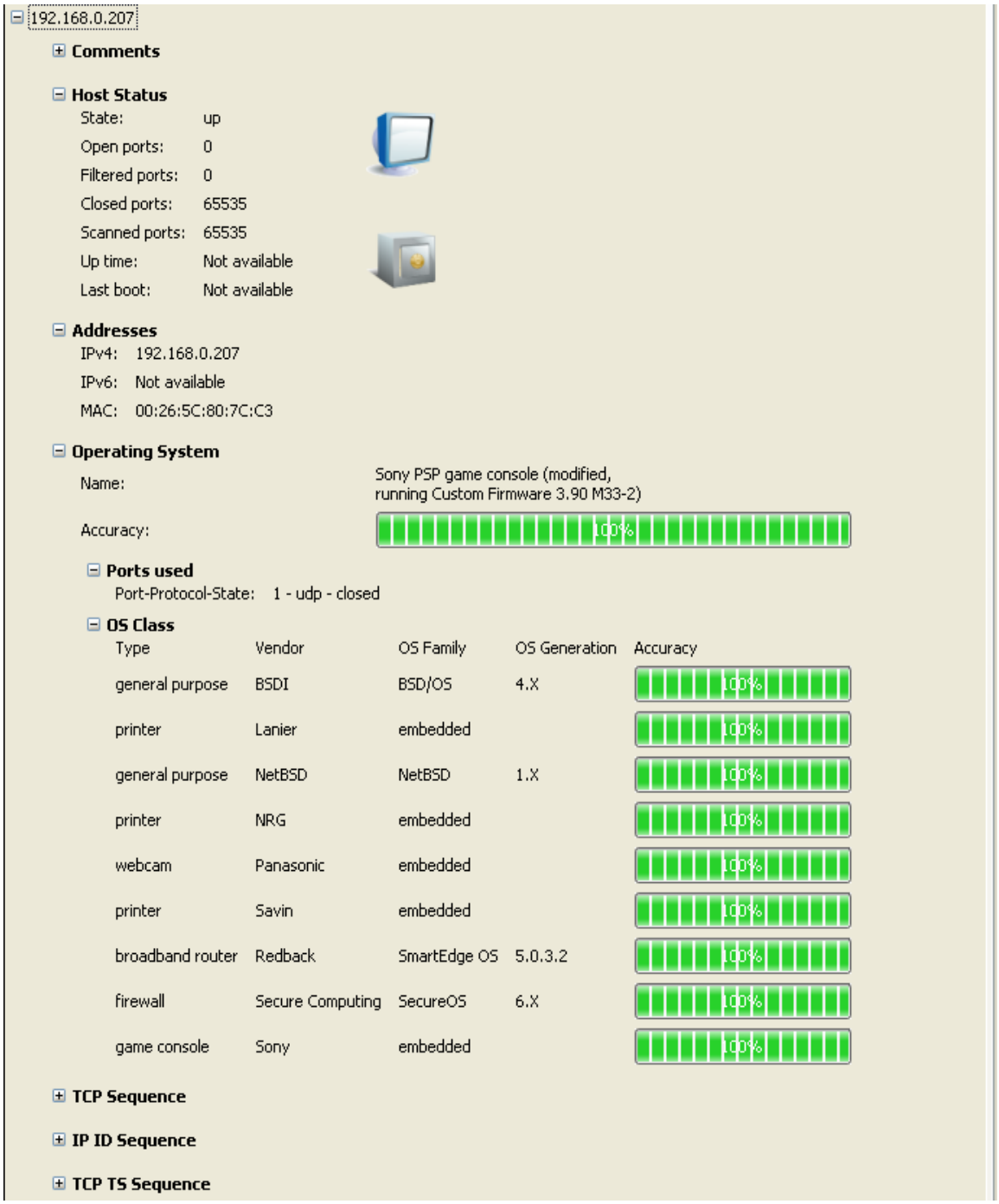
Nintendo Wii

Nmap Scans – Host Details



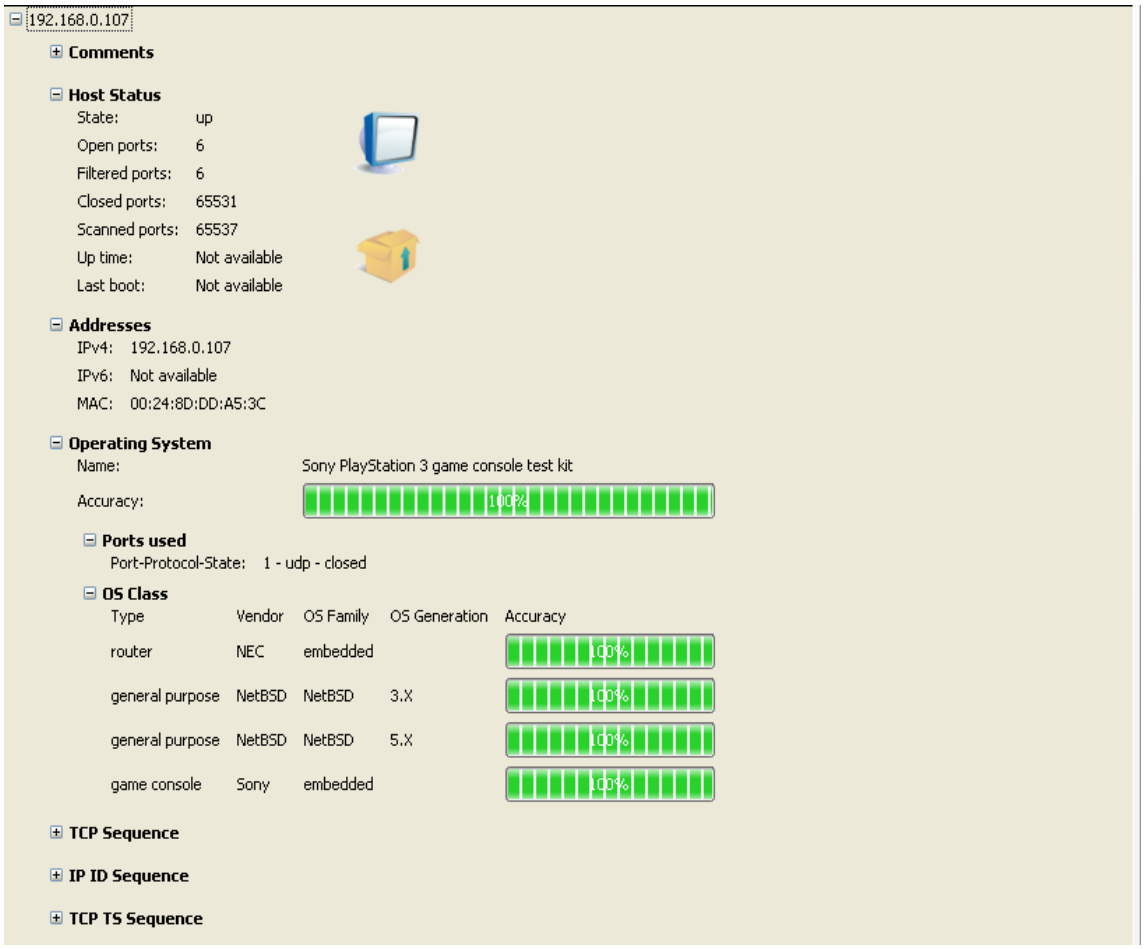
PSPGo

Nmap Scans – Host Details



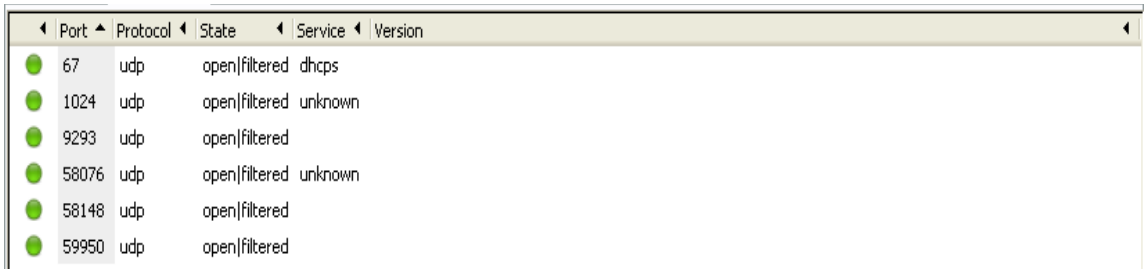
PlayStation 3 Slim

Nmap Scans – Host Details



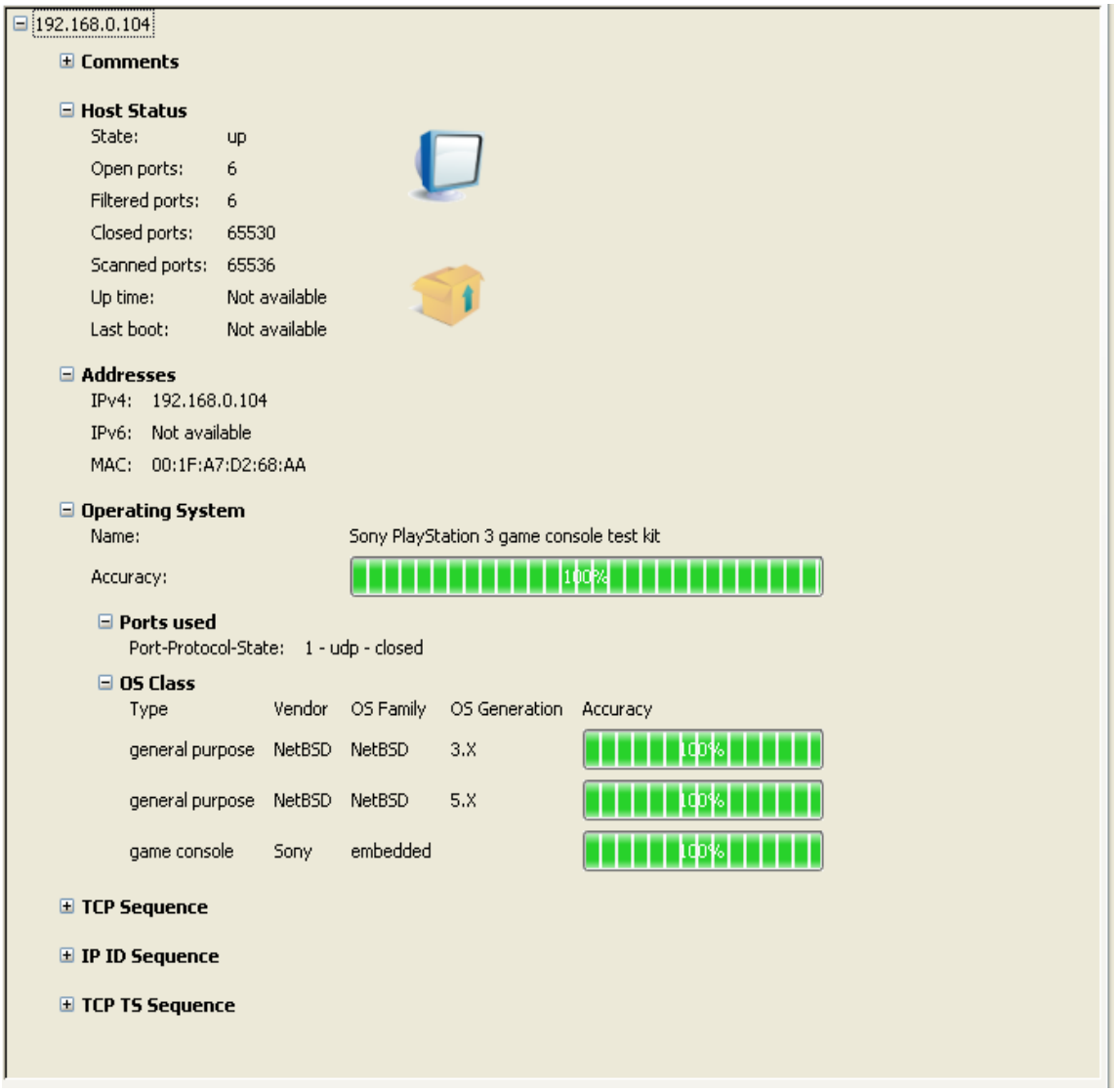
Playstation3 Slim

Nmap Scans – Port Details



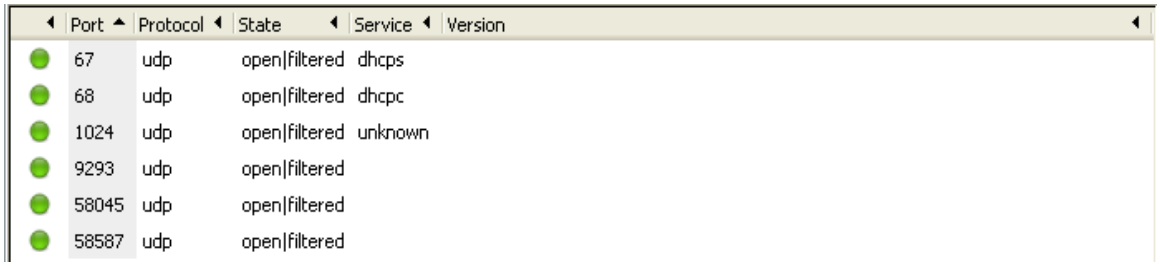
PlayStation 3 Fat

Nmap Scans – Host Details



Playstation3 Fat

Nmap Scans – Port Details



Port	Protocol	State	Service	Version
67	udp	open|filtered	dhcps	
68	udp	open|filtered	dhcpc	
1024	udp	open|filtered	unknown	
9293	udp	open|filtered		
58045	udp	open|filtered		
58587	udp	open|filtered		

Appendix B - Xbox360 Port 1026 Inquiry Responses

Web Browser Response - Root

```
<root>
–
<specVersion>
<major>1</major>
<minor>0</minor>
</specVersion>
–
<device ms:X_MS_SupportsWMDRM="true">
<deviceType>urn:schemas-upnp-org:device:MediaRenderer:1</deviceType>
<manufacturer>Microsoft Corporation</manufacturer>
<manufacturerURL>http://www.microsoft.com/</manufacturerURL>
<modelName>Xbox 360</modelName>
<modelNumber/>
<modelDescription>Xbox 360</modelDescription>
<modelURL>http://www.xbox.com/</modelURL>
<friendlyName>Xbox 360</friendlyName>
<microsoft:X_DeviceCaps>4754</microsoft:X_DeviceCaps>
<microsoft:HandshakeFlags>1</microsoft:HandshakeFlags>
<serialNumber>047635372807</serialNumber>
<UDN>uuid:04763537-2807-2000-0000-0017fa688920</UDN>
–
<serviceList>
–
<service>
<serviceType>urn:schemas-upnp-org:service:RenderingControl:1</serviceType>
<serviceId>urn:upnp-org:serviceId:RenderingControl</serviceId>
<SCPDURL>/Content/RenderingControl</SCPDURL>
<controlURL>/Control/RenderingControl</controlURL>
<eventSubURL>/Event/RenderingControl</eventSubURL>
</service>
–
<service>
<serviceType>urn:schemas-upnp-org:service:ConnectionManager:1</serviceType>
```

```
<serviceId>urn:upnp-org:serviceId:ConnectionManager</serviceId>
<SCPDURL>/Content/ConnectionManager</SCPDURL>
<controlURL>/Control/ConnectionManager</controlURL>
<eventSubURL>/Event/ConnectionManager</eventSubURL>
</service>
</serviceList>
–
<iconList>
–
<icon>
<mimetype>image/png</mimetype>
<width>48</width>
<height>48</height>
<depth>32</depth>
<url>/xbox360.png</url>
</icon>
</iconList>
</device>
</root>
```


Web Browser Response – /Content/ConnectionManager/


```
<scpd>
–
<specVersion>
<major>1</major>
<minor>0</minor>
</specVersion>
–
<actionList>
–
<action>
<name>GetCurrentConnectionIDs</name>
–
<argumentList>
–
<argument>
<name>ConnectionIDs</name>
<direction>out</direction>
<relatedStateVariable>CurrentConnectionIDs</relatedStateVariable>
</argument>
</argumentList>
</action>
–
<action>
<name>GetCurrentConnectionInfo</name>
–
<argumentList>
–
<argument>
<name>ConnectionID</name>
<direction>in</direction>
<relatedStateVariable>A_ARG_TYPE_ConnectionID</relatedStateVariable>
</argument>
```

–
<argument>
<name>RcsID</name>
<direction>out</direction>
<relatedStateVariable>A_ARG_TYPE_RcsID</relatedStateVariable>
</argument>
–
<argument>
<name>AVTransportID</name>
<direction>out</direction>
<relatedStateVariable>A_ARG_TYPE_AVTransportID</relatedStateVariable>
</argument>
–
<argument>
<name>ProtocolInfo</name>
<direction>out</direction>
<relatedStateVariable>A_ARG_TYPE_ProtocolInfo</relatedStateVariable>
</argument>
–
<argument>
<name>PeerConnectionManager</name>
<direction>out</direction>
<relatedStateVariable>A_ARG_TYPE_ConnectionManager</relatedStateVariable>
</argument>
–
<argument>
<name>PeerConnectionID</name>
<direction>out</direction>
<relatedStateVariable>A_ARG_TYPE_ConnectionID</relatedStateVariable>
</argument>
–
<argument>
<name>Direction</name>
<direction>out</direction>
<relatedStateVariable>A_ARG_TYPE_Direction</relatedStateVariable>
</argument>
–
<argument>
<name>Status</name>
<direction>out</direction>
<relatedStateVariable>A_ARG_TYPE_ConnectionStatus</relatedStateVariable>
</argument>
</argumentList>
</action>
–
<action>
<name>GetProtocolInfo</name>
–
<argumentList>
–
<argument>
<name>Source</name>
<direction>out</direction>
<relatedStateVariable>SourceProtocolInfo</relatedStateVariable>
</argument>

```
–
<argument>
<name>Sink</name>
<direction>out</direction>
<relatedStateVariable>SinkProtocolInfo</relatedStateVariable>
</argument>
</argumentList>
</action>
</actionList>
–
<serviceStateTable>
–
<stateVariable sendEvents="no">
<name>A_ARG_TYPE_ProtocolInfo</name>
<dataType>string</dataType>
</stateVariable>
–
<stateVariable sendEvents="no">
<name>A_ARG_TYPE_ConnectionStatus</name>
<dataType>string</dataType>
–
<allowedValueList>
<allowedValue>OK</allowedValue>
<allowedValue>ContentFormatMismatch</allowedValue>
<allowedValue>InsufficientBandwidth</allowedValue>
<allowedValue>UnreliableChannel</allowedValue>
<allowedValue>Unknown</allowedValue>
</allowedValueList>
</stateVariable>
–
<stateVariable sendEvents="no">
<name>A_ARG_TYPE_AVTransportID</name>
<dataType>i4</dataType>
</stateVariable>
–
<stateVariable sendEvents="no">
<name>A_ARG_TYPE_RcsID</name>
<dataType>i4</dataType>
</stateVariable>
–
<stateVariable sendEvents="no">
<name>A_ARG_TYPE_ConnectionID</name>
<dataType>i4</dataType>
</stateVariable>
–
<stateVariable sendEvents="no">
<name>A_ARG_TYPE_ConnectionManager</name>
<dataType>string</dataType>
</stateVariable>
–
<stateVariable sendEvents="yes">
<name>SourceProtocolInfo</name>
<dataType>string</dataType>
</stateVariable>
–
```

```
<stateVariable sendEvents="yes">
<name>SinkProtocolInfo</name>
<dataType>string</dataType>
</stateVariable>
–
<stateVariable sendEvents="no">
<name>A_ARG_TYPE_Direction</name>
<dataType>string</dataType>
–
<allowedValueList>
<allowedValue>Input</allowedValue>
<allowedValue>Output</allowedValue>
</allowedValueList>
</stateVariable>
–
<stateVariable sendEvents="yes">
<name>CurrentConnectionIDs</name>
<dataType>string</dataType>
</stateVariable>
</serviceStateTable>
</scpd>
```

Web Browser Response – /Content/RenderingControl/

```
<scpd>
–
<specVersion>
<major>1</major>
<minor>0</minor>
</specVersion>
–
<actionList>
–
<action>
<name>ListPresets</name>
–
<argumentList>
–
<argument>
<name>InstanceID</name>
<direction>in</direction>
<relatedStateVariable>A_ARG_TYPE_InstanceID</relatedStateVariable>
</argument>
–
<argument>
<name>CurrentPresetNameList</name>
<direction>out</direction>
<relatedStateVariable>PresetNameList</relatedStateVariable>
</argument>
</argumentList>
</action>
–
<action>
<name>SelectPreset</name>
```

```
–
<argumentList>
–
<argument>
<name>InstanceID</name>
<direction>in</direction>
<relatedStateVariable>A_ARG_TYPE_InstanceID</relatedStateVariable>
</argument>
–
<argument>
<name>PresetName</name>
<direction>in</direction>
<relatedStateVariable>A_ARG_TYPE_PresetName</relatedStateVariable>
</argument>
</argumentList>
</action>
</actionList>
–
<serviceStateTable>
–
<stateVariable sendEvents="yes">
<name>LastChange</name>
<dataType>string</dataType>
</stateVariable>
–
<stateVariable sendEvents="no">
<name>PresetNameList</name>
<dataType>string</dataType>
</stateVariable>
–
<stateVariable sendEvents="no">
<name>A_ARG_TYPE_PresetName</name>
<dataType>string</dataType>
–
<allowedValueList>
<allowedValue>FactoryDefaults</allowedValue>
<allowedValue>InstallationDefaults</allowedValue>
<allowedValue>Vendor defined</allowedValue>
</allowedValueList>
</stateVariable>
–
<stateVariable sendEvents="no">
<name>A_ARG_TYPE_InstanceID</name>
<dataType>ui4</dataType>
</stateVariable>
</serviceStateTable>
</scpd>
```

Appendix C - ExoSpace: An OpenSim/Moodle CyberSecurity Simulation

ExoSpace is a virtual world simulation in which a user explores and learns about online security threats and privacy issues. By creating a visual game like interactive environment representing the Internet or 'cyberspace', the appeal is geared towards the users of gaming systems. The home location is reference to the game device itself with the outside representing the online world. Various representations of network communications concepts, user actions and software application functions are given the form of objects to be interacted with. ExoSpace is comprised of the following component systems

- OpenSim 7.1.1 – The virtual world environment software.
- Mono 2.10 – Open Source C# and .Net support environment.
- MySQL 5.0 – The database system utilized to store objects for the OpenSim virtual world.
- Apache 2.3 – Provides web/XMLRPC services for the Moodle LMS and OpenSim
- Moodle 1.2 - Learning management system for security course content.
- Sloodle 1.1 – Provides objects for interaction between the OpenSim world and Moodle
- Wifi 7.0 – Web Based system for management of OpenSim Avatars and content.
- PHP 5.2.6 – Support language for Moodle, WiFi

The system runs under an Ubuntu based 2 GHz AMD Athlon 64 system with 2GB of Ram and 200GB of disk storage. A small course module - Security 101 is created on the Moodle system which can be accessed directly via the web as is typical or from ExoSpace. Sloodle provides the game objects that link back to this Moodle Course content. The Wi-Fi web portal is utilized to create OpenSim avatar accounts and download the required 3D client software. Here the user has the option of selecting a male, female or gender neutral avatar appearance. The administrator of the simulation can then Authorize users for access to ExoSpace from here and get information required to create Moodle accounts if required. If the user already has a Moodle account under a differing name (real name typically), there is a Sloodle registration booth object in ExoSpace which will link their ExoSpace avatar account to that account. Video of ExoSpace, its concepts and the orientation level material is viewable at http://www.youtube.com/user/bigpenguin2000

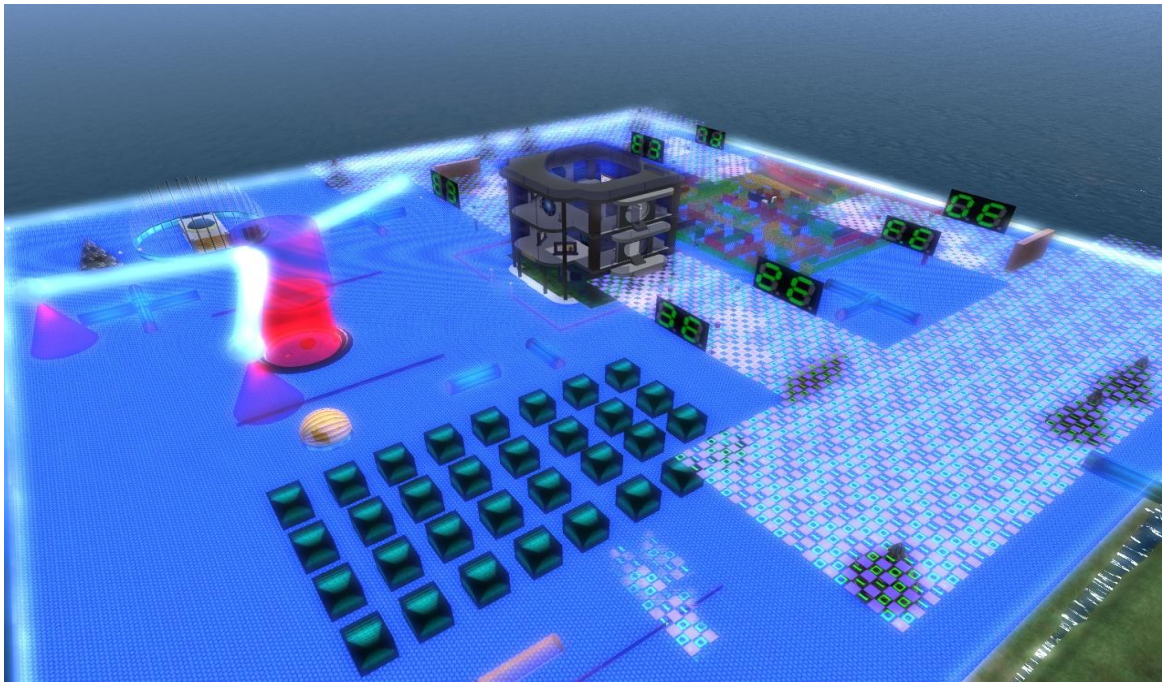


Figure 29 The ExoSpace Virtual World Area

In the ExoSpace simulation the user begins at a home location, it is here that they get the first directions as to what the initial goals are and some clue as to where the elements for goal completion are located. The home object has multiple levels, each level has a different goal/task associated with it and completion of that current level task allows for access to the next level or to the world outside. The player begins in this safe location representative of the game device console. Outside of the house is cyberspace, the equivalent of being online. The house is protected by a firewall, keeping the dangerous hacker/security objects away from the user. Objects in the house provide information (clues) or items required for the user to successfully get access to the next game area. However, limited information resources are provided in the home location, representing the limited connectivity of a standalone device and the user must venture outside of this safe environment to gain more information and gain other objects to progress. While accessing the outside world to get this useful information and additional items they must avoid the vulnerability/threat objects.

On the ground level, these threat objects - 'hackers' randomly roam the environment looking for the user's avatar as a target. Utilizing terrain objects for evasion will provide initial protection from these hacker attacks and later in game a firewall object, programmed to defend against them is obtainable as additional protection. Hacker objects compromise three types of vulnerability threats online computing device users, including game owners may encounter:

- **Scanners** (port probing) these objects look for targets and broadcast their locations to other objects. This is analogous to the hackers who scan portions o the internet looking for target systems and then share that information with other hackers.
- **Infectors** (malware or Trojan viruses) These are hackers who try attach objects to you that slow you down and annoy or cause you to reboot you (get sent back to the home location). Some successful Infectors can also steal your password token, allowing hackers to compromise your safe VPN passage protection. This attack is analogous to the harmful web sites that a user may come in contact with online.
- **Attackers** (Denial of Service Attacks) These hackers will bounce you off of the grid (knocking you offline) possibly damaging you, if the damage level reaches 100% you will be sent back to the starting point. Analogous to a directed hacker attack against the users system.

The Malware and Attacker hacker agents listen actively for any broadcast from any Scanners, should they be within reception range. Scanners sweep a 360 degree ten meter circle looking for avatars as they move about the 256 x 256 meter virtual world. If they sense an avatar they broadcast the name and coordinates of the target they have found. Hackers who hear this information then proceed to move to that location and attempt to find the target with their scanner functions. The limited broadcast distance of the scanners and the necessity for the others hacker agents to hear adds a randomness factor to the attack pattern, much like the necessity of being on a real-time ICQ chat channel to derive such information.

By giving form to these types of threats in a virtual world, we help the learner understand the concepts better and provide an interactive experience. Like the Hackers in ExoSpace, we can create other in world objects which may be utilized to convey other security and privacy related issues. Let us consider a scenario related to web sites. In ExoSpace the user's tasks require them to gain protective knowledge and objects to mitigate vulnerabilities, in the case of the hackers it is a firewall object. To accomplish this goal they need to get more information on security practices from additional sources. Information in the ExoSpace simulation is provided outside the house is from areas known as InfoDomes, which can be considered to correspond to websites. Access to the domes requires the acquisition of the correct keypad code allowing an avatar to pass through a barrier set up to keep threat objects out. While this conveys they idea that these information nodes are safe areas, however as in real life, there is no guarantee that they have not been tampered with.

If we wish to provide a scenario on the value of Antivirus/malware detection software related to websites we can do the following. We change the spelling of the scrolling InfoDome sign to read Inf0D0me, or modify some other physical attribute. These types of modified objects can represent 'phishing' websites, looking like the original, but modified in some way to trick the user. To counteract this, previously encountered InfoNode lesson areas would go over the elements of detecting fraudulent websites. Additionally for this scenario we would create an Antivirus object that the user may obtain that would warn of such tampered objects. Similar approaches could be utilized

for other objects such as media objects or software downloads for example. We can further enhance this type of scenario recognizing the fact that vulnerabilities and risks are not static entities, and as such the defences against them need be constantly updated. For this it would be required that the game player update their firewall (representative of the devices firmware/OS) or Antivirus object. Lest they run across an object threat that their current version of defensive object cannot detect.

ExoSpace Game Play Through

Orientation Level – Second Floor GameStation Home

Starting on the second floor of the home, orientation objects and videos are provided to help the user master moving and interacting with the game environment. This safe area will let the user move around and get used to the client software used in conjunction with the game. Here access to the next game area, the first floor will be granted based on a requirement of possessing two items, reflective of 'two key' security practices which is 'what you know' and "what you have'. To get to the first floor it will be necessary to teleport from the second floor, to do so however you need a Teleporter Key object as it is locked with a force field. The required key object is obtained by providing the correct answers to knowledge quiz based on the concepts related the orientation lessons. Additionally the user is provided with the house firewall keypad code, necessary to pass through the firewall perimeter.

Testing functions in the simulation are provided by the Sloodle quiz chairs based on content for the subject topics which is provided via Moodle or via in world game objects. This form of access granted by task/object segmentation provides for modular approach to covering the concepts at hand while providing a checkpoint to assess knowledge gained. The 3D construct for the quiz chair function is a large transparent sphere with a glowing sphere in the center. This is located above the Quiz Chair and in game we make use of the fact that when an avatar sits in the chair, it becomes phantom. The phantom property of the chair/avatar combination removes it from the OpenSim physics constraint of gravity and makes it easy to levitate and move accordingly (no need for an engine to provide a lift vector). We can take advantage of this function in that it also means the chair and avatar will pass though solid objects in world, hence our sphere. Upon the completion of the quiz when the user stands up, the avatar becomes physical again and is within the sphere, allowing access to the key objects. The user then exits via a teleport cube returning to the floor area.

Threat and Defence Level – First Floor GameStation Home

First floor is based on the learning about the protective nature of VPNs and the dangers posed by hackers as related to the game player/user and the simulation world. The user gains knowledge of Virtual Private Networks (VPNs) and how utilizing those will help to get around in the cyberspace landscape safely. In game, VPNs are

represented by protective pipe objects and accessed with a ‘token’ object that the user’s avatar must have in their possession. Again collecting this object is thru the passing of a quiz, which also provides the user with the stairway keypad access codes so they may return to the second floor level. On this level the user will also find information that will point them to the next location in the level quest of obtaining a firewall object. The firewall object is required as the user needs to get a key code to open the barrier to the top game level; however the system that provides access to this in a hostile hacker infected zone. Ultimately to access the firewall location, the user must to go into cyberspace (online), outside of the protective house and venture to other learning spaces completing tasks related to that areas security lesson.

This area is also where the player first encounters non Moodle information objects in the game world. This illustrates the ability to create game items which contain useful education information about online security, helping the player with the tasks at hand. An example of this on the first floor are the information drawers, which when touched by the players avatar, provide note card item which can be stored in the players inventory for later access. These contain information on the drawers listed topic, as well as opening up a web page connection in the client application connecting to a external sites containing related information.

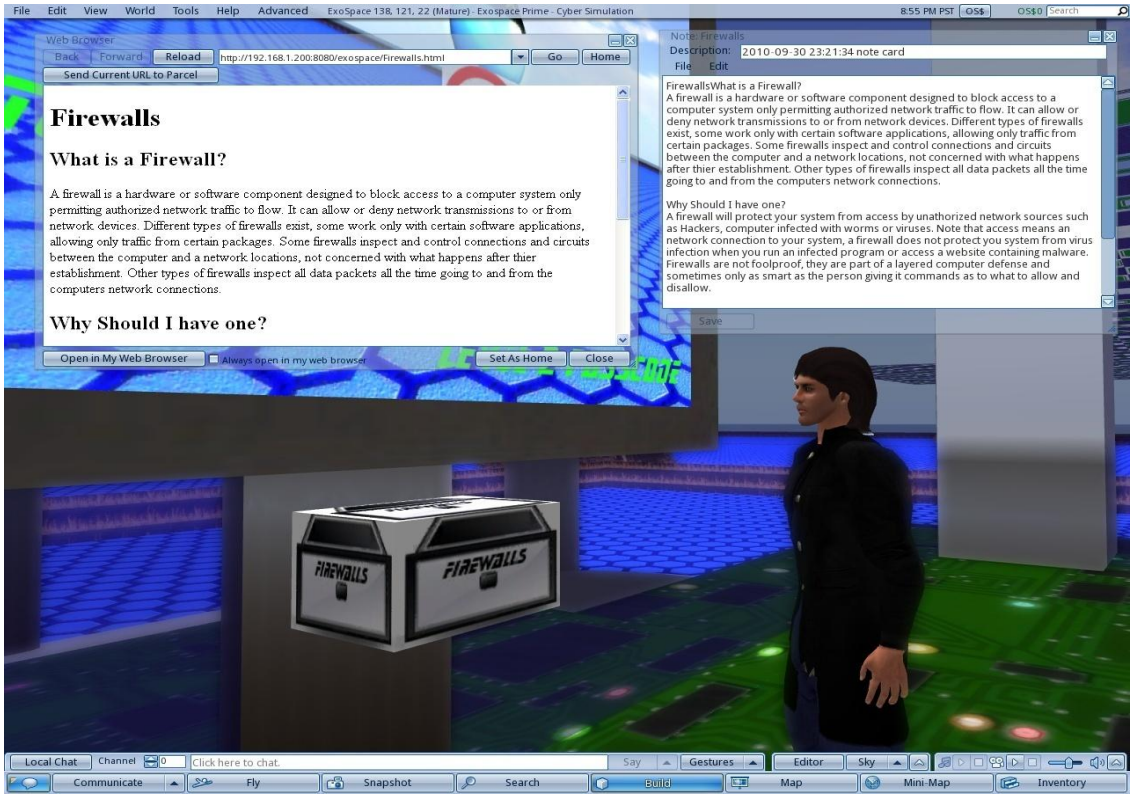


Figure 30 Info Drawer Object, Web Page Top Left and Notecard Top Right

Data Blocks and Web Caches - Data Block Buffer Area

Leaving the GameStation home area the fist stop for the user is the data block zone InfoNode, here the lesson is the danger of remnant information in web browser and network data caches. Web browsers store site information and in some cases usernames and passwords in cached information. With the possibility of theft portable game devices may yield such information to an attacker. Data in a game consoles web browser cache may also be in jeopardy if the owner sells such a device before clearing the cache, or if the device is compromised. Cached data is the form of large blocks containing various pieces of information which can be revealed by the player touching them. The user needs

to enter the data block buffer area and search the data blocks for the password maze door key, left behind by the previous user. When the correct data block is found (there are several which will work) a Maze Door access token is provided and instructions on its usage.

Choosing a Strong Password - Password Maze

Other vulnerabilities covered in game may not have a defence technology against per-se, but rather rely on the correct choice to a knowledge based query to mitigate their risk. Selecting a strong password to protect information is one such choice. Many factors contribute to the possibility of creating a strong password though, and to many online users this variety of options can be confusing. The complexity of the choices required in making a good password decisions is represented in world by a maze, illustrating the sometimes confusing directions one might follow. The user once more visits the InfoDome and reads about the factors in selecting strong password, when completed having acquired the door key the previous area they may enter the maze. While wondering through the maze users will encounter posters with questions related to the choice of a strong password. Correct answers to these questions correspond to TRUE and FALSE arrows on the posters indicating the direction in which to proceed. The goal here is to reach the center of the maze which contains the object holding the access code for the Firewall area. Unlike the GameStation Home area spheres, there is no quiz function here required for getting the key code, as the correct knowledge of the answers led them here. The user will find vending machines here though, which will dispense clothing items to customize their avatars as a reward. This taps into the social aspect of virtual

worlds and the desire for users of them to create a unique identity for their avatars, as such adding an engagement factor along with the reward factor. Armed with the key code, the user proceeds to the Firewall area, the Security Dome.

Another application of this kind lesson representation of this would be the use of unencrypted wireless access points outside the home area. In a scenario with unencrypted access points, we would presume that without a wireless connection object, one cannot access a resource object, let's say a web page object. The user must traverse the level and access ten web page objects. These web page objects represent various functions one would use the Internet for and their associated characteristics. On the game level there are multiple secure or insecure wireless access objects available and web page objects. A distance limitation regulates the selection of access point objects in relation to a specific web page object. Depending on the web resource object location and the available wireless access point objects, the question for the player then becomes is it safe to access that type of web based resource object from this particular location, i.e. should I use a website requiring a password from an unsecured wireless access point? Scenarios like these will require the users to make an informed choice based on the task at hand, the resource object in question and the available wireless connection.

Firewall Acquisition – Security Dome

With access to the Security Dome via the correct key code, the user now has the ability to acquire a firewall for protection while in the central core. The Firewall object is worn

as an attachment to the client display, allowing for it to be the toggled on and off. For protection the Firewall Blocks attacks from the hacker agents, prjecting a shield and provides the ability to 'hide' from other hackers on the first level. This will let the user pass safely into the central core zone mostly unscathed. In the central core where they will find an access portal, which when touched will teleport them to the next game play level. Currently in the ExoSpace simulation this takes the player to floor three of the GameStation Home where a HyperGrid Portal is setup. On this floor of the house we find some Moodle survey screens which are used to provide feedback related to the players experience.

.

APPENDIX D – ExoSpace Objects and associated OSSL Code

Teleporter Door Control.lsl

```
//Teleporter Door Control Script to work with Key
//Walter Ridgewell - 2010
//Based on
//Code example from SecondLife LSL Wiki
// http://wiki.secondlife.com/wiki/Key_Pad_Door
// Copyright © 2007-2009 Linden Research, Inc.
// Licensed under the Creative Commons Attribution-Share Alike 3.0 License
// http://creativecommons.org/licenses/by-sa/3.0/
//
integer channel = 25; // Channel to listen on
float open_time = 10; // Time to keep door open

default
{

   state_entry()
{
   llListen(channel,"Teleporter Access Object",NULL_KEY,"Open");// Listen for Teleporter Object
key "Open" command
   }

   sensor(integer total_number)// Sensor state called when sensor detects Avatar
   {
    llSetStatus(STATUS_PHANTOM,TRUE);// Make it possible to pass through door
      llSetAlpha(0.1,ALL_SIDES);// Make door transparent
      llSleep(open_time);//Keep in this state for allotted time
      llSetAlpha(1.0,ALL_SIDES);//Make door look solid again
      llSetStatus(STATUS_PHANTOM,FALSE);////Door is solid and avatars can't pass
   }


   listen(integer chan, string name, key id, string msg)// Listen state for communcations
{
   llSensor("",NULL_KEY,AGENT,2,PI);// Activate with any agent withing 2 meters -360 degrees

   }
}
```

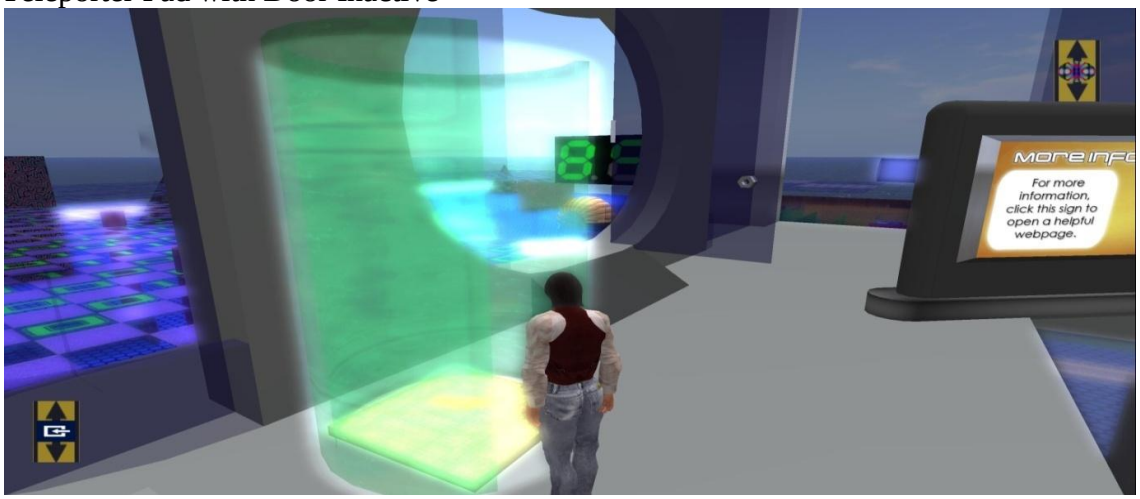
Teleporter Door Key.lsl

```
//Teleporter Door Control Key Script
//Walter Ridgewell - 2010
//Free to copy and modify retaining this notice
//
integer CHANNEL = 25;// Channel to listen on
```

```
default
{
    touch_start(integer total_count)// On touch perform
    {
        llSay(CHANNEL, "Open");// Send 'Open" command to door
        llSay(0,"Teleport Activated");//Tell Avatar command sent
    }
}
```

Teleporter Key Object

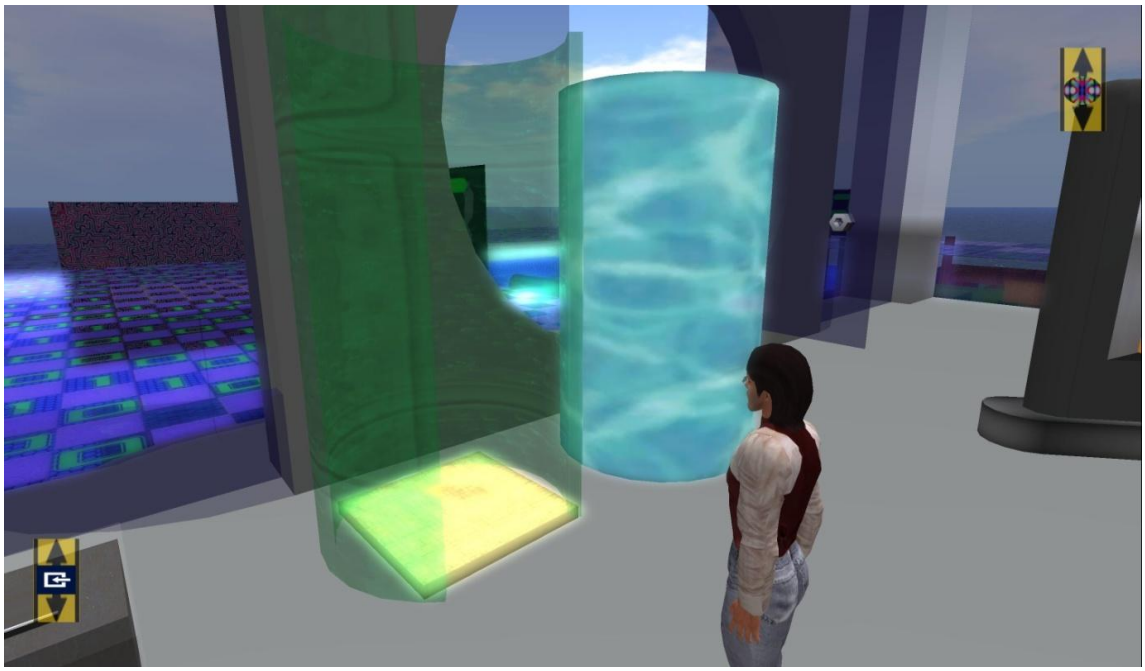
Teleporter Pad with Door Inactive



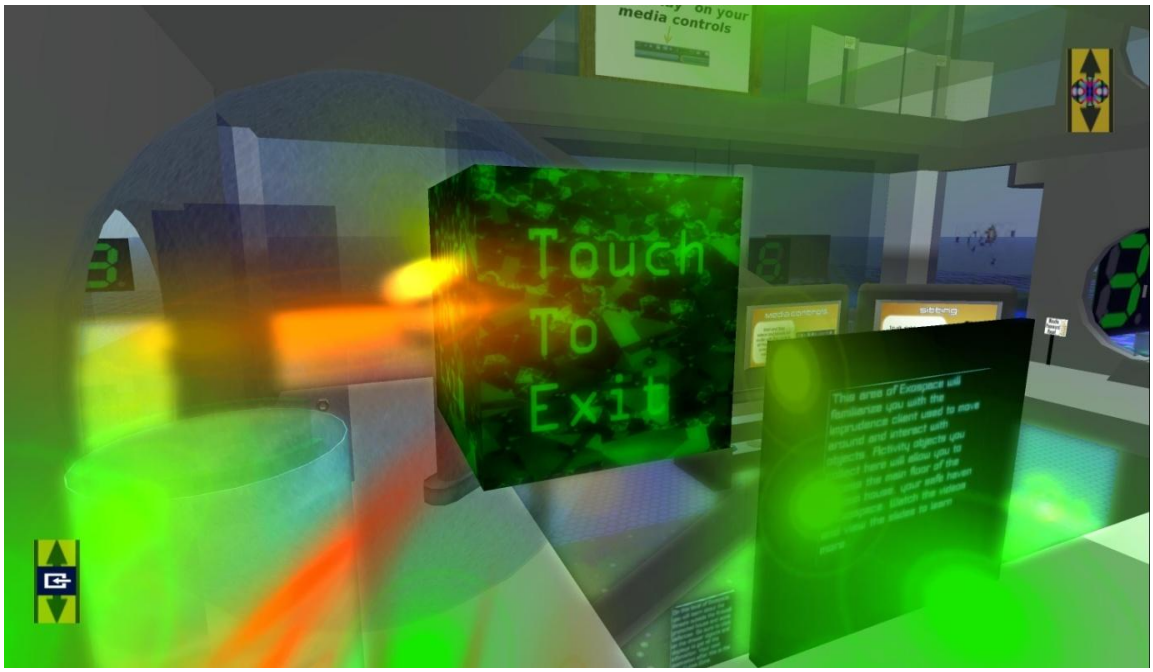
Teleporter Pad with Door Active

Teleporter.lsl

```
// From the book:
//
// Scripting Recipes for Second Life
// by Jeff Heaton (Encog Dod in SL)
// ISBN: 160439000X
// Copyright 2007 by Heaton Research, Inc.
//
// This script may be freely copied and modified so long as this header
// remains unmodified.
//
// For more information about this book visit the following web site:
//
// http://www.heatonresearch.com/articles/series/22/

vector target=<129, 119, 22>;// Exospace Floor 1 Gamstation Home
vector offset;

default
{
   moving_end()
   {
      offset = (target- llGetPos()) * (ZERO_ROTATION / llGetRot());
      llSitTarget(offset, ZERO_ROTATION);
   }

   state_entry()
   {
      llSetText("Teleport pad",<5,0,0>,1.0);
      offset = (target- llGetPos()) * (ZERO_ROTATION / llGetRot());
      llSetSitText("Teleport");
      llSitTarget(offset, ZERO_ROTATION);
   }

   changed(integer change)
   {
      if (change & CHANGED_LINK)
      {
         llSleep(0.5);
         if (llAvatarOnSitTarget() != NULL_KEY)
         {
            llUnSit(llAvatarOnSitTarget());
         }
      }
   }

   touch_start(integer i)
   {
      llSay(0, "Please right-click and select Teleport");
   }
}
```



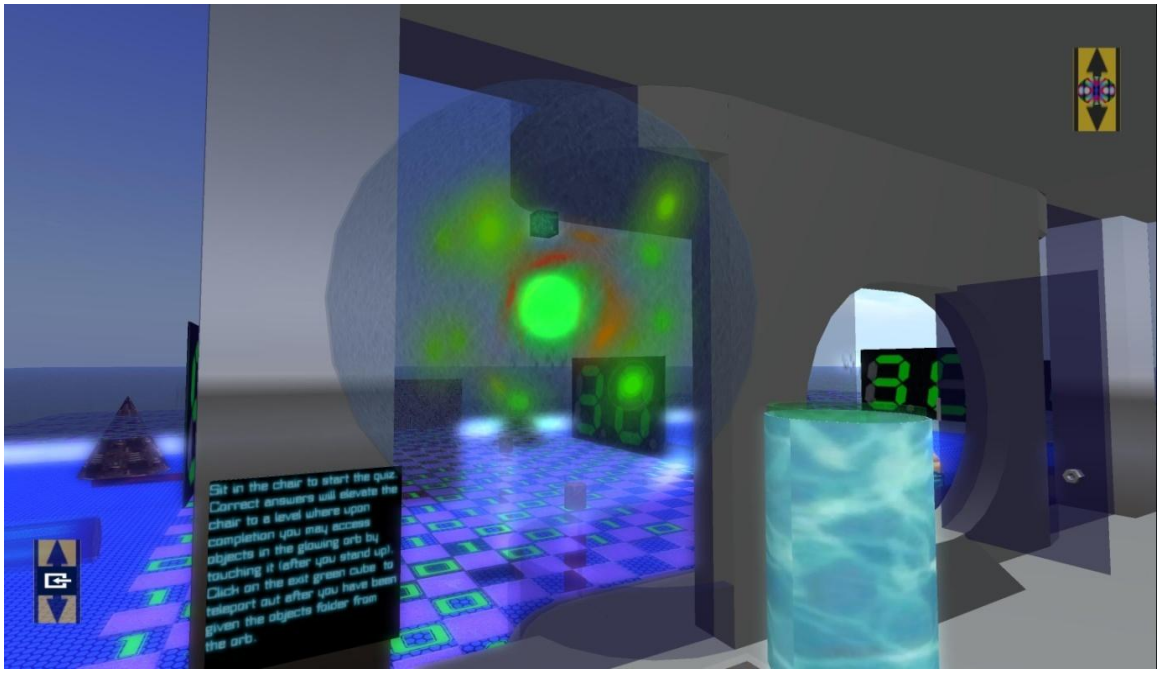
Teleport Pad, Shield Door Moved Aside



Teleport Exit Cube

Give Inventory.lsl

```
//Based on Code example from SecondLife LSL Wiki
// http://wiki.secondlife.com/wiki/LlGiveInventoryList
// Copyright © 2007-2009 Linden Research, Inc.
// Licensed under the Creative Commons Attribution-Share Alike 3.0 License
// http://creativecommons.org/licenses/by-sa/3.0/
//
// When any user clicks this object, this script will give a folder containing everything in the objects
inventory
//

default
{
    touch_start(integer total_number)
    {
        list       inventory;
        integer     num = llGetInventoryNumber(INVENTORY_ALL);
        string      script = llGetScriptName();
        integer     i = 0;

        for (; i < num; ++i) {
            string name = llGetInventoryName(INVENTORY_ALL, i);
            //Don't give them the selling script.
            if(name != script)
            {
                if(llGetInventoryPermMask(name, MASK_OWNER) & PERM_COPY)
                {
                    inventory += name;
                }
                else
                {
                    llSay(0, "Don't have permissions to give you \""+name+"\".");
                }
            }
        }

        if (llGetListLength(inventory) < 1)
        {
            llSay(0, "No items to offer.");
        }
        else
        {
            // give folder to agent, use name of object as name of folder we are giving
            llGiveInventoryList(llDetectedKey(0), llGetObjectName(), inventory);
        }
    }
}
```



QuizSphere Inventory Giving Object

KeyPad Button.lsl

```
//Based on Code example from SecondLife LSL Wiki
// http://wiki.secondlife.com/wiki/Key_Pad_Door
// Copyright © 2007-2009 Linden Research, Inc.
// Licensed under the Creative Commons Attribution-Share Alike 3.0 License
// http://creativecommons.org/licenses/by-sa/3.0/
//

//Button Script
default
{
   touch_start(integer total_number)
   {
      llMessageLinked(LINK_ROOT,1,llGetObjectName(),NULL_KEY);
   }
}
```

KeyPad.lsl

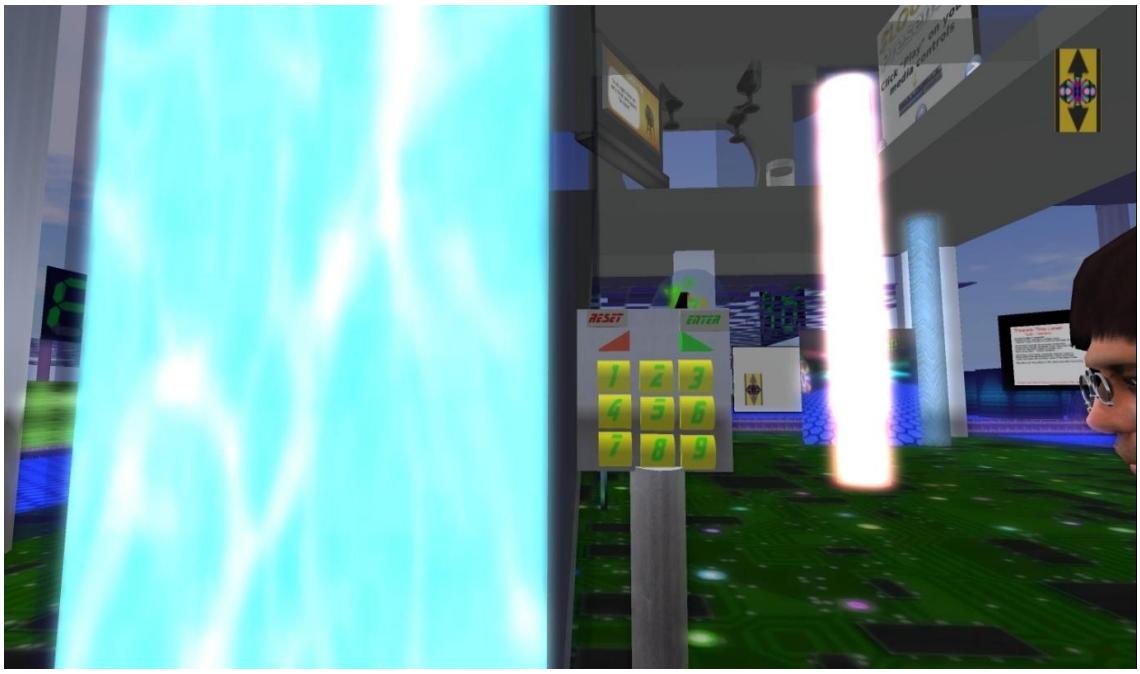
```
//Based on Code example from SecondLife LSL Wiki
// http://wiki.secondlife.com/wiki/Key_Pad_Door
// Copyright © 2007-2009 Linden Research, Inc.
// Licensed under the Creative Commons Attribution-Share Alike 3.0 License
// http://creativecommons.org/licenses/by-sa/3.0/
//

integer channel = 3801; //If you change this remember to change it in the door script as well
list temp;
list code = [1,2,3,4];
integer handle;
```

```
default
{
    on_rez(integer param)
    {
        handle = llListen(0,"",llGetOwner(),"");
        llOwnerSay("What do you wish the code to be?");
        llSetTimerEvent(15);
    }
    link_message(integer send_num, integer num, string msg, key id)
    {
        if(msg == "Reset")
        {
            llWhisper(0,"Reset Pressed");
            temp = [];
        }
        if(msg == "Enter")
        {
            if(temp != code)
            {
                llWhisper(0,"Incorrect Code!!");
                temp = [];
            }
            if(llList2CSV(temp) == llList2CSV(code))
            {
                llWhisper(channel,"open");
                llWhisper(0,"You may enter.");
                temp = [];
            }
        }
        if(msg != "Reset" && msg != "Enter")
        {
            temp += [(integer)msg];
            if(llGetListLength(temp) > llGetListLength(code))
            {
                temp = [];
            }
        }
    }
    timer()
    {
        llOwnerSay("You have run out of time. Setting default " + llList2CSV(code));
        llListenRemove(handle);
        llSetTimerEvent(0);
    }
    listen(integer chan, string name, key id, string msg)
    {
        llListenRemove(handle);
        code = llCSV2List(msg);
        llOwnerSay("Code set to " + llList2CSV(code) + ".");
        llSetTimerEvent(0);
    }
}
```

Door Control.lsl

```
//Based on Code example from SecondLife LSL Wiki
// http://wiki.secondlife.com/wiki/Key_Pad_Door
```

```
// Copyright © 2007-2009 Linden Research, Inc.
// Licensed under the Creative Commons Attribution-Share Alike 3.0 License
// http://creativecommons.org/licenses/by-sa/3.0/
//

integer channel = 3801; //Must be the same as in the key pad script
float open_time = 30; //Change to how long you want it to stay open

default
{
   state_entry()
   {
      llListen(3801,"Key Pad",NULL_KEY,"open");
   }
   listen(integer chan, string name, key id, string msg)
   {
      llSetStatus(STATUS_PHANTOM,TRUE);
      llSetAlpha(0.1,ALL_SIDES);
      llSleep(open_time);
      llSetAlpha(1.0,ALL_SIDES);
      llSetStatus(STATUS_PHANTOM,FALSE);
   }
}
```



Access Keypad (right) and Barrier Door

TextBoard.lsl

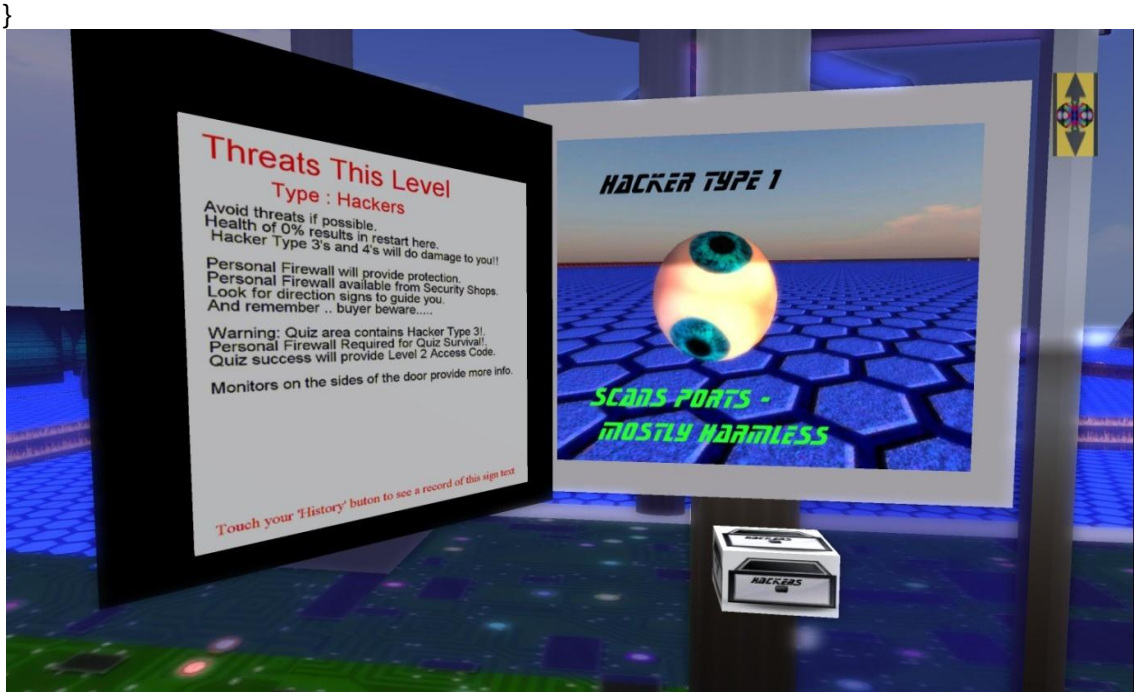
```
//Displays text on whiteborard
//Based on Code example from OpenSimulator.org
// http://opensimulator.org/wiki/OsSetDynamicdata_example1
//
// Licensed under the Creative Commons Attribution-Share Alike 2.5 License
```

```
// http://creativecommons.org/licenses/by-sa/2.5/
//Modified by Walter Ridgewell  2010

string title = "";
string subtitle = "";
string text = "";
string add = "";
integer channel = 0; // if this is >= 0, llSay on that channel on updates

push_text()
{
    compile_text();
    draw_text();
}

compile_text()
{
    title = "Threats This Level";
    subtitle = "Total 4";
    subtitle = "Type : Hackers";

    text = "Avoid threats if possible.\n";
    text += "Health of 0% results in restart here. \n";
    text += " Hacker Type 3's and 4's will do damage to you!! \n";
    text +=  " \n";
    text += "Personal Firewall will provide protection.\n";
    text += "Personal Firewall available from Security Shops. \n";
    text += "Look for direction signs to guide you. \n";
    text += "And remember .. buyer beware..... \n";
    text +=  " \n";
    text += "Warning: Quiz area contains Hacker Type 3!. \n";
    text += "Personal Firewall Required for Quiz Survival!. \n";
    text += "Quiz success will provide Level 2 Access Code. \n";
    text += " \n";
    text += "Monitors on the sides of the door provide more info.\n";

    add = "Touch your 'History' buton to see a record of this sign text";
}

draw_text()
{
    string drawList = "MoveTo 40,80; PenColour RED; FontSize 48; Text " + title + ";";
    drawList += "MoveTo 160,160; FontSize 32; Text " + subtitle + ";";
    drawList += "PenColour BLACK; MoveTo 40,220; FontSize 24; Text " + text + ";";
    drawList += "PenColour RED; FontName Times New Roman; MoveTo 40,900; Text " + add +
";";
    osSetDynamicTextureData("", "vector", drawList, "1024", 0);
}

default {
    state_entry()
    {
        push_text();
    }

    touch_start(integer count)
```

```
    {
        push_text();
        if (channel >= 0) {
            llSay(channel, text);
        }
    }

}
```

DataDrawer.lsl

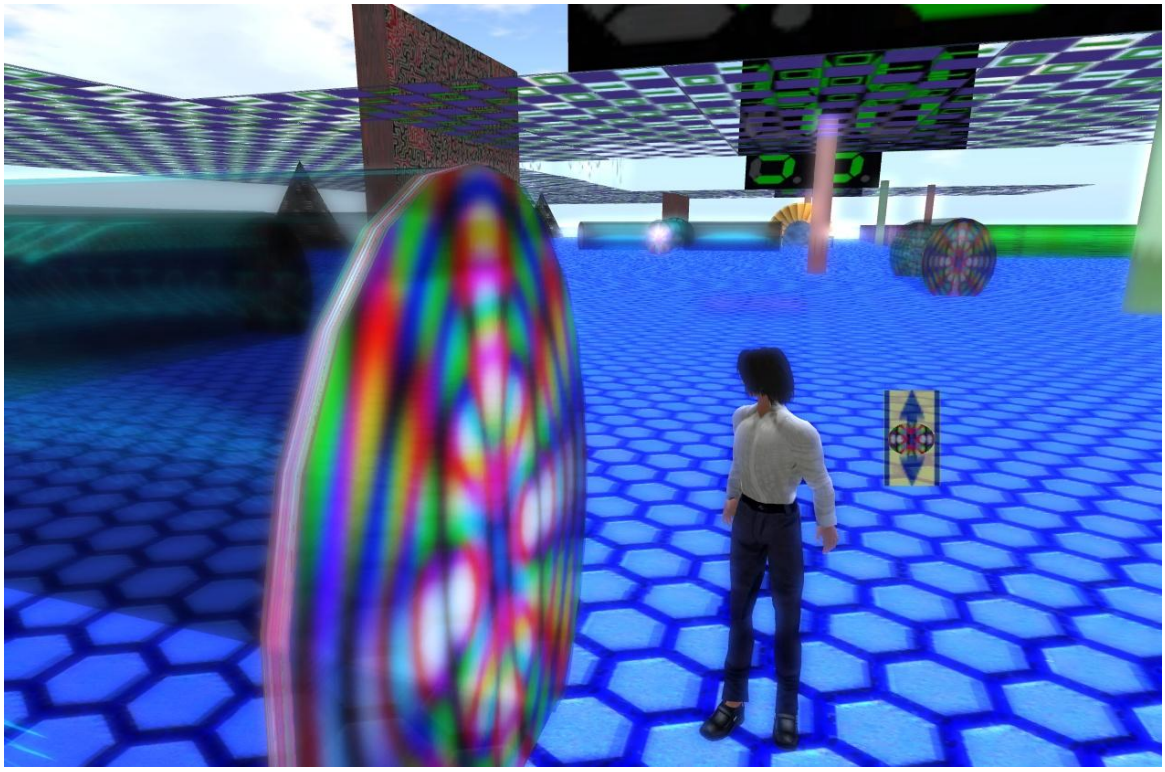
```
//InfoDrawer for giving away notecards and opening web pages
//Walter Ridgewell 2010
//Free to copy and modify retaining this notice
//

key AvatarKey;//Avatars KeyID

default
{
    state_entry()
    {
        llSleep(0.5); //sleep timer to lower resource usage
    }
    touch_start(integer total_number)
    {
        AvatarKey = llDetectedKey(0);//Get the ID of the avatar touching me for later URL command
        llGiveInventory(AvatarKey, "Hackers");//Give my contents away
        llLoadURL(llDetectedKey(0), "About Hackers",
"http://192.168.1.200:8080/exospace/Hackers.html");//Send extrnal URL and open browser on
avatars client
    }
}
```



TextBoard (left) Freeview Display (Right) InfoDrawer (Bottom Right)

VPN PipeDoor.lsl

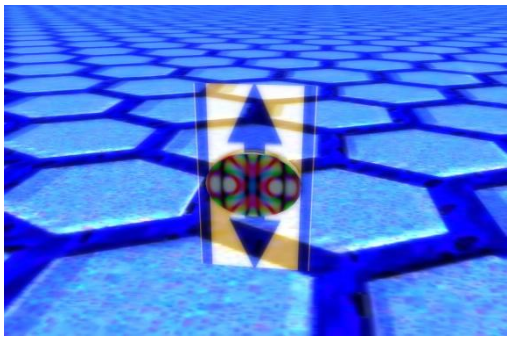
```
//VPN Pipe Door Script to work with Key
//Walter Ridgewell - 2010
//Free to copy and modify retaining this notice
//
//Based on Code example from SecondLife LSL Wiki
// http://wiki.secondlife.com/wiki/Key_Pad_Door
// Copyright © 2007-2009 Linden Research, Inc.
// Licensed under the Creative Commons Attribution-Share Alike 3.0 License
// http://creativecommons.org/licenses/by-sa/3.0/
//


integer channel = 25; // Channel to listen on
float open_time = 1; // Time to keep door open

default
{

   state_entry()
{
   llListen(channel,"VPN Access Card",NULL_KEY,"Activate");// Listen for VPN Access Card
"Activate" command
   }

   sensor(integer total_number)// Sensor state called when sensor detects Avatar
   {
      llSetStatus(STATUS_PHANTOM,TRUE);// Make it possible to pass through door
      llSetAlpha(0.1,ALL_SIDES);// Make door transparent
      llSleep(open_time);//Keep in this state for allotted time
      llSetAlpha(1.0,ALL_SIDES);//Make door look solid again
      llSetStatus(STATUS_PHANTOM,FALSE);////Door is solid and avatars can't pass
   }

   listen(integer chan, string name, key id, string msg)// Listen state for communcations
{
   llSay(0,"VPN Access Requested");// Request door access
// require the door to only to liste to requests from avatars in front of the door
//since other avatars may have keys and the command is sent grid wide
   llSensor("",NULL_KEY,AGENT,2,PI);// Activate with any agent within 2 meters -360 degrees

   }
}
```



VPN Pipe Door and VPN Access Key HUD

VPN Access Card.lsl

```
//VPN Access Card Script to work with Key
//Walter Ridgewell - 2010
//Free to copy and modify retaining this notice
//
integer channel = 25; // Channel to send command on

default
{
    state_entry()
    {
        llCollisionFilter("VPN Pipe Door","",TRUE);// Filter for only the VPN portal calls collision
    }
    collision_start(integer total_number)// activate on start of collision with object
    {
        //llSay(0,"OUCH !");
        llSay(channel, "Activate");// Broadcast command to activate portal
    }
}
```

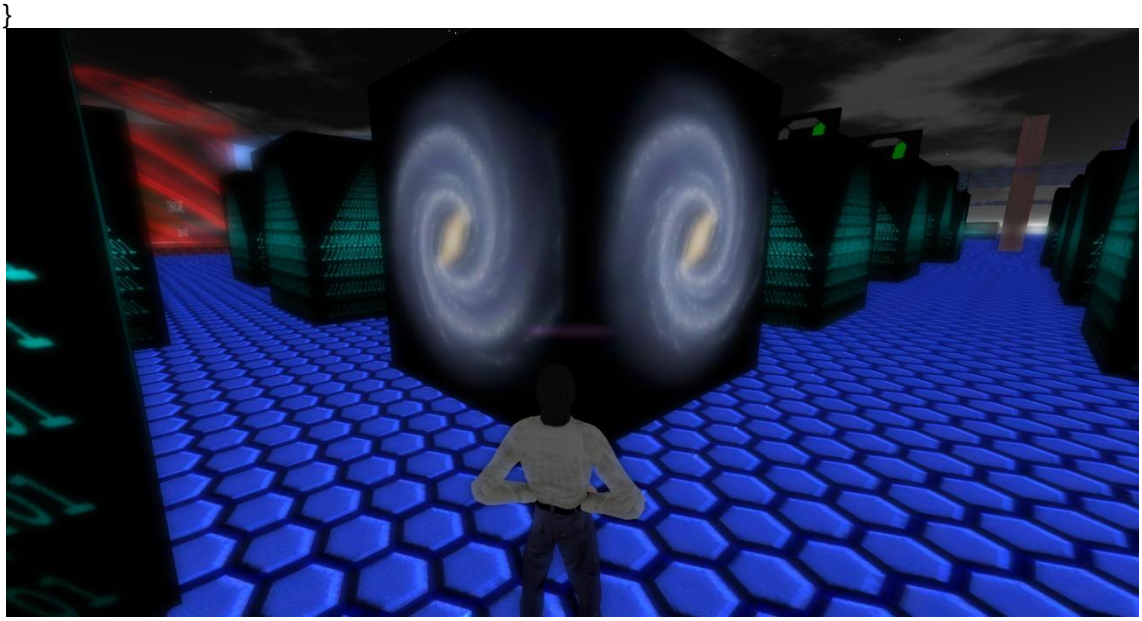


VPN Access Card

DataBlock.lsl

```
//Data Block Script change texture on touch randomly and objects
// if “ExoSpace Maze” image selected
//Walter Ridgewell - 2010
//Free to copy and modify retaining this notice
//
key AvatarKey;

default
{
 on_rez()
   {
          llResetScript();// reset random for sure, initial image 0
}

touch_start(integer total_number)
{
AvatarKey = llDetectedKey(0);    //get identity of Avatar touching me
integer number = llGetInventoryNumber(INVENTORY_TEXTURE);//Count hw many textures I
have in me
float rand = llFrand(number);// Pick random number based on texture quantity
integer choice = (integer)rand;//Take the integer and a index number
string name = llGetInventoryName(INVENTORY_TEXTURE, choice);//select the texture matching
the index number
llSetTexture(name, ALL_SIDES);// set the data cube texture to that number
if (name == "ExospaceMaze")// Is it the maze image
 {
    llGiveInventory(AvatarKey,"Maze KeyCard");// if so give out the keycard
   llGiveInventory(AvatarKey,"Maze NoteCard");//and the information notecard
    }
llSetTimerEvent(5);//hold image for  seconds then reset to default

}

timer ()
{
llSetTexture("BinaryData50", ALL_SIDES);   //default image reset by timer call
}
```

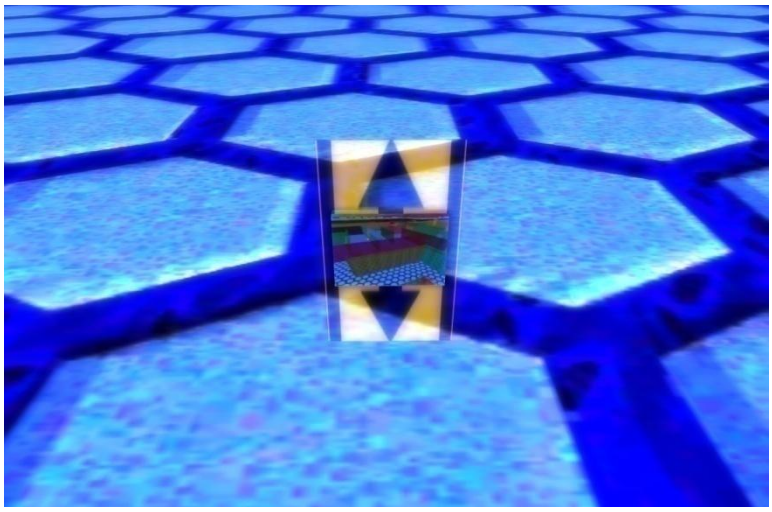
}



DataBlocks

Maze Key Object.lsl

```
//MazeDoor Key Script
//Walter Ridgewell - 2010
//Free to copy and modify retaining this notice
//
//Based on Code example from SecondLife LSL Wiki
// http://wiki.secondlife.com/wiki/Key_Pad_Door
// Copyright © 2007-2009 Linden Research, Inc.
// Licensed under the Creative Commons Attribution-Share Alike 3.0 License
// http://creativecommons.org/licenses/by-sa/3.0/
//
integer channel = 23; // Channel to speak on


default
{
   touch_start(integer total_count)
   {
      llSay(channel, "Open");// Send open command on channel
      llSay(0,"Activate Maze Door");//Inform avatar of command sent
   }
}
```



Maze Key Card

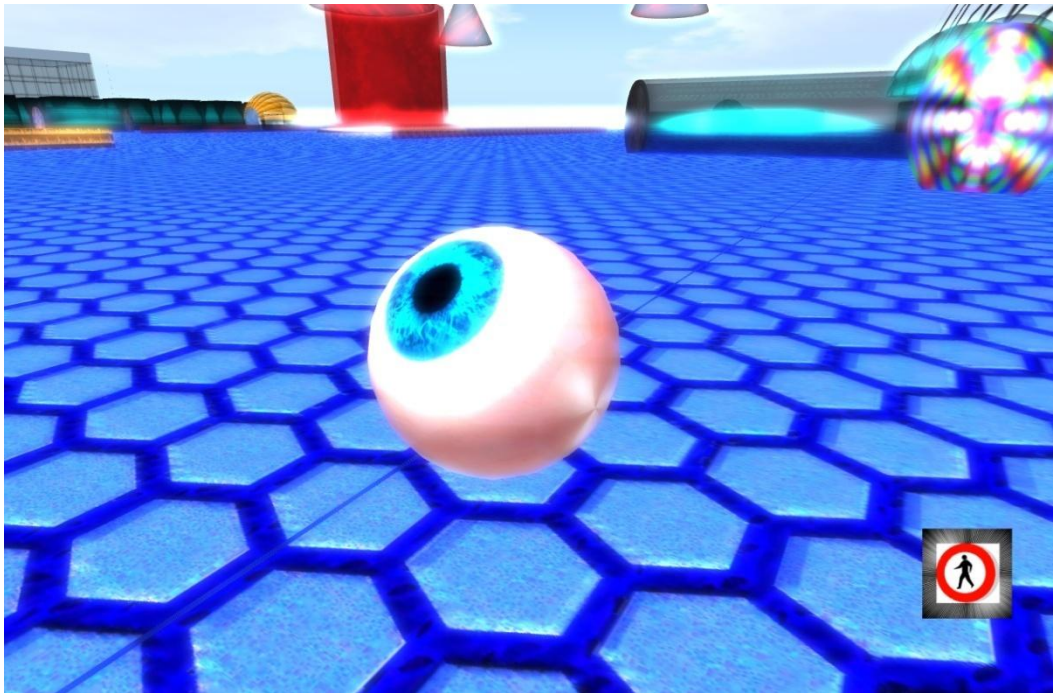
MazeDoor.lsl

```
//Maze Door Control Script to work with Key
//Walter Ridgewell - 2010
//Based on
//Code example from SecondLife LSL Wiki
// http://wiki.secondlife.com/wiki/Key_Pad_Door
// Copyright © 2007-2009 Linden Research, Inc.
// Licensed under the Creative Commons Attribution-Share Alike 3.0 License
// http://creativecommons.org/licenses/by-sa/3.0/
//

integer channel = 23; // Channel to listen on
float open_time = 2; // Length of time to stay open

default
{

   state_entry()
{
   llListen(channel,"Maze KeyCard",NULL_KEY,"Open");// Liste for keycard object
   }

   sensor(integer total_number)// Sensor state called when sensor detects Avatar
   {
      llSetStatus(STATUS_PHANTOM,TRUE);// Make it possible to pass through door
      llSetAlpha(0.1,ALL_SIDES);// Make door transparent
      llSleep(open_time);//Keep in this state for allotted time
      llSetAlpha(1.0,ALL_SIDES);//Make door look solid again
      llSetStatus(STATUS_PHANTOM,FALSE);////Door is solid and avatars can't pass
   }


   listen(integer chan, string name, key id, string msg)
{
// require the door to only to listen to requests from avatars in front of the door
//since other avatars may have keys and the command is sent grid wide
```

```
    llSensor("",NULL_KEY,AGENT,2,PI); // Activate with any agent within 2 meters -360 degrees

    }
}
```

Maze Door and HUD KeyCard

HackerType 1



Hacker Type 1 Looking for Targets

HackerBrain.lsl

```
//HackerBrain Script
//Walter Ridgewell 2010
// Scans for Avatars and broadcasts name and location
//
//Based on code example from Catfart Grayson
//http://forums-archive.secondlife.com/15/0e/190698/1.html
//"The code is open and free for you to use as you want. You may use the code in your own
//commercial products, but may not sell the creature as it stands."
//
// This Command script talks to the movement script which can located be in a second prim //
section
// eg. this script in an antenna and the movement script in a ship, body ect.
//
//
key current_target; // ID key of Avatar spotted
vector current_target_pos;// Target position or Avatar
string target_name;// Name of Target Avatar
vector offset = < -1, 0, 1>;  //1 meter behind and 1 meter above target center.

vector select_new_pos() // Function  to generate new random move location X and Y coordinates
{
    integer good_pos = FALSE;// ready to generate a position flag
    while(good_pos == FALSE)
    {
        float xoffset = llFrand(20) - 10;//20 meter X coord range
        float yoffset = llFrand(20) - 10;//20 meter Y coord range
```

```
        if(llGround(<xoffset,yoffset,0>) > llWater(<xoffset,yoffset,0>))// Am I on the surface?
        {
            if((llGetLandOwnerAt(llGetPos() + <xoffset,yoffset,0>)) == llGetOwner())// Am I in my Sim
            {
                good_pos = TRUE; //set flag
                vector current_pos = llGetPos();// where are we
                float new_x = current_pos.x + xoffset;//generate next X location
                float new_y = current_pos.y + yoffset;//generate next Y location
                float new_z = llGround(<xoffset,yoffset,0>);//Calculate Z above ground
                return <new_x,new_y,new_z>;// return new position to move to
            }
        }
    }
    //Shouldnt actually ever get to this line
    return llGetPos();

}

default                 //We begin
{
    state_entry()
    {
        llSay(0,"Scanning for targets");
        //llWhisper(0,"pop");
        llSetStatus(STATUS_DIE_AT_EDGE,TRUE);// If I go off the edge of the world die
        state search_for_avatar;// Initiate Avatar search
    }
}

state search_for_avatar  // Aavatar search state
{
    state_entry()// begin
    {
        llPlaySound("6_beeps",1.0);// Play some sound effects
        llWhisper(0,"Scanning for targets now...");//State our mode of operation

        list command=["MOVE",select_new_pos(),1];// move to new position
        llMessageLinked( LINK_SET, 150, llDumpList2String(command, "|"), "" );// call movement
script
        llSetTimerEvent(10);//Scan for 10 seconds

    }
    timer()// start of scan event
    {
        llSensor("","",AGENT,30,2 * PI);// search for an Avatar, range 30 meters , 360 degree sweep
    }
    sensor(integer num)// sensor event, if I find something I execute this, note num variable is
number of targets
    {
        integer x = 0;// Process scanned Avatars
        for(x;x < num;x = x +1)// cycle thru Avatars
        {
            if((llGetLandOwnerAt(llDetectedPos(x))) == llGetOwner())//Check to see if I'm in the
correct sim
            {
                current_target = llDetectedKey(0);// Hackers position
```

```
            current_target_pos = llDetectedPos(0);// Targets position
            target_name = llKey2Name(llDetectedKey(0));// Taget name
            //Avatar found
            llSay(0,"Found target "+target_name);// Identify target
            llShout(1,target_name+"|"+(string)current_target_pos); // Tells other hackers position of
discovered Avatar
            llSay(0,"Attack "+llKey2Name(llDetectedKey(0)));// Give attack command
            state aquired_avatar;// Move towards the target
         }
         list command=["MOVE",select_new_pos(),1];//Generate move command
         llMessageLinked( LINK_SET, 150, llDumpList2String(command, "|"), "" );//Send move
command
      }
   }
   no_sensor()// No target found
   {
      list command=["MOVE",select_new_pos(),1];//Generate move command
      llMessageLinked( LINK_SET, 150, llDumpList2String(command, "|"), "" );//Send move
command
   }
   link_message(integer sender,integer num, string data, key id)// Listen for the response from the
second script
   {
      if(num == 150)// Check sender ID
      {
         list Arguments = llParseStringKeepNulls( data, ["|"], [] );// Parse communications from
movement script
         string Command = llList2String( Arguments, 0 );// transfer contents
         if( Command == "ATPOS" )//have we arrived
         {
            llSensor("","",AGENT,30,2 * PI);//if so run 360 degree scan again 30 meter range
            // if target found back to sensor event
         }
      }
   }
}

state aquired_avatar// Found Avatar in scan
{
   state_entry()
   {
      llSensorRepeat("",current_target,AGENT,20,2 * PI, 1);// repetitive 20 meter 360 degree scan
   }
   sensor(integer num)// Target found  sensor call
   {
      current_target_pos = llDetectedPos(0);//Get position
      if(llGetLandOwnerAt(current_target_pos) == llGetOwner())//In my sim?
      {
         list command=["MOVE",current_target_pos,2];// Move towards target
         llMessageLinked( LINK_SET, 150, llDumpList2String(command, "|"), "" );// Check for
response
      }
      else
      {
         state search_for_avatar;// No target found back to search state
      }
```

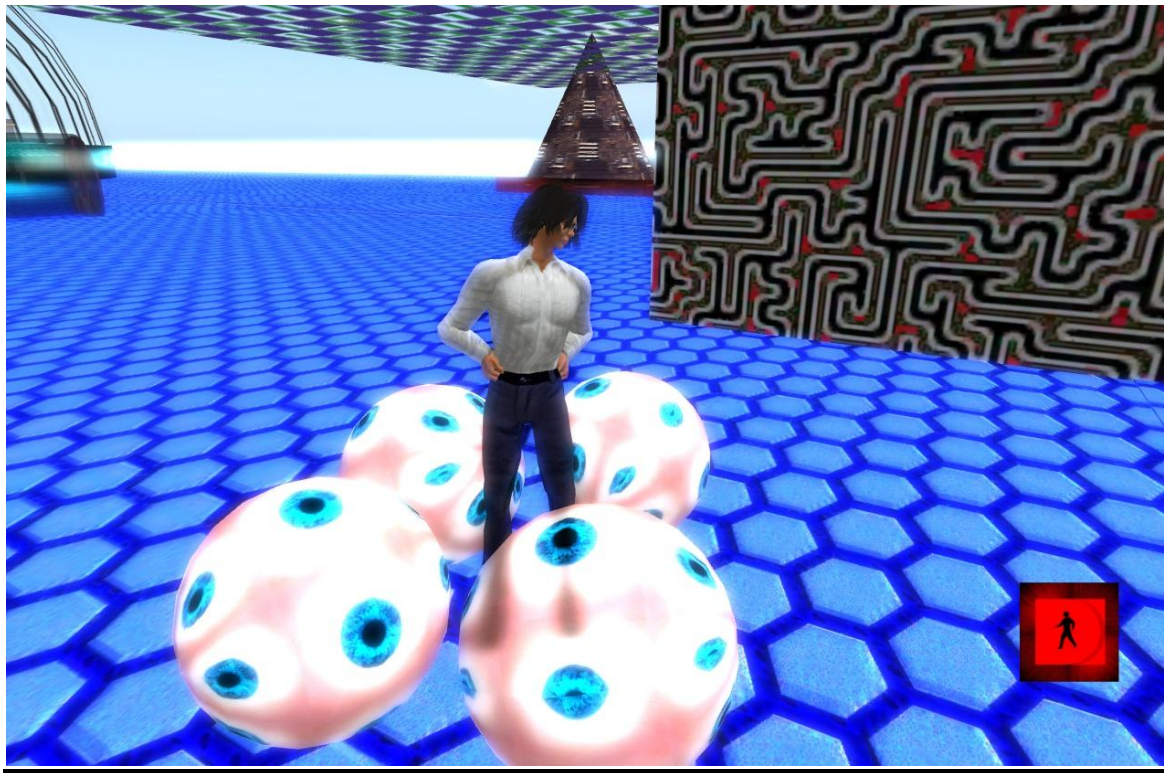
```
    }
    no_sensor()// No sensor hit
    {
        //gone

        state search_for_avatar;// Not target found back to search state
    }
    link_message(integer sender,integer num,string data,key id)// Check for movemnet results
    {
        if(num == 150)
        {
            list Arguments = llParseStringKeepNulls( data, ["|"], [] );
            string Command = llList2String( Arguments, 0 );
            if( Command == "ATPOS" )
            {
                //reached avatar!!
                state reached_avatar;
            }
        }

    }
}

state reached_avatar // We are next to current target
{
    state_entry()
    {
        llSay(0,"Reached target location");// Notify reached target
        llSensorRepeat("",current_target,AGENT,4,2 * PI, 3);// Perform short ranged scan
    }
    sensor(integer num)// Follow target to annoy it... while we can 'see it'

    {
        // Owner detected...
        // Get position and rotation
        vector pos   = llDetectedPos(0);
        rotation rot = llDetectedRot(0);
        // Offset back one metre in X and up one metre in Z based on world coordinates.
        // use whatever offset you want.
        vector worldOffset = offset;
        // Offset relative to owner needs a quaternion.
        vector avOffset = offset * rot;

        pos += avOffset;       // use the one you want, world or relative to AV.

        llMoveToTarget(pos,0.4);
    }
    no_sensor()// If it escapes....
    {
        llSay(0,"Target Lost");
        state search_for_avatar;// Back to looking for targets..
    }
}
```

HackerMovement.lsl

```
//HackerMovement Script
//Walter Ridgewell 2010
//Based on code example from Catfart Grayson
//http://forums-archive.secondlife.com/15/0e/190698/1.html
//"The code is open and free for you to use as you want. You may use the code in your own
commercial products, but may not //sell the creature as it stands."
//
// Moves physical object in small increments
//
integer reached_target = TRUE;// status flag
vector jump_force = <2,0,9.85>;//Impulse movememnt just above gravity 9.8
vector target_position;//Avatar found position
integer target_id;//Avatar ID

impulse()//Generate physical movement pulse
{
   if(reached_target == FALSE)
   {
      llRotLookAt(getRotToPointAxisAt(AXIS_FWD , target_position), 10, 0.5);
      llApplyImpulse(jump_force * llGetMass(),TRUE);
   }
}

vector AXIS_UP = <0,0,1>;//1 meter up
vector AXIS_LEFT = <0,1,0>;//1 meter left-right
vector AXIS_FWD = <1,0,0>;//1 meter fwd-bck

// getRotToPointAxisAt()
// Gets the rotation to point the specified axis at the specified position.
// @param axis The axis to point. Easiest to just use an AXIS_* constant.
// @param target The target, in region-local coordinates, to point the axis at.
// @return The rotation necessary to point axis at target.
// Created by Ope Rand, modifyed by Christopher Omega rotation
rotation getRotToPointAxisAt(vector axis, vector target)
{
   return  llGetRot() * llRotBetween(axis * llGetRot(), target - llGetPos());
}


default// Begin
{
   state_entry()
   {
      llWhisper(0,"Movement");
      llSetStatus(STATUS_PHYSICS,TRUE);//Phsyical movemnet
      llSetStatus(STATUS_ROTATE_X,FALSE);//Don't rotate around X
      llSetStatus(STATUS_ROTATE_Y,FALSE);//Don't rotate around Y
      state move; //Movement
   }
}

state move
{
   state_entry()
   {
```

```
    }

    not_at_target()// Not at scanned targets position ...target not found here off we go
    {
        llRotLookAt(getRotToPointAxisAt(AXIS_FWD , target_position), 2, 0.5);// Turn towards the
last Avatar position
    }

    link_message(integer sender, integer num,string data, key id)// Script communications to brain..
    {
        if(num == 150)// ID
        {
            list Arguments = llParseStringKeepNulls( data, ["-=-"], [] );// Parse command string
            string Command = llList2String( Arguments, 0 );//transfer
            if( Command == "MOVE" )// Movement
            {
                // llWhisper(0,"move");
                llTargetRemove(target_id);// Remove target from the list
                target_id = llTarget((vector)llList2String( Arguments, 1 ),2);//Can I Get target ID
                target_position = (vector)llList2String( Arguments, 1 );// Can I Get to target location
                reached_target = FALSE;//Not there yet
                impulse();// Move again
            }
        }
    }

    at_target(integer tnum,vector targetpos,vector ourpos)// At targets position
    {
        reached_target =TRUE; // We are here
        llMessageLinked(LINK_SET,150,"ATPOS","");// relay that back to brain
    }

    collision_start(integer num)// Hit an object move again..
    {
        impulse();
    }

    land_collision_start(vector pos)// Hit the ground eg. bounce .. move again
    {
        impulse();
    }
}
```

HackerType 2



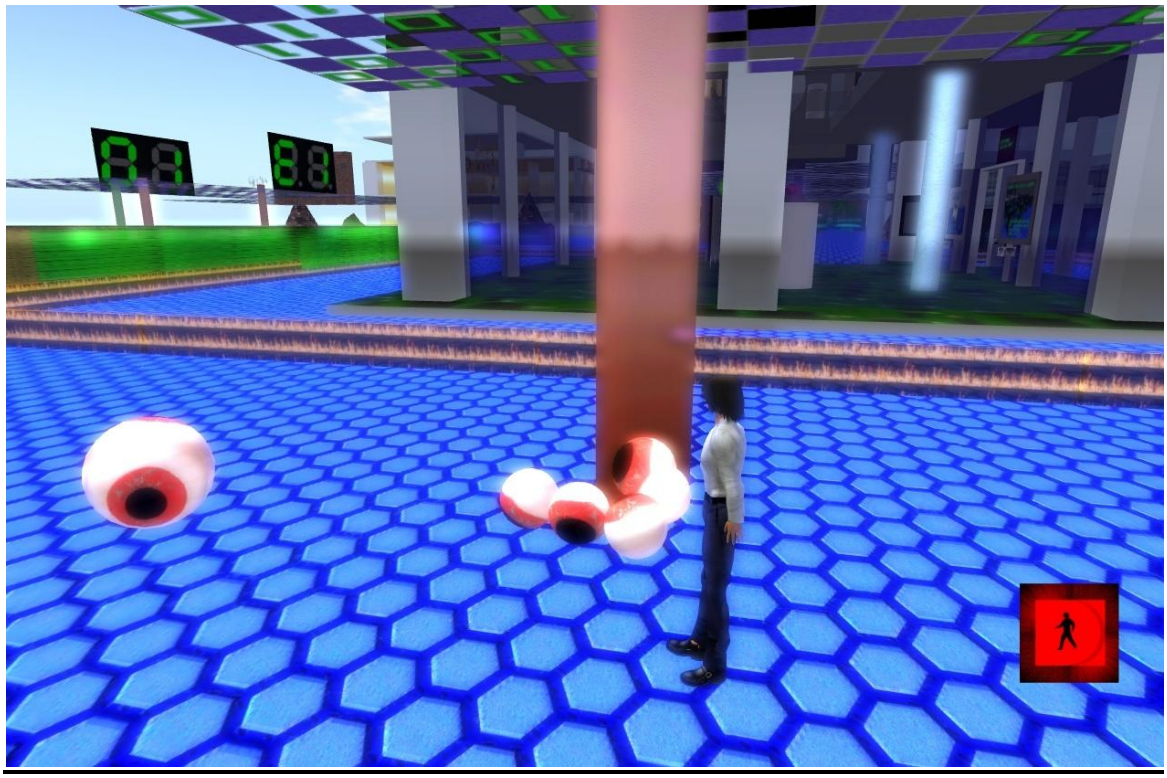
Surrounded by Hacker Type 2 Malware

Malware.lsl

```
// New Script Name: Malware.lsl
// Category: Virus simulation// Description: Swarms Target Avatars spotted by HackerType1
// Modified by: W Ridgewell, 2010
//
//Based on
// Script Name: Zombie-prim.lsl
// Category: Weapons
// Description: Moves to an avatar when their name is spoken
// Comment: The Script
// From the Internet LSL Script Database & Library of Second Life™ scripts.
// http://www.free-lsl-scripts.com  by Ferd Frederix
//
// This program is free software; you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation; either version 2 of the License, or
// (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
```

```
// GNU General Public License for more details.
//
// Other licenses may be located within the source of this script,
// in which case the more restrictive license is to take effect
//
////////////////////////////////////////////////////////////////
//Zombie-prim
//By TSL Resident: Nitsuj Kidd

string target ;//Target ID holder
integer up = 1;// 1 m up
integer force = 1;//1 m forward
default
{
   state_entry()
   {
      llSetStatus( STATUS_DIE_AT_EDGE | STATUS_PHYSICS, TRUE);//If I go offworld
die/physics ally
      llListen(0,"","","");  // Listen on the main channel for all talk
      llSensorRepeat("","",AGENT,30,TWO_PI,2);//scan for avatars, 30m range
   }
   listen(integer chan, string name, key id, string msg)// Listen for specific ‘attack’ msg
   {
      if(llToLower(llGetSubString(msg,0,llStringLength("attack")))=="attack ")//Parse heard comm.
For  attack
      {
         target = llToLower(llDeleteSubString(msg,0,llStringLength("attack")));//If heard, extract
target avatar name
         llOwnerSay("Target set to " + target);// Announce intentions
      }
   }
   sensor(integer num)//called if I can see an Avatar that matches name in message
   {
      integer i;
      for(i=0; i<num; i++)
      {
         if(llSubStringIndex(llToLower(llDetectedName(i)),target)!=-1)// if I’m not at target
         {
            llApplyImpulse(((llDetectedPos(i)-llGetPos())*force)+<0,0,up>,FALSE);//move to
detected targets position utilizing impulse push
         }
      }
   }
}
```

HackerType 3



Hacker Type 2 Firing Virus Infectors – Note Firewall Indicator Is Red (Off = Danger)

VirusAttack.lsl

```
// New Script Name: VirusAttack
// Category: Virus simulation// Description: Fires Virus Objects
// Comment: Targeting set by Hacker Type 1 broadcast, Fires malicious Virus 'bullet' Objects
// Modified by: W Ridgewell
//
//Based on
// Original Script Name: Stalking_lava_Fire.lsl
// Category: Weapons
// From the Internet LSL Script Database & Library of Second Life™ scripts.
// http://www.free-lsl-scripts.com  by Ferd Frederix
//
// This program is free software; you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation; either version 2 of the License, or
// (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
```

```
// GNU General Public License for more details.
//
// Other licenses may be located within the source of this script,
// in which case the more restrictive license is to take effect
//
//////////////////////////////////////////////////////////////////

string victim;// target ID
key sound = "electricity arcs";//Scary sound file
vector start_pos;// initial position
integer channel;// communication channel
vector target_pos;//target position
list target_info;// List of detected targets



warpPos(vector destpos)
{   //R&D by Keknehv Psaltery, 05/25/2006
    //with a little pokeing by Strife, and a bit more
    //some more munging by Talarus Luan
    //Final cleanup by Keknehv Psaltery
    // Compute the number of jumps necessary
    integer jumps = (integer)(llVecDist(destpos, llGetPos()) / 10.0) + 1;
    // Try and avoid stack/heap collisions
    if (jumps > 100 )
        jumps = 100;    //  1km should be plenty
    list rules = [ PRIM_POSITION, destpos ];  //The start for the rules list
    integer count = 1;
    while ( ( count = count << 1 ) < jumps)
        rules = (rules=[]) + rules + rules;   //should tighten memory use.
    llSetPrimitiveParams( rules + llList2List( rules, (count - jumps) << 1, count) );
}



default
{
    state_entry()
    {
        channel = 1;// command channel
        llListen(channel,"",NULL_KEY ,"");//monitor all boradcasts
        start_pos = llGetPos();//record my current position
        llSay(0,"Listening for Target");//state my intention
        llSetText("Monitoring",<0,1,0>,1);//show my intention

    }
    listen( integer channel, string name, key objectkey, string message )
    {
        target_info = llParseString2List(message,["|"],[]);//Listen for a Hacker 1 formatted string
        victim = llList2String(target_info,0);//extract target name
        llShout(0,"Target set to " + victim);//announce
        target_pos=llList2Vector(target_info,1);//extract target location
        llShout(0,"Target position is "+ (string)target_info);//announce
        warpPos(target_pos);//move to target area
        llSensor("",NULL_KEY,AGENT,16,PI);// initiate scan
    }
```

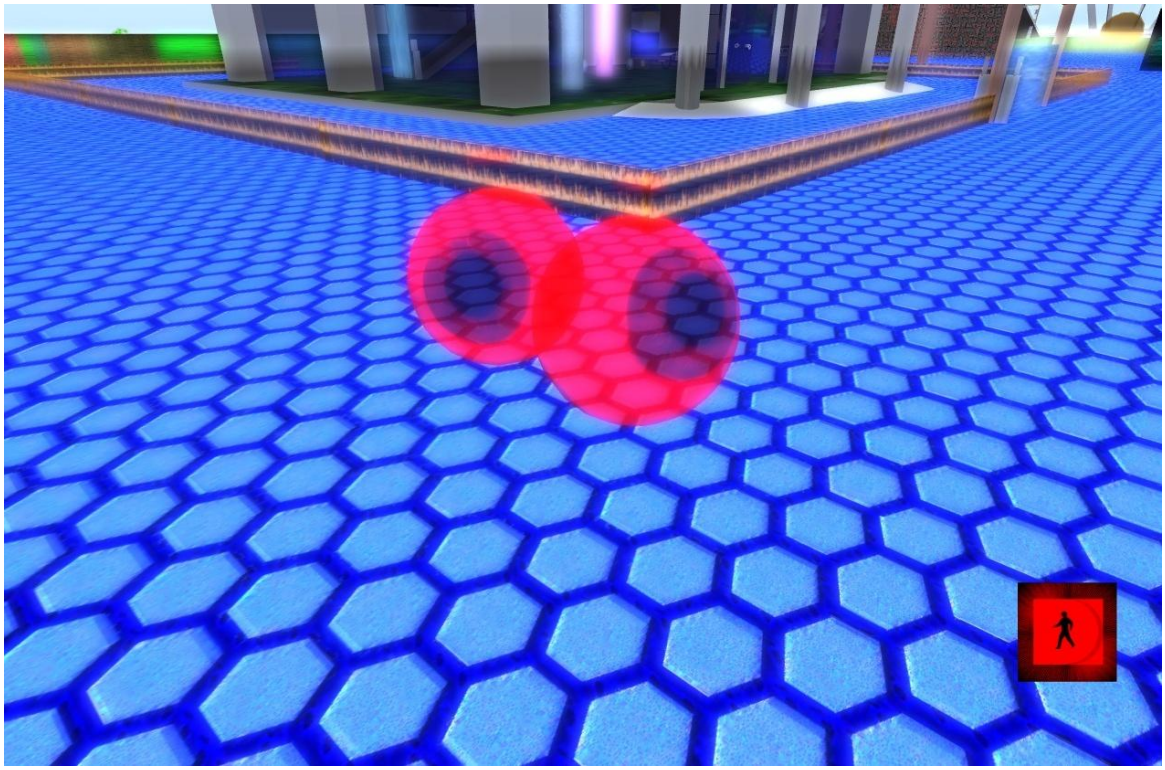
```
    no_sensor()// Nothing seen
    {
        llSay(0,"Switching to scan mode...");//goto search mode
        state searching;//jump
    }
    sensor(integer num)//Seen target
    {
        state tracing;//begin to follow
    }
}

state tracing
{


    state_entry()
    {
        llSetText("Tracing target", <0,0,1>, 1.0);//State mode
         llSensorRepeat(victim,NULL_KEY,AGENT,10,PI,1.0);// begin 10m tracking scan

    }
    no_sensor()//not seen
    {
        state searching;// goto search mode
    }
    sensor(integer num)//seen
    {

        llSetStatus(STATUS_PHYSICS, TRUE); // Enable physics to allow physical movement.
        vector pos = llDetectedPos(0);//get target position
        vector offset =<-5,0,0>;//5 m back
        pos+=offset;//adjust our postion
        llMoveToTarget(pos, 0.5); // Move to the detected position.
        object=llGetInventoryName(INVENTORY_OBJECT,0);  //get name of the 1st object in
inventory ie Virus
        llRezObject(object, llGetPos() + <0.0, 0.0, 0.5>, ZERO_VECTOR, ZERO_ROTATION, 0);
//rez the 1st attack bullet/virus object – repeats per sensor hit

    }

}



state searching
{
    state_entry()
    {
        llSetTextureAnim(FALSE | SMOOTH | PING_PONG | LOOP | REVERSE, ALL_SIDES, 1, 1,
0.70, 0.2, 0.2);//move my textures as a warning I'm upto something
        llShout(0,victim + " Search in Progress");//announce my intention
        llSensorRepeat(victim,NULL_KEY,AGENT,10,PI,1.0);//repeat scans @ 10m
        llSetText("Scanning",<1,0,0>,1);//set my mode display

    }
    no_sensor()//not seen
```

```
    {

        llShout(0,"Target out of range - going back to start position " + (string)target_pos);//bah
gone
        warpPos(target_pos);// back to the original target seen position
        llShout(0,"Exiting Warp Drive....");// done my jump
        state default;//back to monitoring

    }
    sensor(integer num)//seen
    {
       vector here = llGetPos();// get position
        llShout(0,"Located "+(string)victim+" at "+(string)here str);//victory yell.. announce
       llPlaySound("electricity arcs",1.0);//scary sound
       state tracing;//track and attack target
    }
}
```

MalwareBullet.lsl

```
//MalwareBullet fired by Hacker Type 3
//Walter Ridgewell - 2010
//Based on
//Code example from SecondLife LSL Wiki
// http://wiki.secondlife.com/wiki/LlSetDamage
// Copyright © 2007-2009 Linden Research, Inc.
// Licensed under the Creative Commons Attribution-Share Alike 3.0 License
// http://creativecommons.org/licenses/by-sa/3.0/
//
default
{
    on_rez(integer param) // Becomes active when rezzed.
    {
        llSetDamage(20.0); // Set the damage to only 20%
        llSensor("", "", AGENT, 5.0, PI); // Sweep a 5 meter sphere searching for agents.
    }
    sensor(integer num) // If an agent is detected...
    {
        llSetStatus(STATUS_PHYSICS, TRUE); // Enable physics to allow physical movement.
        llSetTimerEvent(10.0); // Set a 10 second timer.
        llMoveToTarget(llDetectedPos(0), 0.5); // Move to the detected position.
    }
    no_sensor() // If no agents are detected...
    {
        llDie(); // Auto destruct.
    }
    timer() // If we missed our target...
    {
        llDie(); // Auto destruct.
    }
}
```

HackerType4



Hacker Type 4 DDOS

DDOS.lsl

```
//Hacker Type 4 Denial Of Service Attack – Bounces Avatars off Grid
//Walter Ridgewell 2010
//Free to copy and modify retaining this notice
//
//Based on
// Original Script Name: Stalking_lava_Fire.lsl
// Category: Weapons
// From the Internet LSL Script Database & Library of Second Life™ scripts.
// http://www.free-lsl-scripts.com  by Ferd Frederix
//
// This program is free software; you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation; either version 2 of the License, or
// (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
//
// Other licenses may be located within the source of this script,
// in which case the more restrictive license is to take effect
//
////////////////////////////////////////////////////////////////////
```

```
string victim; // Target ID
key sound = "electricity arcs";// sound effects
vector start_pos;// Starting postion location
vector target_pos;// Targets position
list target_info;// List of heard target data



default
{
   state_entry()
   {
      llSetStatus( STATUS_DIE_AT_EDGE | STATUS_PHYSICS, TRUE);//Die at edge of world
physics true
      llListen(1,"","","");//listen to command channel
      start_pos = llGetPos();//record my start position
   }

   listen( integer channel, string name, key objectkey, string message )
   {
      target_info = llParseString2List(message,["|"],[]);// Listen for broadcast from hacker 1 target
and postion, put in list
   {
      victim = llList2String(target_info,0);//extract Target ID
      llSay(0,"Target set to " + victim);//announce
      target_pos=llList2Vector(target_info,1);//extract Target Data
      llSay(0,"Target position is "+ (string)target_info);//Announce
   }
      state tracking;//got tracking state
   }
}

state tracking//Tracking state
{


   state_entry()
   {
      llSetPos(target_pos);//goto target postion
      llSetText("Acquiring Target...", <1,0,0>, 1.0);//announce intention
      llSensorRepeat(victim,NULL_KEY,AGENT,10,PI,1.0);//scan for target 10m range

   }
   no_sensor()
   {
      state searching;//not found goto search
   }
   sensor(integer num)
   {
      vector pos = llDetectedPos(0);//Found target get loaction
      llMoveToTarget(pos, 0.05);//move close to target
      llPlaySound("electricity arcs",1.0);//play ominous sound
```

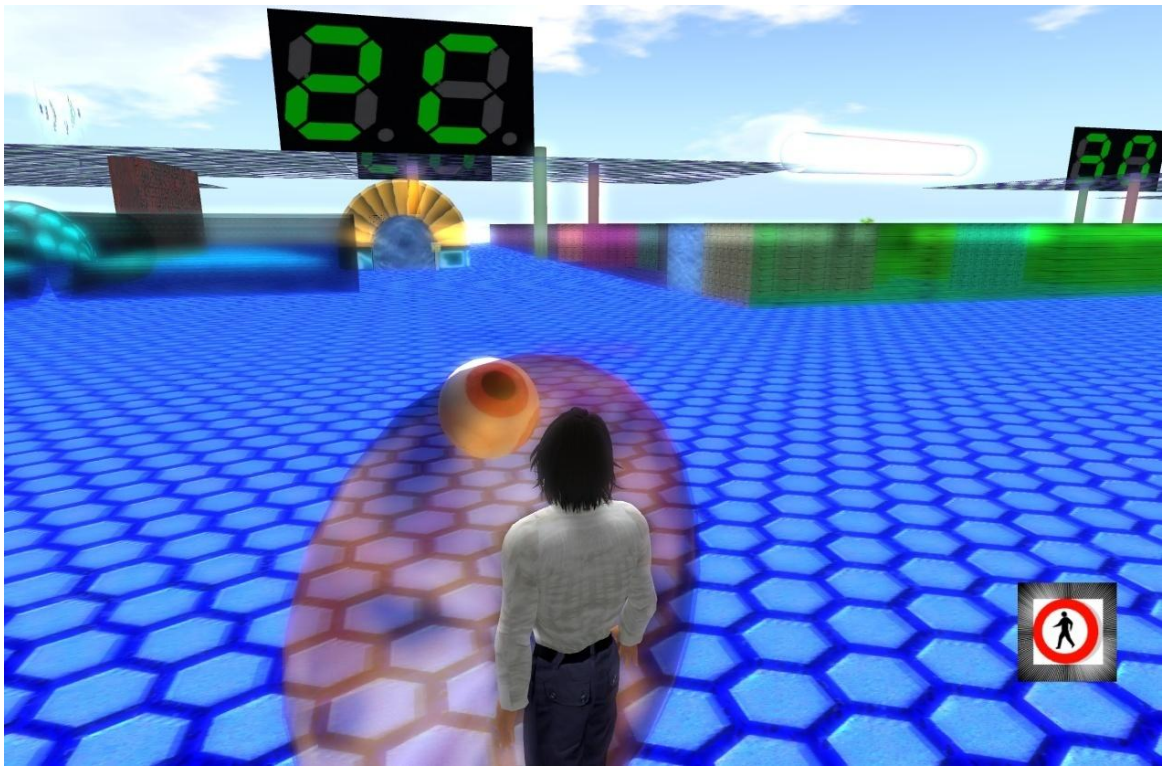
```
      llPushObject(llDetectedKey(0), <0,0,2000>, ZERO_VECTOR, FALSE);//Bounce target off
the grid


   }

}

state searching// Search State
{
   state_entry()
   {
      llSetTextureAnim(FALSE | SMOOTH | PING_PONG | LOOP | REVERSE, ALL_SIDES, 1, 1,
0.70, 0.2, 0.2);// Animate the objects skin
      llSensorRepeat(victim,NULL_KEY,AGENT,20,PI,1.0);//scan for target 10m range

      llSetText("Scanning",<1,0,0>,1);// announce intention

   }

   no_sensor()// Not Found
   {

      llSay(0, "Target out of range - going back to start position " + (string)start_pos);//Going back
to where I was
      llSetPos(start_pos);// Back to where I was
      state default;//return to initial state
   }

   sensor(integer num)
   {
      vector here = llGetPos();// See target again
      llSay(0,"Located "+victim+" at "+(string)here);//Announce discovery
      llSetPos(here);// Goto Target position
      state tracking;//Go after target
   }
}
```

FireWall.lsl

```
//Firewall HUD Device - Rez shield between User, Hackers and Virus Objects
//Walter Ridgewell 2010
//Free to copy and modify retaining this notice
//

string status = "off"; // Set our Active Status
rotation between; // Intercept shield point

default
{
   touch_start(integer num_detected)
   {
      if (status=="off")// Are we turned off
      {
```

```
            llOwnerSay("Firewall On.");//Activate
            status = "on";//change status
            llSetColor(<1,1,1>,ALL_SIDES);// Change color to white
            llShout(1,"Attack Willy Wonka");//Send out decoy command string to other hackers
            llSensorRepeat("","",ACTIVE,5,TWO_PI,0.1);// Activate sensorsweep for shield
        }
        else if (status=="on")// We were turned on
        {
            llOwnerSay("Firewall Off.");//Deactivate Firewall
            status = "off";//change status
            llSetColor(<1,0,0>, ALL_SIDES);//Set colour to red
            llSensorRemove();//Deactivate sensor
        }


    }

    sensor(integer num_detected)//Sensor state
    {
        if(llVecMag(llDetectedVel(0)) > 1.0)//is there a moving object near us
        {
 between = llGetRot() * llRotBetween(<0.5,0,0> * llGetRot(), llDetectedPos(0) -
llGetPos());//determine intercept point
            llRezObject("Personal_FirewallV2",llGetPos() + <0.5,0,0> *
between,ZERO_VECTOR,between,0);//rez defense shield at intercept point

        }
    }
}
```

Firewall HUD


Hacker Type 3 Attack and Firewall Shield