Seguridad y Competencias Profesionales
Tema 5: Seguridad de los programas
Curso 2012–2013

Ingeniería en Informática (2° ciclo)

Departamento de Ingeniería Informática
Universidad de Cádiz

Cádiz, 29 octubre 2012

Índice

Objetivos

El alumno debe ser capaz de:

Conocimiento

Conocer el proceso a través del cual un programa no malicioso falla, y algunos de los tipos más comunes de vulnerabilidades. Enumerar los distintos códigos maliciosos y definir las distintas fases de actuación de los virus.

Comprensión

Prever el software malicioso, y tomar conciencia de la importancia de pensar en aspectos de seguridad al desarrollar el software.

Aplicación

Resolver los problemas generados por el código malicioso y mal escrito e implementar medidas contra ellos.

Bibliografía I


Anónimo.
Maximum Security.
SAMS, 4ª ed., 2003.


Gómez Vieites, A.
Enciclopedia de la Seguridad Informática.
RA-MA, 2006.


Pfleeger Ch. P. & Pfleeger S. L.
Security in Computing.
Prentice Hall PTR, 3ª ed., 2003.


Stallings, W.
Fundamentos de Seguridad en Redes. Aplicaciones y estándares.
Pearson, 2ª ed., 2004.

Bibliografía II

Stallings, W.
Cryptography and Network Security: Principles and Practice.
Pearson, 5ª ed., 2011.

Otros recursos I

Sitios web sobre virus y botnets

- http://alerta-antivirus.red.es
- http://www.virustotal.com
- http://www.f-secure.com/weblog/
- http://www.f-secure.com/en_EMEA/security/security-lab/
- http://ddanchev.blogspot.com/
- http://www.research.ibm.com/antivirus/
- http://www.securelist.com/
- http://vmyths.com/

Otros recursos II

Sitios web sobre vulnerabilidades

- http://cve.mitre.org
- http://www.kb.cert.org/vuls
- http://www.phrack.org

Introducción (I)

Tipos de amenazas

- Fortuita
- Intencionada:
  - Con intervención directa
  - Automática por código dañino

¿Por qué código dañino?

- Impunidad
- Impacto
- Publicidad
- Beneficio económico

Introducción (II)

Características de los ataques

- Indiscriminados
- Masivos
- Ciertos SO
- Ciertas aplicaciones
- Ciertos tipos de usuarios

Problema social

- Mal visto publicar y distribuir información sobre virus
- Falta de formación por parte de los profesionales

¿Qué es un programa seguro?

Concepto de programa seguro

Aquel que consigue confidencialidad, disponibilidad e integridad.

Valoración de un programa

- Estudio del diseño del programa
- Número, naturaleza y tiempo de vida de las vulnerabilidades y defectos

Notas sobre terminología

Defecto	Código que está "mal"
Error	Estado interno no válido
Fallo	Efecto visible (el programa "no va")

Programas erróneos no maliciosos

Un programa legal puede causar daños

Razones

- Mala programación
- Deficiente organización de los recursos
- No comprobación de funcionamiento

Efectos

- Daños sobre datos, software o hardware
- Facilitar las intrusiones (*vulnerabilidades*)
- Fuga de información

Ejemplo: desbordamientos de búfer (I)

Concepto

- La aplicación usa una zona de memoria limitada contigua para almacenar información que lee del usuario
- No comprueba que la información que da el usuario cabe en dicha área, sobreescribiendo otras partes de memoria

Consecuencias

- En el mejor caso, el programa se cierra
- En el peor caso, el atacante consigue ejecutar código arbitrario con los privilegios del programa

Ejemplo: desbordamientos de búfer (II)

Medidas

- Concienciación y diseño y pruebas exhaustivos
- Entre otras técnicas de prueba, *fuzz testing*:
  - Técnica de prueba de software de caja negra que consiste en encontrar *bugs* automáticamente, inyectando datos semialeatorios a un programa.
  - Permite encontrar fallos de implementación software y, si es posible, identificarlos.
  - `http://pages.cs.wisc.edu/~bart/fuzz/`
    `http://video.google.com/videoplay?docid=6509883355867972121`
  - *Fuzzers* (programas): VolPer, KiF, WSFuzzer...

Ejemplo desarrollado

- Phrack.org: "Smashing the Stack for Fun and Profit"
- `http://www.phrack.org/issues.html?id=14&issue=49`

Ejemplo: desbordamientos de búfer (III)

Otro ejemplo

```
/* Escapes all newlines in the input string,
   replacing them with "\n".*/
/* Requires: p != NULL; p is a valid '\0'-terminated string */

void escape(char *p)
{
  while (*p != '\0')
    switch (*p)
    {
      case '\n':
        memcpy(p+2, p+1, strlen(p));
        *p++ = '\\'; *p++ = 'n';
        break;
      default:
        p++;
    }
}
```

Fuente: Stallings, W. *Cryptography and Network Security: Principles and Practice*

Ejemplos recientes de vulnerabilidades

Provenientes de los desarrolladores

- Productos de Mozilla: `http://www.mozilla.org/security/known-vulnerabilities`
- Microsoft Windows Vista: `http://seclists.org/fulldisclosure/2009/Sep/39`

Provenientes de los distribuidores

- OpenSSL y Debian: `http://www.links.org/?p=327`

Catálogo de tipos conocidos

CWE: `http://cwe.mitre.org`

Ejercicios

Ejercicio 5.1

En grupos, localizar en el CWE 4 tipos de vulnerabilidad al azar, identificando:

- Resumen de en qué consiste
- Cómo se podría mitigar o evitar
- Al menos 1 programa comercial que la sufra o la haya sufrido

Código malicioso (I)

Algunos tipos

- Puertas traseras
- Bombas lógicas
- Caballos de Troya
- Bacterias o conejos
- *Malscript* en webs infectadas o vulnerables (XSS, CSRF)
- Virus

Soporte

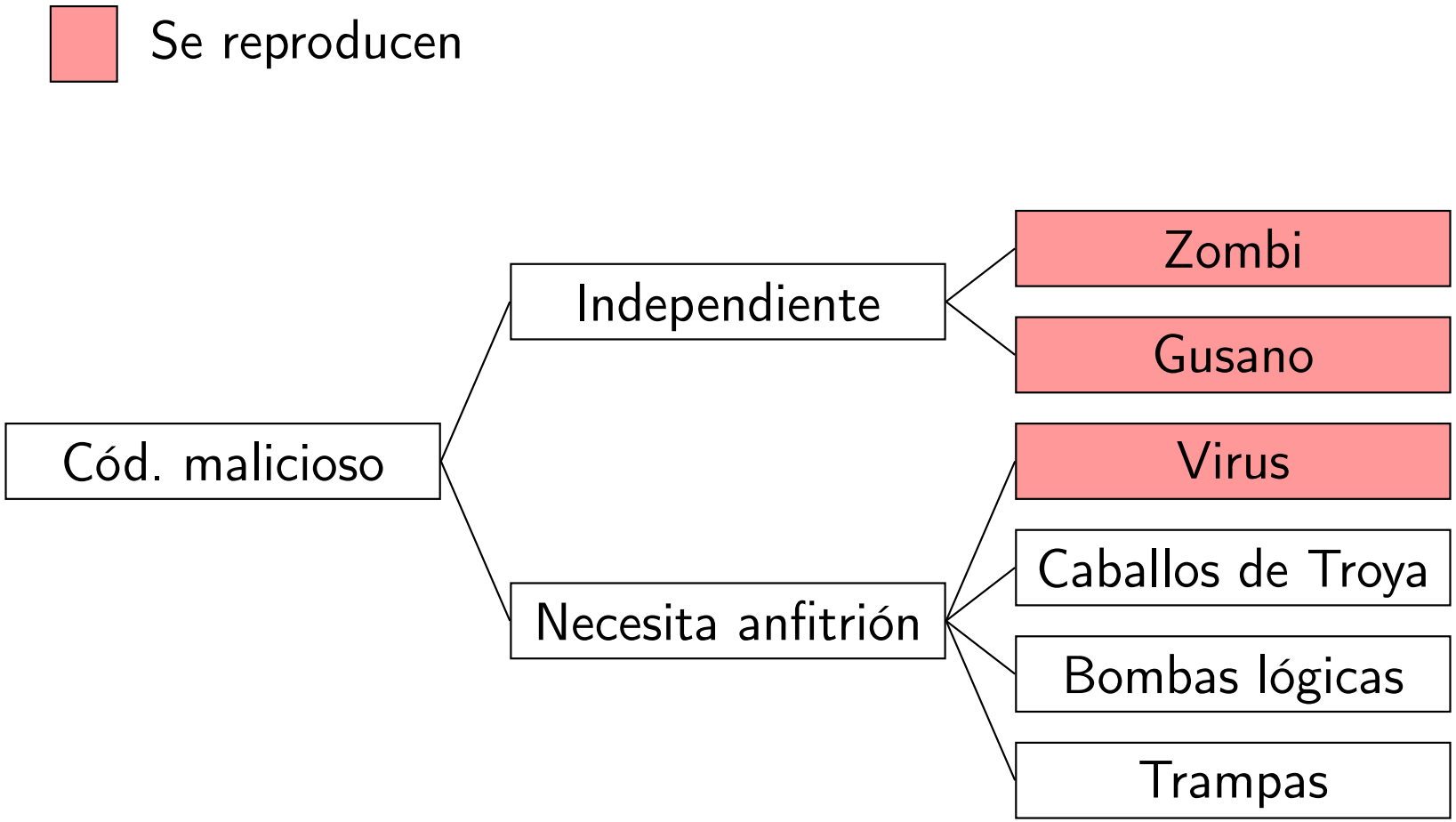
Pueden (o no) necesitar un programa anfitrión.

Reproducción

Pueden (o no) reproducirse.

Código malicioso (II)

Taxonomía parcial del código malicioso



Trampas o puertas traseras

Concepto

Entrada secreta a un programa o sistema informático sin pasar por los procedimientos establecidos.

Activación

- Secuencias especiales de entrada
- Se activa desde un usuario en particular
- Secuencia improbable de acontecimientos

Medidas

- Desarrollo de programas: código abierto, desarrollo colaborativo, etc.
- Actualización del software

Bombas lógicas

Concepto

Programas que ejecutan órdenes destructivas al producirse una condición en el sistema.

Algunas condiciones

- Una determinada fecha
- Un determinado valor en un registro
- Una petición de interrupción

Usos

- Controlar a los clientes en sus pagos
- Programas gratuitos durante un tiempo
- Venganzas

Caballos de Troya

Concepto

Programa o procedimiento de órdenes aparentemente útil, con una funcionalidad visible al usuario (incluso como antivirus) y otra oculta y dañina.

Usos

- Llevar a cabo funciones a usuarios no autorizados
- Destrucción de datos

Camaleones

Concepto

Sustituyen a la pantalla de entrada tradicional en sistemas multiusuario (GDM y KDM en GNU/Linux), interceptando sus credenciales.

Usos

- Robo de credenciales

Ejercicios

Ejercicio 5.2

En grupos, localizar dos ejemplos de cada uno de estos tipos de software malicioso, explicando de forma resumida su funcionamiento:

- Trampa
- Bomba lógica
- Caballo de Troya
- Camaleón

Bacterias o conejos

Concepto

Programa que se reproduce indefinidamente.

Funcionamiento

1. Se copia a sí mismo una y otra vez
2. Ocupa toda la memoria principal y/o secundaria y causa denegación de servicio

Casos conocidos y medidas

- *Fork-bombs*: `:(){:|:&};:` (¡cuidado!)
- Como videojuego: Core War (`http://en.wikipedia.org/wiki/Core_War`)
- Medida: imponer límites de uso máximo de recursos

Zombi o Bot

La *botnet* Conficker tenía cerca de 10 millones de máquinas

Concepto

Programa que obedece a un centro de control remoto, como parte de una *botnet*. Éstas suelen ser controladas por organizaciones criminales y vendidas a atacantes.

Algunos usos

- Ataques de denegación de servicio
- Correo no solicitado (*spam*)
- Intermediarios ante servidor web con contenido malicioso (*fast-flux*)

Medidas

- Pasivas: usar sólo software fiable y actualizado
- Activas: *honeynets*

XSS: *Cross-Site Scripting* (I)

Concepto

Código ejecutable embebido en una página web por un atacante, que es ejecutado en la máquina de un cliente que visita dicho sitio.

XSS no persistente

- No se almacena en el sitio web
- El código aparece debido a una petición especialmente construida (enlace proporcionado en correo, p.ej.)

XSS persistente

- El código se almacena en la web, y se ejecuta a visitantes normales
- Se puede decir que la web ha sido "infectada"

XSS: *Cross-Site Scripting* (II)

Usos

- Robo de *cookies* (contraseñas, sesiones, etc.)
- Modificación de los contenidos mostrados
- Dar confianza al usuario para que permita otros ataques

Medidas en el servidor

- Filtrado y validación de la entrada recibida
- Uso de formatos de entrada menos potentes (*Markdown*)

Medidas en el cliente

- Utilizar efectivamente los mecanismos de "zonas de seguridad" (Internet Explorer)
- Desactivar JavaScript por completo o de forma selectiva (Firefox: NoScript)

CSRF: *Cross-Site Request Forgery*

Concepto

- El usuario ha iniciado sesión en el servicio que queremos atacar
- A través de contenido HTML específico en una web propia o vulnerable, aprovechamos su sesión para realizar otras acciones

Algunas medidas más

- Comprobar la cabecera HTTP *Referrer* (falseable)
- Hacer que se envíen identificadores secretos específicos a cada usuario para cada petición legítima (solución adoptada en algunos *frameworks* web, como Django)
- Autenticar no las sesiones, sino las transacciones

Virus

Concepto

Código ejecutable que infecta otros programas para propagarse, y que al activarse realiza algún tipo de acción no autorizada (normalmente dañina).

Basado en

- Troyanos y bombas lógicas
- Técnica de reproducción de bacterias

Objetivo: su difusión

- Afectan a los sistemas más populares y vulnerables
- Son de acción retardada, para mejorar su propagación

Tipos de virus (I)

Compañía

- Crea un fichero con igual nombre pero con una extensión de mayor prioridad (como `f.com` frente a `f.exe`)
- Comunes en MS-DOS, aún factibles en ciertos entornos
- Intenta que se ejecute este fichero en vez del original
- Es fácil de eliminar borrando dicho fichero dañino

Correo electrónico

- Se propaga al abrir el correo, sin leer el fichero adjunto
- El virus se envía a todas las direcciones de la agenda
- Puede producir daños locales

Tipos de virus (II)

Macros

- Se introduce el código en una macro asociada a un documento
- Escritos en lenguaje de alto nivel: hay virus de macros multiplataforma
- Es fácil detectarlo y eliminarlo

Proceso de infección

1. El fichero llega al sistema (correo, pendrive, etc.)
2. Usuario abre el documento
3. Usuario desencadena la acción que activa la macro
4. Macro puede infectar plantillas (`Normal.dot` en Word)

Tipos de virus (III)

Gusanos

- Aprovechan vulnerabilidades en servicios de red para propagarse
- Después se pueden comportar como virus, troyano, *spyware*, etc.

Características

- Fueron los primeros en burlar las protecciones de forma automática, borrar su rastro
- Su objetivo no es destruir pero pueden hacer mucho daño
- Su construcción exige un buen conocimiento de los sistemas en red

Tipos de virus (IV)

Fases de actuación de un gusano

1. Buscar un fallo e instalar el programa de penetración
2. Cargar el resto del gusano
3. Buscar vías de comunicación y explorar el sistema para buscar información
4. Extraer pistas y claves para propagarse a otros sistemas

Tipos de acciones de un virus (I)

Contaminadores del *Master Boot Record*

1. Localizar sectores libres en disco para almacenar MBR
2. Leer el MBR y guardar el contenido en dichos sectores
3. Marcar estos sectores como erróneos
4. Grabar en MBR el lanzador, que llama al antiguo MBR

Contaminadores de la BIOS

- Prácticamente indetectables, aunque normalmente retirables con algún ajuste de la placa base
- Corrompe BIOS: `http://www.f-secure.com/v-descs/emperor.shtml`
- Infecta BIOS: `http://www.coresecurity.com/files/attachments/Persistent_BIOS_Infection_CanSecWest09.pdf`

Tipos de acciones de un virus (II)

Contaminadores de ficheros de órdenes

Infectan `command.com`, `/bin/sh`, guiones de procesamiento de lotes, etc.

Contaminadores de ficheros ejecutables

Se adaptan bien a cualquier formato de fichero. Pueden afectar a sistemas multiusuario y a sus copias de seguridad.

Multipropósito

Mezclan los anteriores vectores de ataque.

Llegada e instalación

El virus llega al sistema

Vías de entrada

Fichero, correo electrónico, gusano, MBR, macro, etc.

Instalación en algún punto del ejecutable original

Añadidura	al final
Inserción	en zonas no utilizadas o segmentos de datos
Reordenación	para que invoque a una rutina en otro sitio
Sustitución	del ejecutable por el virus

Activación

El virus se activa por primera vez

Paso a período de latencia

Tras activarse, estará en período de latencia hasta que se manifieste

Tipos

Acción directa	Al iniciar la ejecución de un ejecutable infectado
Residentes	Se mantienen en memoria, interceptando acciones del sistema, y pueden sobrevivir a reinicios (los falsean)

Ocultamiento

El virus evita ser descubierto y eliminado

Algunas técnicas

Dispersión	Deja sólo una pequeña parte en el huésped
Compresión	Comprime el fichero y lo sitúa en el espacio libre
Camuflaje	Oculta los síntomas del virus
Sobrepasamiento	Evita las vacunas
Autocifrado	Se cifra, evitando búsquedas heurísticas
Polimorfismo	Cambia de forma al propagarse, suele darse en autocifrados
Blindaje	Dificulta su desensamblado

Reproducción o propagación

Objetivos

- Propagación a otros sistemas
- Duplicar esfuerzos en la consecución de algún fin

Pasos

1. Búsqueda de uno o varios programas huésped sanos
2. Recomposición de las partes del programa
3. Infección del huésped

Manifestación

Acciones realizables

- Formatear discos
- Borrar o sustituir ficheros
- Atacar elementos programables de hardware
- Borrar o dañar la información del directorio de disco
- Atacar la confidencialidad del sistema
- Instalación de puertas traseras o *rootkits*

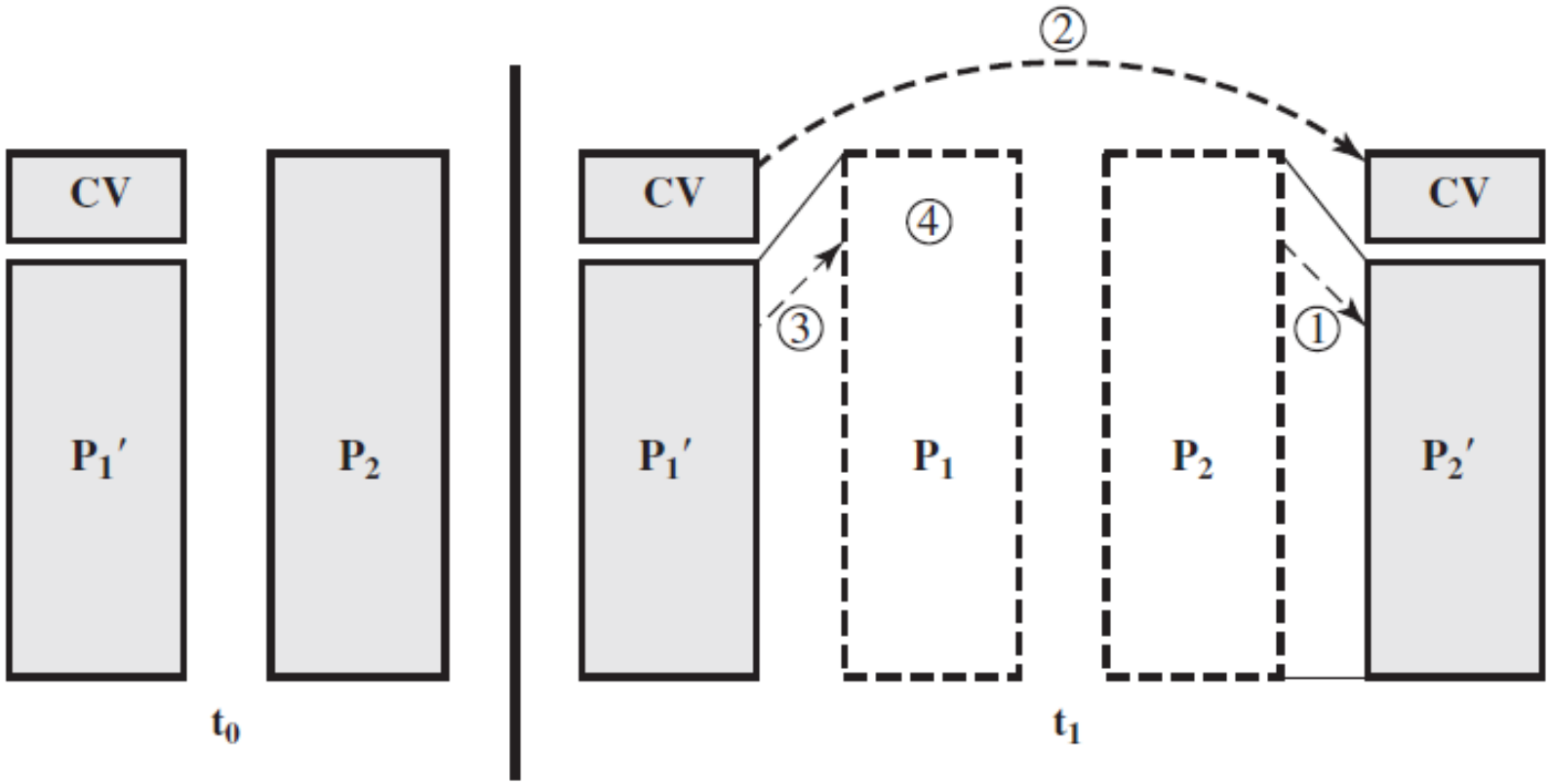
Ejemplo de virus (I)

Estructura

```
program V :=
{ goto main;
  1234567;

  subroutine infect-executable:=
    {loop:
    file := get-random-executable-file;
    if (first-line-of-file = 1234567)
      then goto loop
      else prepend V to file; }

  subroutine do-damage :=
    {whatever damage is to be done}

  subroutine trigger-pulled :=
    {return true if some condition holds}

  main: main-program :=
    {infect-executable;
    if trigger-pulled then do-damage;
    goto next;}
next:
}
```

Ejemplo de virus (II)

Compresión de virus



Fuente: Stallings, W. *Cryptography and Network Security: Principles and Practice*

Ejemplo de virus (III)

Compresión de virus

```
program CV :=
{
  goto main;
  01234567;

  subroutine infect-executable :=
    {loop:
      file := get-random-executable-file;
      if (first-line-of-file = 01234567) then goto loop;
      (1) comprimir fichero;
      (2) añadir CV al fichero;
    }

  main: main-program :=
    {
      if ask-permission then infect-executable;
      (3) descomprimir el resto del fichero;
      (4) ejecutar el fichero descomprimido;
    }
}
```

Utilidades contra el software malicioso (I)

Comprobadores de integridad

- Verifican si algún fichero ha sido cambiado inadvertidamente
- No buscan virus, sólo huellas de ataques

Vacunas

- Muchos antivirus incluyen una
- Residentes en memoria, parecidas a un virus
- Interceptan llamadas y detectan la presencia de virus
- Pueden producir falsos negativos y positivos

Utilidades contra el software malicioso (II)

Antivirus

- Rastrean un soporte, localizando y eliminando virus
- Existen tanto para uso personal como para servidores de correo (ClamAV)

Método de desarrollo

- Conocer la existencia de un nuevo virus y aislarlo.
- Localizar una secuencia de código característica.
- Generar una rutina que lo detecte.

Prevención

Medidas

- Realizar periódicamente copias de seguridad
- No usar copias piratas de programas
- Activar los dispositivos de protección física en los discos
- Trabajar con privilegios de usuario normal
- No arrancar con discos no originales
- Activar las medidas de seguridad de las aplicaciones macro
- Conocer los nuevos códigos malignos
- Tener antivirus
- Difundir la necesidad de protección contra *malware*

Diagnóstico

Síntomas

- El ordenador funciona lento y se bloquea
- Algunos programas no pueden ejecutarse
- Aumento de los sectores ocultos y menos RAM libre
- Cambios en los atributos de los ficheros y aparición de ficheros con el mismo nombre
- Excesiva actividad en los discos
- Sistema de arranque cambiado y aviso del antivirus
- *Word* nos pide guardar cambios que no hemos realizado
- Puertos TCP/IP a la escucha cuyo servicio no está claro

Desinfección

Pasos

1. Apagar el ordenador
2. Arrancarlo con una copia limpia
3. Utilizar una copia limpia de antivirus y ejecutarlos uno a uno para detectar el tipo
4. Limpiar todos los ficheros del disco duro
5. Limpiar todos los soportes externos
6. Restaurar programas, mejor a partir de copias de seguridad
7. Avisar del tipo de virus a otros usuarios que hayan tenido relación con nosotros

Ejercicios

Ejercicio 5.3

En grupos, localizar en el Security Lab de F-Secure información sobre 4 virus, identificando:

- Vía de entrada
- Mecanismos de activación
- Mecanismos de propagación
- Técnicas de ocultación
- Acciones que realiza

Conclusiones

- El código malicioso puede producir graves daños en nuestros recursos informáticos
- El código mal escrito facilita el trabajo de los desarrolladores de código malicioso, o produce daños
- Debemos tomar medidas preventivas y saber qué hacer en cada caso
- Tener en cuenta que cada vez hay más aplicaciones transparentes al usuario
- El intercambio de información directa o vía *web* favorece este mal
- Las medidas no suponen un gasto elevado