

IPAD 破文一篇

前言:

一个偶然的的机会,得到一台 IPAD2,若只当个娱乐玩具,还是挺很不错的。在苹果封闭的体系保护中,不去苹果那儿注册,安装自己开发软件是相当的困难,尤其在没有 MAC 笔记本的情况下,安装 IPAD 的开发环境相当艰苦,我到现在还没安装成功过(要是你成功安装过,欢迎教我一下),呵呵。前不久, jailbreakme 利用 PDF 漏洞实现了越狱,可以安装各种软件和开发工具了,哈哈,可以在没有 XCode 开发环境下,从另一个方面深入地了解一下其内部系统了。

言归正传,在 APP 中找了款免费的 SSH 软件,体验了一下,整体感觉还是不错的,只是使用时会有 150 次按键的限制,有点不爽,这便激发我来解决这一点。特别说明,本文仅以学习研究为目的。

准备:

一台越狱后的 IPAD2, GCC 编译工具, GDB 调试工具, Open SSH 服务,当然还有我们的目标软件 zaTelnet。

一台电脑,装有 Putty 系列工具, IDA.6.1 工具。

保证电脑和 IPAD 的网络畅通。

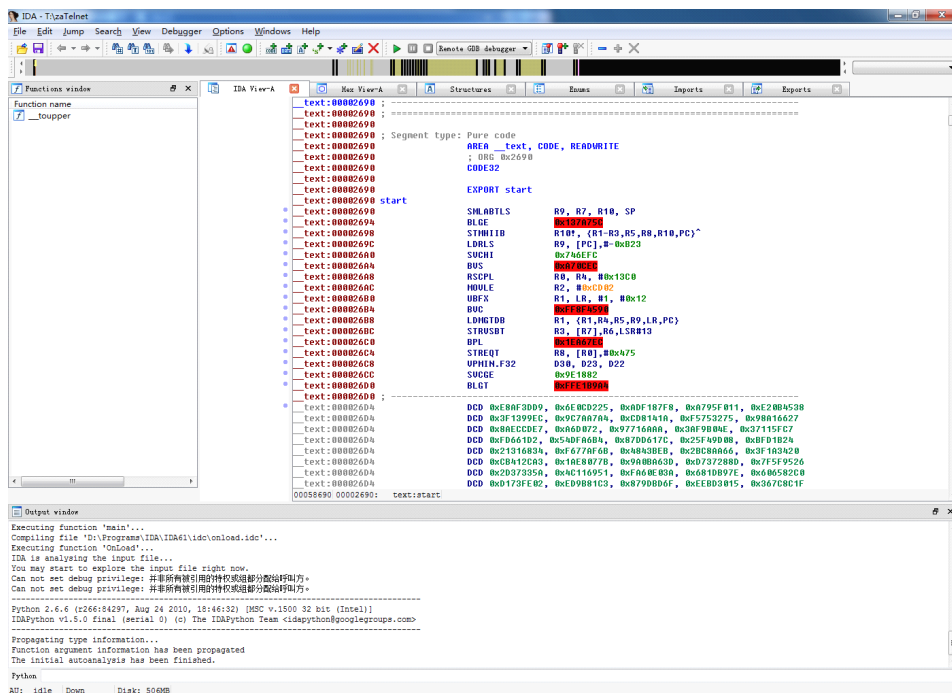
分析:

以 zaTelnet3.0.6 为例

通过 SSH 连接在 IPAD 找到 zaTelnet 所在目录

/private/var/mobile/Applications/DFFD7B8E-D0F6-422C-86C3-5F27FDE8FA1A

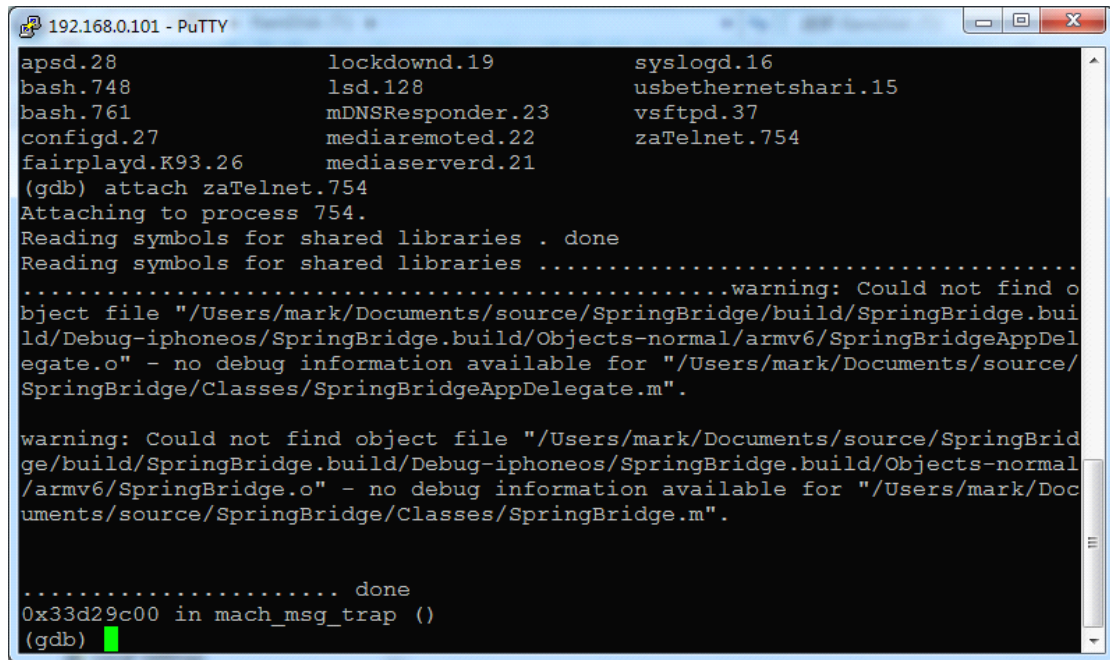
将主程序 zaTelnet 拉下来看看



```
IDA - zaTelnet
File Edit Jump Search View Debugger Options Windows Help
Remote GDB Debugger
Functions window
Function name
7 __toupser
Text:00002690 ;
Text:00002690 ;
Text:00002690 ; Segment type: Pure code
Text:00002690 AREA _text, CODE, READWRITE
Text:00002690 : @MC 0x2690
Text:00002690 CODE32
Text:00002690 EXPORT start
Text:00002690 start
Text:00002690
Text:00002690 SHL ABTLS R9, R7, R10, SP
Text:00002690 BLGE #0x7020
Text:00002690 STMIB R10, {R1-R3, R5, R8, R10, PC}^
Text:00002690 LDRL R9, [PC], #0x23
Text:00002690 SUCHE #0x7020
Text:00002690 BUS #0x7020
Text:00002690 RSCPL R0, R4, #0x130B
Text:00002690 HUBLE R2, #0x2002
Text:00002690 UFX R1, LR, #1, #0x12
Text:00002690 BVC #0x7020
Text:00002690 LDRBTB R1, {R1, R4, R5, R9, LR, PC}
Text:00002690 STRUSBT R3, [R7], R6, LSR#13
Text:00002690 BPL #0x1020
Text:00002690 STRBQT R0, [R0], #0x75
Text:00002690 UPHIN.F32 D3A, D25, D22
Text:00002690 SUCDE #0x9182
Text:00002690 BLGT #0x7020
Text:00002690
Text:00002690 DCD 0xE80F3DD9, 0x6EBCD225, 0xA0F187F8, 0xA795F011, 0xE20B4538
Text:00002690 DCD 0x3F1399E0, 0x9C7A67A4, 0xC08141A, 0xF575275, 0x98A16627
Text:00002690 DCD 0x080CCE7, 0x6D0072, 0x7716A0A, 0x0F9B0BE, 0x7115E7
Text:00002690 DCD 0xF061D2, 0x5ADF60A, 0x87D0617C, 0x25FA9D08, 0xBF01B2A
Text:00002690 DCD 0x2131603A, 0xF670F0B, 0xA8A3BE8, 0x2B8A666, 0x3F1A3A2B
Text:00002690 DCD 0x0A12000, 0x16E4076, 0x9A0B6509, 0xD732000, 0x7F3F9226
Text:00002690 DCD 0x2D37335A, 0x4C116951, 0xF6A6E03A, 0x6810B97E, 0x6B0582C0
Text:00002690 DCD 0xD173FE02, 0xE09881C3, 0x8790B04F, 0xEE808015, 0x347C8C1F
00002690 00002690 text:start
Output window
Executing function 'main'...
Compiling file 'D:\Programs\IDA\ID61\ldc\lcnload.ld'...
Executing function 'OnLoad'...
IDA is analyzing the input file...
You may start to explore the input file right now.
Can not set debug privilege: 并非所有引用由特殊权限或设备驱动程序
Can not set debug privilege: 并非所有引用由特殊权限或设备驱动程序
Python 2.6.6 (3266:94297, Aug 24 2010, 15:46:52) [MSC v.1500 32 bit (Intel)]
IDAPython v1.5.0 final (serial 0) (c) the IDAPython Team <idapython@googlegroups.com>
Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished.
Python
AP: idle Down Disk: 506MB
```

一地鸡毛，像加了壳似的。静态分析是没法做了，只能通过动态调试的来看看。

上 GDB 工具，使用 ATTACH 命令，进入 zaTelnet 进程的领空。

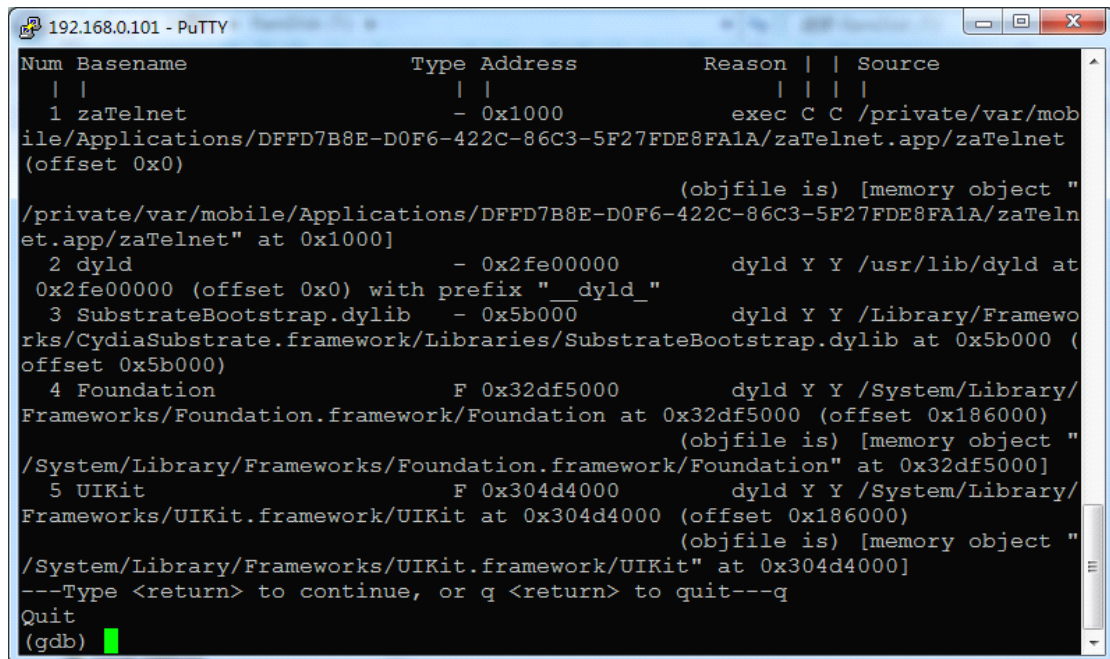


```
192.168.0.101 - PuTTY
apsd.28          lockdownd.19    syslogd.16
bash.748        lsd.128        usbethernetshari.15
bash.761        mDNSResponder.23 vsftpd.37
configd.27      mediaremoted.22 zaTelnet.754
fairplayd.K93.26 mediaserverd.21
(gdb) attach zaTelnet.754
Attaching to process 754.
Reading symbols for shared libraries . done
Reading symbols for shared libraries .....
.....warning: Could not find object file "/Users/mark/Documents/source/SpringBridge/build/SpringBridge.build/Debug-iphoneos/SpringBridge.build/Objects-normal/armv6/SpringBridgeAppDelegate.o" - no debug information available for "/Users/mark/Documents/source/SpringBridge/Classes/SpringBridgeAppDelegate.m".

warning: Could not find object file "/Users/mark/Documents/source/SpringBridge/build/SpringBridge.build/Debug-iphoneos/SpringBridge.build/Objects-normal/armv6/SpringBridge.o" - no debug information available for "/Users/mark/Documents/source/SpringBridge/Classes/SpringBridge.m".

..... done
0x33d29c00 in mach_msg_trap ()
(gdb)
```

再使用 info sharedlibrary 命令，查看加载的模块信息

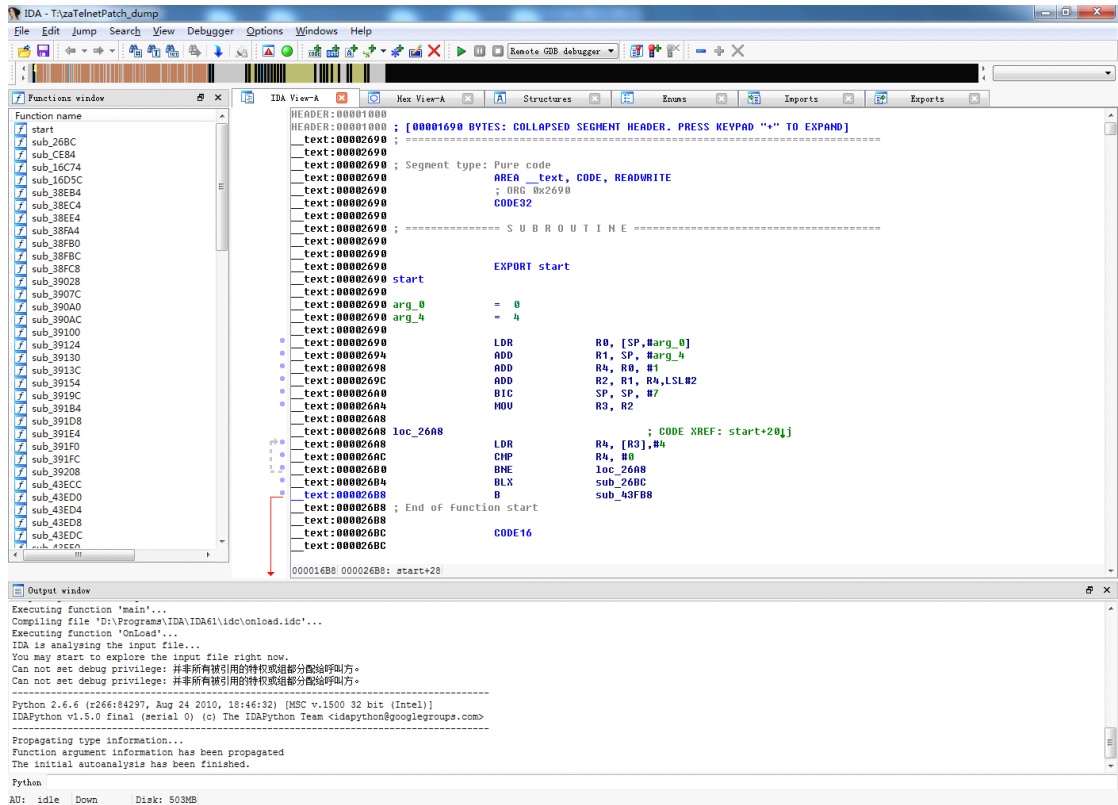


```
192.168.0.101 - PuTTY
Num Basename          Type Address          Reason | | Source
| |                    | |                | | | |
1 zaTelnet             - 0x1000           exec C C /private/var/mobile/Applications/DFFD7B8E-D0F6-422C-86C3-5F27FDE8FA1A/zaTelnet.app/zaTelnet (offset 0x0)
                                   (objfile is) [memory object "/private/var/mobile/Applications/DFFD7B8E-D0F6-422C-86C3-5F27FDE8FA1A/zaTelnet.app/zaTelnet" at 0x1000]
2 dyld                 - 0x2fe00000       dyld Y Y /usr/lib/dyld at 0x2fe00000 (offset 0x0) with prefix "__dyld_"
3 SubstrateBootstrap.dylib - 0x5b000         dyld Y Y /Library/Frameworks/CydiaSubstrate.framework/Libraries/SubstrateBootstrap.dylib at 0x5b000 (offset 0x5b000)
4 Foundation           F 0x32df5000      dyld Y Y /System/Library/Frameworks/Foundation.framework/Foundation at 0x32df5000 (offset 0x186000)
                                   (objfile is) [memory object "/System/Library/Frameworks/Foundation.framework/Foundation" at 0x32df5000]
5 UIKit               F 0x304d4000      dyld Y Y /System/Library/Frameworks/UIKit.framework/UIKit at 0x304d4000 (offset 0x186000)
                                   (objfile is) [memory object "/System/Library/Frameworks/UIKit.framework/UIKit" at 0x304d4000]
---Type <return> to continue, or q <return> to quit---
Quit
(gdb)
```

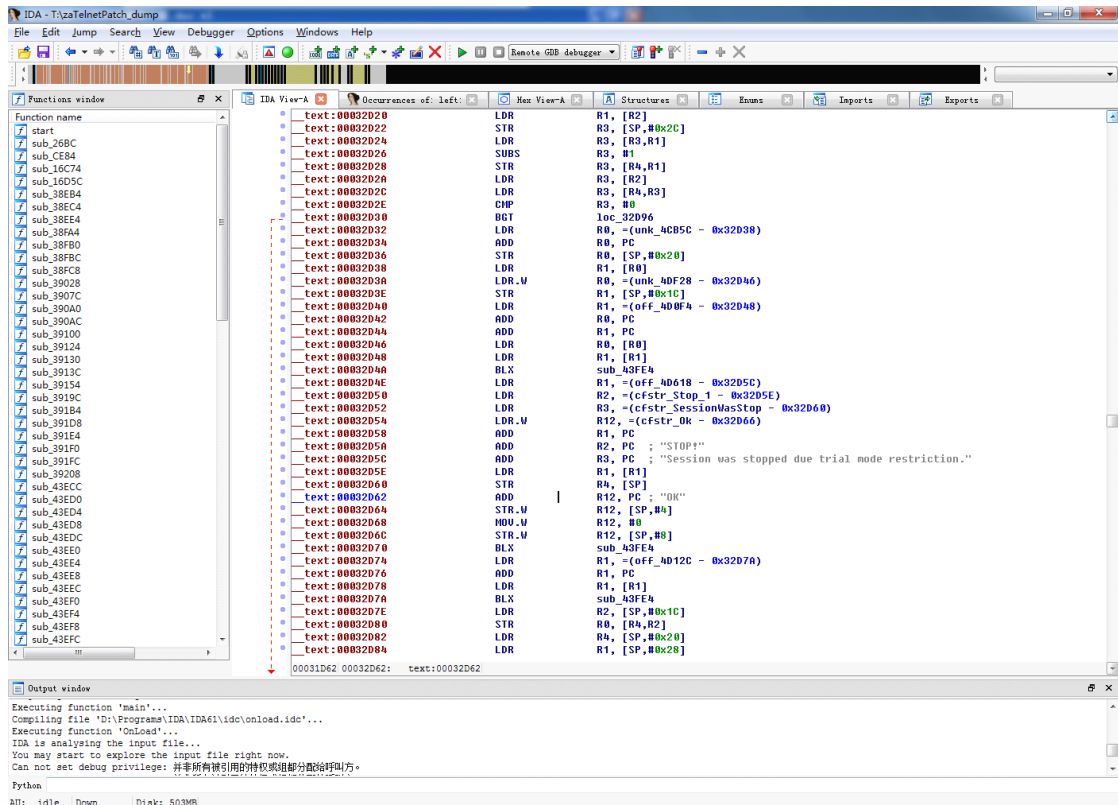
发现进程占用的空间 为 0x1000 ~ 0x5b000

再使用 dump zaTelnetPatch_dump 0x1000 0x5b000 命令，将内存 DUMP 到 zaTelnetPatch_dump 文件中。

再上 IDA 来分析下 zaTelnetPatch_dump 文件。



貌似是解过密的原始代码了。
通过提示特征信息，快速定位。



相关的处理

__text:00032CB8	PUSH	{R4-R7,LR}
__text:00032CBA	ADD	R7, SP, #0xC
__text:00032CBC	PUSH.W	{R8,R10,R11}
__text:00032CC0	VPUSH	{D8-D15}
__text:00032CC4	SUB	SP, SP, #0x74
__text:00032CC6	LDR.W	R3, =(off_44154 - 0x32CD2)
__text:00032CCA	STR	R0, [SP,#0x28]
__text:00032CCC	ADD	R0, SP, #0x40
__text:00032CCE	ADD	R3, PC
__text:00032CD0	STR	R2, [SP,#0x24]
__text:00032CD2	LDR	R3, [R3]
__text:00032CD4	STR	R7, [SP,#0x60]
__text:00032CD6	STR.W	SP, [SP,#0x68]
__text:00032CDA	STR	R3, [SP,#0x58]
__text:00032CDC	LDR.W	R3, =(unk_42C94 - 0x32CE4)
__text:00032CE0	ADD	R3, PC
__text:00032CE2	STR	R3, [SP,#0x5C]
__text:00032CE4	LDR.W	R3, =(loc_32F02 - 0x32CEC)
__text:00032CE8	ADD	R3, PC
__text:00032CEA	ORR.W	R3, R3, #1
__text:00032CEE	STR	R3, [SP,#0x64]
__text:00032CF0	BLX	sub_43F80

__text:00032CF4	LDR.W	R0, =(unk_4DE58 - 0x32D04)	
__text:00032CF8	LDR.W	R1, =(off_4DDF0 - 0x32D06)	
__text:00032CFC	MOV.W	R2, #0xFFFFFFFF	
__text:00032D00	ADD	R0, PC	
__text:00032D02	ADD	R1, PC	
__text:00032D04	LDR	R0, [R0]	
__text:00032D06	LDR	R1, [R1]	
__text:00032D08	STR	R2, [SP,#0x44]	
__text:00032D0A	BLX	sub_43FE4	
__text:00032D0E	TST.W	R0, #0xFF	
__text:00032D12	BEQ.W	loc_32E7E	
__text:00032D16	LDR.W	R2, =(unk_4CB98 - 0x32D22)	
__text:00032D1A	LDR	R3, [SP,#0x28]	
__text:00032D1C	LDR	R4, [SP,#0x28]	
__text:00032D1E	ADD	R2, PC	
__text:00032D20	LDR	R1, [R2]	
__text:00032D22	STR	R3, [SP,#0x2C]	
__text:00032D24	LDR	R3, [R3,R1]	
__text:00032D26	SUBS	R3, #1	按一次计数器减 1
__text:00032D28	STR	R3, [R4,R1]	
__text:00032D2A	LDR	R3, [R2]	
__text:00032D2C	LDR	R3, [R4,R3]	
__text:00032D2E	CMP	R3, #0	

在 GDB 中，使用 00032D26 命令，内存修改代码

```
(gdb) set {unsigned char} 0x00032D26 = 0x00
```

```
(gdb) x/i 0x00032D26
```

```
0x32d26:      subs    r3, #0
```

```
(gdb)
```

继续运行程序

```
(gdb) c
```

Continuing.

见证奇迹的时刻到了。你就会发现，按任何一个按键之后，右上角的计数不再减少了。至此完成的破解。当然实际使用时，总不能架一个 GDB 吧，总需要自动 Patch 吧。

补丁：

进程执行时会检查 /Library/MobileSubstrate/DynamicLibraries/ 目录中 扩展名 plist 的文件中的描述来决定是否动态同名的扩展名为 dylib 的文件。用这个方法就可以实现在 zaTelnet 执行自动加载我们的 Patch 动态库。

附上 Patch 内存的核心代码，大部分都是从网上抄来的，呵呵。

```
static void patch_mem(void *p, unsigned char data)
```

```
{
```

```
    int page = getpagesize();
```

```

uintptr_t address = (uintptr_t)(p);
uintptr_t base = address / page * page;
mach_port_t self = mach_task_self();
kern_return_t error;

if ((page - (uintptr_t)(p) - base) < 12)
    page *= 2;

if (error = vm_protect(self, base, page, FALSE, VM_PROT_READ | VM_PROT_WRITE |
VM_PROT_COPY))
{
    // fprintf(stderr, "vm_protect():%d\n", error);
    return;
}
*(unsigned char*) p = data;
// __clear_cache((char*)(p), (char*)(p + 1));

if (error = vm_protect(self, base, page, FALSE, VM_PROT_READ |
VM_PROT_EXECUTE))
{
    // fprintf(stderr, "vm_protect():%d\n", error);
    return;
}
}

static mach_msg_type_number_t read_mem(void *p, char *data, mach_msg_type_number_t size)
{
    int page = getpagesize();
    vm_address_t address = (vm_address_t)(p);
    mach_port_t self = mach_task_self();
    pointer_t buf;
    kern_return_t error;

    if (error = vm_read(self, address, size, &buf, &size))
    {
        //fprintf(stderr, "vm_read():%d\n", error);
        return 0;
    }
    memcpy(data, (void*)buf, (int)size);
    return size;
}

int _X_HIDDEN patcher_(unsigned char* pFun, unsigned char* pIns)
{

```

```

unsigned char *p;
char buf[16];
p = pFun;
if (read_mem(p, buf, sizeof(buf)) == sizeof(buf))
{
    if (0 == memcmp(buf, "\xF0\xB5\x03\xAF\x2D\xE9\x00\x0D\x2D\xED\x10\x8B", 12))
    {
        p = pIns;
        if (p[0] == '\x01' && p[1] == '\x3b')
        {
            // 修改内存属性
            // p[0] = '\0';
            patch_mem(p, '\0'); // 停止减少按键计数
            return 1;
        }
    }
}
return 0;
}

```

最后：

就不贴执行文件了，重点是分析过程，初次接触苹果系统，水平有限，不足之处，欢迎各位指教。

最后来一张效果图

```

ufomato-iPad:~ root# uname -a
Darwin ufomato-iPad 11.0.0 Darwin Kernel Version 11.0.0: Wed Mar 30 18:52:42 PDT 2011; root:xnu-1735.46~10/RELEASE_ARM_S5L8940X iPad2,1 arm K93AP Darwin
ufomato-iPad:~ root#

```