# Desarrollo de una librería de usuario para detección de TOCTOU en entornos Linux

Razvan Raducu
Tutor: Ricardo J. Rodríguez

Máster Universitario en Investigación en Ciberseguridad
Universidad de León

Septiembre, 2019

# Índice

# Time Of Check To Time Of Use (TOCTOU)

- Condición de carrera
  - Accesos no sincronizados
  - Recurso común
  - Modificación

# Time Of Check To Time Of Use (TOCTOU)

- Condición de carrera
  - Accesos no sincronizados
  - Recurso común
  - Modificación
- Operaciones E/S en sistema de ficheros

# Time Of Check To Time Of Use (TOCTOU)

- Condición de carrera
  - Accesos no sincronizados
  - Recurso común
  - Modificación
- Operaciones E/S en sistema de ficheros
- Referencias a objetos del sistema de ficheros:
  - Descriptores de ficheros
  - Rutas de nombres $\rightarrow$ problemáticas

# Time Of Check To Time Of Use (TOCTOU)

- Condición de carrera
  - Accesos no sincronizados
  - Recurso común
  - Modificación
- Operaciones E/S en sistema de ficheros
- Referencias a objetos del sistema de ficheros:
  - Descriptores de ficheros
  - Rutas de nombres $\rightarrow$ problemáticas
- Este trabajo se centra en Linux:
  - 37% TOP 10M sitios web usan Linux
  - 100% TOP 500 supercomputadores usan Linux

# Características TOCTOU

- Causan desde denegación de servicio a escalado de privilegios

# Características TOCTOU

- Causan desde denegación de servicio a escalado de privilegios
- No determinista. Depende de factores externos como:
  - Carga del sistema
  - Variables de entorno

# Características TOCTOU

- Causan desde denegación de servicio a escalado de privilegios
- No determinista. Depende de factores externos como:
  - Carga del sistema
  - Variables de entorno
- Secuencias de llamadas
  - Establece condición: *Time Of Check*
  - Modificación en base a esa condición: *Time Of Use*
  - **Ventana de vulnerabilidad**

# Características TOCTOU

- Causan desde denegación de servicio a escalado de privilegios
- No determinista. Depende de factores externos como:
  - Carga del sistema
  - Variables de entorno
- Secuencias de llamadas
  - Establece condición: *Time Of Check*
  - Modificación en base a esa condición: *Time Of Use*
  - **Ventana de vulnerabilidad**
- Problemas derivados del sistema de ficheros:
  - Operaciones no aisladas
  - Operaciones no atómicas
  - Inconsistencia

# Características TOCTOU

- Causan desde denegación de servicio a escalado de privilegios
- No determinista. Depende de factores externos como:
  - Carga del sistema
  - Variables de entorno
- Secuencias de llamadas
  - Establece condición: *Time Of Check*
  - Modificación en base a esa condición: *Time Of Use*
  - **Ventana de vulnerabilidad**
- Problemas derivados del sistema de ficheros:
  - Operaciones no aisladas
  - Operaciones no atómicas
  - Inconsistencia
- Programa vulnerable se ejecuta con más privilegios que el atacante
  - Bit SetUID (+s) o SetGID (+g)

# Explotar TOCTOU

- Engañar al programa manipulando el sistema de ficheros

# Explotar TOCTOU

- Engañar al programa manipulando el sistema de ficheros
- Cambiar el objeto por un enlace simbólico

| SUID program | Attacker |
|---|---|
| `access("file")` | |
| | `unlink("file")` |
| | `link("sensitive", "file")` |
| `fd = open("file")` | |
| `read(fd, ...)` | |

Ataque TOCTOU. Fuente: [1]

[1] M. Payer and T. R. Gross, "Protecting applications against TOCTTOU races by user-space caching of file metadata" VEE'12, pp. 215-226, 2012.

# libTTThwart

- Implementada en C

# libTTThwart

- Implementada en C
- Librería compartida cargada en "`/etc/ld.so.preload`". Alcance de sistema.

# libTTThwart

- Implementada en C
- Librería compartida cargada en "`/etc/ld.so.preload`". Alcance de sistema.
- Sólo se activa si RUID != EUID o RGID != EGID

# libTTThwart

- Implementada en C
- Librería compartida cargada en "`/etc/ld.so.preload`". Alcance de sistema.
- Sólo se activa si RUID != EUID o RGID != EGID
- Intercepta llamadas para comprobar integridad del sistema de ficheros comparando características y realizar la llamada original (si procede)

# libTTThwart

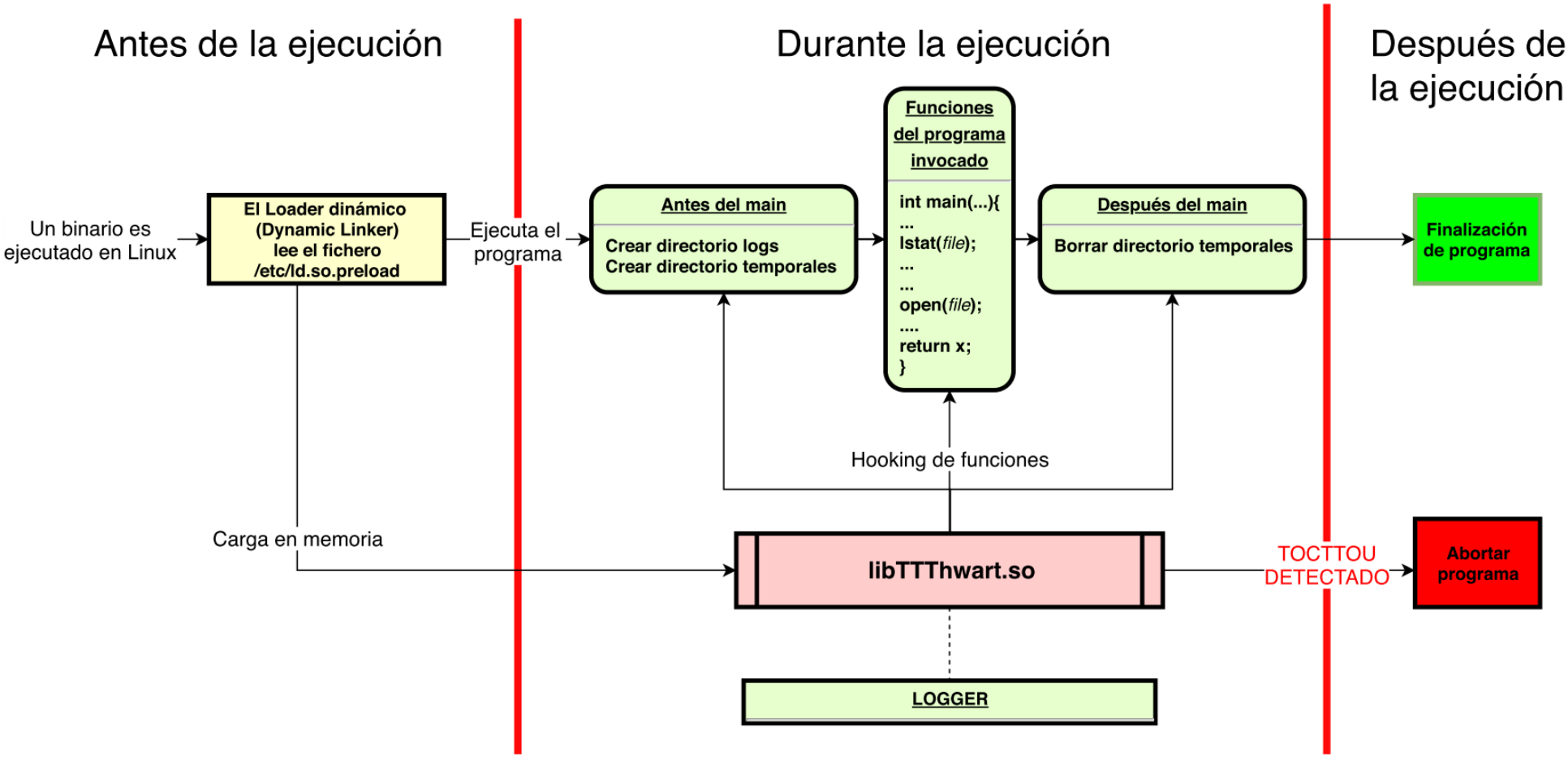
- Implementada en C
- Librería compartida cargada en "`/etc/ld.so.preload`". Alcance de sistema.
- Sólo se activa si RUID != EUID o RGID != EGID
- Intercepta llamadas para comprobar integridad del sistema de ficheros comparando características y realizar la llamada original (si procede)
- Registra todos los cambios en los objetos referenciados

# libTTThwart

- Implementada en C
- Librería compartida cargada en "`/etc/ld.so.preload`". Alcance de sistema.
- Sólo se activa si RUID != EUID o RGID != EGID
- Intercepta llamadas para comprobar integridad del sistema de ficheros comparando características y realizar la llamada original (si procede)
- Registra todos los cambios en los objetos referenciados
- Se interceptan 46 funciones

`https://github.com/RazviOverflow/libTTThwart`

# Diagrama de sistema



# Ejemplo interceptado

```c
int access(const char *path, int mode){

    if(LIBRARY_ON){
        const char *sanitized_path = sanitize_and_get_absolute_path(path);
        print_function_and_path(__func__, path, sanitized_path);
        get_fs_and_initialize_checking_functions(sanitized_path);
        if(file_does_exist(sanitized_path)){
            upsert_file_data_in_array(&g_array, sanitized_path, get_inode(sanitized_path), g_temp_dir);
        } else {
            upsert_nonexisting_file_metadata_in_array(&g_array, sanitized_path, NONEXISTING_FILE_INODE);
        }
        free((void *)sanitized_path);
    }
    if(original_access == NULL){
        original_access = dlsym_wrapper(__func__);
    }
    return original_access(path, mode);
}
```

# Metadatos registrados

Los metadatos que se comprueban son:

| **Metadato** | **Descripción** |
|---|---|
| `st_dev` | ID del dispositivo en el que está el objeto |
| `st_ino` | Inodo del objeto |
| `st_mode` | Tipo del objeto y sus permisos |

# Estudio reutilización de inodos

| Sistema de ficheros | Reutilización de inodos |
|---|---|
| BTRFS | No |
| EXT2 | Sí |
| EXT3 | Sí |
| EXT4 | Sí |
| FAT16 | No |
| FAT32 | No |
| NTFS | No |
| HFS+ | No |
| JFS | No |
| NILFS2 | Sí |
| REISERFS | Sí |
| XFS | Sí |
| RAMFS | No |
| TMPFS | No |

# Demostración

Demo de explotación de TOCTOU y uso de la librería libTTThwart

# Evaluación

- Microbenchmarking
  - 4 test diferentes, en diferentes sistemas de ficheros
  - Ejecutado 50K veces con y sin libTTThwart activada
  - En el mejor de los casos un impacto del **89% (1.89x)**
  - En el peor de los casos un impacto del **266% (3.66x)**
- CPU SPEC2006
  - Ejecutada 8 veces. 4 sin y 4 con libTTThwart activada
  - Resultado de la potencia de la CPU sin diferencia
  - Incremento del **3%** en el tiempo total de ejecución

# Limitaciones de libTTThwart

- Vulnerabilidades intrínsecas a la API de Linux
  - Diversas ventanas de vulnerabilidad más estrictas, pero igualmente explotables
- Creación de nuevos procesos mediante `fork()` y `exec()`
- Depende GNU C Library (GLIBC)
- Requiere binarios con enlazamiento dinámico

# Trabajo relacionado

- Modelos teóricos:
  - Modelos basados en no interferencia y propiedad conmutativa
  - Envolver a la condición de carrera con $k-$condiciones
  - Ataque laberinto

# Trabajo relacionado

- Modelos teóricos:
  - Modelos basados en no interferencia y propiedad conmutativa
  - Envolver a la condición de carrera con $k-$condiciones
  - Ataque laberinto
- Modelos con implementación:
  - Módulos del núcleo del sistema o librerías compartidas que interceptan las llamadas al sistema

# Trabajo relacionado

- Modelos teóricos:
  - Modelos basados en no interferencia y propiedad conmutativa
  - Envolver a la condición de carrera con $k-$condiciones
  - Ataque laberinto
- Modelos con implementación:
  - Módulos del núcleo del sistema o librerías compartidas que interceptan las llamadas al sistema
- Otras soluciones genéricas:
  - Analizadores estáticos

# Trabajo relacionado

- Modelos teóricos:
  - Modelos basados en no interferencia y propiedad conmutativa
  - Envolver a la condición de carrera con $k-$condiciones
  - Ataque laberinto
- Modelos con implementación:
  - Módulos del núcleo del sistema o librerías compartidas que interceptan las llamadas al sistema
- Otras soluciones genéricas:
  - Analizadores estáticos

## Librería libTTThwart

- Fácil de instalar. No requiere software adicional. Uso transparente
- Cantidad de funciones interceptadas
- Tiene en cuenta la reutilización del inodo
- Registra diversos metadatos

# Conclusiones

- TOCTOU es aún un gran reto debido a su no determinismo y sutileza. Difícil de detectar y de reproducir

# Conclusiones

- TOCTOU es aún un gran reto debido a su no determinismo y sutileza. Difícil de detectar y de reproducir
- Las soluciones propuestas actualmente no son definitivas, no impiden que la vulnerabilidad suceda

# Conclusiones

- TOCTOU es aún un gran reto debido a su no determinismo y sutileza. Difícil de detectar y de reproducir
- Las soluciones propuestas actualmente no son definitivas, no impiden que la vulnerabilidad suceda
  - Las soluciones basadas en la API son a su vez vulnerables

# Conclusiones

- TOCTOU es aún un gran reto debido a su no determinismo y sutileza. Difícil de detectar y de reproducir
- Las soluciones propuestas actualmente no son definitivas, no impiden que la vulnerabilidad suceda
  - Las soluciones basadas en la API son a su vez vulnerables
  - Es necesario modificar el núcleo del sistema operativo

# Conclusiones

- TOCTOU es aún un gran reto debido a su no determinismo y sutileza. Difícil de detectar y de reproducir
- Las soluciones propuestas actualmente no son definitivas, no impiden que la vulnerabilidad suceda
  - Las soluciones basadas en la API son a su vez vulnerables
  - Es necesario modificar el núcleo del sistema operativo
    - Uso exclusivo de descriptores de ficheros

# Conclusiones

- TOCTOU es aún un gran reto debido a su no determinismo y sutileza. Difícil de detectar y de reproducir
- Las soluciones propuestas actualmente no son definitivas, no impiden que la vulnerabilidad suceda
  - Las soluciones basadas en la API son a su vez vulnerables
  - Es necesario modificar el núcleo del sistema operativo
    - Uso exclusivo de descriptores de ficheros
    - Impedimento de creación de enlaces simbólicos

# Conclusiones

- TOCTOU es aún un gran reto debido a su no determinismo y sutileza. Difícil de detectar y de reproducir
- Las soluciones propuestas actualmente no son definitivas, no impiden que la vulnerabilidad suceda
  - Las soluciones basadas en la API son a su vez vulnerables
  - Es necesario modificar el núcleo del sistema operativo
    - Uso exclusivo de descriptores de ficheros
    - Impedimento de creación de enlaces simbólicos
- Cambios en el núcleo del sistema operativo complicados

# Conclusiones

- TOCTOU es aún un gran reto debido a su no determinismo y sutileza. Difícil de detectar y de reproducir
- Las soluciones propuestas actualmente no son definitivas, no impiden que la vulnerabilidad suceda
  - Las soluciones basadas en la API son a su vez vulnerables
  - Es necesario modificar el núcleo del sistema operativo
    - Uso exclusivo de descriptores de ficheros
    - Impedimento de creación de enlaces simbólicos
- Cambios en el núcleo del sistema operativo complicados
- Responsabilidad de los programadores

# Trabajo futuro

- Superar el límite de un único proceso utilizando implementaciones basadas en memoria compartida

# Trabajo futuro

- Superar el límite de un único proceso utilizando implementaciones basadas en memoria compartida
- Proteger la estructura de datos con mecanismos de sincronización y exclusión mutua

# Trabajo futuro

- Superar el límite de un único proceso utilizando implementaciones basadas en memoria compartida
- Proteger la estructura de datos con mecanismos de sincronización y exclusión mutua
- Mejorar el rendimiento de la librería rediseñando el algoritmo de búsqueda

# Trabajo futuro

- Superar el límite de un único proceso utilizando implementaciones basadas en memoria compartida
- Proteger la estructura de datos con mecanismos de sincronización y exclusión mutua
- Mejorar el rendimiento de la librería rediseñando el algoritmo de búsqueda
- Ampliar la lista de funciones interceptadas contemplando nuevas versiones de Linux

# Gracias por su atención
# ¿Preguntas?

Razvan Raducu
rrad@unileon.es