Ingeniería Inversa
Clase 4

Análisis de Malware

martin.uy
Open by default.

Análisis de Malware

ADVERTENCIA

Ejecutar malware real durante el análisis dinámico tiene riesgos. Se recomienda un ambiente completamente aislado.

En el caso de usar un hipervisor, utilizar la versión más actualizada y agregar la menor cantidad posible de periféricos (ej. evitar aceleración 3D).

Análisis de Malware

- Se van a ver algunas técnicas genéricas empleadas por malware binario durante esta clase, desde las perspectivas del análisis y del desarrollo
  - Cada caso es diferente (propósito y objetivo)
  - No todo malware es binario
  - Es una disciplina en constante evolución: nuevas APIs, nueva funcionalidad, nuevos anti-virus (AV) y heurísticas de detección

Análisis de Malware

- Componentes
  - Agente
    - Ejecuta las operaciones en el target (endpoint) atacado: cómputo, robo de datos, secuestro de datos, etc.
    - Puede tener diferentes niveles de autonomía
      - Responder a comandos ejecutados de forma interactiva (agente más liviano y genérico)
      - Actuar de forma autónoma utilizando técnicas más sofisticadas de exfiltración de datos (agentes inteligentes y específicos)
        - Evitar “ruido” en el endpoint y en la red

Análisis de Malware

- Componentes
  - Agente
    - Es deseable que sea de tamaño reducido y Position-Independent-Code, para ejecutar inclusive inyectado en procesos (y acceder directamente a su memoria)
    - Expuesto a anti-virus y a dar pistas sobre el atacante
    - Si es binario, tiene que estar diseñado para cada plataforma (sistema operativo y arquitectura)

Análisis de Malware

- Componentes
  - Centro de comandos (C&C)
    - Centraliza los comandos e información de los diferentes agentes
    - No está expuesto a anti-virus
  - Canal de comunicaciones
    - Comunicaciones cifradas
    - Canales encubiertos
  - Se suele llamar “botnet” al conjunto de agentes y su C&C

Análisis de Malware

- Defensa
  - Prevenirlo es complejo: una organización necesita trabajar en la detección
  - Tradicionalmente las soluciones de defensa se enfocaban en analizar fingerprints (en el endpoint o en la red). Hoy es insuficiente
  - Es necesario el análisis dinámico o “user behavior”

Análisis de Malware

- ¿Cuáles son las ventajas y desventajas de un canal TCP/UDP/IP desde el C&C al agente?

Análisis de Malware

- TCP/UDP/IP hacia el agente
  - ¿Cuál es la dirección IP del agente? ¿Puede cambiar? (IP dinámica) ¿Está atrás de un NAT? ¿es una IP interna? ¿hay redirección de puertos?
  - Un firewall puede bloquear fácilmente una conexión TCP o un paquete UDP entrante, a menos que vaya a un puerto conocido de un host conocido
  - Un proceso tiene que estar escuchando en un puerto (alto, si el agente no consiguió elevar privilegios) y eso levanta sospechas

Análisis de Malware

- TCP/UDP/IP hacia el agente
  - El agente tiene que encargarse de cifrar los datos que exfiltra (evitar sospechas a un IDS)
  - Es un canal con buen ancho de banda para exfiltrar grandes cantidades de datos
  - El C&C tiene el control de cuándo se levanta y baja el canal. El agente no tiene que estar consultando para reconexiones (polling)

Análisis de Malware

- TCP/UDP/IP hacia el agente
  - Port knocking. Si el agente consiguió privilegios para leer los datos en crudo de la interfaz de red, puede sensar una clave (ejemplo: secuencia de SYN a diferentes puertos) para decidir escuchar al C&C en cierto puerto
    - Evitar “escuchar siempre” para mayor discreción
  - El C&C puede cambiar su IP (proteger al atacante)

Análisis de Malware

- ¿Cuáles son las ventajas y desventajas de un canal TCP/UDP/IP desde el agente al C&C?

Análisis de Malware

- TCP/UDP/IP hacia el C&C
  - ¿Cuál es la dirección IP del C&C? ¿Cómo la sabe el agente? ¿Hay que hacer una resolución de DNS?
    - Una IP fija del C&C puede dejar inutilizable rápidamente a un agente
    - Una resolución de DNS puede ser sospechosa (blacklisting de dominios)
    - El atacante puede perder el dominio y la botnet quedar inutilizable
    - No hay tanta flexibilidad para el C&C

Análisis de Malware

- TCP/UDP/IP hacia el C&C
  - Agentes sofisticados pueden buscar titulares del día en Internet y probar con un dominio basado en esas palabras
  - El atacante registra el dominio en el día y reestablece la conectividad

Análisis de Malware

- TCP/UDP/IP hacia el C&C
  - No hay problemas de NAT si el endpoint atacado tiene conectividad a Internet
  - El C&C puede utilizar un puerto conocido y generalmente permitido para conexiones salientes (ej. 80, 443)
    - Sin embargo, un firewall con deep inspection puede encontrar un protocolo inesperado y por lo tanto sospechoso

Análisis de Malware

- TCP/UDP/IP hacia el C&C
  - El agente no tiene por que estar escuchando siempre. Decide cuándo establecer la conexión -podría pollear siempre pero sería sospechoso-
  - El agente tiene que encargarse de cifrar los datos exfiltrados
  - Canal con buen ancho de banda

Análisis de Malware

- ¿Cuáles son las ventajas y desventajas de un canal HTTP/S, DNS, ICMP y otros canales encubiertos desde el agente al C&C?

Análisis de Malware

- HTTP/S, DNS, ICMP y otros canales encubiertos
  - Abuso del protocolo. Ej. Consultas DNS mal formadas
  - Suelen depender de resolver un dominio del atacante (o quedar atados a una IP fija)
  - Tienen las ventajas de ser iniciados como tráfico saliente y a discreción del agente
  - Suelen pasar los firewalls y ser menos sospechosos si están implementados emulando comportamiento humano (ej. cuidar la cadencia de las transferencias)
  - Tienen menos ancho de banda

Análisis de Malware

- Multi-staging / bootstrapping
  - Frecuentemente el malware se deploya en múltiples etapas
    - Script que descarga un binario
    - Payload de explotación (reducido en tamaño) que termina cargando un payload completo
    - El primer stage permite profilear el sistema. Ejemplo: infección previa, sistema operativo, arquitectura, etc. y tomar decisiones en relación a la obtención y carga de la siguiente etapa

Análisis de Malware

- Multi-staging / bootstrapping
  - Puede ser interesante para no dejar tantos rastros persistentes: sólo persiste el bootstrapper y no la lógica (¿de negocio?) del agente
  - Evitar un pasaje por el sistema de archivos para evitar un análisis de anti-virus
    - Los antivirus instalan hooks a nivel de kernel para detectar operaciones sobre el sistema de archivos y actuar en consecuencia

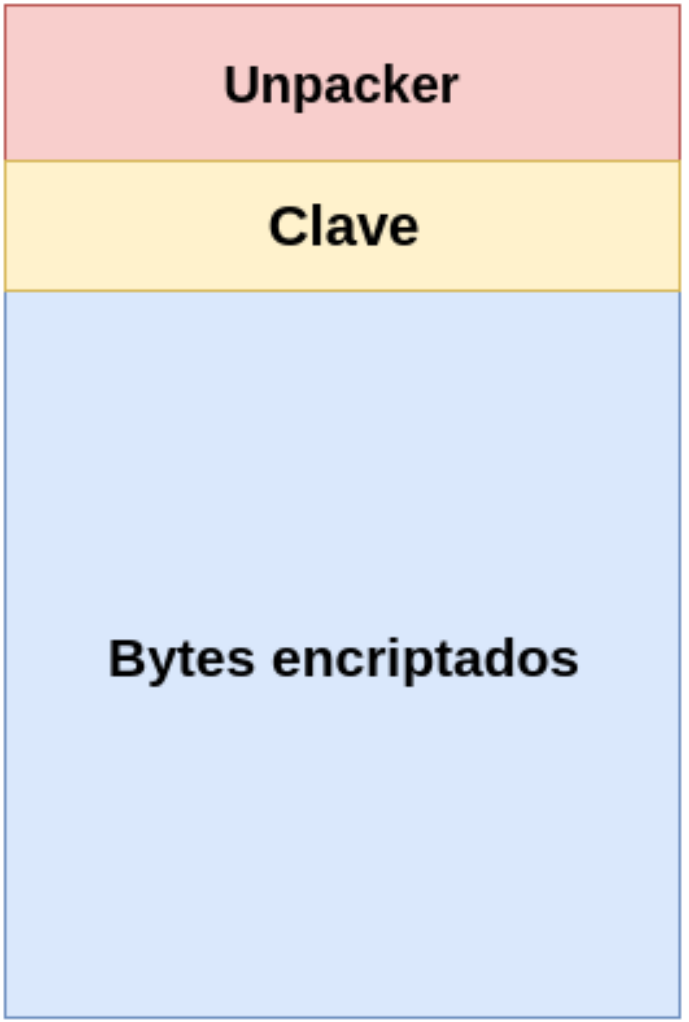
Análisis de Malware

- La forma más simple para empezar a entender un malware sería analizar los strings que contiene el binario
  - Por ejemplo: una URL al 2nd stage
  - `strings bin` (Linux)
  - El malware sofisticado no expone sus strings (datos, símbolos de funciones importadas, etc.) ni código en claro: los tiene cifrados u ofuscados para resistir las formas más básicas de análisis estático

Análisis de Malware

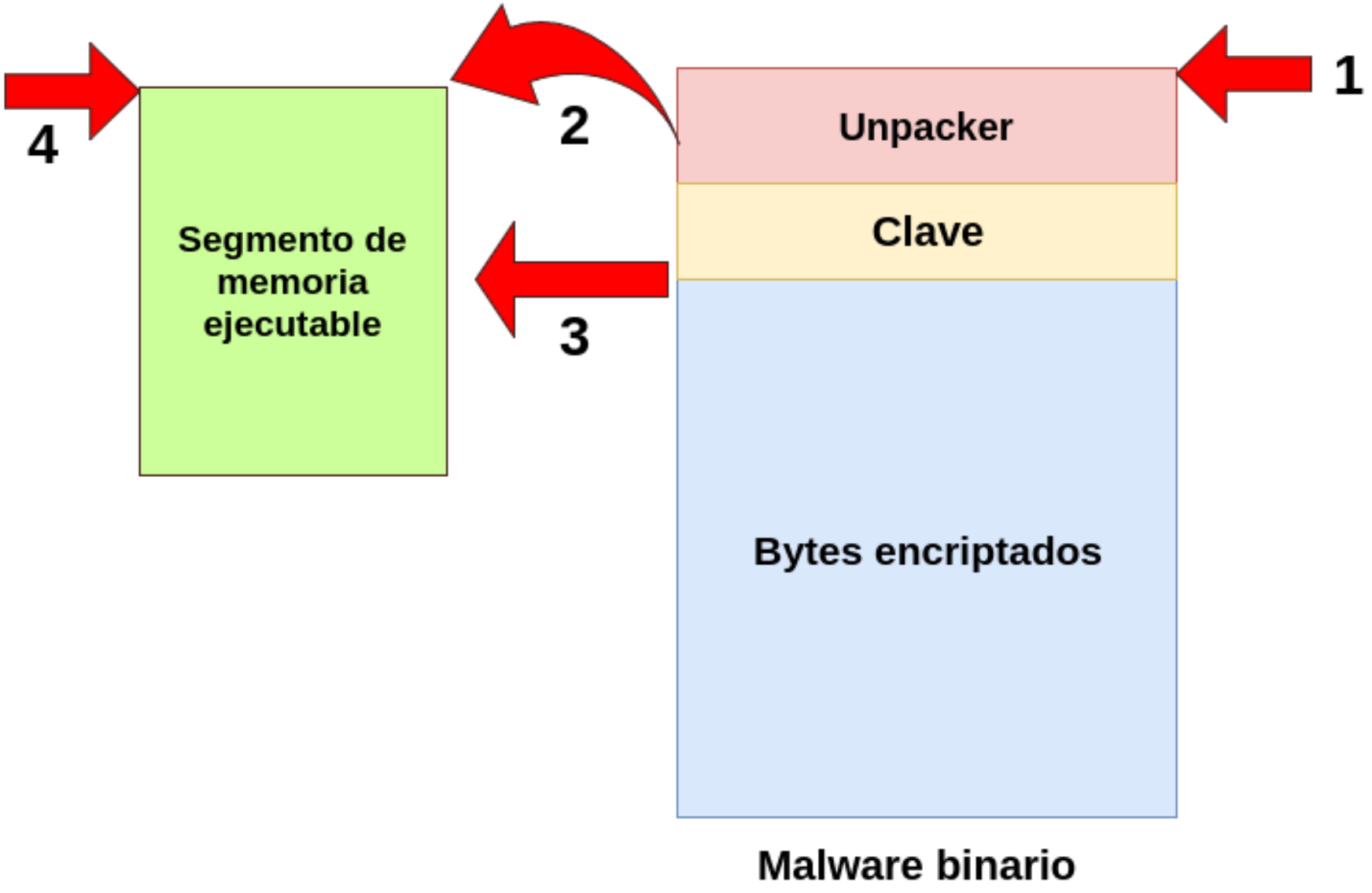
- Unpacking
  - Al comenzar la ejecución, el malware debe auto-desempacarse (bootstrapping)
    - Total o parcialmente
  - La ejecución comienza por el desempacador que utiliza una clave contenida en el binario u obtenida de alguna fuente
    - Criptografía trivial: solo para ofuscación y evitar el fingerprinting más básico de IDSs y anti-virus

Análisis de Malware



Malware binario

Análisis de Malware



Análisis de Malware

1. El unpacker empieza a ejecutarse (entry-point del binario)
2. El unpacker aloca un segmento de memoria ejecutable (permisos de escritura y, posteriormente, de ejecución)
3. El unpacker lee o recibe la clave y los bytes cifrados. Descifra y escribe los bytes en claro en el segmento de memoria previamente alocado
4. El unpacker salta a ejecutar los bytes descifrados

Análisis de Malware

- ¿No es sospechoso que un binario aloque memoria con permisos de escritura, escriba bytes, cambie los permisos de la memoria a ejecución y salte allí? ¿Cuál sería un caso de uso legítimo para que un proceso haga eso?

Análisis de Malware

- ¿No es sospechoso que un binario aloque memoria con permisos de escritura, escriba bytes, cambie los permisos de la memoria a ejecución y salte allí? ¿Cuál sería un caso de uso legítimo para que un proceso haga eso?

JIT compilers (OpenJDK, Flash, etc.) 

Análisis de Malware

- La operación de cifrado puede ser algo tan simple como un XOR con la clave o algo más avanzado
- Diferentes agentes pueden contener la misma clave o diferentes, según el nivel de sofisticación
- Puede desempacarse todo o irse desempacando de a poco, en función de lo necesario, para dificultar su dumpeo. La clave puede recibirse en tiempo de ejecución y no persistirse inclusive
- ¿Cuáles son los retos de desarrollar malware con estas restricciones? ¿Cómo hacerlo?

Análisis de Malware

- ¿Es fácil…
  - Desarrollar una máquina criptográfica?
  - Incluir un BLOB con bytes cifrados en el binario?
  - Alocar memoria con permisos de ejecución?
  - Descifrar los bytes, escribirlos en la memoria alocada y saltar a ejecutar?
  - Desarrollar el código a ser packeado como BLOB?

Análisis de Malware

- ¿Es fácil…
  - Desarrollar una máquina criptográfica?
  - Incluir un BLOB con bytes cifrados en el binario?
  - Alocar memoria con permisos de ejecución?
  - Descifrar los bytes, escribirlos en la memoria alocada y saltar a ejecutar?
  - Desarrollar el código a ser packeado como BLOB?

Análisis de Malware

- El código ejecutable (a ser packeado) debe ser autocontenido como si fuera shellcode
  - ¿Por qué?
    - No queremos pasar el código despackeado por el file system para evitar anti-virus. De lo contrario, podríamos tener un binario generado por el compilador y cargarlo normalmente
    - Por lo tanto, no hay “loader” que resuelva automáticamente librerías externas ni otras tareas
    - Cargar un ELF o PE completo es complejo de implementar manualmente y aumentaría el tamaño del malware

Análisis de Malware

- No queremos tener una IAT (Import Address Table) o una GOT (Global Offset Table) con las funciones que utiliza el malware porque revela fácilmente sus intenciones
- Podemos linkear estáticamente todas las funciones externas (glibc, en Linux) pero agrandaría el tamaño

Análisis de Malware

- Datos y código son parte de un mismo stream de bytes: no hay secciones
- El código es PIC (Position-Independent-Code)
  - Direccionamiento relativo al instruction pointer
  - Preparado para ejecutarse sea cual sea la dirección virtual en la que se aloque o despackee

Análisis de Malware

```
#include <stdio.h>
#include <sys/types.h>
#include <netinet/in.h>

extern int sockfd;

static int f ();

int main(){
    sockfd = socket(AF_INET, SOCK_STREAM, 0);

    int i = f();
    ...
}
```

Análisis de Malware

- Se puede:
  - Incluir headers y usar constantes
  - Utilizar variables locales
  - Utilizar funciones y variables estáticas
- No se puede:
  - Referenciar variables globales externas
  - Llamar funciones externas (linkear librerías dinámicas)

Análisis de Malware

- Si seguimos estas reglas y compilamos PIC, el segmento .text contiene los bytes de “shellcode” para cifrar
  - Podemos programar malware en C (en lugar de escribirlo en assembly) y packearlo fácilmente

Demo 4.1

Compilar shellcode y packear malware

Análisis de Malware

- Resolución de funciones
  - Llamar a la API del SO (en lugar de ejecutar syscalls directamente) tiene sus ventajas: abstracciones de alto nivel
    - ¿Cuántas syscalls necesitaríamos implementar para resolver un dominio, o hacer un request HTTP?
    - Podríamos simplemente llamar a las APIs de getaddrinfo y curl

Análisis de Malware

- Resolución de funciones
  - Necesitamos para eso resolver funciones
    - Linux
      - Si libdl.so está mapeado en el proceso:
      - resolver dlsym leyendo la memoria de la lib (en base a la estructura ELF y a la base address disponible en /proc/<PID>/maps)
      - Usar dlsym y dlopen para cargar librerías y resolver símbolos
      - Si el loader dinámico está en el proceso, podríamos utilizarlo para resolver funciones
      - Podemos implementar nuestro propio resolver en base a la estructura ELF y la base address de las libs del proceso

Demo 4.2

Resolver dlsym y, con dlsym, getaddrinfo

Análisis de Malware

- Resolución de funciones
  - Usar funciones de librerías
    - Windows
      - Pointer a TIB (fs:0x30) → PEB
      - Recorrer módulos (dlls) cargadas en el proceso hasta localizar kernel32.dll
      - Resolver GetProcAddress browseando la memoria (estructura PE)
      - Usar LoadLibrary para cargar librerías y GetProcAddress para resolver símbolos

Demo 4.3

Resolver kernel32.dll en Windows

Análisis de Malware

- PEB – Process Environment Block
  - Estructura no documentada, usada por ntdll
  - Parte de la información con la que se carga es provista por el kernel
  - Residente en espacio de usuario (por lo tanto, el proceso puede leerla)
  - Contiene información global del proceso: lista de módulos cargados (PPEB_LDR_DATA), si está siendo debuggeado, ID de sesión, parámetros, etc.

Análisis de Malware

```
typedef struct _PEB_LDR_DATA {
  BYTE       Reserved1[8];
  PVOID      Reserved2[3];
  LIST_ENTRY InMemoryOrderModuleList;
} PEB_LDR_DATA, *PPEB_LDR_DATA;

typedef struct _LDR_DATA_TABLE_ENTRY {
    ...
    PVOID DllBase;
    PVOID EntryPoint;
    PVOID Reserved3;
    UNICODE_STRING FullDllName;
    ...
} LDR_DATA_TABLE_ENTRY, *PLDR_DATA_TABLE_ENTRY;
```

Análisis de Malware

- Al malware no le gusta que lo debuggeen en un sandbox
  - IsDebuggerPresent (Windows API)
    - Campo “BeingDebugged” en PEB
  - ¿Quién es el proceso padre? ¿Hay un debugger (TracerPid) en /proc/<PID>/status? (Linux)
  - Consulta a los recursos físicos (RAM, disco, etc.). Un sistema con pocos recursos puede no ser real
  - Consulta a los drivers (Vendor IDs)
  - Uptime del sistema (GetTickCount Windows API)

Análisis de Malware

- Al malware no le gusta que lo debuggeen en un sandbox
  - Benchmarking: ¿cuánto lleva hacer una operación computacionalmente costosa?
  - ¿Hay una instrucción emulada? ¿Cuánto tiempo lleva su ejecución?
  - Para medir tiempo en x86/x86_64 se puede utilizar la instrucción no-privilegiada RDTSC (read timestamp)
    - Podemos usar un reloj aproximado con un thread paralelo decrementando una variable

Análisis de Malware

- Al malware no le gusta que lo debuggeen en un sandbox
  - Si el malware detecta que está siendo debuggeado, puede comportarse de manera no-sospechosa
- El malware puede permanecer latente y actuar únicamente bajo ciertas condiciones
- Debido a lo anterior, las herramientas completamente automáticas tienen sus limitaciones -especialmente contra el malware más sofisticado-. El análisis manual y patchear al malware para que exhiba su comportamiento puede ser necesario (¡y divertido!)

Análisis de Malware

- Herramientas para monitorear el filesystem, el tráfico de red o el registry pueden ser complementarias
  - Sin embargo, si el malware cifra las comunicaciones de red o los archivos que escribe, puede ser necesario debuggearlo/instrumentarlo para ver los datos antes del cifrado

Análisis de Malware

- Limitaciones del fingerprinting
  - Packing (con diferentes claves)
    - Habría que fingerprintear el malware despackeado y saltear las técnicas de detección de sandbox
  - Polimorfismo
    - Supongamos que el malware necesita setear el registro EAX en 0.
      ¿Qué puede hacer para eso?

Análisis de Malware

- Supongamos que el malware necesita setear el registro EAX en 0. ¿Qué puede hacer para eso?
  - XOR EAX, EAX
  - MOV EAX, $0
  - SUB EAX
  - SHIFT lógico
  - Etc.

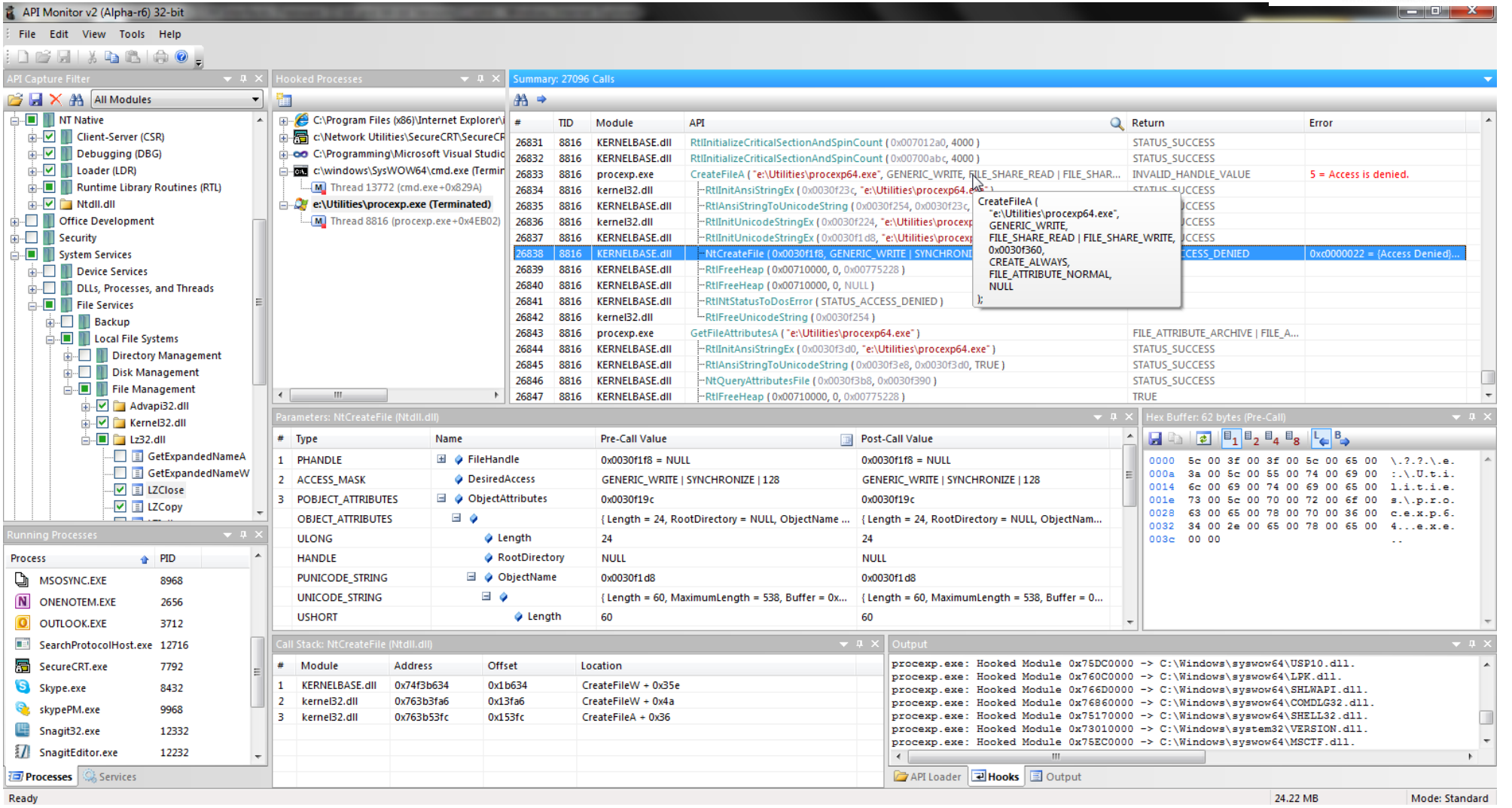
Análisis de Malware

- Cada instrucción se puede reescribir de forma diferente dando el mismo valor semántico
- Se pueden inclusive agregar instrucciones inocuas en el medio (ofuscación)

Análisis de Malware

- Otros objetivos del malware:
  - Worming → infectar targets cercanos
  - Escalación de privilegios - rootkit
  - Persistencia
    - Init.d (Linux)
    - Servicio (Windows)
    - Windows Management Instrumentation (WMI)
    - Firmware / bootloader patching

Análisis de Malware



Windows API Monitor

Lab 4.1

Analizar el malware del Lab 4.1 (ELF, x86) y describir su comportamiento

ADVERTENCIA:

Ejecutar en máquina virtual únicamente. El binario hace daño REAL. Se recomienda usar snapshots.

Referencias

- https://www.virustotal.com
- https://github.com/ytisf/theZoo/tree/master/malwares