

KARL

Jak płatek śniegu, unikalny kernel.

Adam Wołk



a.wolk@fudosecurity.com



awolk@openbsd.org



<https://blog.tintagel.pl>



@mulander



ASLR

Address Space Layout Randomization

ASLR

Randomizacja przestrzeni adresowej poprzez losowe rozmieszczenie kluczowych sekcji procesu:

- Załadowanego pliku binarnego
- Lokalizacji stosu (stack)
- Lokalizacji sterty (heap)
- Załadowanych bibliotek

ASLR

Celem jest utrudnienie wykorzystania błędów bezpieczeństwa w praktyce, utrudnia:

- Skoki do istniejących funkcji
- Lokalizacje gadżetów ROP
- Odnoszenie się do wcześniej umieszczonego kodu/danych w pamięci

Relatywne adresy przestają być przewidywalne, zatem wymagana jest faza odkrywczą w trakcie wykonywania kodu atakującego. Prowadzi do mniej stabilnych ataków.

KARL != KASLR

KASLR

Kernel Address Space Layout Randomization

KASLR

Losowe umieszczenie kernela w przestrzeni adresowej:

- W całości, losując tylko miejsce załadowania (iOS)
- Sekcjami, np. kernel (.text, .rodata, .data, .bs) i moduły ładowane do kernela (MacOSX)
- Z podziałem na pod-sekcje, jedyna implementacja NetBSD z podziałem na 33 pod-sekcje

KARL

Kernel Address Randomized Link

KARL

- Dodany w czerwcu 2017
- To nie jest ASLR, kernel jest nadal zawsze ładowany w tej samej lokacji KVA
- System zawiera mechanizm linkujący, odpalany przy:
 - Pierwszej instalacji
 - Przy każdym starcie systemu
 - Po instalacji patchy przez **syspatch(8)**

KARL

Kernel posiada sekcje bootstrap, jest ona ładowana w tej samej znanej lokacji ale po inicjalizacji zostaje zneutralizowana instrukcjami TRAP lub od-mapowana (zależnie od architektury).

KARL

- **/bsd** - kernel ładowany przy **następnym** starcie systemu
- **/var/db/kernel.SHA256** - suma kontrolna z /bsd, weryfikowana przy następnym uruchomieniu. Przy niepasującej sumie, re-linkowanie nie nastąpi
Przydatne przy rozwoju kernela/debugowaniu.
- **/bsd.booted** - obecnie załadowany kernel

KARL - start systemu

- Ładujemy /bsd lub /bsd.booted jeżeli podnosimy się z hibernacji
- /etc/rc startuje usługi, na samym końcu uruchamia
 /usr/libexec/reorder_kernel &
- Weryfikacja sumy kontrolnej
- Re-linkowanie kernela
- Podmiana /bsd i zapis nowej sumy kontrolnej

KARL

- ~1300 plików *.o uczestniczy w składaniu kernela
- Wynikiem tej operacji jest unikalny kernel
- Utrudnia lokalizację gadżetów ROP i budowanie z nich ciągów do wykonywania
 - Wymaga błędów ujawniających lokalizacje gadżetu w pamięci
 - Każdy system ma inny kernel - wymaga unikalnego payloadu

KARL

```
fishtank# cat /usr/share/compile/GENERIC.MP/relink.log
(SHA256) /bsd: OK
LD="ld" sh makegap.sh 0xc0000000
ld -T ld.script -X --warn-common -nopie -o newbsd ${SYSTEM_HEAD} vers.o ${OBS}
text  data  bss  dec  hex
8470312 2672744 667648 11810704      b43790
mv newbsd newbsd.gdb
ctfstrip -S -o newbsd newbsd.gdb
mv -f newbsd bsd
umask 077 && cp bsd /nbsd && mv /nbsd /bsd && sha256 -h /var/db/kernel.SHA256 /bsd
```

Kernel has been relinked and is active on next reboot.

```
SHA256 (/bsd) = 5747180e5469ef093cb4d9d3f9fa36ffa1a698aaebb54bf48413b78856a88d9f
```

```
fishtank#
```

Reorder libraries

Reorder libraries

- Podobny mechanizm jak KARL
- Re-link bibliotek przy starcie systemu
 - libc
 - libcrypto



Warszawa
Aleje Jerozolimskie 178

JOIN FUDO SQUAD


OUR TECH POSITIONS

- QA ENGINEER
- C DEVELOPER
- PYTHON DEVELOPER
- JAVA SCRIPT DEVELOPER
- TECH SUPPORT ENGINEER

FUDO STRENGTHS

- PRODUCT UNIQUENESS & QUALITY FOCUS
- CONSTANT SEARCH FOR NEW POSSIBILITIES
- WORKING (ALMOST :) ALL TOGETHER
- OPENNESS FOR EMPLOYEE FEEDBACK
- TEAM!

"Everything is possible with those developers!"

 hr@fudosecurity.com



Thank you!

Adam Wołk

✉ a.wolk@fudosecurity.com

✉ awolk@openbsd.org

🌐 <https://blog.tintagel.pl>

🐦 @mulander