Információs rendszerek biztonságtechnikája

Sebezhetőségek

Vakulya Gergely

A sebezhetőségek

- A sebezhetőség bizonyos rendszerek, szoftverek gyengesége, amit egy támadó a saját céljaira, illetve károkozásra tud kihasználni.
- A sebezhetőséget kihasználó szoftver az exploit.
- A következők időbeli alakulása lényegében tetszőlegesen alakulhat:
  - A sebezhetőségek fennállásának időintervalluma
  - Azok felfedezése
  - A javítások elkészítése (a sebezhetőség befoltozása)
  - A sebezhető rendszerek tényleges javítása
  - A sebezhetőség nyilvánosságra hozása
  - A sebezhetőség ismert, vagy feltételezett kihasználása

A CVE adatbázis

- CVE: Common Vulnerabilities and Exposures
- Nyilvános sérülékenység-adatbázis.
- A publikált sérülékenységek egy CVE számot kapnak, amiről egyértelműen be lehet őket azonosítani (emellett lehet fantázianevük is).
- Legfontosabb információk:
  - Milyen szoftvert (vagy egyéb rendszert érint)
  - Milyen verziókat érint
  - Rögzítés dátuma
  - Leírás (működési mechanizmus)
  - Referenciák (például az egyes szállítók információihoz)
- Több gyűjtőoldal:
  - https://cve.mitre.org
  - https://www.cvedetails.com
  - https://www.cve.org/
  - https://nvd.nist.gov

A sebezhetőségek fő típusai

- Kódfuttatás (**code execution**): A támadó (esetenként tetszőleges) kódot képes futtatni a sebezhető rendszeren.
- Védelem, vagy ellenőrzés megkerülése (**bypass**): A támadó valamilyen ellenőrzés kihagyására képes a sebezhető rendszert késztetni.
- Privilégiumszint emelés (**privilege escalation**): A támadó egy számára nem biztosított privilégiumhoz (például root joghoz) jut.
- Szolgáltatásmegtagadás (**denial of service, DoS**): A támadó a rendszer szolgáltatásait leállítja, lelassítja, vagy azokat valamilyen módon akadályozza.
- Adatszivárgás (**information leak**): A támadó számára nem nyilvános adatokhoz jut.

Támadási mechanizmusok

- Puffertúlcsordulásos támadás (buffer overflow, stack buffer overflow)
- SQL-injektálás (SQL injection, SQLi)
- Cross-side scripting, XSS
- Kéréshamisítás (Cross-site request forgery, CSRF vagy XSRF)
- Versenyhelyzet (Race condition)
- Sidechannel attack
- Memory corruption
- Bemenet ellenőrzés (Input validation)
- Kriptográfiai támadások

Ezek kombinációi is gyakoriak.

CVE-2019-18634, sudo pwfeedback exploit (Linux és egyéb rendszerek)

- A sudo 1.7.1 (2009-04-11) és 1.8.26 (2018-11-13) verziói között.
- A `sudo` parancs a jelszó bekérésekor nem ad kimenetet.
- Azonban beállíthatjuk, hogy a jelszó begépelése közben csillagok jelenjenek meg.
- Ha ez a beállítás fennáll, megfelelően összeállított bemenet beírásával a támadó root jogosultsághoz juthat.
- Típus: Helyi privilégiumszint növelés (local root).
- Mechanizmus: Stack buffer overflow.
- Megjegyzés: A ‘sudo‘ root joggal fut.

CVE-2016-5195, Dirty COW (Linux)

- Linux kernel, 2.x-4.x, a 4.8.3 előtt.
- COW jelentése: Copy On Write
- Versenyhelyzet (race condition) alapú sebezhetőség.
- A kihasználásával a támadó egy olyan, a root tulajdonában levő fájlt tud írni, amire csak olvasási joga van.
  - Átírhat konfigurációs fájlokat, például jogosultságokat adva magának.
  - Backdoort helyezhet el valamelyik setuid root-os programban.

A Dirty COW működése

- Unix alatt „minden file”. A `/proc` alatt például virtuális fájlok vannak. A `/proc/self/mem` az aktuális folyamat memóriájának leképezése.
- `mmap()` függvény: Egy fájlt a memóriára képez le.
- Saját COW másolatot is készíthetünk egy fájlról.
- Copy on write: Csak akkor másolja le ténylegesen, ha módosítjuk.
- Race condition: Két eseménynak egymáshoz képest speciálisan időzítve kell lefutnia a hibához.
- Két thread-et futtatunk:
  - Az egyikben az `madvice()` függvénnyel azt mondjuk a kernelnek, hogy az adott területet nem fogjuk a közeljövőben használni.
  - A másikban a saját memóriát leképező `/proc/self/mem` azon részére írunk, ahova a támadott fájlt map-eltük.
- Nagyon sok próbálkozás után egy bizonyos ponton a kernel „elrontja” az írás és a másolás sorrendjét és felülíródik a megnyitt fájl.

CVE-2017-5754 (Meltdown) és CVE-2017-5753 / CVE-2017-5715 (Spectre)

- Operációs rendszertől független hardveres sebezhetőségek.
- Sebezhető a Pentium-II-től a 8. generációs Core processzorokig lényegében minden Intel CPU, ezek AMD megfelelői, a modern ARM architektúrák és potenciálisan egyéb CPU-k.
- Mecanizmus: Race condition, sidechannel attack, timing attack.
- Következmény: Elvileg nem hozzáférhető memóriaterületek olvasása (például kriptográfiai kulcsok, jelszavak, személyes adatok stb.).

A modern CPU-k sebességének egyik kulcsa

Pipelining

- Az utasítások végrehajtását az órajel vezérli.
- Naiv megközelítés: 1 órajel alatt 1 utasítás.
- Az utasítások azonban több lépésből állnak (például: betöltés, dekódolás, végrehajtás).
- Pipelining: Amíg az egyik utasítást végrehajtja a CPU, a következőt már dekódolja, a rákövetkezőt már betölti.
- Feltétel: Tudni kell, hogy melyik utasítás következik. Elágazásoknál ez nem egyértelmű.

Az elágazás problémájának megoldása

- Megvárjuk, amíg kiderül, merre folytatódik a program futása
- Vagy megpróbáljuk megjósolni (branch prediction)

Branch prediction és speculatíve execution

Branch prediction

- A CPU megjósolja, hogy egy elágazás merre fog folytatódni.
- Stratégiák:
  - Mindig az ugrást választja
  - Mindig a nem ugrást választja
  - Azt az ágat választja, ami az előző alkalommal, vagy előző alkalmakkor is végrehajtódott

Speculative execution

- A CPU bizonyos, **valószínűleg szükséges** utasításokat **előre** végrehajt.
- Ha ténylegesen végre kellett hajtania az utasításokat, akkor időt spórolt.
- Ha mégsem kellett volna végrehajtani, akkor vissza kell csinálni a tévesen végrehajtott műveletet.

A cache memória

- Adott időben egy folyamat a memóriának csak egy kis részét használja (lokalitás).
- A memória elérését a cache gyorsítja.
- Egy adott memóriarekesz első elérésekor a kérés a memóriából kerül kiszolgálásra és az adat beíródik a cache-be is.
- Az újabb hivatkozáskor már a sokkal gyorsabb cache-ből lehet kiolvasni az adatot.
- A cache a memóriához képest nagyságrendekkel kisebb és gyorsabb.
- A cache általában az LRU (Last Recently Used) algoritmust használja.
- A cache általában lap alapú, illetve a memórialapozást is gyorsíthatja. Így egy byte beolvasása annak egy lapnyi környezetét is gyorsítja.

A sebezhetőség két építőeleme

A branch prediction szándékos félrevezetése

- A branch prediction működése szándékosan befolyásolható, vagy megjósolható.
- Könnyen készíthető olyan elágazás, amit nagy valószínűséggel rosszul fog jósolni a CPU.
- A branch prediction sok esetben tanítható is.

Kétszeresen indirekt címzés

`y = a[b[x]]`

- `b` egy tömb, aminek az `y`-adik elemét kiolvassuk.

A támadás menete

```
if(x < 10)
    y = a[b[x]]
```

- Az `a` tömbhöz van hozzáférési jogunk.
- A `b` tömb elejéhez van hozzáférési jogunk, a végéhez nincs.
- Kis értékű `x`-re a kiolvasás sikeres. Ezekkel tanítjuk a branch prediction-t.
- Nagy `x` esetén az eredmény: segmentation fault.

Azonban az elágazás igaz ága spekulatíven lefutott, kiolvasva a nem hozzáférhető területet. Persze mivel a feltétel nem teljesült, a processzor visszaállította az eredeti állapotot.

Kivéve a cache-t! Most az `a` tömb egyik eleme benne van a cache-ben, tehát gyorsabban kiolvasható.

Néhány további részlet

A módszer tulajdonságai

- Folyamatok egymáshoz képesti lefutásának hibáját használjuk ki, így ez egy **race condition attack**.
- Egy olyan csatornán keresztül lehet információhoz jutni, ami nem kötődik konkrétan a támadott rendszerhez, szoftverhez, protokollhoz, tehát ez egy **side-channel attack**.
- Magát az adatot időzítés segítségével olvassuk ki, így ez egy **timing attack**.

Egyéb kiegészítések

- Ez így leírt módszer csak a támadás elvét mutatja be, valójában sok konkrét sebezhetőség és támadás létezik.
- A bemutatott kódrészlet például figyelmen kívül hagyja a lapméretet.

Védekezés a Meltdown / Spectre típusú sebezhetőségekkel szemben

- Elvileg teljeskörű javítás csak az érintett CPU cseréjével lehetséges (9. gen. Core CPU-k már védettek).
- Szoftveres javítás: **mitigation** (enyhítés, csillapítás), valójában nem

DoS / DDoS típusú támadások

Slow loris (CVE-2007-6750 és sok egyéb)

Összesítés

- DoS (Denial of Service) típusú támadás webszerverek ellen.
- Számos webszerver verzió érintett.
- Inkább konfigurációs jellegű probléma.

Működése

CVE-2017-0199, Office arbitrary code execution

CVE-2017-0144, Eternal Blue

CVE-2020-0796 aka CoronaBlue aka SMBGhost

CVE-2023-3269, StackRot

WiFi routerek jelszavainak megjósolhatóságára vonatkozó sebezhetőségek

- A hardver eszközök általában nagy sorozatban készülnek. Egy típuson belül azonosnak tekinthetőek.
- De mi van, ha mégis különbözniük kell?
  - MAC cím
  - Egyedi jelszó
  - Egyéb egyedi azonosító
- A MAC cím egyediségére vannak jól bevált módszerek (maszk ROM)
- A router háttértára flash memória. Egy firmware frissítés az itt levő jelszót törölné.
- Ötlet: Származtassuk a jelszót valamilyen egyébként is eltárolt adatból. A WiFi SSID-jét hasonlóképpen.

WiFi routerek jelszavainak megjósolhatóságára vonatkozó sebezhetőségek

- Sebezhetőség:
  - Ha ismerjük a generátor adatot, kiszámíthatjuk a generált adatokat.
  - A generált adatok között is összefüggés lehet.
- Két példa:
  - CVE-2023-47352: Technicolor TC8715D routerek esetén a támadó az SSID és a BSSID (router MAC címe) alapján megjósolhatja a WPA2 jelszót.
  - CVE: N/A, UBEE EVW3226 routerek esetében lényegében ugyanez a sebezhetőségtípus
  - Mindkét router alapértelmezett jelszava admin/admin; az adminisztratív felület elérhető WiFi-ről is
- Megoldási lehetőségek:
  - Magasabb elvárások a gyártóval kapcsolatban
  - Ne bízzunk az alapértelmezett jelszavakban!

Authentication bypass típusú sebezhetőségek

- Általánosan: A webfelületek nem ellenőrzik minden esetben a felhasználó privilégiumait, azonosságát
- Példa: Az adminisztratív felületre egy login oldal irányít át; ott már további ellenőrzés nem történik
- Huawei HG866 Authentication Bypass, CVE: N/A
-

CVE-2023-2598, Linux IO uring memory out of bounds read/write

Összefoglaló

- CVE-2023-2598
- Sebezhető szoftver: Linux kernel 6.3-rc1 - 6.4-rc1 (utóbbiban javítva)
- Mechanizmus: Memory corruption, out of bounds r/w
- Következmény: Memória szivárgás, privilégiumszint növelés, tetszőleges kódvégrehajtás
- https://anatomic.rip/cve-2023-2598/

IO uring

Rendszerhívások

- Az operációs rendszer szolgáltatásait adó API
- Jelentős részük valamilyen IO művelet
- Például fájlműveletek, hálózati kommunikáció
- A rendszer hívás overheaddel jár

IO uring

- Az IO uring megoldás a rendszerhívások által kialakuló overheadet csökkenti
- Aszinkron működés: nem blokkol
- Egy körpufferben kell átadni a szükséges rendszerhívásokat, azok paramétereivel
- A probléma a memóriapufferek átadásánál és használatánál lép fel

Néhány szó a Linux memóriaszervezéséről

- Lapszervezés: 4 kByte méretű lapokkal
- A lapszervezés megakadályozza a külső tördelődést, viszont a folytonos területek mégis speciálisan vannak kezelve
  - Fej (head) lap
  - Farok (tail) lapok
- Folio-k (ívek)
  - Az összetett lapok kezelésére használják
  - Egyszerű lap esetén maga a lap
  - Összetett lap esetén a fejlap
  - Semmiképp nem lehet faroklap

A hiba

A virtuális memória

- A virtuális memória lényege, hogy a logikai és fizikai memória között leképezést ad.
- Egy logikai memórialaphoz egy fizikai lapot lehet rendelni

A hiba forrása

- Ha az IO uring-nek olyan puffer van átadva, ahol egy folytonos logikai terület mindegyik lapjához ugyanaz a fizikai lap van rendelve, akkor a kód elrontja a számolást és az 1 db fizikai lap után következőket is használni fogja

CVE-2016-0728 (Linux kernel) CVE-2014-0038 (Linux kernel)
CVE-2016-3714 (ImageTragick)