Sisteme de operare avansate

Exploiting Concurrency Vulnerabilities in System Call Wrappers

Robert N. M. Watson
Computer Laboratory
University of Cambridge
robert.watson@cl.cam.ac.uk

Interceptare apeluri de sistem

- Folosirea de wrappere pentru apeluri de sistem
  - nivel kernel
  - nivel userspace
- Verificări de securitate suplimentare
- Folosită în research şi comercial (anti-viruşi)
- Există probleme de securitate
  - datorită concurenţei

Concurenţa în SO moderne

- Fizică
  - multiprocesoare
- Logică
  - în userspace: multi-procese, multi-threading, async IO, semnale
  - în kernel: întreruperi, kernel thread-uri

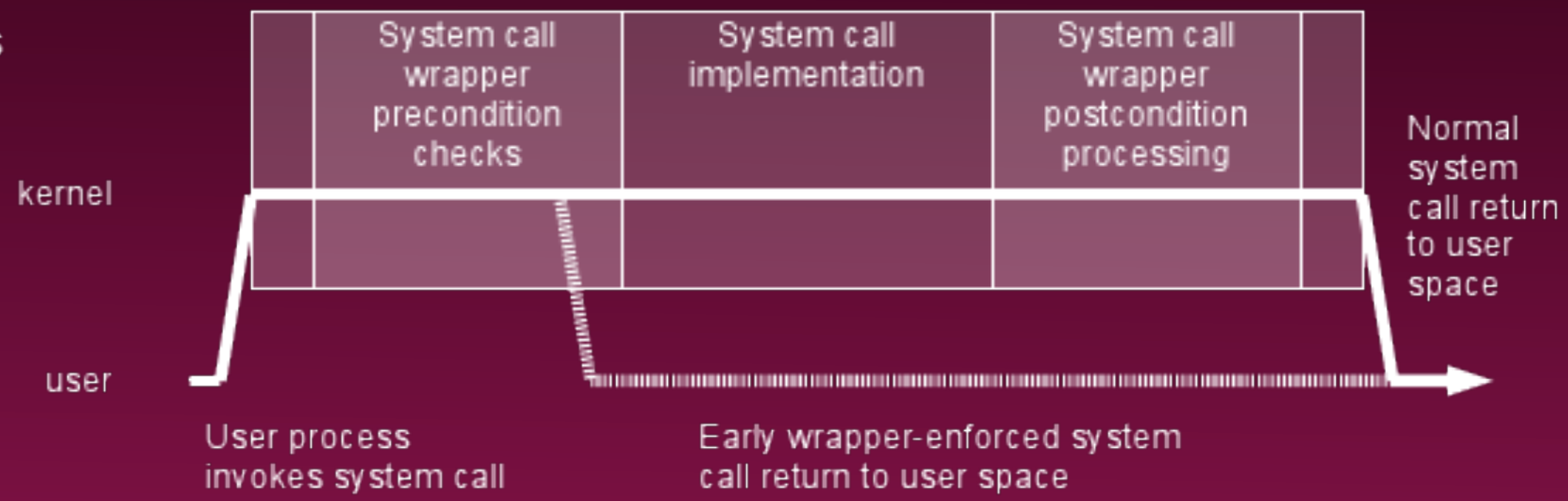
System call wrappers

- Construite ca un reference monitor
  - tamper proof
  - always invoked
  - small enough to be tested
- System Call Wrappers
  - rezidă de obicei în kernel
  - necesită modificări minime în kernel
  - sunt invocate la fiecare apel de sistem
  - portabile accros APIs (UNIX)

Implementare tipică în kernel

- Interceptează trap-ul apelului de system şi inserează
  - verificări pre apel de sistem
    - inspectare argumente, eventual înlocuirea acestora
    - respingerea unui apel de sistem
  - verificări post apel de sistem
    - menţinerea de informaţii pentru cazurile în care e nevoie de stare
    - logarea rezultatului apelului de sistem
    - modificarea rezultatului

System call wrappers [2]



Politica de securitate

- Verificarea se face în concordanţa cu o politică de securitate
- Definită de utilizator
- În general statică
- Politica poate fi:
  - compilată în modului de kernel
  - definită şi încărcată din userspace

Vulnerabilităţi datorate concurenţei

- Politica de securitate nu este respectate datorită unor probleme de concurenţă
- Efecte:
  - denial of service
  - leaking of sensitive data
  - incorrect access control

Atacuri asupra bazate pe concurenţă asupra SCW-relor

- Buguri de sincronizare în SCW
- Buguri ce exploatează non-atomicitatea dintre kernel şi SCW
  - nesincronizare dintre SCW şi kernel la copierea argumentelor (syntactic race condition)
  - nesincronizarea dintre SCW şi kernel la interpretarea argumentelor (semantic race condition)

Sintactic race conditions

- “Portabile” pe mai multe kernele sau SCW-uri
- Sunte probleme fundamentale
  - nu pot fi rezolvate fără schimbări majore în arhitectură
- Time-of-audit-to-time-of-use (TOATTOU)
  - logurile nu reflectă corect acţiunile efectuate

Sintactic race conditions [2]

- Time-of-check-to-time-of-use (TOCTTOU)
  - operaţiile de verificare nu sunt atomice cu operaţiile pe care le protejează
- Time-of-replacement-to-time-of-use (TORTTOU)
  - argumentele apelurilor de sistem sunt înlocuite între verificare în SCW şi folosirea lor în kernel

Tehnici de exploatare

- Identificarea unor resurse partajate între user, SCW şi kernel
- Argumentele directe sunt pasate în registri dar argumentele indirecte...
- .. sunt pasate prin Userspace memory

Argumente indirecte

- Exemple:
  - file paths
  - adrese de socket
  - resource limits
- Aceste argumente sunt copiate de două ori
  - de SCW pentru a face verificările
  - de kernel pentru a executa apelul de sistem

Condiţii de exploatare race-uri

- Concurenţă
  - se poate obţine cu procese multiple, thread-uri, semnale, async IO
- Accesul la memorie de către cel puţin două entităţi
  - pentru procese multiple: memorie partajată prin moştenire (minherit, rfork, clone)
  - thread-uri

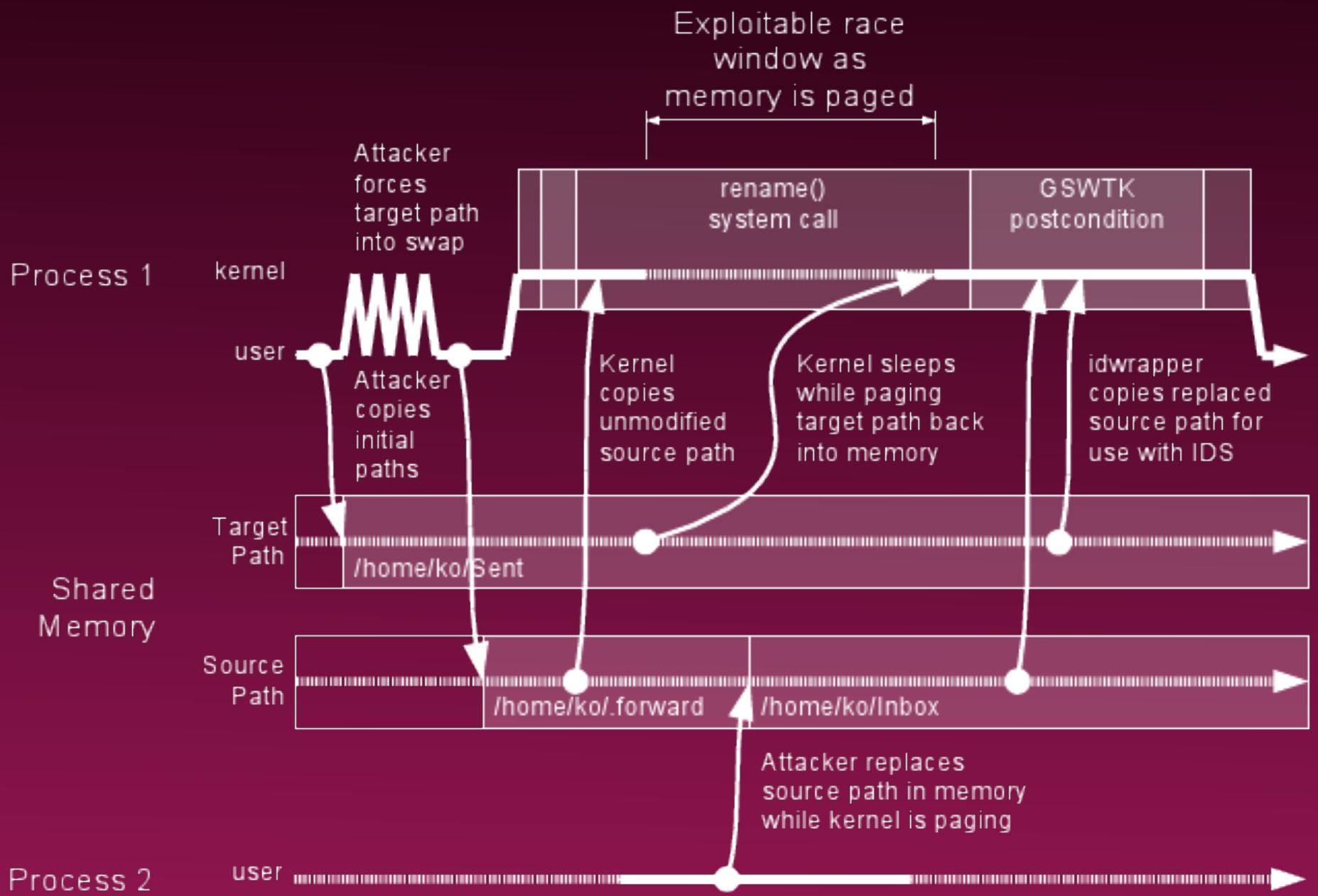
Exploatare race-uri pe sisteme uni-procesor

- Atacatorul trebuie sa determine kernel-ul să cedeze procesorul între pre-condition şi kernel sau între kernel şi post-conditions
  - voluntară: blocking IO pe un socket sau disk
  - involuntară: page fault
    - daca pagina cu argumente este în swap, deschide o fereastră de câteva ms în care se poate schimba argumentul

Atac cu page fault-uri

- rename(from, to)
- from este în memorie, to este swapat
- SCW va accessa from şi verifica accesul după care va incerca să acceseze to, se va genera un page fault şi se va porni operaţia de swap-in
- Între timp, controlul ajunge la un helper thread care modifică from

Atac cu page fault-uri [2]



Cazul unui singur argument

- Argumentul se plasează între două pagini una în memorie alta în swap
- Se suprascrie doar bucata din prima pagină
  - în ideea ca deja a fost citită de SCW

Atac cu cedare voluntară

- connect asteaptă primirea unui SYN-ACK
- În acest timp atacatorul poate schimba argumentele (de exemplu adresa la care se face conectarea)
- Post-condiţia va evalua o adresa falsă

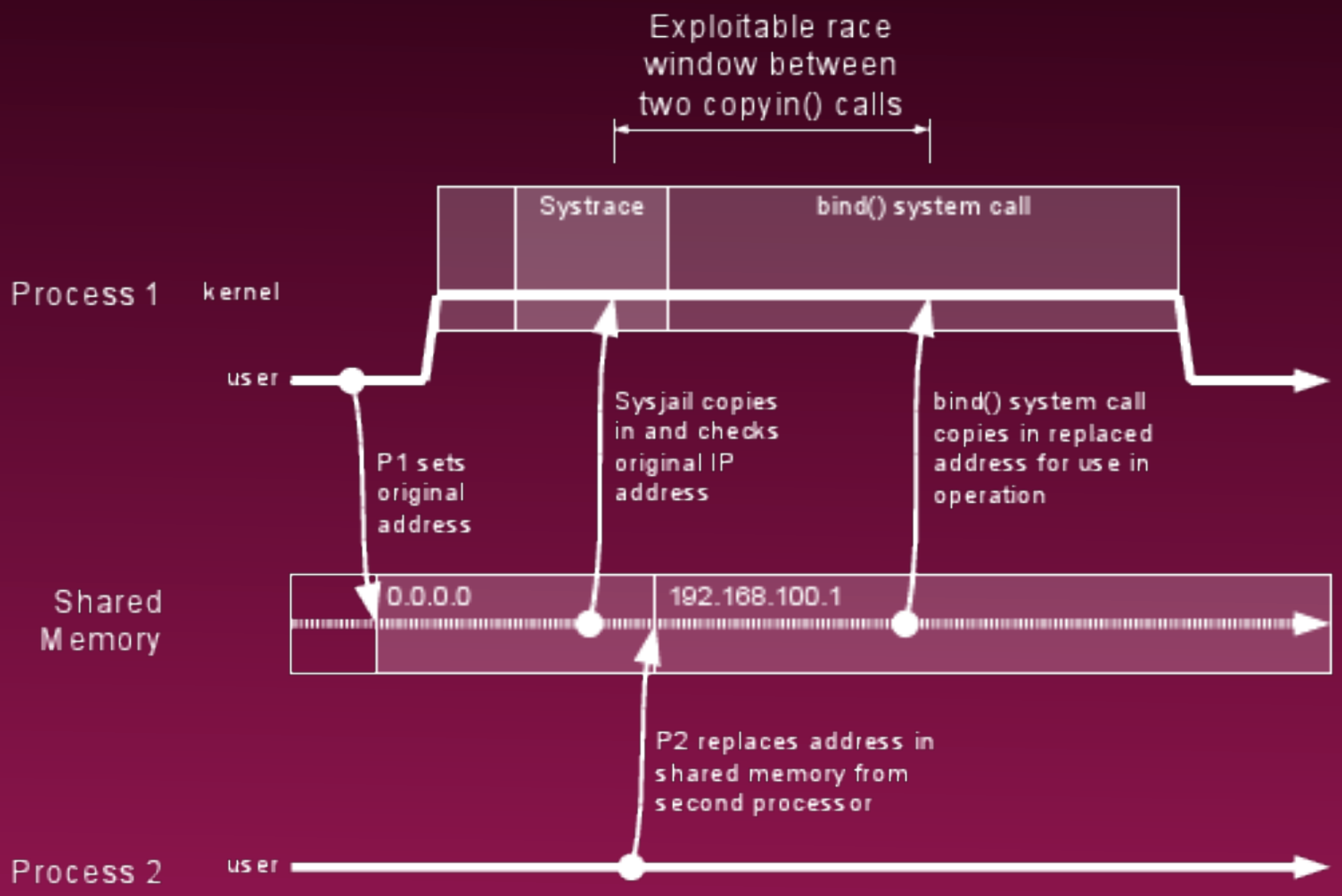
Exploatare race-uri pe sisteme multiprocesor

- Se foloseşte de paralelismul oferit de hardware
- Trebui determinată fereastra de timp în care se poate face înlocuirea
  - se foloseşte un timer de mare precizie (TSC) pentru a temporiza înlocuirea + binary search pentru a determina fereastra de vulnerabilitate
  - la argumentele care sunt inlocuite de SWC, urmărirea în buclă a faptului că argumentul e modificat

Exploatare race-uri pe sisteme multiprocesor [2]

- Rezultatele experimentale indică o fereastră de la 5K-15K cicli (GSWTK) până la 100K (systrace)
- Mai mult decât suficentă pentru atacuri

Exemplu


Exploitable race window between two copyin() calls
Systrace
bind() system call
Process 1
kernel
user
P1 sets original address
Sysjail copies in and checks original IP address
bind() system call copies in replaced address for use in operation
Shared Memory
0.0.0.0
192.168.100.1
P2 replaces address in shared memory from second processor
Process 2
user

Generic Software Wrapper Toolkit

- Sistem generic, bazat de wrappere configurabile
- Wrapperele sunt scrise folosind un limbaj gen C extins cu suport SQL
- Rulează pe Solaris, FreeBSD, BSD/OS, şi Linux
- Există o multitudine de wrappere, de la sisteme de control al accesului până la sisteme de detecţie a intruziunilor

Din 23 de wrappere inspectate 16 au fost identificate ca vulnerabile şi exploate cu succes

Wrapper	Description	Vulnerabilities
`callcount`	Count system calls	None: relies on the system call number.
`conwatch`	Track IP connections by processes.	Postcondition TOATTOU race on `connect()` and `bind()` allows masking address/port used.
`dbfencrypt`	Encrypt files with '$' in their names; prevent rename so that encryption policy on a file cannot be changed.	Postcondition TOCTTOU race allows incorrect name in policy check; precondition TORTTOU races on I/O write unencrypted data and bypass rename checks.
`dbexec`	Authorize execution of programs based on a pathname database.	Precondition TOCTTOU race allows bypass by substituting the name during the wrapper check.
`dbsynthetic`	Synthetic file system sandbox substituting pathnames from a database.	Precondition TORTTOU race bypasses path replacement; postcondition TORTTOU race leaks true paths
`life`	Prints the process life cycle.	Precondition TOATTOU race replaces `exec()` paths.
`noadmin`	Deny all privileged operations.	None: relies on the kernel's process credential.
`aks.wr`	Audit file operations	Pre/postcondition TOATTOU races avoid audit.
`seq-kernel.wr`	Sequence-based intrusion detection	None: relies on the system call number.
`imapd.wr`	Detect anomalous access by `imapd`.	Postcondition TOATTOU path races prevent alerts.

Systrace

- Sistem prin care un proces poate controla un target process prin inspectarea şi modificarea argumentelor apelurilor de sistem
- OpenBSD, NetBSD; ports pentru Linux, Mac OS X, FreeBSD
- Au fost testate module sudo monitor mode şi sysjail

sudo monitor mode

- se interceptează execve şi se face audit la comenzile executate
- Politica este menţinută în user-space
  - sunt necesare context switch-uri pentru a citi şi acţiona conform politicii
- Pe sisteme MP fereastra de vulnerabilitate a fost determinată la 430K cicli
- Atacul a reuşit înlocuirea argumentelor, mascând astfel acţiunile în log

sysjail

- ruleză procesele ce sunt necesare a rula ca root
  - pentru a minimiza impactul asupra unei posibile vulnerabilităţi în aplicaţie
- se ataşează la toate procesele ce rulează în jail
- validează sau în anumite cazuri rescrie argumentele apelurilor de sistem

Exemplu bind

- adresa la bind trebuie să fie
  - egală cu cea configurată pe jail
  - INADDR_ANY, caz în care jail-ul o va înlocui cu cea configurată pe jail
- S-a reuşit exploatare pe un sistem MP prin inlocuirea adresei de bind

CernNG

- Similar cu GSWTK
- S-au exploatat vulnerabiliţăţi de tipul TOATTOU şi TOCTTOU
- În ciuda unor metode de protecţie targetate împotriva unor astfel de atacuri (bazate pe memorie virtuală)

Cum se pot preveni aceste race-uri?

- Tehnici de mitigare
  - se modifică SCW-urile pentru a ne proteja la aceste tipuri de atacuri
- Modificarea sistemului de operare dar păstrarea SCW-urilor
- Renunţarea la SCW-uri

Tehnici de mitigare

- Detectarea modificării argumentelor la post-condiţie şi roll-back
  - rollback-ul nu este practic pentru apeluri de sistem complexe
  - detectarea modificării argumentelor poate fi şi ea exploatată cu aceleaşi tehnici

Tehnici de mitigare [2]

- Marcarea paginilor care menţin argumentele ca RO pe durată apelului de sistem
  - se poate face doar la nivel de pagina; dacă alte thread-uri stochează date în acea pagină...
  - trebuie projate toate paginile mapate asociate cu acea pagina fizică
  - memory mapped files: trebuie avute în vedere şi operaţia de write()
  - trebuie avut în vedere şi faptul că aplicaţia poate face remap cu RW

Tehnici de mitigare [3]

- Stack gap
  - argumentele indirecte se copiză într-un zonă rezervată, pe stivă
- Probleme
  - de implementare: zona este accesibilă din alte thread-uri şi poate fi mapată chiar şî în alte procese
  - copieri în plus

Tehnici de mitigare [4]

- Concluzii
  - greu de implementat
  - probleme fundamentale
  - În practică ineficiente: atât “stack gap”-ul cât şi sistemul de VM protection oferit de CerbNG au putut fi ocolite

Message Passing Systems

- Argumentele apelurilor de sistem sunt trimise sistemului de operare într-un bundle
- Elimină race-urile sintactice, nu şi cele semantice
- Trap-ul ce se ocupa cu tratarea apelurilor de sistem trebui să aibă ştie formatul argumentelor

Renunţarea la SCW

- Integrarea unui framwork de securitate în kernel
- Modelul cel mai larg adopdat în curent de comunitatea “OS security”: LSM, SeLinux, SEBSD, SEDarwin