Curs 5

Exploiting Concurrency Vulnerabilities in System Call Wrappers

Robert N. M. Watson

Operating Systems Practical

30 Octombrie, 2013

Cuprins

Outline

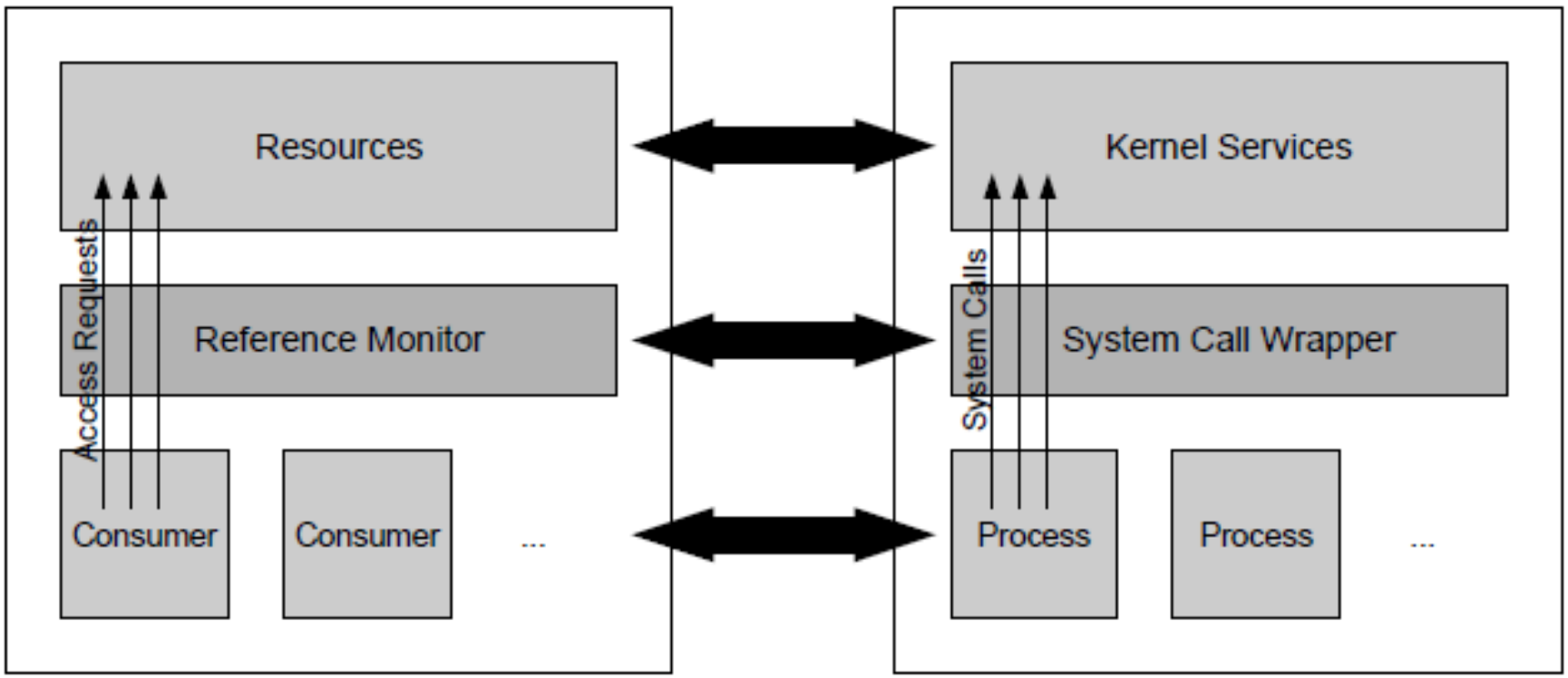
Interceptare apeluri de sistem

- Folosirea de wrappere pentru apeluri de sistem
  - nivel kernel
  - nivel userspace
- Verificări de securitate suplimentare
- Folosită în research și comercial (anti-viruși)
- Există probleme de securitate
  - datorită concurenței

Concurența în SO moderne

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

Arhitectura unui engine de interceptare apeluri de sistem

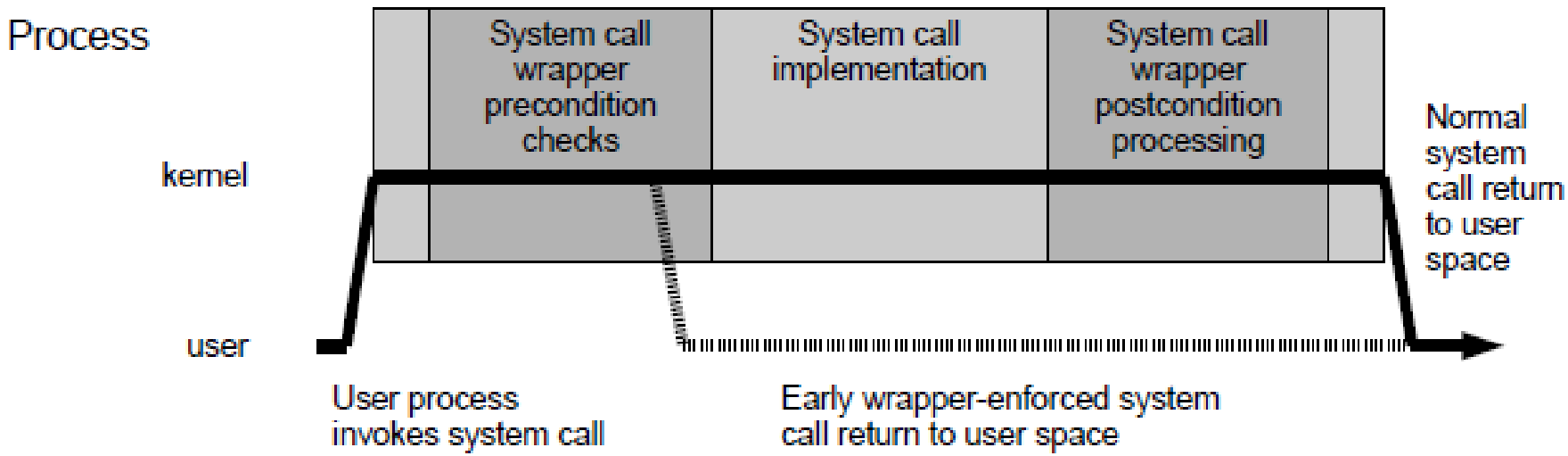


Outline

System call wrappers

- Implementate cu ajutorul unui monitor
  - rezidă de obicei în kernel
  - necesită modificări minime în kernel
  - sunt invocate la fiecare apel de sistem
  - portabile accros APIs (UNIX)
- Avantaje:
  - nu poate fi accesat de aplicații
  - complexitate scăzută -¿ ușor de analizat
  - este întotdeauna invocat

System call wrappers (2)



Implementare tipică în kernel

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

Politica de securitate

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

Vulnerabilități datorate concurenței

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

Atacuri asupra bazate pe concurență asupra SCW-relor

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

Semantic race conditions

- Sunt posibile datorită complexității unor operații
- Kernelul poate implementa operația într-un mod diferit de cel în care SCW se așteaptă să îl implementeze
- Odată identificate sunt ușor de fixat (nu sunt probleme fundamentale)

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 acțiunile efectuate

Sintactic race conditions (2)

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

Tehnici de exploatare

- Identificarea unor resurse partajate între user, SCW și 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

Condiții de exploatare race-uri

- Concurență
  - se poate obține cu procese multiple, thread-uri, semnale, async IO
- Accesul la memorie de către cel puțin două entități
  - pentru procese multiple: memorie partajată prin moștenire (minherit, rfork, clone)
  - thread-uri

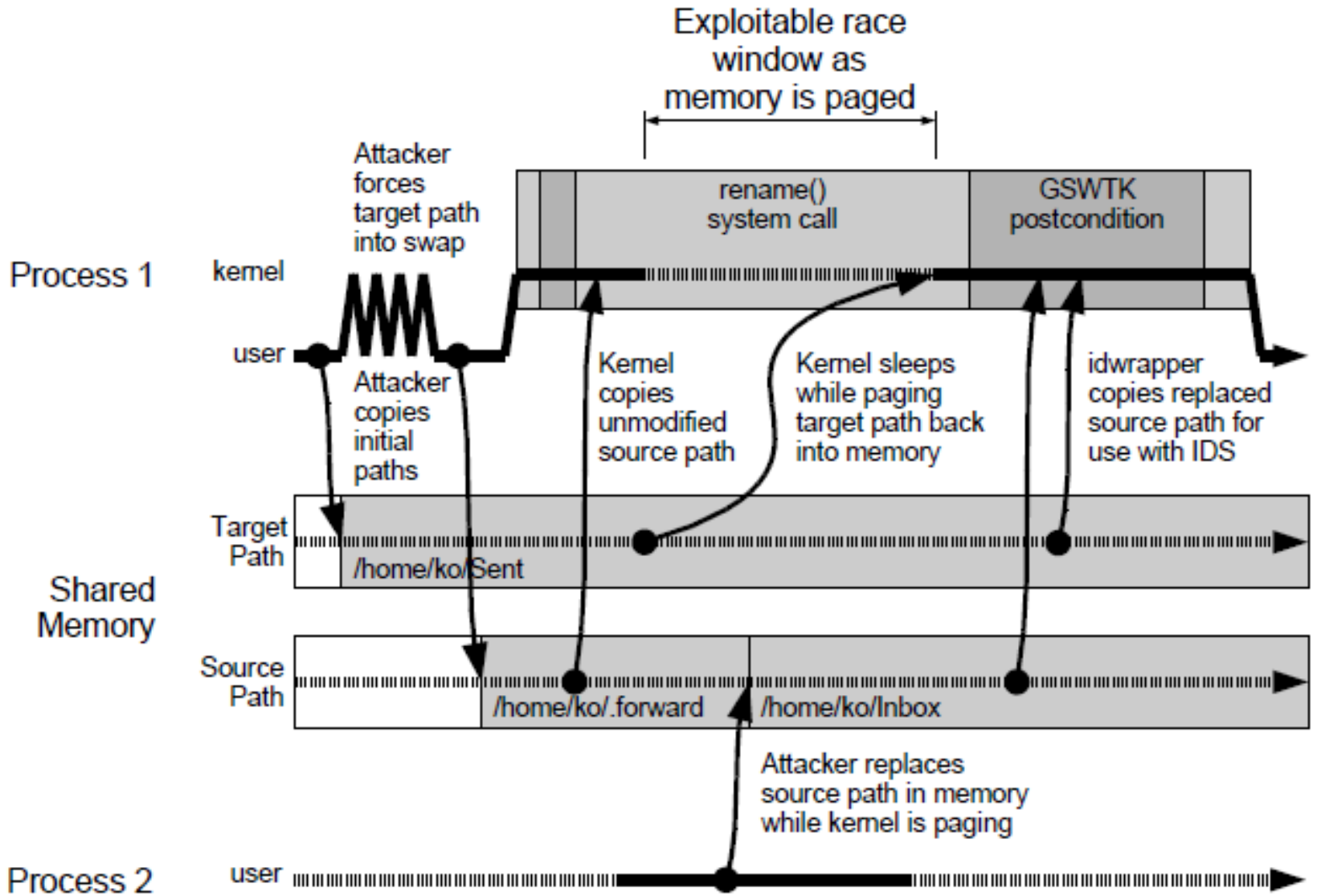
Exploatare race-uri pe sisteme uni-procesor

- Atacatorul trebuie sa determine kernel-ul să cedeze procesorul între pre-condition și kernel sau între kernel și 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 in memorie, to este swapat
- SCW va accessa from și verifica accesul după care va incerca să acceseze to, se va genera un page fault și se va porni operația 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

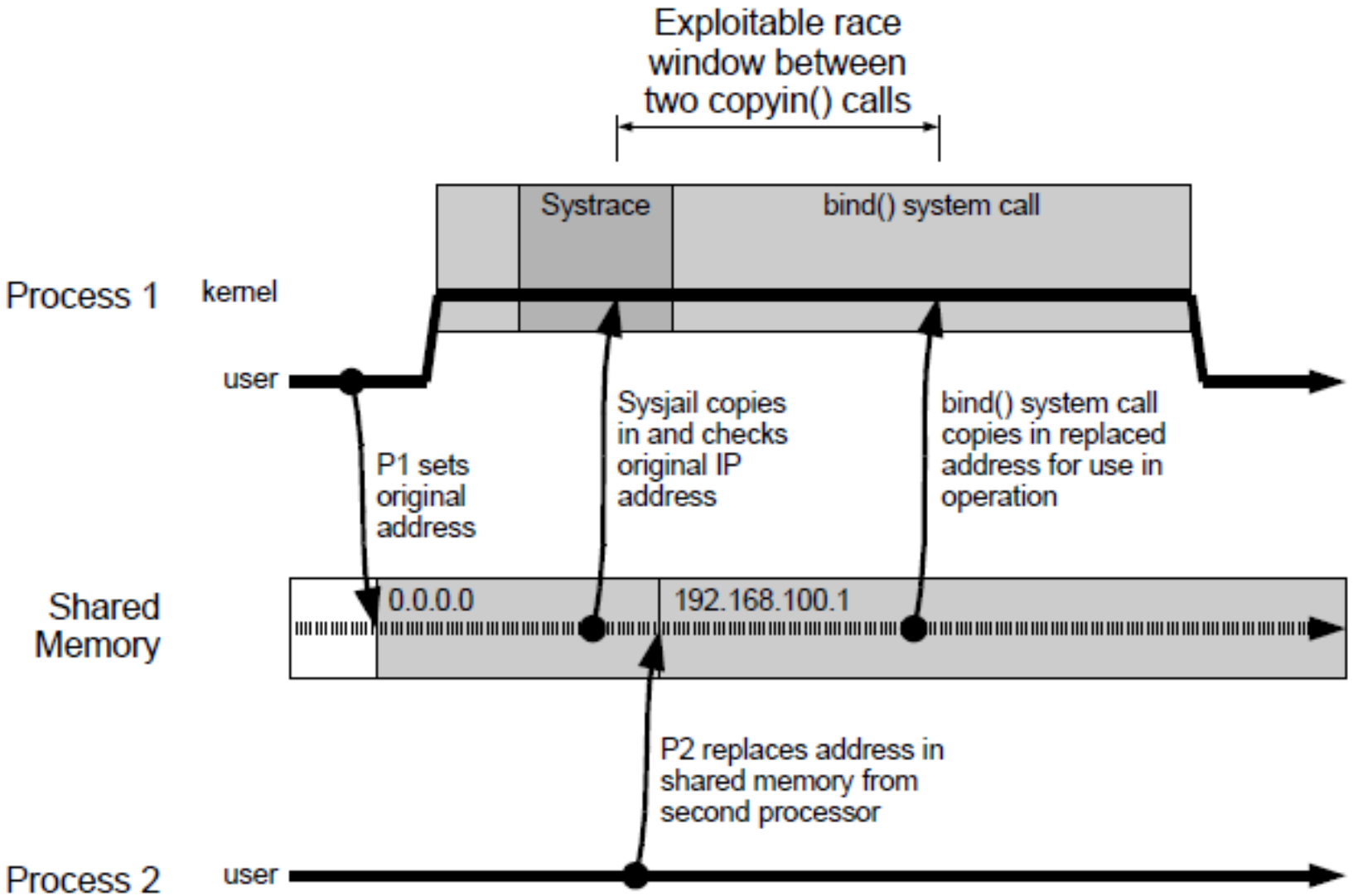
- connect asteaptă primirea unui SYN-ACK
- În acest timp atacatorul poate schimba argumentele (de exemplu adresa la care se face conectarea)
- Post-condiția va evalua o adresa falsă

Exploatare race-uri pe sisteme multiprocesor

- Se folosește de paralelismul oferit de hardware
- Trebui determinată fereastra de timp în care se poate face înlocuirea
  - se folosește 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


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, și Linux
- Există o multitudine de wrappere, de la sisteme de control al accesului până la sisteme de detecție a intruziunilor

- Din 23 de wrappere inspectate 16 au fost identificate ca vulnerabile și 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 și modificarea argumentelor apelurilor de sistem
- OpenBSD, NetBSD; ports pentru Linux, Mac OS X, FreeBSD
- Au fost testate module sudo monitor mode și sysjail

- Se interceptează execve și se face audit la comenzile executate
- Politica este menținută în user-space
  - sunt necesare context switch-uri pentru a citi și acționa conform politicii
- Pe sisteme MP fereastra de vulnerabilitate a fost determinată la 430K cicli
- Atacul a reușit înlocuirea argumentelor, mascând astfel acțiunile în log

- Rulează procesele ce sunt necesare a rula ca root
  - pentru a minimiza impactul asupra unei posibile vulnerabilități în aplicație
- Se atașează 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 reușit exploatare pe un sistem MP prin inlocuirea adresei de bind

- Similar cu GSWTK
- S-au exploatat vulnerabilități de tipul TOATTOU și TOCTTOU
- În ciuda unor metode de protecție 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
- Renunțarea la SCW-uri

Tehnici de mitigare

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

Tehnici de mitigare (2)

- Marcarea paginilor care mențin 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 și operația de write()
  - trebuie avut în vedere și faptul că aplicația 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 și poate fi mapată chiar și în alte procese
  - copieri în plus

Tehnici de mitigare (4)

- Concluzii
  - Greu de implementat
  - Probleme fundamentale
  - În practică ineficiente: atât “stack gap”-ul cât și 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 și cele semantice
- Trap-ul ce se ocupa cu tratarea apelurilor de sistem trebui să aibă știe formatul argumentelor

Renunțarea la SCW

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

Outline

Cuvinte cheie

- apeluri de sistem
- concurență
- kernel Linux
- politica de securitate
- wrappers
- race
- page fault
- stack gap

Outline

Resurse

- Robert N. M. Watson - Exploiting Concurrency Vulnerabilities in System Call Wrappers

Outline

Întrebări

?