Centro de
Investigación en
Computación

Static Analysis of Malware

Análisis Estático de malware

Sistemas

Técnicas

Etc.

Course

Análisis y Detección de Malware

Instructor

Acosta Bermejo Raúl

Lecture notes

2024-B
10 de octubre del 2024

Instituto
Politécnico
Nacional

Table of contents (outline)

Tabla de contenido

1. Código Fuente
2. Signatures
3. Hash values (ssdeep)
4. Entropía
5. Listas blancas y negras
6. Ofuscación
7. Varias Herramientas
8. Análisis de Código Binario

Source Code
Código Fuente

Técnicas y Herramientas

Static Analysis

Definición

- There are various techniques to analyze static source code for potential vulnerabilities that maybe combined into one solution.
- These techniques are often derived from compiler technologies.
- There are a many tools:
  - They suffer from false negatives and false positives.
  - Examples: CBMC, K8-Insight, PC-lint, Prevent, Satabs, SCA, Goanna, Cx-enterprise, Codesonar.
- SAMATE - Software Assurance Metrics And Tool Evaluation.

Link

- https://en.wikipedia.org/wiki/Static_program_analysis
- https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis

Static Analysis

Herramientas

List of tools for static code analysis

- By language
  - Aprox. 35 tools for multi-language in wikipedia.
  - Aprox. 25 tools for C/C++.
  - Aprox. 15 tools for Java.
- Formal methods
  - Tools that use sound, i.e. no false negatives, formal methods approach to static analysis (e.g., using static program assertions).
  - Aprox. 10 tools.

Static Analysis

Definición

- Code Quality and Code Security
  - Vulnerabilities in the code.
  - Metrics.

Link

- https://www.sonarqube.org/

Community

EDITION

Used and loved by 200,000+ companies

FREE & OPEN SOURCE

Download for free

All the following features:

- ✓ Static code analysis for 17 languages
  Java, C#, JavaScript, TypeScript, CloudFormation, Terraform, Kotlin, Ruby, Go, Scala, Flex, Python, PHP, HTML, CSS, XML and VB.NET
- ✓ Detect Bugs & Vulnerabilities
- ✓ Review Security Hotspots
- ✓ Track Code Smells & fix your Technical Debt
- ✓ Code Quality Metrics & History
- ✓ CI/CD integration
- ✓ Extensible, with 50+ community plugins

Static Analysis

Definición

Herramientas

- Analizan el Código tomado de:
  - Servidor de Código: Git (GitLab, GitHub), Mercury, SVN, CVN.
  - Localmente
- Calculan diferentes métricas:
  - Complejidad del Código: líneas de código por función, clase, archive, etc.
  - Apego a estándares según el lenguaje: indentación, formateo, etc.
  - Apego a buenas practicas.

Static Analysis

Definición

Flow Analysis

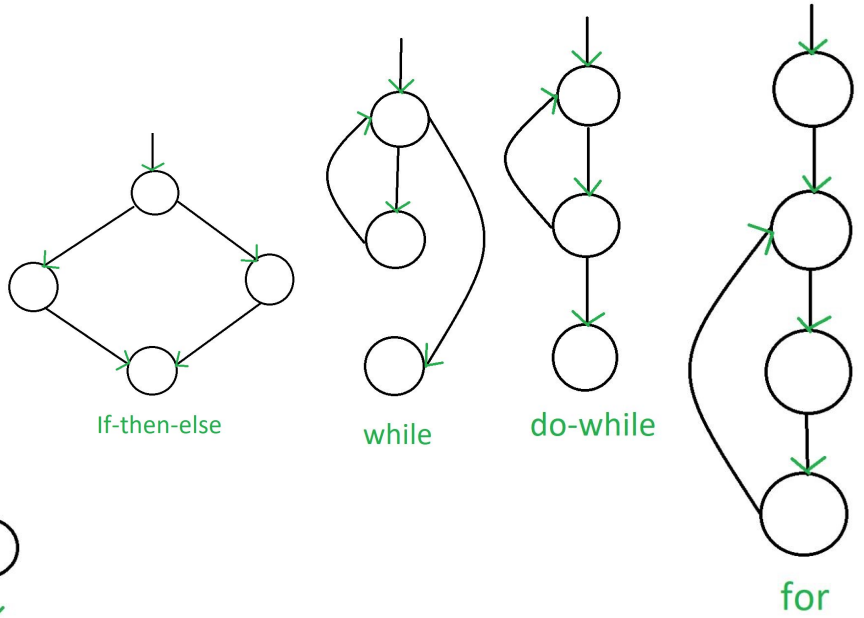
- **Basic block** A sequence of consecutive instructions where control enters at the beginning of a block, control leaves at the end of a block and the block cannot halt or branch out except at its end.
- **Control Flow Graph (CFG)**
  - It was originally developed by *Frances E. Allen*.
  - An abstract graph representation of software by use of nodes that represent basic blocks. A node in a graph represents a block; directed edges are used to represent jumps (paths) from one block to another. If a node only has an exit edge, this is known as an ‘entry’ block, if a node only has a entry edge, this is know as an ‘exit’ block.

  - Other types of graphs: Data-Dependence Graph
  - https://www.sciencedirect.com/topics/computer-science/control-flow-graph.

Static Analysis

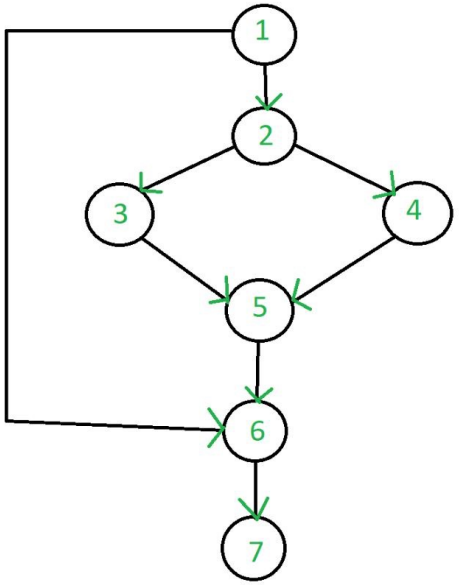
Definición

CFG



Example

```
if A = 10 then
   if B > C A = B
   else A = C
   endif
Endif
print A, B, C
```



Static Analysis

Definición

Function Flow Graph (FFG)

Call Flow Graph (CFG)

- It is a directed graph whose vertices represent the functions of which a software program is composed, and the edges symbolize function calls. A vertex is represented by either one of the following two types of functions:
  - Local functions, implemented by the programmer to perform specific tasks.
  - External functions: provided by the O.S. or system and external libraries.
- One particularity of the graph is that only local functions can invoke external functions, not the other way around. Function call graphs are generated from the static analysis of the disassembly file.

Static Analysis

Definición

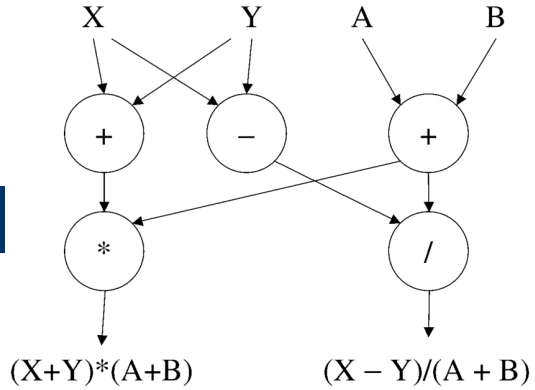
Comparing Graph

The problem can be tackle in many ways:

- Classical: isomorphism, isomorphic subgraph.
- Libraries of graphs
  - Isomorphism: https://www.boost.org/doc/libs/1_51_0/libs/graph/doc/isomorphism.html
  - Subgraph: https://www.boost.org/doc/libs/1_51_0/libs/graph/doc/table_of_contents.html

Static Analysis

Definición



Data Flow Analysis

- **Data Flow Graph (DFG)**
  It is a collection of arcs and nodes in which the nodes are either places where variables are assigned or used, and the arcs show the relationship between the places where a variable is assigned and where the assigned value is subsequently used.
  https://www.sciencedirect.com/topics/computer-science/data-flow-graph
- **Taint Analysis**
  It attempts to identify variables that have been 'tainted' with user controllable input and traces them to possible vulnerable functions also known as a 'sink'. If the tainted variable gets passed to a sink without first being sanitized it is flagged as a vulnerability.
- **Lexical Analysis**
  It converts source code syntax into ‘tokens’ of information in an attempt to abstract the source code and make it easier to manipulate.

Static Analysis

Definición

Herramientas para crear el CFG

1. Avrora
   - http://compilers.cs.ucla.edu/avrora/cfg.html.
2. CFGgrind
   - https://github.com/rimsa/CFGgrind
3. Dyninst
   - https://dyninst.org/manuals/dyninstAPI
     Requiere un conocimiento profundo de la herramienta.
   - Ejemplo: https://github.com/passlab/DCFG
4. Programas en Github
   - https://github.com/Kazhuu/asm2cfg
   - https://github.com/zestrada/playground/blob/master/objdump_cfg/objdump_to_cfg.py

Static Analysis

Definición

Bibliografía

Artículos, Tésis, Sitios, etc:

- “Representation and Analysis of Software”, Mary Jean Harrold **et al**.
- “Control Flow Graphs for Real-Time System Analysis”, Reconstruction from Binary Executables and Usage in ILP-Based Path Analysis Dissertation Zur Erlangung des Grades Doktor der Ingenieurwissenschaf.
- “Control Flow Graph Based Attacks”, ZHEN LI, Master’s Thesis at NADA, 2014.
- A Method to Evaluate CFG Comparison Algorithms Patrick P.F. Chan and Christian Collberg.

- https://schaumont.dyn.wpi.edu/ece4530f19/lectures/lecture18-notes.html

Static Analysis

Definición

Bibliografía

Software y Tésis

- PRACTICAL DYNAMIC RECONSTRUCTION OF CONTROL FLOW GRAPHS
  - https://repositorio.ufmg.br/bitstream/1843/36523/8/thesis.pdf.
  - Se usa el software Dyninst su API
    ParseAPI Programmer's Guide

- Varios blogs
  - https://reverseengineering.stackexchange.com/questions/10604/how-to-generate-cfg-from-assembly-instructions
  - Propone usar **ejecución simbólica**: representación abstracta del código.

Static Analysis

Definición

Symbolic Execution

- Papers
  - A Survey of Symbolic Execution Techniques, ROBERTO BALDONI, EMILIO COPPA, DANIELE CONO D'ELIA, CAMIL DEMETRESCU, and IRENE FINOCCHI
- Teoria
  - https://en.wikipedia.org/wiki/Symbolic_execution
  - Una de las herramientas más usadas es Angr que usa python.
  - **Angr genera el CFG**: https://docs.angr.io/built-in-analyses/cfg

Signatures

Firmas de archivos

Funciones hash

Signatures

De archivos

A signature is a typical footprint or pattern associated with a malicious attack on a computer network or system. This pattern can be a series of bytes in the file (byte sequence) in file, network traffic, etc.

A variable-length plaintext is “hashed” into a (typically) fixed-length hash value (often called a “message digest” or simply a “hash”).

Son conocidas como *message digest* (resumen de mensaje).

funcion (objeto) = valor único / tamaño fijo
objeto = archivo, paquete de datos, etc. / tamaño variable

Links

- https://www.sciencedirect.com/topics/computer-science/message-digest

Funciones hash

Más usadas

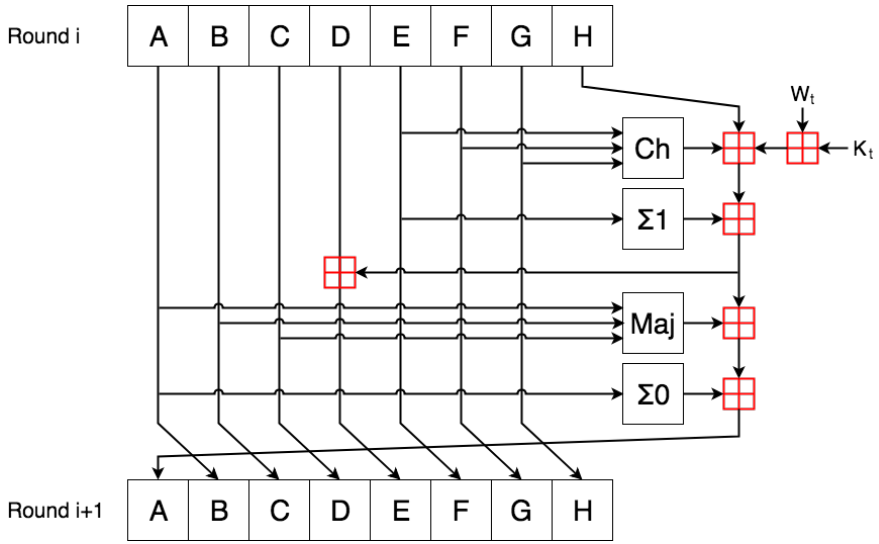
La competición de funciones hash de la NIST seleccionó una nueva función hash, el **SHA-3**, en 2012 A diferencia de SHA-2 con SHA-1, el algoritmo SHA-3 no es derivación del SHA-2.

- **MD5 (***Message-Digest Algorithm 5***)**
  - Diseñado por el Ronald Rivest del MIT desarrollado en 1991 y reemplazo a MD4 después de que Hans Dobbertin descubriese su debilidad (colisión de hash).
  - Se usan 128 bits representados típicamente como un número de 32 símbolos hexadecimales.
  - MD5("Generando un MD5 de un texto") = 5df9f63916ebf8528697b629022993e8
- **SHA1** (*Secure Hash Algorithm 1*)
  - Se usan 160 bits (20 bytes)representados como un número de 40 símbolos hexadecimales.
  - Desde el 2005 es considerado no seguro. El NIST en el 2011 ya no lo recomendó,
- **SHA2**
  - Es un conjunto de funciones hash criptográficas (SHA-224, SHA-256, SHA-384, SHA-512) diseñadas por la Agencia de Seguridad Nacional (NSA) y publicada en 2001.

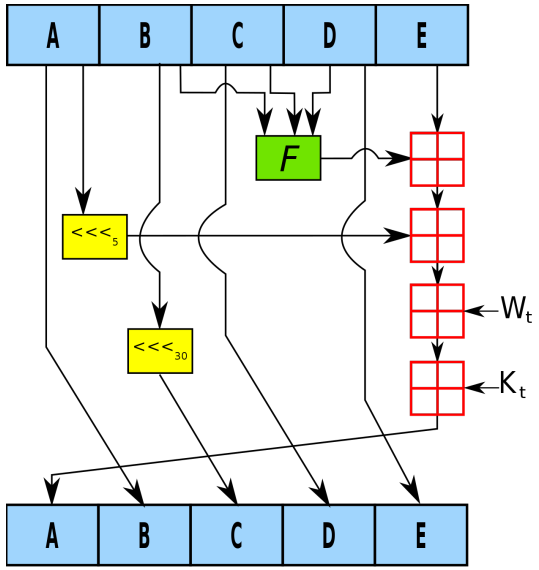
Funciones hash

Más usadas

SHA2



SHA1



https://es.wikipedia.org/wiki/SHA-2

Funciones hash

Más usadas

Herramientas

Existen varias:

1. Comandos
   $ md5sum

2. Aplicaciones
   Cada SO tiene las suyas

3. Librerias
   En OpenSSL ligada a libcrypto.
   #include <openssl/md5.h>

Patterns

Otros tipos

Cadenas

Se buscan patrones en archivos y el más fácil de hacer es el de cadenas:

1. Comandos
   $ strings binary_file
   $ strings /bin/ls

2. Aplicaciones
   Buscan patrones complejos.
   Herramienta YARA. Se verá mas adelante con detalle.

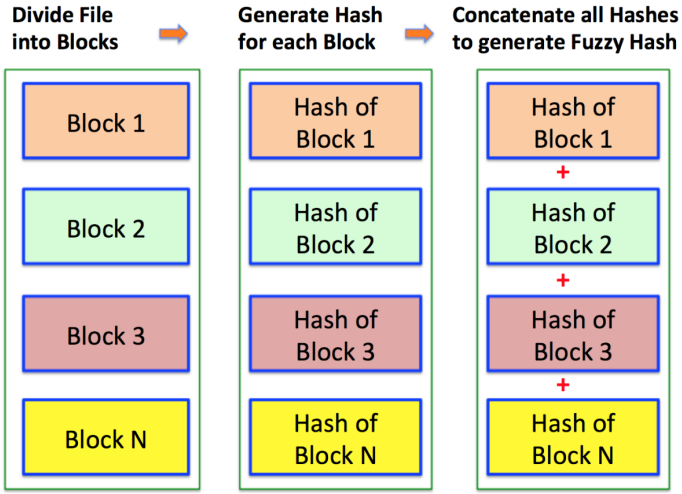
Other hash values

Ssdeep

Similitud de archivos

Ssdeep

Introducción



It is a program for computing CTPH:

Context Triggered Piecewise Hashes

- CTPH can match inputs that have homologies. Such inputs have sequences of identical bytes in the same order, although bytes in between these sequences may be different in both content and length.
- Locality-Sensitive Hashing (LSH). It is an algorithmic technique that hashes similar input items into the same "buckets" with high probability.
- It also provides a library (libfuzzy) to generate/compare fuzzy hashes.
  - Although “better fuzzy hashes” are available (Sdhash, TLSH), ssdeep is still one of the primary choices because of its speed and being a de facto standard.
- Links
  - https://ssdeep-project.github.io/ssdeep/index.html
  - https://tlsh.org/
  - https://github.com/trendmicro/tlsh

Ssdeep

Introducción

Ejemplos de comandos:

- $ ssdeep /bin/ls

  ssdeep,1.1--blocksize:hash:hash,filename
  768:5V62vDzfENAM8ucSL1N+zlPiZlik35qI8ToZp0q3rWOlCg1YYcuWetFSQTp
  DRI8l:rDP5SiZbw5VN6q3KoCCYO7jpOI,"/bin/ls"
- $ ssdeep –r /dir

  Calcula el valor hash de todos los archivos en el directorio.
- Comparar archivos

  $ ssdeep -b file1.txt > hashes.txt Aqui se guarda el hash.

  $ ssdeep -b -m hashes.txt file2.txt

  file2.txt matches hashes.txt:file1.txt (99)
- Comparar los valores hash de varios archivos

  $ ssdeep -x list1.txt list2.txt

Binary Diffing

Introducción

Binary Diffing

- It is a process where two files are compared at instruction level, looking for differences in code.
  - Diaphora
    https://github.com/joxeankoret/diaphora
  - BinDiff
    https://www.zynamics.com/bindiff.html
  - DarunGrim
- URL
  - https://cybersecurity.att.com/blogs/labs-research/code-similarity-analysis-with-r2diaphora.

Entropía

Teoría y Herramientas

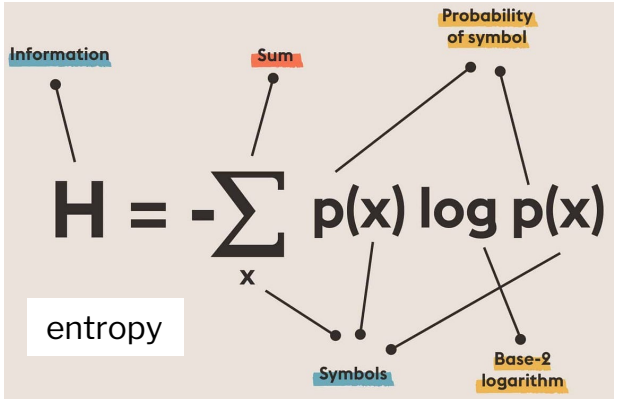
Estudio estadístico

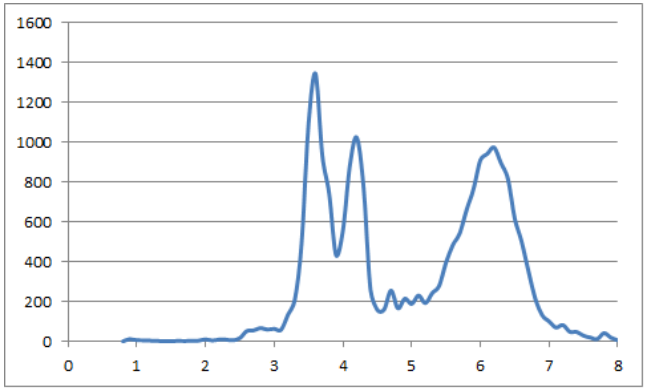
Entropía

Introducción

Definition

- Entropy is a measure of randomness within a set of data.
- When referenced in the context of information theory and cybersecurity, most people are referring to Shannon Entropy.
- This is a specific algorithm that returns a value between 0 and 8 where values near 8 indicate that the data is very random, while values near 0 indicate that the data is very homologous.
- Formula





Entropía

Introducción

There are many sites with code like this:

- CPP: https://github.com/merces/entropy
- PYTHON: https://cocomelonc.github.io/malware/2022/11/05/malware-analysis-6.html

```
float CalculateEntropy(char *pData,int nDataSize)
{
    float fEntropy=0;
    float bytes[256]={0.0};
    float temp;

    for(int i=0;i<nDataSize;i++) {
        bytes[(unsigned char)pData[i]]+=1;
    }

    for(int j=0;j<256;j++){
        temp=bytes[j]/(float)nDataSize;
        if(temp){
            fEntropy+=(-log(temp)/log(2))*bytes[j];
        }
    }

    fEntropy=fEntropy/(float)nDataSize;
    return fEntropy;
}
```

Entropía

Introducción

Práctica

- Analice varias muestras de malware de MAREA.
- Calcule la entropía con 2 algoritmos diferentes.
- Compare los resultados con los valores de VirusTotal.
- ¿A que conclusión llega?
- Ahora tome archivos *goodware* y cífrelos con algún algoritmo y compare la entropía antes y después.

- Haga las pruebas descritas en:
  - https://cocomelonc.github.io/malware/2022/11/05/malware-analysis-6.html

Listas

Negras y Blancas

Sitios web

Listas

Blancas y Negras

Descripción

Son direcciones que ya han sido analizadas y se sabe que:

- Negras: que tienen malware o son usadas por malware
  URL Blocklist, blacklist, etc.
- Blancas: que no representan ningún problema de seguridad.
  White list

Ejemplos:

- URLhaus
  - The data set is available in various formats: CSV.
  - Update each 5 min, past 30 days.
  - https://urlhaus.abuse.ch/

Ofuscación

Archivos ofuscados
Empaquetados y/o cifrados

Ejemplos

Introducción

Definición

También se usa para la protección de La propiedad intelectual.

Ofuscación

- Se refiere a encubrir el significado de una comunicación haciéndola más confusa y complicada de interpretar.
- Acto deliberado de realizar un cambio no destructivo, ya sea en el código fuente, en el código intermedio (bytecodes) o en el código máquina cuando el programa está en forma compilada o binaria.
- Se cambia el código manteniendo el funcionamiento original, para dificultar su entendimiento y los intentos de ingeniería inversa y desensamblado. [Wikipedia].
- Como un efecto lateral, la ofuscación, en ocasiones, hace que los programas resultantes sean más grandes.

- Links
  - https://es.wikipedia.org/wiki/Ofuscación
  - https://www.preemptive.com/obfuscation/

Introducción

Definición

Ofuscación de código

Collberg et al. [CTL97, CTL98] presented the first formal definition of the obfuscation problem. They stated that a transformation function T maps a program P to a new program P' such that P' is resilient to deobfuscation and P' contains the same behaviour as P.

Collberg et al. put obfuscation transformations into three categories:

- Lexical Transformations
- Control-flow Transformations.
- Data Transformations.

Introducción

Definición

Ofuscación de código

- **Lexical Transformations**. It modifies information that is related to the structure of the program, e.g. by renaming identifiers, changing or removing debugging information and comments. This type of obfuscation transformation can also be used to protect the intellectual property rights of software.
- **Control-flow Transformations** It makes changes to the control flow of the program, e.g. through code reordering and jump insertion, and through the insertion of opaque predicates. The results of these transformations are hard for a deobfuscator to compute.
- **Data Transformations** It obfuscates data and data structures in the program, e.g. variable-splitting transformations that split a single variable into two or more variables with new operations to produce an equivalent result to the original variable.

Introducción

Definición

Ofuscación de código

Techniques

- Code reordering.
- Garbage insertion.
- Data Transformations.
- Equivalent code replacement.
- Variable renaming.
- Code and data encapsulation.

Introducción

Definición

Otras técnicas

- La **esteganografía** trata del estudio y aplicación de técnicas que permiten ocultar mensajes u objetos, dentro de otros, llamados *portadores*, de modo que no se perciba su existencia.
- Trata de ocultar mensajes dentro de otros objetos y de esta forma establecer un canal encubierto de comunicación, de modo que el propio acto de la comunicación pase *inadvertido* para observadores que tienen acceso a ese canal. [Wikipedia].

- Links
  - https://es.wikipedia.org/wiki/Esteganografía
  - https://en.wikipedia.org/wiki/Steganography

Buen ejemplo de que las versiones de dos idiomas son diferentes.

Archivos

Ofuscados

Empaquetamiento

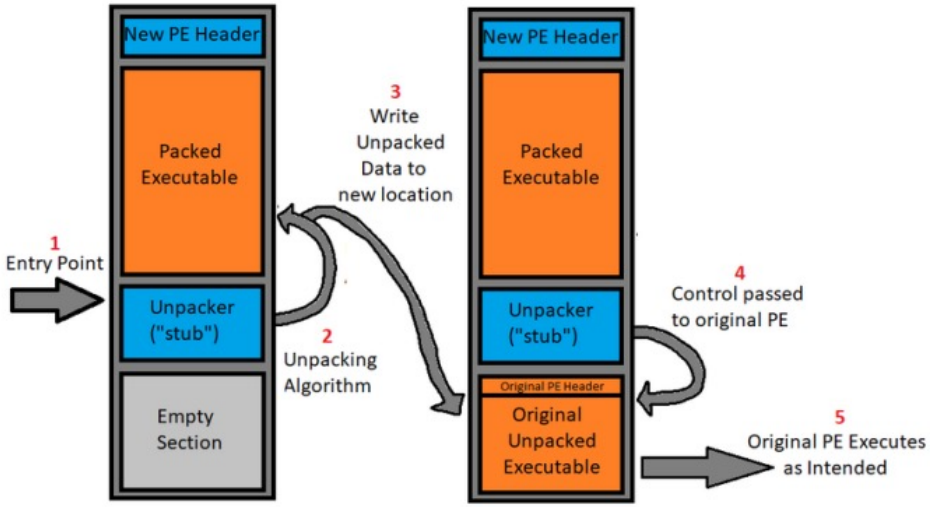
- Los ejecutables ofuscados y empaquetados frecuentemente se caracterizan por presentar niveles altos de entropía.
- Además, el tamaño de las secciones también puede ser un interesante indicativo. Por ejemplo, si el vsize (virtual size, tamaño que ocuparía en memoria) es mucho mayor que el size (tamaño que ocupa en disco) es un indicativo de código empaquetado.
- En varios documentos lo explican, por ejemplo:
  - **BlackHat**: https://www.blackhat.com/ presentations/ bh-dc-07/ Kendall_McMillan/ Presentation/bh-dc-07-Kendall_McMillan.pdf
- Empaquetadores/Compresores de ejecutables (el código)
  - Se empaqueta el código binario y se crea un nuevo Exe.
  - El más usado es UPX (open source): https://upx.github.io/
  - Variantes: UPXScrambler, UPXFreak, UPXSh!T, UPXCrypter y UPXModifier.

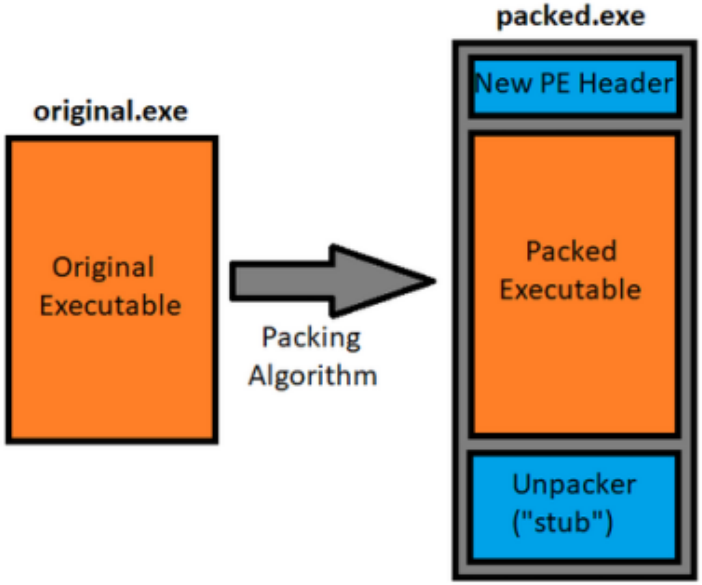
https://malware.news/t/the-basics-of-packed-malware-manually-unpacking-upx-executables/35961

Archivos

Ofuscados

Explicación gráfica





Como se detecta un archivo empaquetado?

Packer detectors

PEID, ExeInfo PE or RDG Packer Detector

Si no lo detecto y lo analizo
que se obtiene?

Archivos

Ofuscados

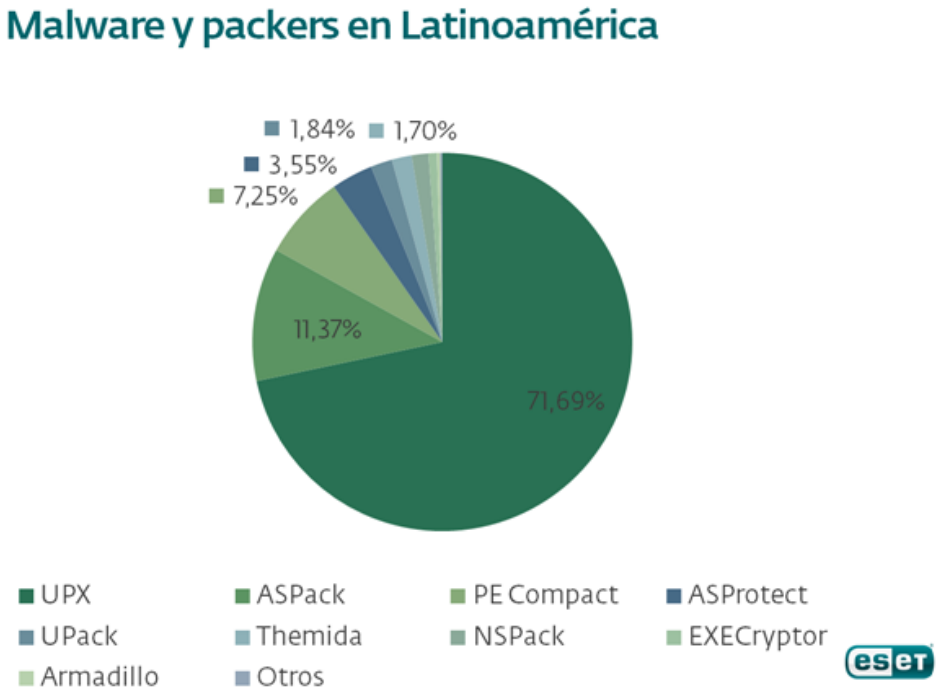
Empaquetamiento

- Reportes
  - https://www.welivesecurity.com/la-es/2011/03/22/el-reinado-de-upx/

En el 2011, el **75%** del total de muestras se encuentra protegida con packers de diferentes tipos.

Después UPX esta ASpack que no es open source pero comprime entre 40% y 70%.

http://www.aspack.com/



Archivos

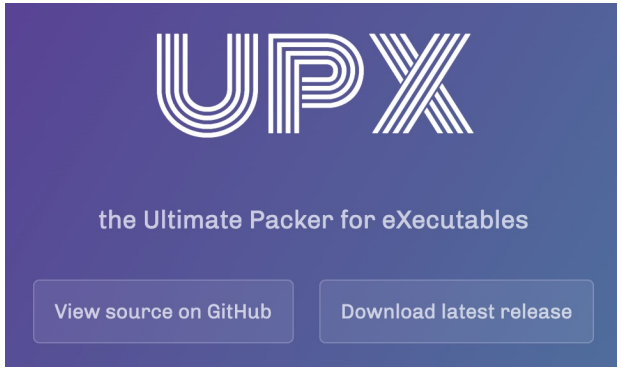
Ofuscados

Lista de empaquetadores

- UPX (Gratuito)
  - Multiplataforma
  - https://upx.github.io/
- Aspack (Comercial $)
  - Windows 32 y 64 bits
  - http://www.aspack.com/
- Themida (Comercial $)
  - https://www.oreans.com/Themida.php
- PE Compact (Comercial $)
  - https://github.com/dehoisted/PECompact-Cracked
  - https://bitsum.com/pecompact.htm
- NSPack (Comercial $)
  - https://sansorg.egnyte.com/dl/LuOgfrk5V9

Archivos

Ofuscados



UPX

- It is a free, secure, portable, extendable, high-performance executable packer for several executable formats: Windows programs and DLLs, macOS apps and Linux executables.
  - It reduce the file size of programs and DLLs by around 50%-70%
  - very fast decompression: more than 500 MB/sec on any reasonably modern machine
- The term UPX is a shorthand for the Ultimate Packer for eXecutables.
- Referencias
  - https://upx.github.io/
  - **Source Code**: https://github.com/upx/upx
  - It may be distributed and used freely, even with commercial applications.

Archivos

Ofuscados

Práctica

- Crear un archivo ejecutable.
  - Empaquetarlo con UPX y analizar el resultado.

    Como se detectan archivos empaquetados?
  - Desempaquetarlo y verificar su resultado. Es el mismo que el inicial?
- Referencias de base:
  - https://github.com/cybertechniques/example-techniques-obfuscation-packing-upx/blob/master/README.md

Archivos

Ofuscados

Anti-Unpacking

- UPX
  - https://cujo.com/blog/upx-anti-unpacking-techniques-in-iot-malware/

Varias Herramientas

Patrones sospechosos

Algunas

Herramientas

Introducción

Patrones sospechosos

- Ejemplos de patrones son:
  - Técnicas anti-debug
  - El uso de sandboxes
- La detección se hace mediante:
  - Librerías, por ej. el del sandbox.
- Ejemplos
  - Pafish
    - https://github.com/a0rtega/pafish?ref=ciberseguridad.blog
  - CAPE (Malware Configuration And Payload Extraction)

    It is a malware sandbox released on github around September 2016. Built on Cuckoo (More precisely, spender sandbox), it can automatically extract payload and configuration information from many well-known malware.
    - https://github.com/ctxis/CAPE
    - https://soji256.medium.com/build-a-cape-sandbox-to-analyze-emotet-3d507599dda6

Binary Code Analysis

Análisis de Código Binario

Técnicas y herramientas

Binary code analysis

Ejecutables

El analisis de los binarios involucra los siguientes temas:

- **Tipos de archivos ejecutables**
  Windows (PE), UNIX (ELF), MacOS (Mach-O).
- **Estructura de los ejecutables**
  Encabezado.
  Cuerpo.
- **Desensamblado**
  Herramientas.
- **Análisis de alto nivel del ejecutable**
  OPCODEs
  Syscalls
  API calls

Binary code analysis

Ejecutables

Especificación de los ejecutables:

1. Windows (PE)
2. UNIX (ELF)
3. MacOS (Mach-O)

Veremos los
Slides de Herramientas

Binary code analysis

Ejecutables

Al analizar una muestra de malware de la cual se tiene el **archivo ejecutable**, uno puede extraer:

- **Byte-value**
  Es el valor binario de la instrucciones en ensamblador.
- **Opcode** (OPeration CODE)
  Es el nemónico de la instrucción en ensamblador.
  Se hace estática y dinámicamente.
- **Syscall** (System Call) Es difícil => se hace dinámicamente
  Son las invocaciones que hace el programa al API del Sistema Operativo.
- **APIcall** (**A**pplication **P**rogram **I**nterface Call) Es difícil => se hace din.
  Son las invocaciones que hace el programa al API de una librería del sistema. Por ejemplo, libc en Linux o win32.dll en Windows.

Binary Code

System Call

El análisis de Syscalls se centra en:

1. **Categorizar** las syscalls
   Por ejemplo: Time, filesystem, process memory, communication, etc.
2. **Simplificar** su procesamiento
   Cada syscall recibe parámetros mediante el uso de la pila: push, pop. Aquí se tiene bytes de información que manipulan con la pila.

Binary Code

API call

El análisis de API call se centra en :

1. **Categorizar** las APIs
   Por ejemplo: network management (socket), DLL, etc.
2. **Simplificar**
   Lo mismo que syscall pero con semántica (objetos).
3. Usar información:
   1. Espacial: argumentos, valores de retorno.
   2. Temporal: flujo de las llamadas.
      Se usa la Secuencia de las llamadas.

Las API pueden ser de Librerías o del SO.

The end

Contacto

Raúl Acosta Bermejo

http:www.cic.ipn.mx
http://www.ciseg.cic.ipn.mx/

racostab@ipn.mx
racosta@cic.ipn.mx

57-29-60-00
Ext. 56652