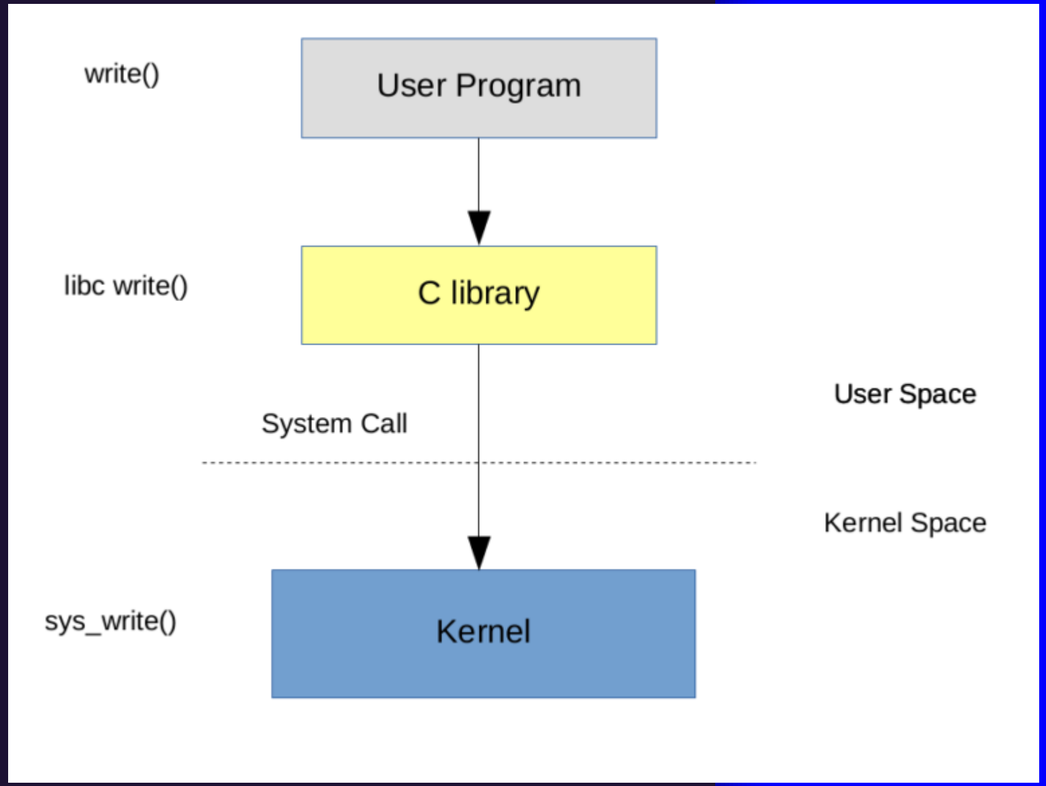
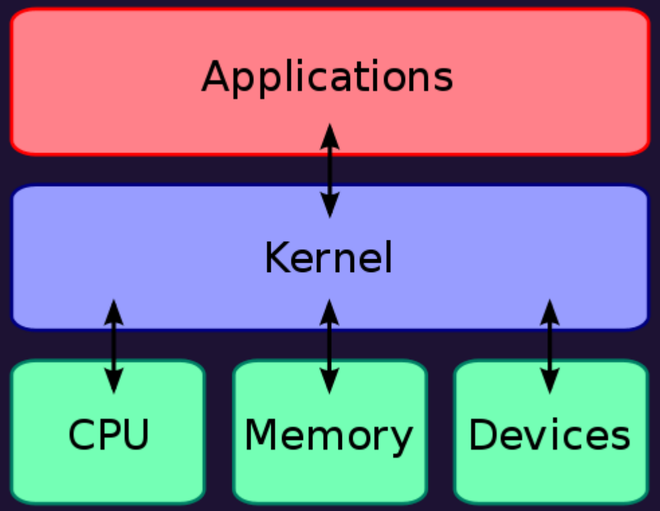
Exploiting Concurrency Vulnerabilities in System Call Wrappers

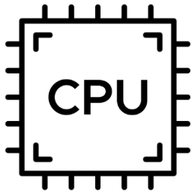
Antonio Roblero Alejandro



Concurrencia y kernel

Los núcleos del sistema operativo son programas altamente concurrentes que consumen servicios de concurrencia internamente y los ofrecen a los programas, la mayoría de los sistemas de escritorio y servidos suelen soportar multiprocesamiento.





CPU

El sistema tiene la capacidad de ejecutar múltiples procesos al mismo tiempo, pero también conduce a la competencia de recursos y el intercambio entre varios procesos en el sistema.

Concurrencia

Debido a la introducción de interrupciones, mecanismos de excepción y preferencia del modo kernel, estas rutas de ejecución (procesos) del kernel se ejecutan de manera intercalada.

Manejo de concurrencia

La gestión de la concurrencia es de los problemas centrales en la programación de sistemas operativos. Los errores relacionados con la concurrencia son algunos de los más fáciles de crear y algunos de los más difíciles de encontrar.

Wrapper

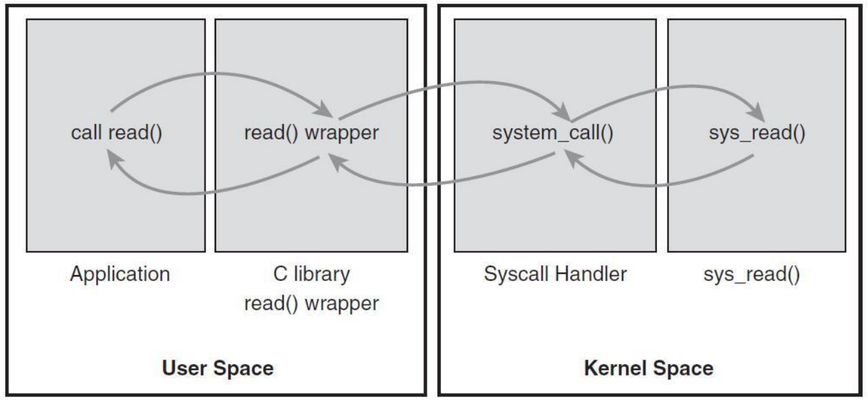
Los contenedores controlan la interacción de la función original con el entorno, desde la vista de patrón de diseño, se le puede considerar un proxy o un decorador.

Interposición a llamadas del sistema con wrappers

Los contenedores de llamadas al sistema pueden cumplir con el rol del monitor ya que se ejecutan en el dominio del kernel, se invocan en la ruta de la llamada del sistema y evitan modificar el kernel de forma compleja.

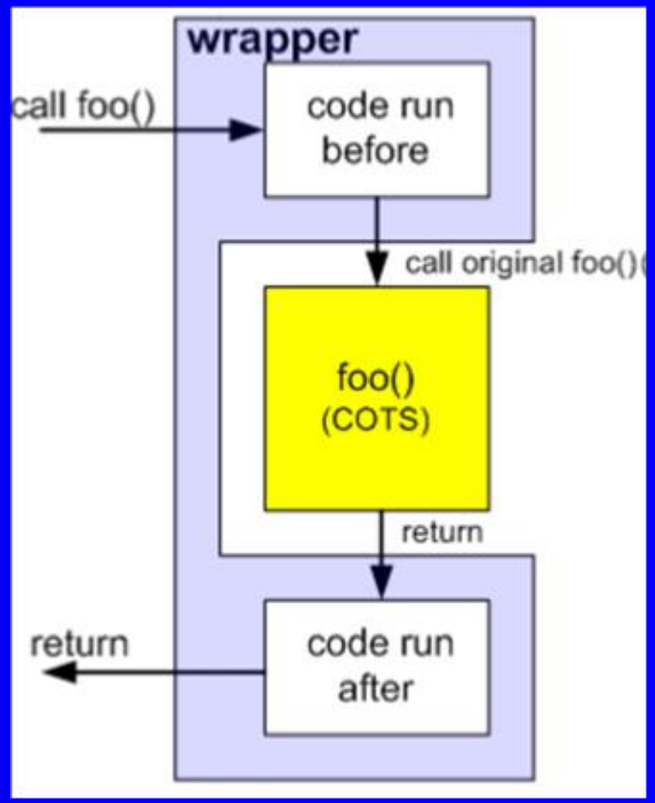
Wrapper y seguridad

Los contenedores de llamadas al sistema y su posible uso para supervisor los programas.



Los envoltorios de llamadas al sistema implementan el procesamiento de condiciones previas y posteriores alrededor de la llamada al sistema.

- Condición previa: permite que el contenedor inspeccione o sustituya los argumentos antes de pasarlo al kernel.
- Condición posterior: ayuda a que el contenedor rastree y registre los resultados, transforma los valores devueltos, etc.



Vulnerabilidades de la concurrencia

- **Condición de carrera.**

  Las condiciones de carrera surgen como resultado del acceso compartido a los recursos. Cuando dos subprocesos de ejecución tienen una razón para trabajar con los mismos recursos, siempre existe la posibilidad de que se produzcan confusiones.

- **Vulnerabilidades debido a la concurencia.**
  - La carrera de datos.
  - Perdida de información.
  - Problemas de rendimiento.

Ataques de concurrencia en wrappers

1

Time-of-check-to-time-of-use (TOCTTOU):

Vulnerabilidades, en las que las comprobaciones de control de acceso no son atómicas con las operaciones que protegen

2

Time-of-audit-to-time-of-use (TOATTOU):

Vulnerabilidades, para ocultar los datos de cualquier registro que se pueda realizar, cubriendo las pistas de un exploit de una aplicación de detección de intrusos..

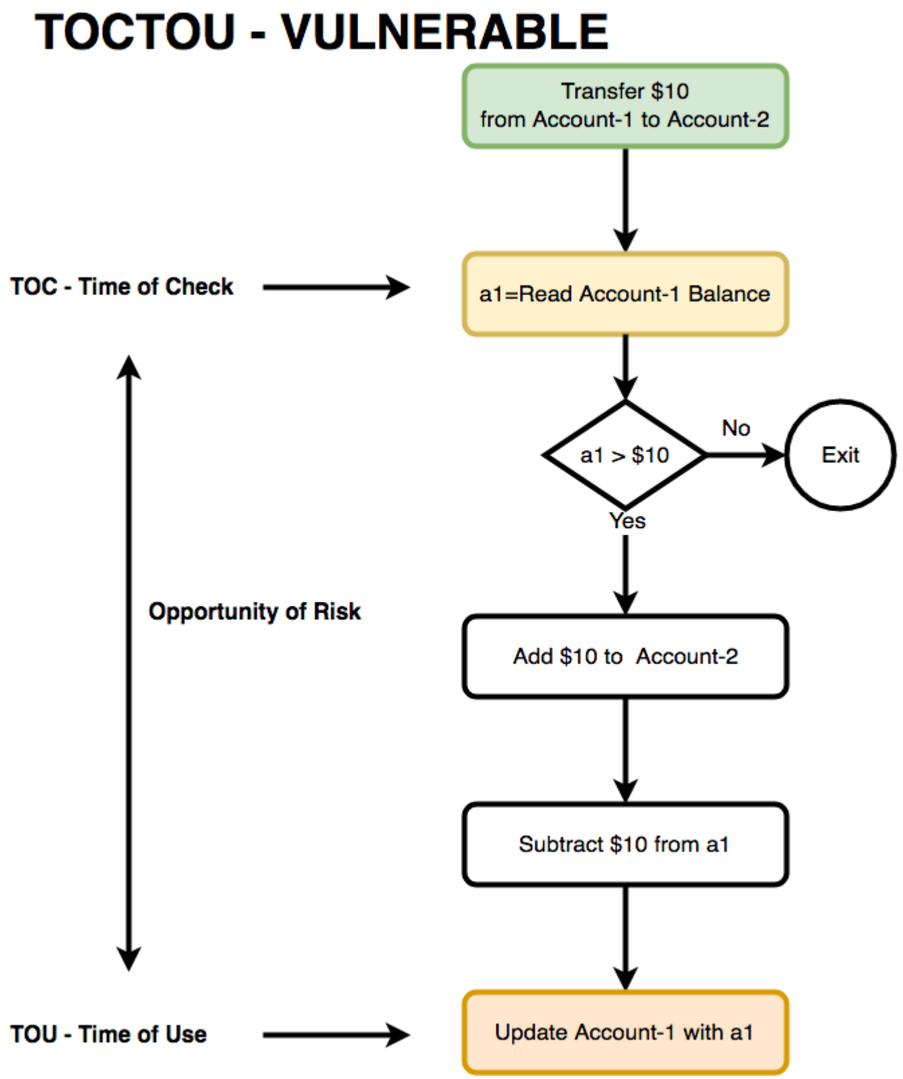
3

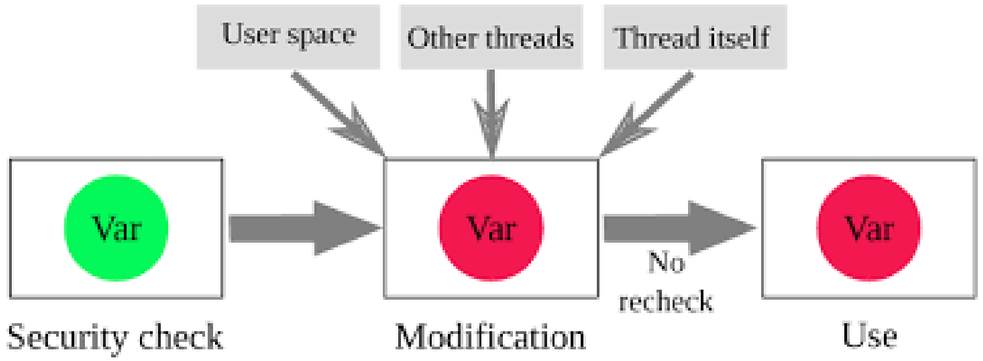
Time-of-replacement-to-time-of-use (TORTTOU):

Vulnerabilidades, en las que los atacantes modifican los argumentos de las llamadas al sistema después de que un envoltorio los ha reemplazado, pero antes de que el kernel haya accedido a ellos.

Time-of-check-time-of-use (TOCTTOU).

La falla ocurre cuando un programa busca una característica de un objeto, y luego toma alguna acción que asume que la característica aún se mantiene cuando en realidad no es así.





Descripción del problema

La falla ocurre cuando un programa busca una característica particular de un objeto, y luego toma alguna acción que asume que la característica aún se mantiene cuando en realidad no es así. Una subclase de defectos de TOCTTOU surge cuando se asume erróneamente que los identificadores de objeto permanecen ligados a un objeto.

Demostración del problema.

La falla simbólica arquetípica de TOCTTOU en un programa privilegiado en el sistema operativo UNIX surge cuando un programa setuid to root requiere guardar datos en un archivo propiedad del usuario que ejecuta el programa. El programa no debe alterar el archivo a menos que el usuario pueda alterar el archivo sin privilegios especiales.

- Se parte del hecho que el usuario puede modificar el archive “/tmp/X”
- Antes de que ocurra la llamada al sistema *open,* se elimina “/tmp/X” y se crea un enlace fijo para “/etc/passwd”
- Entonces *open* accede a los datos asociados con “/etc/passwd” cuando abre “/tmp/X”, ya que “/tmp/X” y “/etc/passwd” apuntan al mismo archivo.

```
if (access(filename, W_OK) == 0)
{
    if ((fd = open(filename, O_WRONLY)) == NULL)
    {
        perror(filename);
        return(0);
    }
    /* now write to the file */
```

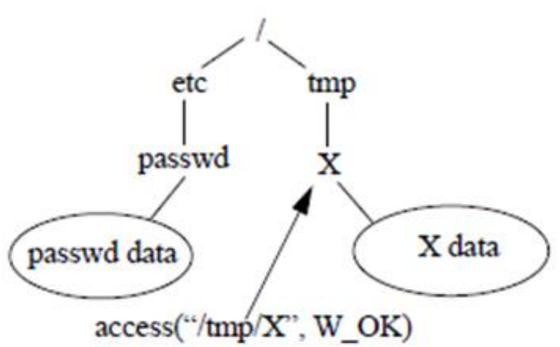


Figure 1a.

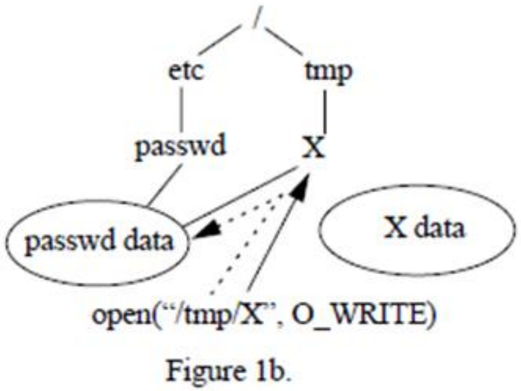


Figure 1b.

Caracterización de TOCTTOU

Condición de carrera

Una falla TOCTTOU ocurre en dos eventos donde el segundo depende del primero. Durante el intervalo entre dos eventos, ciertas suposiciones de los resultados de la primera llamada al sistema influyen en el segundo.

Condición de programación

Se denomina condición de programación a la existencia del intervalo entre el primer y el segundo evento.

Intervalo de programación

Al intervalo mismo se le llama intervalo de programción

Condición de entorno

Cuando se tiene una condición de carrera, si el atacante puede afectar las suposiciones creadas por el primer evento del programa, se tiene una condición de entorno.

Retomando el problema.

- La llamada del sistema *access* crea la suposición de que el usuario está autorizado a modificar el archivo “/tmp/X”.
- *Open* actúa sobre esa suposición. Entonces la condición de programación se mantiene.
- Si el atacante puede alterar la referencia del nombre “/tmp/X”, entonces la condición de entorno también se cumple y existe una falla TOCTTOU explotable.

```
if (access(filename, W_OK) == 0)
{
    if ((fd = open(filename, O_WRONLY)) == NULL)
    {
        perror(filename);
        return(0);
    }
    /* now write to the file */
```

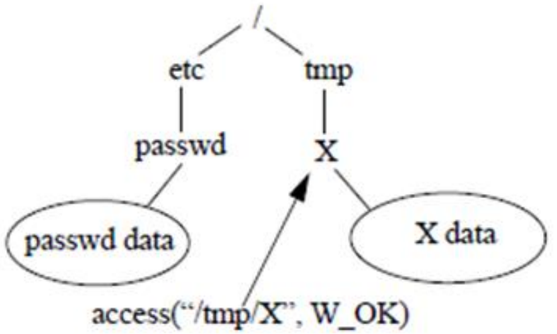


Figure 1a.

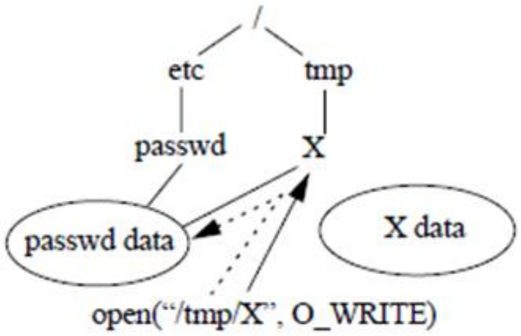


Figure 1b.

Técnicas de mitigación.

Lee Badger ha sugerido un enfoque de detectar y mitigar los cambios de explotación en el estado del kernel a través de una condición posterior, tomando medidas correctivas si se ha producido una infracción.

Provos proporciona facilidades similares en Systrace, copiando argumentos indirectos en el “espacio de pila”, un área reservada de la memoria del proceso, lo que permite que los contenedores sustituyan los argumentos indirectos de mayor tamaño que el argumento original.

Ghormley aborda las carreras de argumentos en el marco de interposición de SLIC a través de búferes en el núcleo que extienden el espacio de direcciones.

Conclusiones.

- Los ataques a la concurrencia son muy viables.
- No usar de manera irresponsible los contenedores a las llamadas del sistema
- Las vulnerabilidades debido a las condiciones de carrera pueden ser prevenidas con un poco de pensamiento, las primitivas de control de concurrencia del kernel, etc.
- Las ténicas de mitigación adolecen de vulnerabilidades, así como degradación semántica o del rendimiento

Referencias.

- Watson, R. N. (2007). Exploiting Concurrency Vulnerabilities in System Call Wrappers. WOOT.
- Fraser, T., Badger, L., & Feldman, M. Hardening COTS software with generic software wrappers. IEEE.
- Provos, N. (2003, August). Improving Host Security with System Call Policies. In USENIX Security Symposium
- Ghormley, D. P., Rodrigues, S. H., Petrou, D., & Anderson, T. E. (1998, June). SLIC: An Extensibility System for Commodity Operating Systems. In USENIX Annual Technical Conference (Vol. 98).
- Bishop, M., & Dilger, M. (1996). Checking for race conditions in file accesses. Computing systems.
- Zhao, S., Gu, R., Qiu, H., Li, T. O., Wang, Y., Cui, H., & Yang, J. (2018, June). Owl: Understanding and detecting concurrency attacks. In 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. IEEE.
- Snyder, D. (2001). On-line intrusion detection using sequences of system calls.
- Garfinkel, T. (2003). Traps and pitfalls: Practical problems in system call interposition based security tools. In In Proc. Network and Distributed Systems Security Symposium.
- Edge, J. (2007). Exploiting races in system call wrappers [LWN.net]. Consultado el 19 de Julio de 2021, en https://lwn.net/Articles/245630/.
- Threads. (2021). Consultado el 19 de Julio de 2021, en https://comic.browserling.com/53
- Beautiful Free Images & Pictures | Unsplash. (2021). Retrieved 19 July 2021, from https://unsplash.com/