TP1 - TokenSnare
Diseño e Implementación de Honeytokens

---

1. Introducción: El Arte del Engaño Digital

En ciberseguridad, la defensa no siempre es pasiva. Las tecnologías de engaño (*deception technology*) son una estrategia proactiva para detectar intrusiones de manera temprana. Un **honeytoken** es un recurso digital trampa, diseñado con un único propósito: activar una alarma silenciosa en el momento en que un atacante interactúa con él. Puede ser un archivo PDF en un servidor de archivos, una clave de API falsa en un repositorio de código, o una cuenta de usuario inactiva. Cuando se accede a este cebo, delata la presencia de un actor no autorizado en el sistema.

El objetivo de este trabajo práctico es ir más allá de la teoría y construir desde cero un sistema funcional de generación y alerta de honeytokens, llamado **TokenSnare**. Inspirados por herramientas como `Canarytokens`, desarrollarán su propia implementación, investigando los mecanismos de "llamada a casa" (*call home*) que hacen posible esta técnica.

2. Objetivos de Aprendizaje

Al completar este trabajo, los estudiantes habrán logrado:

- **Comprender en profundidad** el concepto de honeytokens y su rol en una estrategia de ciberdefensa activa.
- **Investigar y analizar** los diversos mecanismos técnicos de *call home* aplicables a distintos formatos de archivo (documentos, ejecutables, imágenes, etc.).
- **Desarrollar una aplicación de línea de comandos (CLI)** funcional para generar diferentes tipos de honeytokens.
- **Implementar un backend simple** capaz de recibir las notificaciones de los tokens activados y generar las alertas correspondientes.
- **Adquirir experiencia práctica** en el empaquetado y despliegue de aplicaciones multicomponente utilizando contenedores Docker.

---

3. Parte 1: Investigación - Mecanismos de "Call Home"

Antes de escribir código, es crucial entender cómo un simple archivo puede "llamar a casa". Esta fase consiste en investigar y documentar las técnicas que permiten incrustar un mecanismo de notificación en diferentes tipos de archivos. Para cada uno de los formatos que planeen implementar, deberán explicar teóricamente cómo funciona el *callback*.

Ejemplos de mecanismos a investigar:

- **Documentos PDF:** El uso de JavaScript embebido o referencias a recursos externos (imágenes, fuentes) que se cargan al abrir el documento.
- **Documentos de Microsoft Office (Word/Excel):** El uso de macros (VBA) o la inclusión de imágenes/objetos vinculados a una URL externa.
- **Clonado de Sitio Web:** Inserción de código HTML/JavaScript (ej. `<img>`, `<script>`) en una página clonada que apunta al servidor de alertas.
- **Códigos QR:** Generación de un código QR que contenga una URL única apuntando al servidor.
- **Binarios/Ejecutables:** Un programa simple que, al ejecutarse, realiza una petición HTTP/DNS a un endpoint controlado.

- **Otros formatos:** Investigar las capacidades de formatos como CSS, archivos de configuración, etc.

El resultado de esta investigación será la base teórica que se incluirá en el informe final.

---

4. Parte 2: Desarrollo de la Herramienta "TokenSnare"

El núcleo del trabajo es el desarrollo de un sistema compuesto por dos partes principales: un generador de tokens (CLI) y un servidor de alertas (backend).

A. Componentes a Desarrollar:

1. **El Generador (`tokensnare-cli`):**
    - Una aplicación de línea de comandos que permita al usuario generar diferentes tipos de honeytokens.
    - Debe interactuar con el servidor de alertas para registrar un nuevo token y obtener una URL única de *callback* para incrustar en el archivo.
    - **Ejemplo de uso:** `python tokensnare-cli.py --type pdf --output carnada.pdf --message "Informe Confidencial Q3"`
2. **El Servidor de Alertas (`tokensnare-server`):**
    - Un servidor web simple (ej. usando Flask, FastAPI, Express.js) con al menos dos endpoints:
        - Uno para registrar nuevos tokens desde la CLI.
        - Otro para recibir las peticiones *GET* de los honeytokens activados.
    - Cuando un token es activado (alguien accede a la URL única), el servidor debe registrar el evento en un log o en la consola, incluyendo información como la fecha/hora, la dirección IP de origen y el tipo de token.

B. Requisitos Mínimos de Implementación:

- **Mínimo de 5 formatos/tecnologías:** Se debe implementar la generación y detección para al menos cinco tipos de honeytokens distintos (ej: PDF, Excel con macro, clon de web, código QR, binario para Windows/Linux).

C. Desafíos Opcionales (para nota adicional):

- Implementar al menos una característica extra que aporte valor. Por ejemplo, permitir que el usuario defina el contenido visible del token (el texto de un documento, el título de una web clonada, etc.) para hacerlo más creíble.
- Investigar e intentar implementar un honeytoken en un formato no convencional (ej. un archivo EPUB que intente cargar un CSS remoto, un token en una imagen SVG, etc.). Deberán documentar los desafíos encontrados, el soporte de los visores/clientes y si la implementación fue exitosa.

---

5. Entregables y Criterios de Evaluación

Se deberán entregar dos componentes principales:

1. **Informe del Proyecto (PDF):**
    - **Portada:** Datos de la materia, Cuatrimestre y año, TP elegido, nombre del grupo y detalle de los integrantes.
    - **Introducción Teórica:** Explicar brevemente qué son los honeytokens y la investigación realizada sobre los mecanismos de *call home* (Parte 1).

- **Diseño e Implementación:** Describir la arquitectura de la solución (CLI + Server), las decisiones de diseño tomadas y explicar técnicamente cómo se implementó la generación para cada uno de los 5+ tipos de tokens.
- **Manual de Uso y Prueba:** Instrucciones claras sobre cómo ejecutar la solución completa utilizando contenedores Docker y Docker Compose. Se deben incluir los comandos exactos y alguna captura de pantalla del sistema en funcionamiento (generando un token y luego activándolo para ver la alerta en el servidor).
- **Formato y Extensión:**
  Fuente: Arial 10.
  Espaciado: Simple.
  Extensión mínima: 12 carillas.

2. **Código Fuente:**
   - El código completo de la CLI y del servidor, junto con los `Dockerfile` y `docker-compose.yml` necesarios para levantar el entorno.
   - Se debe entregar como un único archivo `.zip` o un enlace a un repositorio Git (ej. GitHub, GitLab).

Criterios de Evaluación:

- **Funcionalidad y Variedad de Tokens (50%):** Correcto funcionamiento de la solución y diversidad/complejidad de los 5 tipos de honeytokens implementados.
- **Calidad del Código y Contenerización (25%):** Estructura del código, claridad y correcta implementación del entorno con Docker.
- **Calidad del Informe y la Investigación (20%):** Profundidad de la investigación teórica y claridad en la documentación del diseño y las pruebas.
- **Funcionalidad Adicional / Desafío Opcional (5%):** Valor agregado por las características extra o la resolución de algún desafío opcional.

Recursos de Referencia:

- Honeytokens and Canarytokens Setup
- Canarytokens.org (para inspiración)