**TECNOLOGICO NACIONAL DE MEXICO**

**INSTITUTO TECNOLOGICO DE CHIHUAHUA ll**

****

**DEPARTAMENTO DE SISTEMAS Y COMPUTACION**

**INGENIERIA EN SISTEMAS COMPUTACIONALES**

**ESTRUCTURA DE DATOS**

**TAREA#2 FORMATO DE TEXTO**

**EVALUACION PARCIAL #1**

**ELENA GUADALUPE DIAZ LOPEZ**

**18550623**

**INDICE Pág.**

Introducción…………………………………………………………………………….3

Objetivo…………………………………………………………………………………3

Errores de software informático gravemente épicos………………………………4

Protección contra copia maliciosa de cd de sony………………………………...10

Conclusiones………………………………………………………………………….12

Recomendaciones……………………………………………………………………12

Referencias…………………………………………………………………………...12

**INDICE DE FIGURAS**

**Fig.1-** Therac-25………………………………………………..................................4

Fig.2- World of Warcraft………………………………………………………………5

Fig.3- Puente Queensboro……………………………………………………………6

**Fig.4-** USS Yorktown (CG-48)…………………………………………………………7

Fig.5- Stanislav Petrov………………………………………………………………..8

Fig.6- Lanzamiento de misiles Patriot……………………………………………..10

**INTRODUCCION**

En este documento a continuación presentado se comentan algunas de las situaciones con errores de Software mas caóticamente importantes, se describen los detalles documentados de estos mimos sucesos con evidencia recabada verídicamente.

**OBJETIVO**

El principal objetivo de este documento es hacer conciencia de la importancia que tiene la calidad de un software y las consecuencias que se pueden llegar a obtener en caso de que este mismo tenga fallas fatídicas o de tales dimensiones como en los siguientes casos se mencionan.

**ERRORES DE SOFTWARE INFORMATICO GRAVEMENTE EPICOS.**

La mayoría de los errores de software son pequeños inconvenientes que pueden ser superados o trabajados por el usuario – pero hay algunos casos notables donde un simple error ha afectado a millones, a un grado u otro, e incluso causó lesiones y pérdida de vidas. El software está escrito por los seres humanos, y cada pieza de software por lo tanto tiene errores, o "características no documentadas" como un vendedor podría llamarlos. Es decir, el software hace algo que no debería, o no hace algo que debería. Estos errores pueden deberse a un mal diseño, malentendido de un problema o simplemente un simple error humano, como un error tipográfico en un libro. Sin embargo, mientras que un libro es leído por un humano que normalmente puede inferir el significado de una palabra mal escrita, el software es leído por computadoras, que son comparativamente estúpidas, y harán sólo lo que se les dice. Aquí hay diez casos en los que las consecuencias de estos errores eran enormes ,de una manera u otra:10

**Therac-25**

**1985-1987**

****

Fig.1 Therac-25

Therac-25 era una máquina para administrar radioterapia, generalmente para el tratamiento del cáncer pacientes. Tenía dos modos de operación. El primero consistió en un haz de electrones dirigido directamente ael paciente en pequeñas dosis durante un corto período de tiempo. El segundo apuntaba primero al haz de electrones a niveles energía de alto a un "objetivo" metálico, que esencialmente convertiría el haz en rayos X que luego pasaron al paciente. En modelos anteriores de la máquina Therac, para este segundo modo de operación, había seguridad física para asegurarse de que este objetivo estaba en su lugar ya que, sin él, los haces de energía muy alta se podían disparar por error directamente al paciente. En el nuevo modelo, estos seguros físicos fueron reemplazados por los de software. Desafortunadamente, hubo un error en el software: un "desbordamiento aritmético" a veces ocurrió durante las comprobaciones de seguridad automáticas. Esto significa básicamente  
que el sistema estaba usando un número dentro de sus cálculos internos que era demasiado grande para que manejara. Si, en este preciso momento, el operador estuviera configurando la máquina, las comprobaciones de seguridad fallarían y el objetivo metálico no se llevaría a su lugar. El resultado fue que los haces 100 veces más altos que la dosis prevista se inyectarían a un paciente, dándoles envenenamiento por radiación. Esto sucedió en 6 ocasiones conocidas, causando la muerte posterior de 4 pacientes.

**World of Warcraft “Corrupted-Blood” Glitch**

**Septiembre 13, 2005**

****

Fig.2 World of Warcraft

El exitoso World of Warcraft (WoW), un juego de ordenador en línea creado por Blizzard Entertainment, sufrió un fallo embarazoso tras una actualización de su juego en septiembre13, 2005, causando la muerte masiva (ficticia). Después de una actualización del contenido del juego, se introdujo un nuevo personaje   
enemigo, Hakkar, que tenía la capacidad de infligir una enfermedad, llamada Sangre Corrupta, sobre los personajes que jugarían que drenarían su salud durante un período de tiempo. Esta enfermedad podía pasar de un jugador a otro, al igual que en el mundo real, y tenía el potencial de matar a cualquier personaje contratarlo. Este efecto estaba destinado a ser estrictamente localizado en el área del juego que Hakkarhabitó. Sin embargo, una cosa fue pasada por alto: los jugadores fueron capaces de teletransportarse a otras áreas del juego mientras todavía estaban infectados y transmitir la enfermedad a otros , que es exactamente lo que sucedió. No puedo encontrar ninguna figura en el conteo de cuerpos, pero ciudades enteras dentro del mundo del juego eran áreas sin salida, con cadáveres de jugadores muertos llenando las calles. Afortunadamente, la muerte del jugador no es permanente en WoW y  
el evento pronto terminó cuando los administradores del juego restablecen los servidores y aplicaron más actualizaciones de software. Particularmente interesante es la forma en que las reacciones de los jugadores en el juego podrían reflejar sus reacciones a un incidente similar en la vida real.

**North American Blackout**

**Agosto 14, 2003**

****Fig.3 Gente caminando a través del puente Queensboro en la ciudad de Nueva York durante el apagón

Afectando a alrededor de 55 millones de personas, principalmente en el noreste de los Estados Unidos, pero también Ontario Canadá, este fue uno de los mayores apagones de energía en la historia. Empezó cuando una planta de energía a lo largo de la orilla sur del lago Erie, Ohio se desconectó debido a la alta demanda que puso al resto de la red de energía bajo mayor estrés. Cuando las líneas eléctricas están bajo una carga eléctrica más pesada, se calientan, lo que significa que el material que compone el cable (generalmente aluminio y acero) se expande. Varias líneas de energía colgaban más abajo a medida que se expandían y capturaban árboles, derribándolos y poniendo el sistema menos de presión. Esto condujo a un efecto en cascada que finalmente redujo la red de energía a un 20% de la salida normal.  Mientras que las causas de este apagón no tenían nada que ver con un error de software, podría haber sido evitado si no fuera por un error de software en el sistema de alarma del centro de control. En lo que se llama un escenario de "condición de carrera", dos partes del sistema competían sobre el mismo recurso y no pudieron resolver el conflicto, lo que provocó que el sistema de alarma se congelara y procesamiento. Desafortunadamente, el sistema de alarma falló 'en silencio', lo que significa que se rompió, pero no notificó a nadie que se había roto. Esto significaba que no se proporcionaban alertas de audio o visuales al personal sala de control, que dependía de esas cosas para la conciencia situacional. Las secuelas fueron bien reportado sin energía y dejó muchas áreas sin energía durante varios días y afectó a la industria, los servicios públicos, la comunicación. También fue culpado como al menos un factor que contribuyó a varias muertes.

**Incidente del USS Yorktown**

**Septiembre 21, 1997**

****

Fig.4 USS Yorktown (CG-48)

En el mundo del desarrollo de software, hay varios errores comúnmente conocidos que los programadores encuentro y tienen que atender. Un ejemplo de ello es el error "divide por cero", donde se realiza un cálculo que divide cualquier número por cero. Tal cálculo no es posible resolver, al menos  
no sin usar matemáticas más altas, y la mayoría de software – para todo, desde supercomputadoras a calculadoras de bolsillo – se escribe para tener en cuenta este escenario. Fue con cierta vergüenza, entonces, que el USS Yorktown sufrió un fallo total de su sistema de propulsión y estuvo muerto  
el agua durante casi 3 horas cuando un miembro de la tripulación escribió un "0" en la base de datos a bordo sistema de gestión que luego se utilizó en un cálculo de división. El software se instaló como parte de una operación más amplia para utilizar computadoras para reducir la energía del hombre necesaria para ejecutar algunas naves. Afortunadamente, la nave estaba realizando maniobras en el momento del incidente, en lugar de desplegarse  
en un entorno de combate, lo que podría haber tenido consecuencias más graves.

**Cold War Missile Crisis**

**Septiembre 26, 1983**

****

Fig.5 Stanislav Petrov recibió el Premio Dresde, 2013.

Stanislav Petrov era el oficial de servicio de un búnker secreto cerca de Moscú responsable de la vigilancia el sistema de satélites soviéticos de alerta temprana. Justo después de la medianoche, recibieron una alerta de que Estados Unidos había lanzado cinco misiles balísticos intercontinentales Minuteman. Como parte de la doctrina destrucción mutuamente asegurada que entró en prevalencia durante la Guerra Fría, la respuesta a un ataque   
un poder sería un ataque de venganza por parte del otro. Esto significaba que si el ataque era genuino, necesario responder rápidamente. Sin embargo, parecía extraño que Estados Unidos atacara con sólo un puñado de ojivas: aunque causarían daños masivos y pérdidas de vidas, ni siquiera sería casi   
suficiente para acabar con la oposición soviética. Además, las estaciones de radar en tierra no estaban captando ningún contacto, aunque estos no podían detectar más allá del horizonte debido a la curvatura de Tierra, lo que podría haber explicado el retraso. Otra consideración fue el sistema de alerta temprana sí mismo, que se sabía que tenía defectos y había sido puesto en servicio en primer lugar. Petrov pesar todos todos estos factores y decidió gobernar la alerta como una falsa alarma. Aunque Petrov no tenía su dedo en el botón de la bomba nuclear como tal, si hubiera transmitido una recomendación a sus superiores de que ellos toman el ataque como real, podría haber llevado a una guerra nuclear total. Ya sea basado en la experiencia, intuición, o simplemente la suerte, la decisión de Petrov fue la correcta. Más tarde se determinó que el software de detección temprana  
había recogido el reflejo del sol de la parte superior de las nubes y lo malinterpretó como lanzamientos de misiles.

**PROTECCIÓN CONTRA COPIA MALICIOSA DE CD DE SONY.**

La guerra aparentemente interminable entre los medios de comunicación y los piratas fluye y fluye cada año. Tan pronto como se encuentran nuevas formas de proteger y distribuir de forma segura los medios, se descubren nuevas formas de eludir y comprometer estas medidas. Algunos argumentarían que Sony BGM fue un paso demasiado lejos en 2005, cuando introdujeron una nueva forma de protección contra copias en algunos de sus CDs de audio. Cuando jugado utilizando un ordenador Windows, estos CDs instalarían una pieza de software llamada un 'rootkit'. A rootkit es una forma de software que se entierra profundamente en un ordenador y altera ciertos procesos fundamentales. Aunque no siempre malicioso en la naturaleza, un rootkit se utiliza a menudo para plantar sigilosamente malicioso  
y difícil de detectar (o eliminar) software, tales como virus, troyanos, etc. En el caso de Sony BMG, el objetivo era controlar la forma en que un ordenador Windows utilizaba los CDs de Sony para evitar copiarlos o   
convertirlos a MP3, lo que les ayudaría a reducir la piratería de sus medios. El rootkit logrado esto – pero al tomar medidas para ocultarse del usuario, permitió que los virus y otro software malicioso se ocultaran junto con él. La implementación mal pensada, y una creciente percepción de que Sony BMG no tenía ningún negocio manipulando a los usuarios de forma furtiva, significaba que todo el scheme resultó contraproducente. Resultó en el rootkit ser clasificado como malware por muchos seguridad informática, así como varios trajes de ley y una retirada del producto de los CDs infractores.

**Patriot Missile Bug**

**Febrero 25, 1991**



Fig.6 Lanzamiento de misiles Patriot

Durante la Operación Escudo del Desierto, el ejército estadounidense desplegó el Sistema de Misiles Patriot como defensa contra aviones y misiles, en este caso misiles iraquíes Al Hussein (SCUD). El software de rastreo para  
el misil Patriot utiliza la velocidad de su objetivo y el momento actual para predecir dónde estará el objetivo será de un instante a otro. Dado que varios objetivos pueden viajar a velocidades de hasta MACH 5, estos cálculos deben ser muy precisos. En ese momento, había un error en el software de apuntamiento - lo que significaba que con el tiempo, el reloj interno se 'desviaría' (al igual que cualquier reloj) más y vez el tiempo preciso que el sistema se dejaba en funcionamiento. El error ya era realmente conocido  
acerca y simplemente se solucionó reiniciando regularmente el sistema, y así restablecer el sistema clock. Desafortunadamente, los responsables no entendían claramente cómo "regularmente" debían reiniciar  
el sistema, y se dejó funcionando durante 100 horas. Cuando un misil iraquí fue lanzado, apuntando a un aeródromo EEE.UU. en Dhahran, Arabia Saudita, fue detectado por el sistema de misiles Patriot. Sin embargo, por este punto, el reloj interno se había desviado en 0,34 de segundo, por lo que cuando trató de calcular dónde el misil sería el siguiente, estaba mirando un área del cielo a más de medio kilómetro de distancia de los misiles ubicación verdadera. Rápidamente asumió que no había ningún misil enemigo después de todo y canceló la interceptación. El misil se llevó a su destino donde mató a 28 soldados e hirió a otros98.

**CONCLUCIONES**

Como era de esperarse este documento es solo una prueba mas de la importancia que tiene la buena calidad, el buen majo, el completo conocimiento de un Software antes de emplearlo o llevarlo acabo, ya que como en los casos anteriores se muestra las consecuencias pueden llegar a ser de muy altas perdidas. Me sigue sorprendiendo como de cosas que un humano crear pueden llegar a desatarse tales consecuencias, que solo basta el mínimo error para desplomar.

**RECOMENDACIONES**

En lo personal me gustaría recomendar el ampliar un poco mas el área de prueba de este tipo de Software, aun que ya en la actualidad sea menos probable que se presenten este tipo de errores, sigue siendo posible que pueda llegar a pasar un error como estos nadie nunca esta exentó a este tipo de consecuencias que pueden llegara a ser incluso más devastadores que las mencionadas en los casos anteriores. Nunca está de mas el garantizar/comprobar el buen y completo funcionamiento de herramientas con tal importancia.

**REFERENCIAS**

Jones Andrew- Seriously Epic Computer Software Bugs-24 DIC 2012

URL: http://listverse.com/2012/12/24/10-seriously-epic-computer-software-bugs/