Financira
Evropska unija
NextGenerationEU

THE RECOVERY
AND RESILIENCE
PLAN

# VARNOST PROGRAMOV

Predavanja #7

Matevž Pesek

# Teme!

- Vprašanja?
- Popravki abstraktov
- Ne pozabite na rok! (27. april / 15. maj)

# DANAŠNJE TEME

- Časovno odvisni napadi
- Hkratni dostop
- Time-of-use
- Time-of-check

# Od prejšnjič

- Kaj je avtentikacija?
- Kaj je avtorizacija?
- Kaj je enkripcija?
- Kaj je JWT in zakaj ga uporabljamo?

# ČASOVNO ODVISNI NAPADI

Hkratni dostop

# Kaj so časovno odvisni napad?

- Navadno izkoriščanje časovno-odvisnih ranljivosti, ki sistem privede v stanje, kjer lahko pridobimo neavtoriziran dostop ali izvedemo operacijo.

- Potencialni rezultati napada:
  - Dostop
  - Izvedba operacije
  - Sprememba virov (db, datoteke)
  - Logične napake (npr. večkratne operacije)
  - Splošno nepričakovano obnašanje sistema

# Problem hkratnega dostopa

- Hipotetični primeri
  - Dvakratni klik na gumb za plačilo pri spletnem nakupu
  - elektronske volitve
- Ne-tako-hipotetični primeri
  - Hkratna izvedba vpisa v sistem (izkoriščanje asinhronosti)
  - DDOS

# Primeri

- Juniper (networking)
  - CVE-2020-1667
    - When DNS filtering is enabled on Juniper Networks Junos MX Series with one of the following cards MS-PIC, MS-MIC or MS-MPC, an incoming stream of packets processed by the Multiservices PIC Management Daemon (mspmand) process might be bypassed due to a race condition. Due to this vulnerability, mspmand process, responsible for managing "URL Filtering service", can crash, causing the Services PIC to restart.
    - While the Services PIC is restarting, all PIC services including DNS filtering service (DNS sink holing) will be bypassed until the Services PIC completes its boot process.
  - CVE-2021-31382
    - A Race Condition vulnerability between the chassis daemon (chassisd) and firewall process (dfwd) of Juniper Networks Junos OS, may update the device's interfaces with incorrect firewall filters.
- TIBCO (is)
  - CVE-2018-18808
    - contains a race-condition vulnerability that may allow any users with domain save privileges to gain superuser privileges
- GIT (web)
  - CVE-2022-4037
    - Account Takeover. An issue has been discovered in GitLab CE/EE affecting all versions before 15.5.7, all versions starting from 15.6 before 15.6.4, all versions starting from 15.7 before 15.7.2. A race condition can lead to verified email forgery and takeover of third-party accounts when using GitLab as an OAuth provider.

# Primeri (2)

- Laravel (PHP)
  - CVE-2022-24800
    - Remote code execution by exploiting race condition in the temporary storage directory
- PortfolioCMS
  - CVE-2021-36532
    - Race condition vulnerability discovered in portfolioCMS 1.0 allows remote attackers to run arbitrary code via fileExt parameter to localhost/admin/uploads.php

# PRIMERI

# TOCTOU - Canary

- Primer iz vaj
- S prvo prekoračitvijo preberemo canary
- Z drugo prekoračitvijo zapišemo canary in ROP verigo
- To deluje, ker se prekoračitev preveri ob koncu izvajanja funkcije

```
char buffer[64];

printf("Enter your buffer: ");
read(0, buffer, 0x64);

printf("Hello, %s! What's your surname?\n", buffer);
read(0, buffer, 0x64);

printf("Got it, %s!\n", buffer);
```

# TOCTOU – Linux jedro

- Poznamo lokacijo podatka
- Kako dobimo dolžino podatka “flag”?
- Kako pridemo do “for” zanke?
- Kako prepričamo program, da nam izpiše podatek “flag”?

```
if ( a2 == 26214 )
{
  printk("Your flag is at %px!\n", flag);
  result = 0LL;
}
else if ( a2 == 4919 && is_userspace(*v5)
  && *(v5 + 8) == strlen(flag) )
{
  for ( i = 0; i < strlen(flag); ++i )
    if ( *(*v5 + i) != flag[i] )
      return 22LL;
  printk("%s\n", flag);
  result = 0LL;
}
else
  result = 14LL;
return result
```

https://ypl.coffee/0ctf-2018-finals-baby-kernel/

# TOCTOU – Linux jedro

- Preberemo lokacijo podatka
- S pomočjo niti (thread) vedno znova nastavljamo prebrano lokacijo
- Pred vsakim kernel klicom (ioctl) nastavimo ”flag” na podatek iz uporabnikovega prostora
- Ponavljamo dokler nam ne uspe pretentati “if” stavek in “for” zanko

```
struct guess_t {
  char *flag;
  long len;
} guess;
void *malicious(void *t) {
  struct guess_t *guess = t;
  while (finish == 0)
    guess->flag = addr;
}
...
pthread_t t;
pthread_create(&t, NULL, malicious, &guess);
for (;;) {
  if (ioctl(fd1, 0x1337, &guess) == 0)
    break;
  guess.flag = buf;
}
```

https://ypl.coffee/0ctf-2018-finals-baby-kernel/

# SIDE-CHANNEL NAPADI

In hkratni dostop

# Side-channel napadi - strojna oprema

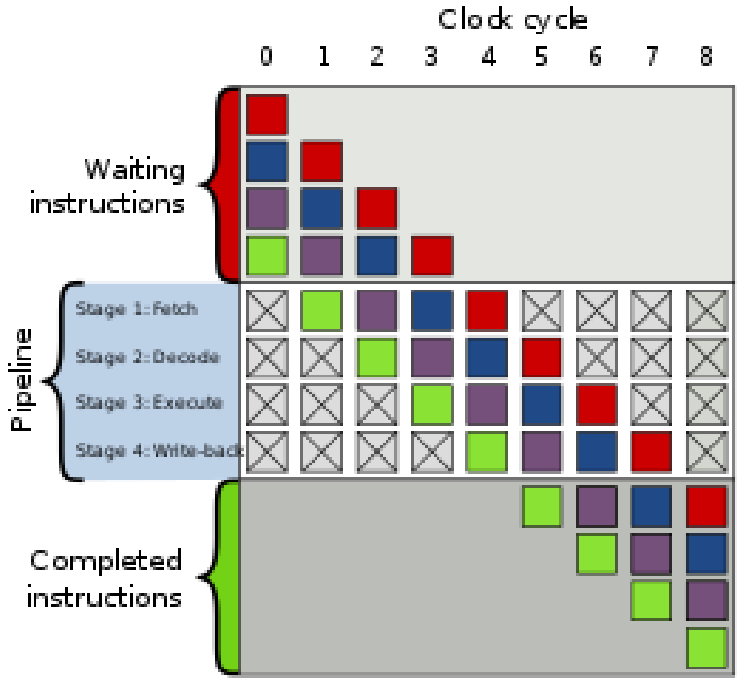
- Kaj so side-channel napadi?
  - Izkoriščanje samega procesa/protokola za pridobivanje dodatnih informacij izven predvidenega delovanja aplikacije/sistema
  - Primeri: časovno odvisne informacije, poraba energije, elektromagnetno sevanje, zvok (npr. disk, tipkovnica, …)

# Strojna oprema – znani primeri

- Meltdown in spectre
  - T.i. “side-channel napadi”
  - Izkoriščanje predikcije skokov na nivoju strojne kode (branch prediction)
- Tipične posledice:
  - Spletno mesto lahko bere podatke, shranjene v brskalniku, za drugo spletno mesto ali sam pomnilnik brskalnika
  - Izkoriščanje ranljivosti CPE za neavtoriziran dostop do pomnilnika (npr. zasebni ključi v drugi aplikaciji)
  - Izstopanje iz peskovnika (sandbox)
  - V kombinaciji s side-channel-om, pridobivanje informacij

# Spectre

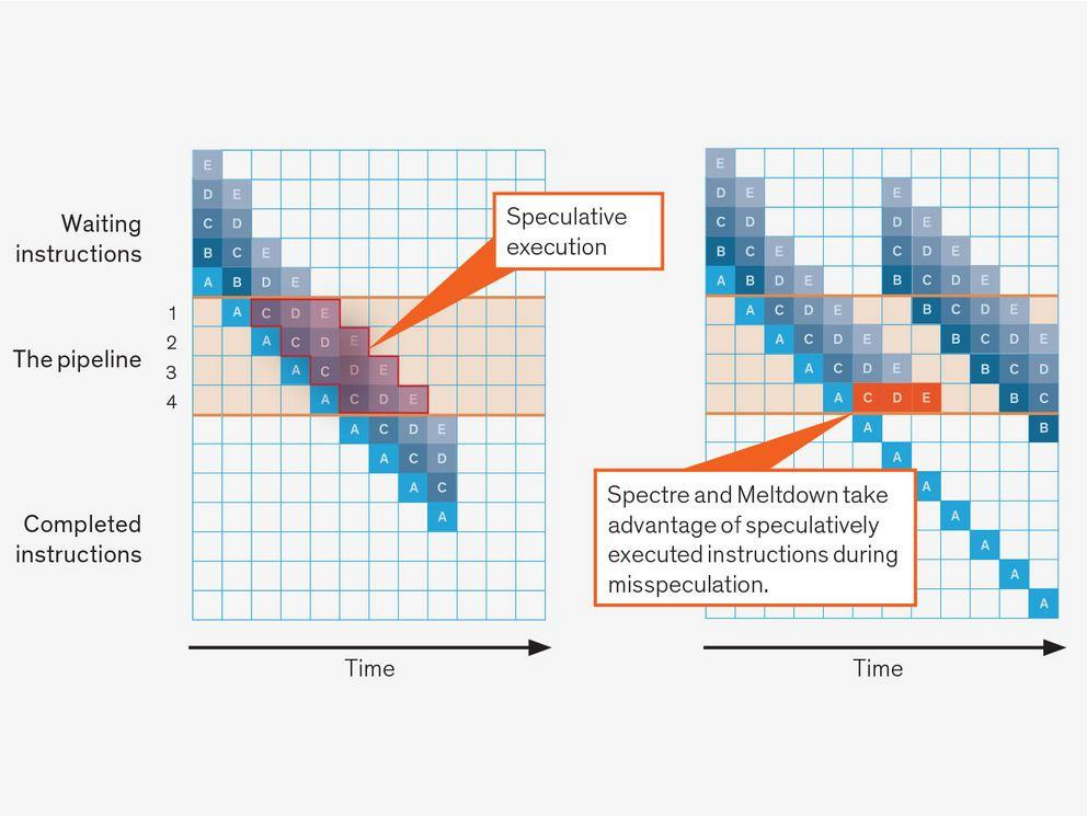
- Skupina ranljivosti,
  - zavede program v dostop do poljubnih lokacij v pomnilniškem prostoru programa
  - Napadalec lahko prebere vsebino dostopnega pomnilnika in tako potencialno pridobi občutljive podatke
  - Primer iz članka:
    - Javascript sandbox




https://en.wikipedia.org/wiki/Branch_predictor

# Meltdown

- Dostop do poljubnega dela pomnilnika
  - Intel, PowerPC, nekateri ARM procesorji
- SW rešitev
  - Upočasni delovanje za 5-30%
- Tip napada
  - Hkratni dostop (race condition)
  - Zaobide preverjanje privilegijev
  - Dostopa do podatkov, ki so naloženi v medpomnilniku (drug proces)




CVE-2017-5754, *Rogue Data Cache Load* (RDCL)
https://spectrum.ieee.org/how-the-spectre-and-meltdown-hacks-really-worked

# Časovno odvisni napadi v programski kodi

- Time-of-check time-of-use
  - TOCTOU
  - Abstrakten primer
    - Preverimo vrednost
    - [preteče nekaj časa]
    - Uporabimo vrednost
  - Primeri izkoriščanja
    - Dostop do datotek/vtičnic (socket)
    - Podatkovne baze
    - Linux jedro

```
if(!access(file,W_OK)) {
    f = fopen(file,"w+");
    operate(f);
    ...
}
else {

fprintf(stderr,"Unable to open file %s.\n",file);
}
```

# Primer (unix)

| Victim | Attacker |
| --- | --- |
| `if (access("file", W_OK) != 0) {`<br>`    exit(1);`<br>`}` | |
| | After the access check, before the open, the attacker replaces `file` with a symlink to the Unix password file `/etc/passwd`:<br>`symlink("/etc/passwd", "file");` |
| `fd = open("file", O_WRONLY);`<br>`write(fd, buffer, sizeof(buffer));`<br>Actually writing over `/etc/passwd` | |

https://en.wikipedia.org/wiki/Time-of-check_to_time-of-use

# HTTP časovno odvisni napad

- Uporabimo časovno odvisnost od parametrov, da pridobimo podatke iz Sistema
- Merimo pretečen čas

```
## NEOBSTOJEČ UPORABNIK

$ for i in $(seq 1 10); do { time curl -s
http://localhost:8000/login.php?login=true -d
'username=test&password=test' >/dev/null; }
2>&1 | grep real; done
```

- Ugotovimo, ali uporabniško ime obstaja brez, da bi vedeli za geslo
- Primer [levo]

```
## OBSTOJEČ UPORABNIK

$ for i in $(seq 1 10); do { time curl -s
http://localhost:8000/login.php?login=true -d
'username=admin&password=test' >/dev/null; }
2>&1 | grep real; done
```

# PRIMERI

# Sidechannel – pin_checker

- S surovo silo rabimo 10^8 (100.000.000) poizkusov
- Preverjanje se pogosto procesira sekvenčno

```
Please enter your 8-digit PIN code:
10000000
Checking PIN...
Access denied.
echo 10000000  0.00s user 0.00s system 44% cpu
0.001 total
./pin_checker  0.13s user 0.00s system 99% cpu
0.128 total
```

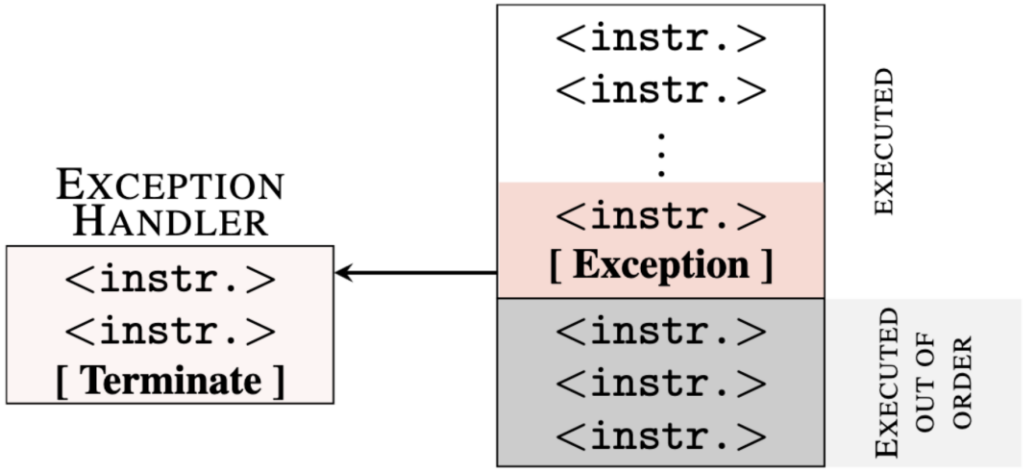
- Če postopoma preverjamo vsako števko in sledimo času izvajanja, lahko ugotovimo pin
- S tem rabimo preveriti samo 8*10 (80) poizkusov

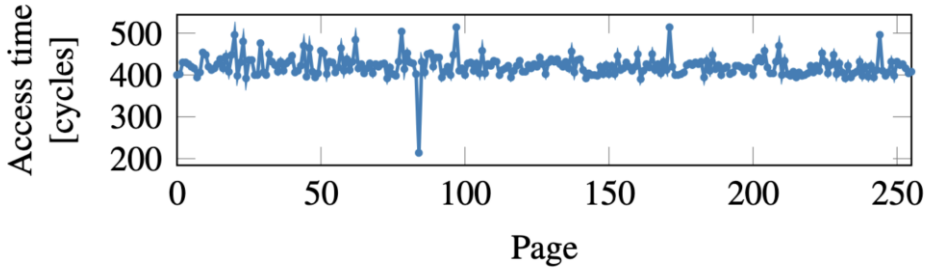
```
Please enter your 8-digit PIN code:
40000000
Checking PIN...
Access denied.
echo 40000000  0.00s user 0.00s system 44% cpu
0.001 total
./pin_checker  0.25s user 0.00s system 99% cpu
0.249 total
```

https://picoctf2022.haydenhousen.com/forensics/sidechannel

# Sidechannel – Meltdown

- Sprožimo napako med izvajanjem
- Zaradi spekulativnega izvajanja se začnejo izvajati ukazi "izven zaporedja"
- Ukazi še vedno naložijo podatke v medpomnilnik
- Te lahko nato "prenesemo" iz enega programa v drugega (npr. časovni napadi)




https://meltdownattack.com/meltdown.pdf

# HVALA

Vaje

- Časovno odvisni napadi - implementacija