MacOS X – xnu is under attack!

Никита Тараканов

CISS Research Team

XNU – IS NOT UNIX

- На самом деле немалая часть ядра, основана на форке FreeBSD 5.0 (/bsd)
- Mach-O (/osfmk)
- IOKit framework (/iokit)

/bsd

- Реализация POSIX вызовов (syscalls.h).
- На данный момент, уже намного сильно отличается от первоначального форка (MacOS X 10.0 Cheetah)
- Однако, есть до сих пересечения с *BSD

CISS

Mach-O(/osfmk)

- Форк от Mach 3 OS (1985) .
- Реализаци low-level механизмов:VMM,IPC, multitasking, console I/O и т.д.
- PPC,x86,x86_64 – отдельные реализации для разных платформ.

Memory Management

- Другая модель разделения r3 и r0 АП, в отличии от Windows, linux – xnu проецируется в собственное АП
- Каждый системный вызов вызывает TLB flush, что по идее пагубно влияет на производительность

CISS

Векторы атак

- BSD syscalls(syscalls.h)
- Ioctl, fcntl
- Mach-O вызовы
- Сторонние драйверы(IOKit framework)

Примеры уязвимостей в /bsd

- CVE-2008-0177 – IPv6 IPComp remote –

  также подрвежены NetBSD,FreeBSD

- CVE-2008-1517 - Pthread desyncronization
- CVE-2009-1235 – HFS fcntl Memory Overwrite

Примеры уязвимостей в /osfmk

- CVE-2008-4218 – get_ldt memory disclosure
- CVE-2011-02182 – set_ldt desing flaw

Уязвимости в сторонних kext

- Хотя Apple предлагает отличный framework для разработки драйверов(представляет из себя урезанный C++), до сих встречаются простейшие ioctl баги(Vmware Fusion)
- Общее кол-во найденных уязвимостей, несравнимо мало по сравнению с ioctl в драйверах Windows

CISS

XNU debugging how-to

- Если вы счастливый обладатель 2 девайсов от Apple, то всё просто man nvram
- Команда nvram некорректно работает в виртуальных машинах, поэтому есть какой-то другой путь…

XNU debugging how-to

- Ключевой файл для настройки отладки: /Library/Preferences/SystemConfiguration/com.apple.Boot.plist – представляет из себя xml файл, в котором можно прописать определённые настройки загрузки XNU

XNU debugging how-to

- Ключевой файл для настройки отладки: /Library/Preferences/SystemConfiguration/com.apple.Boot.plist – представляет из себя xml файл, в котором можно прописать определённые настройки загрузки XNU
- Свойство debug-flags = 0x1 – debugmode

CISS

XNU debugging how-to

- Большинство значений не раскрыты в документации, однако, Apple’вский протокол отладки (remote-kdp) открыт, соответсвенно по коду можно понять значения, и что из-за этого изменится в загрузке XNU (/osmfk/kdp)

CISS

XNU debugging how-to

- Com.apple.Boot.plist:
- <key>Kernel Flags</key>
- <string>-v debug-flags=0x1</string>
- Простейший вариант конфига для отладки, но при такой настройке отладка малоифнормативна и неудобна

CISS

XNU debugging how-to

- Gdb не простой, а от Apple! ☺
- Нету много функционала, который есть в обычном gdb
- Символы для ядра идут отдельным пакетом(Kernel Debug Kit) – символы в основном только для ядра, а не для kext

CISS

XNU debugging how-to

- DEMO

XNU уязвимости

- Stack Overflow
- Heap Overflow
- Integer issues -> различные memory corruption
- Race conditions

XNU эксплуатация

- Null pointer dereference – is dead, из-за особенности АП - неэксплутабельно
- Stack Overflow – ROP по kernel адресам
- Heap overflow – различные сценарии эксплуатации zone allocator’a

XNU эксплуатация

- Собираем информацию об АП ядра и различных kext – kextstat
- nm /mach_kernel | grep nsysent
- Nsysent +CONST – адрес sysent

XNU эксплуатация

- Sysent – таблица симтемных вызовов(syscall.h) полученная из sycall.master
- Есть “дыры” в индексах – можно вставить свой обработчик

XNU эксплуатация

- Форсируем запись своего значения по контролируемому адресу – тем самым создаём свой fake системный вызов
- Копируем payload (iso_font техника)
- Syscall(some_empty_index, NULL)

MacOS X 10.6.X

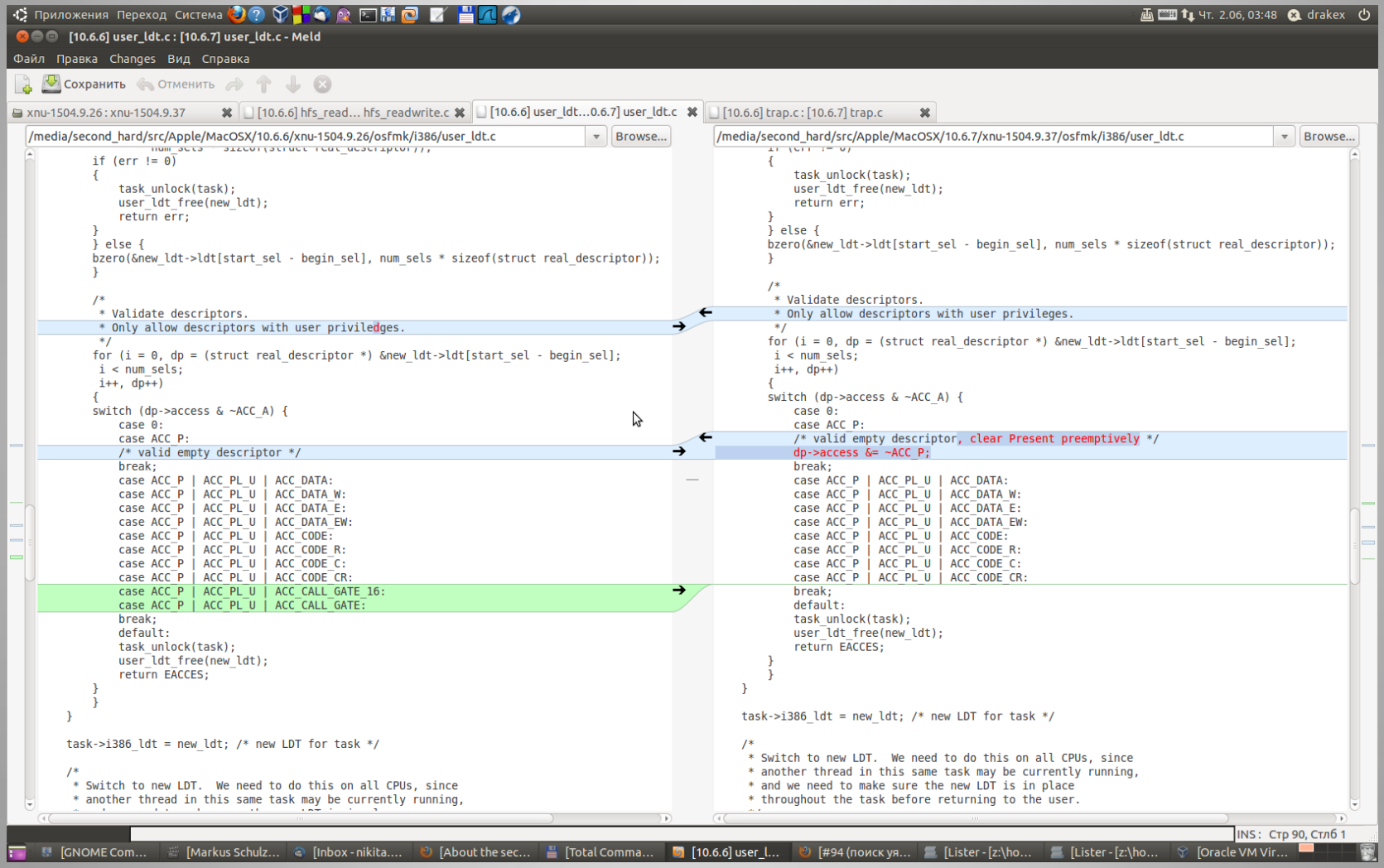
- Наконец-то убрали бажный AppleTalk
- X86_64 – некоторые процессы все равно 32
- ASLR, Heap explotation hardening
- В исходних смесь из ppc,x86,x86_64
- По умолчанию используется 32-е ядро

CISS

MacOS X 10.7.X Lion

- Ориентирована на x86_64, однако все равно есть сторонние 32-е приложения
- Ядро грузится по умолчанию в 64-е АП
- Heap explotation hardening

set_ldt() fix


Приложения Переход Система
Чт. 2.06, 03:48 drakex
[10.6.6] user_ldt.c : [10.6.7] user_ldt.c - Meld
Файл Правка Changes Вид Справка
Сохранить Отменить
xnu-1504.9.26 : xnu-1504.9.37
[10.6.6] hfs_read... hfs_readwrite.c
[10.6.6] user_ldt...0.6.7] user_ldt.c
[10.6.6] trap.c : [10.6.7] trap.c
/media/second_hard/src/Apple/MacOSX/10.6.6/xnu-1504.9.26/osfmk/i386/user_ldt.c
Browse...
/media/second_hard/src/Apple/MacOSX/10.6.7/xnu-1504.9.37/osfmk/i386/user_ldt.c
Browse...
INS : Стр 90, Стлб 1

Вопросы!?

- Защита – уменьшение возможности получить какие-либо сведения об АП ядра
- Аналог для linux - grsecurity
- Спасибо за Ваше внимаение!
- email Nikita.tarakanov.researcher@gmail.com
- Twitter: @NTarakanov

CISS

Little PR

- Join CISS HOT Summer – для получения реальных знаний и умений в сфере ИБ

- из файла MacOSX\10.6.7\xnu-1504.9.37\osfmk\x86_64\idt_table.h:
- USER_TRAP_SPC(0x80,idt64_unix_scall)
- USER_TRAP_SPC(0x81,idt64_mach_scall)
- USER_TRAP_SPC(0x82,idt64_mdep_scall)
- USER_TRAP_SPC(0x83,idt64_diag_scall)

- Кстати с 0x82-м прерыванием связана уязвимость CVE-2011-0182 (в функции i386_set_ldt()).