ASLR
Lecture 18b

COMPSCI 702
Security for Smart-Devices

Nalin Asanka Gamagedara Arachchilage

Slides from Muhammad **Rizwan** Asghar

April 22, 2021

DO WE NEED MORE PROTECTION?

Is mandatory code signing enforcement sufficient to withstand against malicious apps?

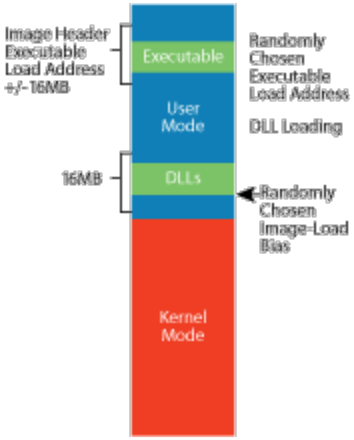
PREVENTING MALICIOUS APPS

- Prevent malicious apps at submission time
  - Static and dynamic analysis
- Prevent malicious apps at install (or run) time
  - Code Signing Enforcement (CSE)
- Operating System (OS) also prevents malicious apps
  - Data Execution Prevention (DEP)
  - Address Space Layout Randomisation (ASLR)

MOTIVATION BEHIND ASLR

- DEP does not defend against *‘return-to-libc’* exploits
- ‘return-to-libc’ exploits require
  - Address of malicious code in memory
  - Addresses of routines to be called
- To withstand against *‘return-to-libc’* attacks, the idea is to introduce artificial diversity
  - Do not load the code at a fixed address

ASLR



- Loading code in memory at different addresses so that it is harder to locate it

- Works best if all code is compiled with the Position Independent Execution (PIE) flag
  - Apps and libraries
  - Many third party apps are not compiled for PIE

ASLR ADVANTAGES

- ASLR enables randomisation of:
  - Binary executable
  - Heap
  - Stack
  - Libraries
  - Dynamic linker

ASLR without PIE

Executable	Heap	Stack	Libraries	Linker
0x2e88	0x15ea70	0x2fdff2c0	0x36adadd1	0x2fe00000
0x2e88	0x11cc60	0x2fdff2c0	0x36adadd1	0x2fe00000
0x2e88	0x14e190	0x2fdff2c0	0x36adadd1	0x2fe00000
0x2e88	0x145860	0x2fdff2c0	0x36adadd1	0x2fe00000
0x2e88	0x134440	0x2fdff2c0	0x36adadd1	0x2fe00000
		Reboot		
0x2e88	0x174980	0x2fdff2c0	0x35e3edd1	0x2fe00000
0x2e88	0x13ca60	0x2fdff2c0	0x35e3edd1	0x2fe00000
0x2e88	0x163540	0x2fdff2c0	0x35e3edd1	0x2fe00000
0x2e88	0x136970	0x2fdff2c0	0x35e3edd1	0x2fe00000
0x2e88	0x177e30	0x2fdff2c0	0x35e3edd1	0x2fe00000

Slide originally from: http://www.rsaconference.com/writable/presentations/file_upload/mbs-402.pdf

ASLR with PIE

Executable	Heap	Stack	Libraries	Linker
0xd2e48	0x1cd76660	0x2fecf2a8	0x35e3edd1	0x2fed0000
0xaae48	0x1ed68950	0x2fea72a8	0x35e3edd1	0x2fea8000
0xbbe48	0x1cd09370	0x2feb82a8	0x35e3edd1	0x2feb9000
0x46e48	0x1fd36b80	0x2fe432a8	0x35e3edd1	0x2fe44000
0xc1e48	0x1dd81970	0x2febe2a8	0x35e3edd1	0x2febf000
		Reboot		
0x14e48	0x1dd26640	0x2fe112a8	0x36146dd1	0x2fe12000
0x62e48	0x1dd49240	0x2fe112a8	0x36146dd1	0x2fe60000
0x9ee48	0x1d577490	0x2fe9b2a8	0x36146dd1	0x2fe9c000
0xa0e48	0x1e506130	0x2fe9d2a8	0x36146dd1	0x2fe9e000
0xcde48	0x1fd1d130	0x2feca2a8	0x36146dd1	0x2fecb000

Slide originally from: http://www.rsaconference.com/writable/presentations/file_upload/mbs-402.pdf

PARTIAL VS FULL ASLR

PIE	Main Executable	Heap	Stack	Shared Libraries	Linker
No	Fixed	Randomised per execution	Fixed	Randomised per device boot	Fixed
Yes	Randomised per execution	Randomised per execution (more entropy)	Randomised per execution	Randomised per device boot	Randomised per execution

ROP LIMITATIONS

- For ROP exploits to succeed, they need to find the base address of a module
- Offsets within a page
- Specific areas of memory

ADDRESS LEAKAGE

- Getting address information from an app
- If you wrote the app yourself, you can explicitly print such values
- Lots of exploits have been designed to leak addresses
- Even PIE is not enough, we really need every function (or method) scattered randomly through the address space
  - Otherwise, if we find one address, we can easily determine others
- It is also possible to prevent ROP attacks by recording and checking addresses when making function calls
  - This slows code down noticeably
  - There are ways to defeat this too

ASLR IN iOS

- Apple introduced ASLR in iOS 4.3
  - Released in March 2011
- Full ASLR support in iOS 5 and later

ASLR IN ANDROID

- Android 4.0 provides ASLR
- Full ASLR was supported in Android 4.1
- Android 5.0 dropped non-PIE support and requires all dynamically linked binaries to be position independent

SUMMARY

- Both DEP and ASLR make it difficult to mount attacks
- Although ASLR hardens the attack, it is still vulnerable to ROP
  - With address leakage

RESOURCES

- **iOS Hacker's Handbook**
  Charlie Miller, Dionysus Blazarkis, Dino Dai Zovi, Stefan Esser,Vincenzo Iozzo, Ralf-Philipp Weinmann
  John Wiley & Sons, Inc., 2012

- **Apple iOS 4 Security Evaluation**
  Dai Zovi, Dino A
  Black Hat USA 2011
  http://media.blackhat.com/bh-us-11/DaiZovi/BH_US_11_DaiZovi_iOS_Security_WP.pdf

- **Too Much PIE is Bad for Performance**
  Payer, Mathias
  2012
  http://e-collection.library.ethz.ch/eserv/eth:5699/eth-5699-01.pdf?pid=eth:5699&dsID=eth-5699-01.pdf

Questions?

Thanks for your attention!