Keyboard Monitoring:

Estudio de Técnicas de Captura de Teclado en Entornos Controlados

Autor: Víctor Barroso

GitHub: @muddysec

1. Introducción

Este documento presenta un proyecto académico desarrollado en el contexto de una práctica del máster en ciberseguridad. El objetivo principal es analizar y replicar de forma controlada una técnica comúnmente utilizada en ataques reales: la captura de pulsaciones de teclado mediante un script de Python. Esta práctica se llevó a cabo exclusivamente con fines formativos y en entornos de laboratorio, sin implicar usuarios reales ni sistemas de terceros.

2. Entorno de desarrollo

- **Lenguaje:** Python 3.10
- **Librerías clave:** `pynput`, `smtplib`, `threading`, `email`
- **Entorno:** Máquina virtual Windows 10 sin conexión a redes externas durante las pruebas
- **Editor:** Visual Studio Code
- **Control de código:** Se mantuvo versión limpia sin datos reales ni credenciales activas

3. Captura de pulsaciones

La funcionalidad principal del script se basa en la captura de eventos de teclado utilizando la librería `pynput`. Inicialmente se empleó el evento `on_release`, aunque posteriormente se optó por `on_press` por su mayor fiabilidad.

Características técnicas:

- Registro de cada tecla pulsada en tiempo real
- Filtrado de teclas modificadoras (Shift, Ctrl, etc.) para evitar registros innecesarios
- Mapeo de teclas especiales como Enter, Espacio o Backspace
- Mecanismo de anti-repetición (debounce temporal de 50 ms) para evitar registros múltiples al mantener pulsada una tecla

4. Almacenamiento local de registros

El script guarda las pulsaciones en un archivo de texto plano (`pulsaciones_grabadas.txt`).

El archivo se abre y cierra de forma controlada en cada operación de escritura para evitar bloqueos y problemas de concurrencia.

5. Envío automatizado del registro por correo

Cada 2 horas, el sistema verifica si el archivo de pulsaciones contiene información. En caso afirmativo, lo envía por correo electrónico como archivo adjunto mediante `smtplib`.

Detalles técnicos:

- Uso de `email.message.EmailMessage` para componer el correo
- Adjuntos nombrados con fecha y hora para control de versiones
- Acceso SMTP a través de TLS (puerto 587)
- Credenciales gestionadas mediante variables de entorno (`.env`) para evitar exposición en el código

> Esta funcionalidad se diseñó para ilustrar técnicas reales, pero **no debe utilizarse en producción ni sistemas ajenos**. Actualmente, Gmail y otros

proveedores requieren el uso de contraseñas de aplicación o acceso OAuth para este tipo de integración.

6. Simulación de ingeniería social mediante instalador falso

Para replicar técnicas de ingeniería social, se creó una simulación de empaquetado malicioso:

1. Se convirtió el script `.py` a un ejecutable `.exe` utilizando **PyInstaller** con la opción `--onefile` para empaquetar todas las dependencias.

2. Posteriormente, se utilizó la herramienta **Ignition Key** para simular un instalador personalizado:

    - Se cambió el icono del ejecutable por uno representando una aplicación meteorológica

    - Se modificó el nombre y la descripción del archivo para simular una app legítima de la AEMET

    - El objetivo fue observar cómo una interfaz verosímil puede ocultar comportamientos ocultos al usuario

Esta simulación se realizó exclusivamente en entornos de laboratorio. No se distribuyó el ejecutable fuera de la máquina de pruebas.

7. Reflexión ética

El desarrollo de este proyecto se enmarca dentro del estudio de técnicas ofensivas con el fin de mejorar la seguridad defensiva. Comprender cómo actúan los atacantes permite diseñar mejores sistemas de protección.

El uso de herramientas como keyloggers está prohibido en la mayoría de contextos legales y éticos, salvo en ejercicios controlados con consentimiento informado. Este proyecto **no promueve el uso ofensivo** ni proporciona mecanismos para su explotación masiva.

8. Contramedidas y defensa

Entre las defensas posibles frente a esta clase de amenaza destacan:

- Uso de antivirus con detección basada en comportamiento
- Restricción de permisos de ejecución de scripts desconocidos
- Supervisión de procesos ocultos o inusuales en segundo plano
- Políticas de control de dispositivos USB y ejecutables no firmados
- Formación continua de usuarios frente a técnicas de ingeniería social

9. Lecciones aprendidas

- La implementación práctica de técnicas ofensivas mejora la comprensión del ciclo de ataque.
- El diseño del script supuso desafíos técnicos como la duplicación de pulsaciones o la gestión correcta del archivo de log.
- Se profundizó en conceptos como la ejecución diferida, el empaquetado de scripts y el uso de variables de entorno para proteger credenciales.

10. Código fuente y estructura

El código fue desarrollado siguiendo buenas prácticas:

- Separación clara de funciones (captura, envío, almacenamiento)
- Eliminación de variables globales inseguras
- Uso de `with open(...)` para manejar archivos de forma segura
- Control de errores en cada sección para asegurar robustez

Archivos que se deben incluir en el repositorio:

```
📁 keylogger-monitoring
 ├ 📜 main.py            # Código principal del keylogger (educativo)
 ├ 📜 README.md          # Instrucciones, contexto ético y técnico
 ├ 📄 proyecto.pdf       # Documento explicativo (este documento)
 ├ 📄 .env.example       # Variables de entorno de ejemplo (sin datos reales)
 └ 📄 .gitignore         # Para excluir logs, .env real, ejecutables y basura
```

Archivos que no deben subirse al repositorio público:

- `pulsaciones_grabadas.txt` → generado automáticamente durante la ejecución
- `.env` → contiene credenciales sensibles
- `main.exe` → ejecutable generado localmente para pruebas

11. Conclusión

Este proyecto ha servido para poner en práctica conceptos clave de la seguridad ofensiva en un entorno ético y controlado. Aporta una visión crítica sobre cómo pequeños scripts pueden comprometer sistemas, y refuerza la necesidad de formación especializada, tanto en la ofensiva como en la defensa.

El conocimiento técnico obtenido es relevante para futuros roles en Red Team, análisis forense o auditoría de seguridad, siempre con un compromiso firme con el uso responsable del conocimiento.