Race condition

Ko želva stavi na srečo, zajec pa na “symlink” napad.

Jure Škofič

ACROS d.o.o.
www.acrossecurity.com

16.6.2010

Kazalo

Nevarna dirka

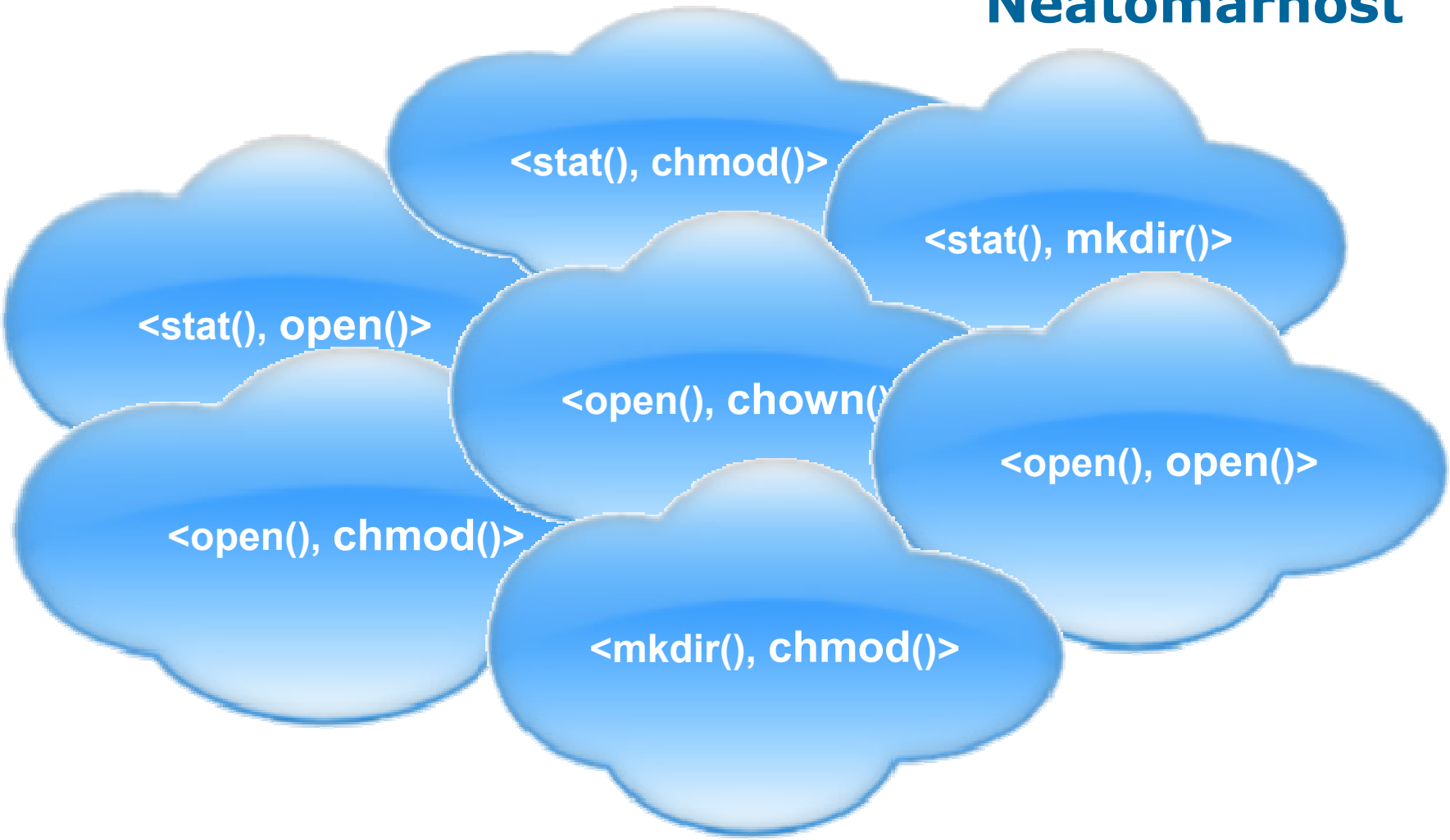
1. TOCTTOU Race condition nastopi ko:

- Preverimo stanje objekta.
- Stanje objekta se spremeni.
- Izvedemo akcijo nad objektom, na podlagi preverjenega stanja.

2. Linux okolje in TOCTTOU race condition ranljivosti

- Race condition ranljivosti najdemo na večini modernih večopravilnih operacijskih sistemih.
- Okolje Linux nudi vsa potrebna orodja za iskanje in analizo tovrstnih varnostnih napak.
- Linux nudi tudi vsa potrebna orodja za učinkovito izkoriščanje race condition ranljivosti.

Neatomarnost



Pogoji za uspešno izkoriščanje race conditionov na datotečnih objektih

- Aplikacija mora nad datotečnim objektom izvesti enega izmed TOCTTOU parov.
- Datotečni objekt se mora nahajati na lokaciji, kjer ima napadalec "write" privilegije.
- Ime datotečnega objekta ne sme biti naključno.

Pogoste prakse razvijalcev

1. Uporaba neatomarnih operacij nad datotečnimi objekti.

2. Kreiranje datotečnih objektov v začasnih mapah, kjer imajo vsi uporabniki privilegije pisanja in izvajanja kode.

3. Uporaba nenaključnih ali pseudo-naključnih imen za datotečne objekte.

Koraki uspešnega napada na datotečne objekte

1. Sinhronizacija s tarčo
2. Pridobitev procesorske časovne rezine
3. Podtikanje datotečnega objekta
4. Izvajanje sovražne kode

Primer

Sinhronizacija

Kako lahko napadalec ve, kdaj se izvaja prvi izmed TOCTTOU parov sistemskih klicev?

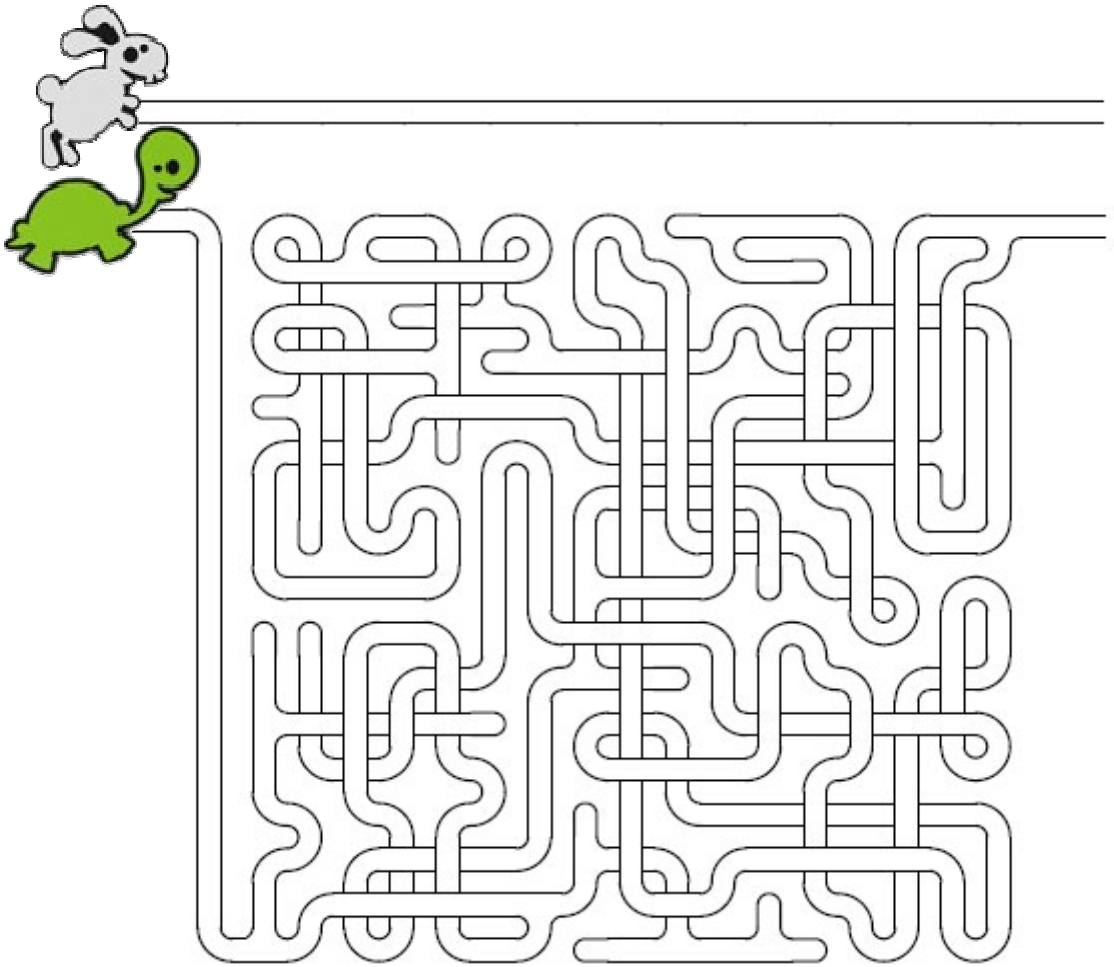
1.Kreira symlink brez asociacije (symlink, ki kaže “v prazno”)

2.Nad simbolično povezavo periodično izvaja klic lstat() in preverja zadnji čas dostopa

3.Klic lstat() ne spremeni časa dostopa na simbolični povezavi.

4.Ko proces izvede operacijo (npr. stat(), open()) nad datotečnim objektom, spremeni čas dostopa datotečnega objekta.

Symlink labirint



Problematika datotečnega predpomnilnika

- Dovolj časa za izvedbo napada napadalec dobi samo, če mora jedro operacijskega sistema razreševati pot, ki se ne nahaja v predpomnilniku (ang. “file cache”), temveč na trdem disku. Tako je jedro operacijskega sistema prisiljeno iz trdega diska prenesti 327 MB podatkov.

- Težavo predstavlja datotečni predpomnilnik. Že samo ustvarjanje labirinta povzroči, da se le ta zapiše v predpomnilnik.

- To težavo napadalec lahko obide tako, da ustvari več labirintov, od katerih uporabi samo prvega. Drugi služijo izpodrivanju iz datotečnega predpomnilnika.

- Druga možnost je zagon procesa iskanja, ki preišče zadostno količino podatkov, da labirint izpodrine iz predpomnilnika.

Praktična predstavitev napada

Potek napada

Zajec ustvari labirint

Zajec izprazni "file cache"

Zajec začne s pripravami

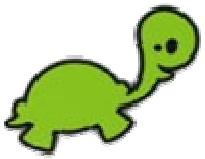
Prvi symlink v labirintu preimenuje v exploit_me.sh

Začne periodično preverjati zadnji čas dostopa na expoit_me.sh s klicem lstat()

Želva nad symlinkom exploit_me.sh izvede klic stat()

Želva pade v labirint

Želva izvede exploit_me.sh



Zajec izbriše symlink in ga nadomesti z zlonamernim exploit_me.sh

Vprašanja?

Hvala za pozornost.

Jure Škofič
jure.skofic@acrossecurity.com

ACROS d.o.o.
www.acrossecurity.com