[quote]

[align=center][b][color=orangered][size=200%]' 免 责 声 明 '[/size][/color][/b][/align]

======

[b][size=150%]该[/size]教材的出版时间是2009年，由于游戏与软件的迭代特性，本书的内容已经与现在你实际接触到的游戏与软件内容有所差异，有些内容可能已经过时，甚至是被淘汰“过期的”，请大家自己斟酌参考。若出现由于原文与本帖翻译有所不同而导致的错误或损失，概不负责。[/b]

[/quote]

[quote]

[align=center][b][color=orangered][size=200%]' 扇 贝 碎 碎 念 '[/size][/color][/b][/align]

======

[b][size=150%]这[/size][/b]是国外教材《Beginning Lua with World of Warcraft Addons》的第十二个章节。全书共14个章节和2个附录。这是我翻译的第五个章节，其余章节的翻译内容我会在以后陆续发帖分享。[b]也希望阅读的人不要吝啬你们的点赞&#128077;，只要点一下便能给予我莫大的支持和鼓励。[/b]

对于这本教材，我是不推荐把这本书作为新版本的插件开发指南。但若从入门或业余角度来看，这本书依然是本相当优秀的教程，可用对魔兽和插件有个整体的了解。虽然阅读过时的教程会特别令人感到不适，特别是当发现输入的代码是已经被游戏淘汰的时候。但希望大家抱着个宏观的状态来阅读，如果遇到无法进行下去的代码错误，请参考你自己正在使用的插件或最新的FrameXML或其他API。从正在使用的插件代码切入可能是最高效的方法。

我已经出了我到目前为止所翻译的几个章节的合集，有兴趣的可以自行浏览[tid=24502396][color=orangered][合集]Lua与魔兽世界插件——《Beginning Lua with World of Warcraft Addons》[/color][/tid]

本次的翻译可能稍微粗糙了一些，[del]因为忙着练级和刷橙装[/del]，大家多担待一下，我尽量把翻译错误更正一下。

[size=110%][b]最后，如果有人在阅读过程中，如果发现哪里翻译错误，或错别字需要更正，请发评论或发私信告知，谢谢大家。[/b][/size]

接下来让我们正式开始这次的阅读吧。

[/quote]

[quote]

[align=center][b][color=orangered][size=200%]' 目&#8195;&#8195;录 '[/size][/color][/b][/align]

======

[size=180%][b]&#9632; 第十一章 使用安全模板 (Working with Secure Templates)[/b][/size]

[b][size=130%]&#9679; 安全及受污染代码 (Secure and Tainted Code)[/size][/b]

[b]&#9675; 受限函数 (Restricted Function)[/b]

[b]&#9675; 钩住安全函数 (Hooking Secure Functions)[/b]

[b]&#9675; 污染如何扩散 (How Taint Spreads)[/b]

[b][size=130%]&#9679; 为单位框体使用安全模板 (Using Secure Templates for a Unit Frame Mod)[/size][/b]

[b]&#9675; 受保护框体的限制 (Restrictions on Protected Frames)[/b]

[b]&#9675; 可用的安全模板 (Available Secure Templates)[/b]

[b]&#9675; 建立带有属性的单位框体 (Building Unit Frames with Attributes)[/b]

· 为我们的单位框体建立一个模板 (Building a Template for Our Unit Frames)

· 使用模板 (Using the Template)

· OnClick处理程序背后的代码 (The Code Behind the OnClick Handler)

[b]&#9675; 使用团长模板 (Using Group Header Templater)[/b]

[b][size=130%]&#9679; 使用安全动作按钮模板 (Using Secure Action Button Templates)[/size][/b]

[b]&#9675; 一个简单的安全动作按钮 (A Simple Secure Action Button)[/b]

[b]&#9675; 高级安全动作按钮 (An Advanced Secure Action Button)[/b]

· 创建一个带有XML的安全按钮 (Creating a Secure Button with XML)

· 摆弄属性 (Playing Around with Attributes)

[b][size=130%]&#9679; 总结 (Summary)[/size][/b]

[/quote]

======

[align=center][color=orangered][b][size=200%]&#9632; 第十一章 使 用 安 全 模 板[/size][/b]

[size=130%][b](Working with Secure Templates)[/b][/size][/color][/align]

你是否曾经想过，是什么阻碍了你开发一个插件来完成你游戏的自动化？虽然这样的一个插件会破坏EULA(最终用户许可协议)，你不会想这样干，因为你不想失去你的《魔兽世界》账号。但到底是什么技术阻止了你开发这样一个插件呢？

整个UI是用Lua和XML编写的，因此有一些函数允许你移动你的角色和施法。例如有一个函数CastSpellByName(“name”)，它允许你施放一个法术。但试着用一个简单的法术来进行攻击：

[quote]

/script CastSpellByName(“Attack”)

[/quote]

你将收到一个错误信息，说你的操作已被阻止，因为只对暴雪用户界面有效。在本章中，我们将探索游戏如何决定哪些属于默认UI，哪些不属于。我们也会讨论在什么情况下我们可以使用一个插件来施放法术和执行某些受限的动作。

=======

[color=blue][b][size=160%]&#9679; 安 全 及 受 污 染 代 码[/size]

[size=130%](Secure and Tainted Code)[/size][/b][/color]

在《魔兽世界》中，所有的Lua值和引用要么是安全的，要么是受污染的。从插件或通过斜杠命令创建的一切都将受污染，而游戏创建的一切都是安全的。当执行路径(执行路径是当前运行的代码，在游戏开始时启动，在所有调用的函数返回时结束)也可以是安全的，也可以是受污染的，但是它总是以安全的开始。一旦它遇到任何受污染的东西，他就会被污染。

游戏在加载时使用数字签名来检查默认UI的完整性，只有经过暴雪数字签名的代码才会在加载时被标记为安全的(secure)，所以暴雪UI创建的每个值也都被标记为安全的。这也意味着你所有的插件自动被标为受污染的(tainted)，你创建的值也是受污染的。

这对你来说意味着什么？有一些特定的函数，称之为受限函数(restricted functions)，只能从不受污染的执行路径调用。CastSpellByName就是这些受限函数之一。但我们不需要在这里讨论这些函数，因为我们将永远无法调用它们中的任何一个。让我们看一个演示受污染和安全代码的示例。

[b][size=130%]&#9675; 受限函数(Restricted Function)[/size][/b]

在补丁2.0中引入污染系统(taint system)之前，我曾经写过一个插件，可以在某些情况下防止某些法术的施放。我为什么要这样做？这个插件是黑翼之巢(Nefarian in Blackwing Lair)的boss模块。以防你不知道，这个相当古老的boss，奈法利安(Nefarian)，会不时地对某个职业上一个减益(debuff)，腐化该职业的一些法术和能力。对于我这样的神牧来说，这个减益效果会把所有治疗法术变成伤害法术，所以当你中了这个减益后继续治疗坦克是非常糟糕的一个情况。该插件发现了这一点，并阻止所有的治疗法术。

在这个插件中，阻止施法过程中钩子(hook)起到了一些作用。其中一个钩子位于函数CastSpellByName中。让我们试着为这个函数写一个简单的钩子。注意，这个函数只在使用/cast斜杠命令时调用(例如在宏中)，而不是在点击动作条中的按钮时调用。动作条的工作方式略有不同(我们将在本章的后面看到它们)，为了让其简单一些，我将只向你展示CastSpellByName的钩子。

下面的代码可以很容易地在游戏中被一个插件执行，如TinyPad：

[code=lua]

local old = CastSpellByName

CastSpellByName = function(...)

print(...)

return old(...)

end

[/code]

这将打印出传递给CastSpellByName函数的所有参数，然后调用它。实际的插件有一个对变量的简单检查，如果调试是活动的(active)，该变量被设置。现在尝试执行下面的斜杠命令或使用一个施放法术的宏。

[quote]

/cast Attack

[/quote]

你将得到与直接调用CastSpellByName相同的错误消息，即使对它的实际调用来自默认UI。我们的简单钩子刚刚打破了所有可以施放法术的斜杠命令，/castrandom和/castsequence也同样会受到影响。

让我们看看通过跟踪调用发生了什么。执行路径从聊天框的DnEnterPressed事件中作为安全执行开始。之后遍历文件FrameXML\ChatFrame.lua中的大量代码，这些代码将你的命令标识为斜杠命令，直到最终到达表SecureCmdList中的斜杠命令处理程序。这项过程就像普通的斜杠命令处理程序一样，不用担心不同的名字。处理程序解析斜杠命令，确定你是试图使用一个物品还是一个法术，并获取可选的目标参数。当Lua到达以下处理程序时，执行仍是安全的，该处理程序也定义在文件FrameXML\ChatFrame.lua中。

[code=lua]

SecureCmdList[“CAST”] = function(msg)

local action, target = SecureCmdOptionParse(msg);

if (action) then

local name, bag, slot = SecureCmdItemParse(action);

if (slot or GetItemInfo(name)) then

SecureCmdUseItem(name, bag, slot, target);

else

CastSpellByName(action, target);

end

end

end

[/code]

它现在可以调用存储在全局变量CastSpellByName中的函数了，该全局变量通常是安全的。但我们用自定义钩子函数重写了它，这是不安全的。

现在，执行路径在进入函数的那一刻就被污染了。这个受污染的执行现在调用了真正的函数CastSpellByName，它仍然是安全的(但是存储在本地变量old中的引用是受污染的，你会看到污染是如何一点点扩散的)。这个函数仍然是安全的，这并不能帮助我们——它是从一个受污染的执行路径调用的，因此会失败。消除这种污染的唯一方法是重新加载你的用户界面。

我们可以使用issecurevariable(tb1, key)来检查存储在表中的值是否被污染。第一个参数是可选的，默认情况下是全局环境\_G。因此，我们可以使用以下代码检查CastSpellByName的完整性。

[code=lua]

print(issecurevariable(“CastSpellByName”))

[/code]

它将打印nil，后面跟着空字符串。第二个返回值是引起污染的插件，空字符串意味着它被用户输入的/script命令污染了。正如你所看到的，不可能使用传统的钩子方法钩住一个受保护的函数。但是还有另一种方法来钩住函数。

[b][size=130%]&#9675; 钩住安全函数(Hooking Secure Functions)[/size][/b]

接口API提供了hooksecurefunc(tb1, key, hook)函数，它可以用来挂起一个安全函数而不影响它的污染。同样，第一个参数tb1是可选的，默认情况下是全局环境。最后一个参数是每次调用钩子函数都会执行的函数。钩子接收与原始函数相同的参数。

它实际上不是一个“真正的”钩子，因为它既不可能修改传递给原始函数的参数，也不可能改变它的返回值。所以不可能写一个像我签名描述的插件。但是这样的post-hooks通常就足够了，特备是当你相对默认UI元素进行小的更改时。使用这种钩子的一个例子是DBM的战场模块。它钩住更新得分框体的函数，并将框体中玩家的颜色设置为各职业的颜色。

重载你的UI，如果你还没用这个功能摆脱旧钩子和污染，这样我们旧可以在任何，这样我们可以。让我们测试一下CastSpellByName现在是否安全。

[code=lua]

print(issecurevariable(“CastSpellByName”))

[/code]

它现在打印“1”且后面跟着nil(如果变量是被污染的，则是被变量污染的模块)，则意味着一切正常。我们现在应用我们的新钩子，它也会打印它接收到的参数：

[code=lua]

hooksecurefunc(“CastSpellByName”, function(...))

print(...)

end)

[/code]

我们现在可以再次检查它是否仍然安全：

[code=lua]

print(issecurevariable(“CastSpellByName”))

[/code]

我们仍然认为结果是1和nil，所以它没有被污染。你现在可以错误你的施法或一个简单的斜杠命令，例如/cast Attack，它工作正常，并且调用了钩子函数。

注意，只有当函数存储在表中或全局变量(从技术上讲，全局变量只是表中的一个条目)时，这种连接函数的方法才是有效。显然，你不会遇到存储在局部变量中的安全函数，因为这个局部变量只是在创建它的文件中可见，此文件是数字签名的，因此不能再其中写入代码。但是你可能会遇到这样的情况，你需要再不损坏框架的情况下挂钩它的事件处理程序。你可以使用此框架上的方法框架：SetHook(处理程序，钩子)向其他事件处理程序之一添加安全钩子。我们在前面讨论事件处理程序时看到了这个方法。

你应该总是使用hooksecurefunc或frame:SetHook，如果你想挂起暴雪定义的还没被污染的功能。这样做的原因是污染会迅速蔓延到整个默认UI。一个被污染的值可以迅速感染其他的功能和值。让我们看看污染是如何扩散的，以及我们如何跟踪它。

[b][size=130%]&#9675; 污染如何扩散 (How Taint Spreads)[/size][/b]

当执行受污染的函数或访问受污染的值时，安全执行会失去其安全状态并受到污染。当在受污染的执行路径中创建或修改一个值时，该值被污染。注意，仅仅从受污染的执行路径访问安全值并不会污染它。你总是在访问值时创建一个被污染的副本(请注意，包含表之类的复杂对象的变量的值始终是对该表的引用，因此你将获得对安全表的污染引用，而不是该表的污染副本)。

因此，一个污染变量可以从访问之初就污染了执行路径，因此可以迅速在整个默认用户界面中传播。污染发生后由该代码创建的所有值也将被污染。然后其他代码将访问这些值，并且也会受到污染，以此类推。你可以通过保留单个值来快速破坏你没想到会破坏的东西。

到目前为止，我们使用了很多由默认UI的Lua代码直接调用的代码，比如斜杠命令处理程序或下拉菜单。但它们没有破坏任何东西，因为暴雪在其代码中处理了污染问题。这段代码要么确保重要的值不被污染，要么使用了函数securecall(func, ...)。这个函数接收另一个函数，并使用给定的参数执行它，在函数返回后恢复执行路径的旧的污染状态。这意味着在你自己的代码中使用此函数是没有意义的，因为污染状态将恢复到污染状态，这对你没有任何帮助。

在游戏的早期版本中，污染扩散是一个大问题，尤其是在引入污染系统后的几个月。然而，暴雪现在采取了许多预防措施，通过挂住不处理与受保护功能有关的功能的随机功能来防止插件破坏默认UI。默认UI的所有关键部分确保它们不访问可能被插件污染的变量。不使用hooksecurefunc就不直接处理任何与受保护函数相关的函数是非常安全的。

但是，当你想post-hook一个安全函数时，你仍然应该使用hooksecurefunc。它仍可能会发生的情况，污染一个看起来无害的变量或函数会破坏部分默认UI。但是暴雪会很快发现并修复这些问题。在我的示例中，污染世界地图框体的WorldMapFrame\_Update函数可能会破坏设置焦点选项，这当然是一个意外的结果。不过，这个问题在3.1.1补丁中得到了修复，现在可以安全地污染WorldMapFrame\_Update了。

但是，如果你发现了代码破坏了默认UI，则运行一下命令并重新加载UI，以获取更多调试信息：

[quote]

/console taintlog 1

[/quote]

你可以执行由于插件而被阻止的功能，注销后，你将在logs文件夹中获得taint.log文件。该文件包含有关受保护的失败调用和造成异常调用的其他信息。

例如，函数UnitHealth中的以下钩子会污染单元框体的关键部分，并且绝对不能在插件中使用此类钩子。

[code=lua]

local old = UnitHealth

UnitHealth = function(...)

return old(...)

end

[/code]

[quote]

注意：污染问题很难不在插件中出现。这意味着你拥有的任何插件都可以破坏这个例子，即使它看起来与实际问题完全无关。当你从某个用户那里收到有关污染的错误报告时，请记住这一点。当尝试调试一个污染问题时，总是要求一个已安装的插件列表和污染日志。

[/quote]

现在尝试与怪物交战并击杀它。你会在聊天框中看到一条消息，说由于插件导致默认UI操作被阻止。损坏的功能是默认UI的目标的目标框体，如果你的目标更改了目标，它将无法正确更新。这已记录在taint.log文件中。

[quote]

[img]./mon\_202012/05/-10er0nQ5-2f3vZmT3cSlj-bc.png[/img]

[/quote]

日志文件首先告诉你是哪个插件造成了污染，这个插件在本例中式MACRO\_TAIN，因为我从TinyPad运行了这个钩子，它使用RunScript(str)以Lua代码的形式执行输入的文本。通过RunScript加载的代码被认为是魔兽世界的一个宏，因为命令/script内部也使用RunScript。接下来的几行显示了污染该变量的执行路径的调用堆栈。

日志中的下一行告诉你发生的实际问题。TargetofTarget\_Update()函数试图执行一个收保护的操作，但在访问全局变量UnitHealth时，由于执行受到污染而被阻止。下一行显示失败调用的调用堆栈和失败的实际调用TargetofTargetFrame: show()。处理受保护方法，在战斗中不能从受污染代码中调用。尝试执行这样的调用会在聊天框中产生一条通知，并在污染日志中产生一个条目。我们将在下一节讨论受保护的方法和框体。我们现在知道我们不能做什么，看看我们能做什么。在某些情况下，仍然可以创建一个额外插件来施放法术或标记单位。

=======

[color=blue][b][size=160%]&#9679; 为 单 位 框 体 使 用 安 全 模 板[/size]

[size=130%](Using Secure Templates for a Unit Frame Mod)[/size][/b][/color]

安全模板是而已用于框体的普通XML模板。这些模板已经具有某些功能，这些功能只能从安全代码中调用，比如针对团队成员施放法术。这些模板的可能用途是单位框体和操作按钮。但它们也被认为是受保护的框架，在战斗中不能调用某些方法。让我们写一个简单的例子，为你和你当前的目标显示单位框体。

游戏提供了一个安全模板，可以用于此：SecureUnitButtonTemplate。如上所述，你从这个模板创建的每一个框体都将是一个受保护的框体，并且当你在战斗中，受保护的框架不能做某些动作。在查看可用模板之前，让我们先看看受保护框架的限制。

[b][size=130%]&#9675; 受保护框体的限制 (Restrictions on Protected Frames)[/size][/b]

在战斗中，不能从受污染代码中调用受保护框体的下列方法。当你不在战斗状态时，所有这些方法在正常的框体中仍然有效。

[quote]

1、frame:Show()

2、frame:Hide()

3、frame:SetPoint(point, relativeTo, relativePoint, x, y)

4、frame:SetWidth(width)

5、frame:SetHeight(height)

6、frame:SetAttribute(attribute, name) (you will see the purpose of this method in just a bit)

[/quote]

受保护框体的子框体不一定受到保护，因此可用将普通框体作为子框体添加到安全框体中。但是所有的父节点和你锚定的安全框体都将自动受到保护，因为如果它们没有被保护，你可用简单地显示/hide或移动父节点或锚点来绕过这些限制。这有时会导致意想不到的问题，因为框体可能会突然受到保护。因此，当选择父类和锚点的保护框体时要小心。可用使用方法frame:IsProtected()来检查一个框体当前是否受到保护。

这些限制的目的是为了防止像CT-HealMonitor这样的插件，它存在于2006年，在安全模板被引入《魔兽世界》2.0之前。这个插件显示了一些单位框体，这些框体是根据它们它们缺失的健康情况进行排序的，也就是说，健康值最低的玩家排名最高。治疗们可用简单地点击列表中排名靠前的团员，并一直对他进行治疗。当然，你仍然可用创建一个显示玩家生命值的模块，但是你不能在战斗中点击这些玩家。还有一个额外的可用自动选择最好的治疗法术和最优的目标。这样的插件已经不可能了。你只能通过从安全模板创建的受保护框架上显示单击来执行预定义的受保护操作。

受保护的框体没有分配给他们特殊的方法，但是安全模板没有被污染，因此可以调用受保护的函数。不同的安全模板提供了与受保护功能相关的不同预定义功能。在开始使用示例模块之前，让我们看看所有可用的模板，以对整个系统进行概述。

[b][size=130%]&#9675; 可用的安全模板 (Available Secure Templates)[/size][/b]

基本上有两种类型的插件，需要充分利用安全模板：单位框体插件和动作条插件。动作按钮只需要一个简单的模板，它允许你执行当你点击它们时可能被阻止的动作，而单位框体需要很多不同的模板。下面的列表显示了所有可用的安全模板：

[quote]

[list]

[\*]SecureActionButtnTemplate：可用于实现按钮，当它们被单击时执行受限制的功能。

[\*]SecureUnitButtonTemplate：一个单位框体的模板，它被绑定到一个特定的单位，当你单击框体的目标。它也可以显示和隐藏自动如果单位的存在或消失。

[\*]SecureGroupHeaderTemplate：用于显示raid组的单元按钮列表的抽象标题。这个框体不应该直接使用，它是以下模板的一个模板。

[list]

[\*]Secupartyheadertemplate：一个模板，可以包含SecureUnitButtonTemplate以显示你的整个组，此模板继承自SecureGroupHeaderTemplate。

[\*]SecureRaidGroupHeaderTemplate：可用于团队的子组的模板，它的工作原理类似于团长(party header)

[\*]SecureGroupPetheaderTemplate：一个宠物列表的抽象标题，不应该直接使用，if充当以下模板的模板。

[\*]SecurePartyPetHeaderTemplate：你的角色里的宠物模板。

[/list]

[\*]SecureRaidButtonTemplate：团队中的宠物模板。

[/list]

[/quote]

所有这些安全模板只是提供功能，没有设计。你完全可以根据你的需要自由地设计它们。你可以将普通框体作为子框体添加到这些安全模板中，并且对这些子框体没有任何限制。

现在让我们创建一个简单的例子，它只显示两个单位框体：一个给你自己，一个给你的目标。我们需要从SecureUnitButtonTemplate继承这两个框体，但是我们如何将其绑定到一个特定的单位，以及如何告诉框体它应该做的受限的操作？仅仅使用一个简单的变量或函数调用就会破坏模板的重要代码。还有另一种通过安全框架的属性与此框架通信的方法。

[b][size=130%]&#9675; 建立带有属性的单位框体 (Building Unit Frames with Attributes)[/size][/b]

安全框体的属性是通过调用方法SetAttribute(attribute, value)来设置的，其中attribute是一个字符串，用于标识属性和value。GetAttribute(attribute)可用于检索属性的当前值。SetAttribute是受保护框体的一种受保护方法，这意味着在战斗中我们不能改变属性。在我们的示例中，最重要的属性是unit按钮模板和属性单位。这将设置框体所绑定的单位。

现在我们已经了解了如何使用模板，所有让我们从这里开始构建示例模块。为我们的插件创建一个新文件夹并命名为MyUnitFrames或类似的，然后添加适当的TOC文件，并向该TOC文件添加XML。我们现在为我们的两个单位框体构建一个从SecureUnitButtonTemplate继承模板。

[b]· 为我们的单位框体建立一个模板(Building a Template for Our Unit Frames)[/b]

从一个安全模板创建是没有问题的，当然，创建的模板也将受到保护。这个保护实际上也是一个XML属性，它在文件FrameXML\SecureTemplates中的模板SecureFrameTemplate中设置。XML，用于所有安全框体。XML属性总是继承自模板。XML看起来是这样的。

[code=xml]

<Frame name=”SecureFrameTemplate” protected=”true” virtual=”true”/>

[/code]

您不能在模板中将其更改为talse，因为在框架或模板已被保护之后，你无法删除它的保护属性。在我们开始创建框架之前，将标志跟元素添加到XML中。

[code=xml]

<Ui xmls=http://www.blizzard.com/wow/ui xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance xsi:schemaLocation=”http://www.blizzard.com/wow/ui/..\FrameXML\UI.xsd”>

</Ui>

[/code]

将以下内容添加到XML文件中(在标记之间)，以创建一个类似于简单单位框体的简单模板。它仅由竖屏，名称的字体字符串和健康栏(health bar)组成。

[code=xml]

<Button name=”MyUnitFrameTemplate” virtual=”true” movable=”true” inherits=”SecureUnitButtonTemplate”>

<Size>

<AbsDimension x=”96” y=”32”>

</Size>

<Layers>

<Layer level=”ARTWORK”>

<Texture name=”$parentPortrait”>

<Size>

<AbsDimension x=”32” y=”32”/>

</Size>

<Anchors>

<Anchor point=”LEFT”>

</Anchors>

</Texture>

<FontString name=”$parentName” inherits=”GameFontNormal”>

<Anchors>

<Anchor point=”TOPLEFT” relativePoint=”TOPRIGHT” relativeTo=”$parentPortrait”/>

</Anchors>

</FontString>

</Layer>

</Layers>

<Frames>

<StatusBar name=”$parentHealth” minValue=”0” maxValue=”1”>

<Size>

<AbsDimension x=”64” y=”16”/>

</Size>

<Anchors>

<Anchor point=”BOTTOMLEFT” relativePoint=”BOTTOMRIGHT” relativeTo=”$parentPortrait”/>

</Anchors>

<BarTexture file=”Interface\TargettingFrame\UI-StatusBar”/>

</StatuBar>

</Frames>

<Scripts>

<OnDragStart>

self:StartMoving()

</OnDragStart>

<OnDrageStop>

self:StopMovingOrSizing()

</OnDragStop>

</Scripts>

</Button>

[/code]

我们在模板中只定义了两个脚本处理程序，这些处理程序处理拖动框体。注意，启动和停止移动或调整大小不是受保护的函数，这意味着在战斗中使用鼠标移动画面是没有问题的。但是在战斗中通过调用它移动到SetPoint是无效的。

我们还需要一个OnLoad处理器，它通过调用一些函数来初始化框体，例如，我们需要调用RegisterForDrags方法。我们还需要为事件UNIT\_HEALTH提供一个事件处理程序，该事件在每次单位的运行状况发生改变时触发，我们将更新那里的有生命条(health bar)。

但是在模板中定义这两个处理程序实际上并不明智，因为我们从这个模板创建的两个框体差别太大了。但是稍后在框架中定义这些处理程序会生成一些重复的代码，正如我所说的，重复的代码是不好的。当然可以在OnLoad处理程序中定义部分处理程序，并使用带有一些初始化函数的Lua文件来避免几行重复代码。但是，这个示例的目的是演示使用几行Lua代码的XML单位框体。这几行重复的代码不值得为避免它们而增加额外的复杂性。

[b]· 使用模板(Using the Template)[/b]

现在，我们已经讨论了安全的单位按钮模板的基本内容，现在从模板中创建实际的两个框体了。请注意，框体非常简单，它们只显示一个竖屏(portrait)，以及一个简单的生命条。默认情况下，两个框体都以屏幕中间开始，但它们可以在任何时候被拖动。你可以添加自己的素材作为纹理和一个法力条(mana bar)去获得功能齐全的单位框体模块。将其添加到示例中将增加不必要的复杂性，因为我们需要处理很多任务，例如根据法力类型(法力/能量/怒气等)正确着色。

让我们从玩家框体开始，它需要在OnLoad处理程序中执行以下任务：注册点击和拖动事件，将$parentName设置为玩家名称，并将$parentPortrait设置为玩家肖像的纹理。它还需要调用安全模板提供的初始化函数：SecureUnitButton\_OnLoad(self, unit, menuFunc)。这里，unit是应用该框体的单位的ID，这个方法会调用self:SetAttribute(“unit”, unit)所以我们不需要这样做。menuFunc是一个函数，当玩家右键点击玩家框体时调用。我们将使用默认的玩家下拉菜单。

事件处理程序使用API函数SetPortraitTexture(texture, unit)，该函数将纹理作为它的第一个参数，将单位ID作为它的第二个参数。这个函数创建一个单位的肖像，并将其作为纹理，如果单位的外观发生变化(例如，当你装备了一个新的头盔)，它也会更新这个肖像(portrait)。

我们还需要事件UNIT\_HEALTH和一个事件处理程序，如果被arg1 == “player”调用事件，该事件处理程序将更新状态栏。下面是代码：

[code=xml]

<Button name=”MyPlayerFrame” parent=”UIParent” inherits=”MyUnitFrameTemplate”>

<Anchors>

<Anchor point=”RIGHT” relativePoint=”CENTER”/>

</Anchors>

<Scripts>

<OnLoad>

self:RegisterForClicks(“AnyUp”)

self:RegisterForDrag(“LeftButton”)

self:RegisterEvent(“UNIT\_HEALTH”)

\_G[self:GetName()..”Name”]:SetText(UnitName(“player”))

SetPortraitTexture(\_G[self:GetName()..”Portrait”], “player”)

SecureUnitButton\_OnLoad(self, ”player”, function()

ToggleDropDownMenu(1, nil, PlayerFrameDropDown, self, 106, 27)

end)

</OnLoad>

<OnEvent>

local arg1 = ...

if arg1 == “player” then

\_G[self:GetName()..”Health”]:SetValue(

UnitHealth(arg1)/UnitHealthMax(arg1)

)

end

</OnEvent>

</Script>

</Button>

[/code]

我们现在有一个功能齐全的玩家框体，工作得很好。我们添加一个目标框体，这个稍微有点复杂，因为我们需要处理目标的改变。我们可以在这里使用事件PLAYER\_TARGET\_CHANGED并在事件处理程序中更新目标的肖像、名称和生命值状况。但是如果我们没有选择任何目标会发生什么呢？我们不能只在框架上调用方法self:Hide(),因为在战斗中不允许隐藏一个受保护的框体。

游戏提供了所谓的安全状态驱动(secure state driver)来做到这一点。我们可以为我们的框体调用RegisterUnitWatch(frame)函数来告诉安全模板API来隐藏框体。当单位的属性被设置，单位不再存在时，它会隐藏框体，并且当它出现时显示它。

将以下按钮添加到XML以创建目标框体。OnLoad和OnEvent处理程序都与玩家框体的处理程序非常相似，因此代码应该很容易理解。请注意，无需使用OnLoad设置目标的肖像和名称，因为登陆时不会选择目标。

[code=xml]

<Button name=”MyTargetFrame” parent=”UIParent” inherits=”MyUnitFrameTemplate”>

<Anchors>

<Anchor point=”LEFT” relativePoint=”CENTER”/>

<Anchors>

<Attributes>

<Attribute name=”unit” type=”string” value=”target”/>

</Attributes>

<Scripts>

<Onload>

self:RegisterForClicks(“AnyUp”)

self:RegisterForDrag(“LeftButton”)

self:RegisterEvent(“UNIT\_HEALTH”)

self:RegisterEvent(“PLAYER\_TARGET\_CHANNGED”)

SecureUnitButton\_OnLoad(self, “target”, function()

ToggleDropDownMenu(1, nil, TargetFrameDropDown, self, 120. 10)

end)

RegisterUnitWatch(self)

</OnLoad>

<OnEvent>

if event == “UNIT\_HEALTH” then

local arg1 = ...

if arg1 == “target” then

\_G[self:GetName()..”Health”]:SetValue(

UnitHealth(arg1)/UnitHealthMax(arg1)

)

end

elseif event == “PLAYER\_TARGET\_CHANGEED” then

\_G[self:GetName()..”Health”]:SetValue(

UnitHealth(“target”)/UnitHealthMax(arg1)

)

SetPortraitTexture(\_G[self:GetName()..”Portrait”], “target”)

\_G[self:GetName()..”Name”]:SetText(UnitName(“target”))

end

</OnEvent>

</Scripts>

</Button>

[/code]

[quote]

注意：我们的插件破坏了默认目标下来菜单的“设置焦点”选项。这样做的原因是，处理此下拉菜单的默认UI函数依赖于安全代码，因此我们通过创建一个被污染包装器函数来调用它。暴雪没有将“设置焦点”作为安全按钮。在一个真实的插件中，你必须创建一个安全操作按钮模板的下来菜单，你将在本章后面看到安全模板如何改变焦点目标。

[/quote]

这两个框体现在功能齐全了(除了提到的焦点问题)。但当我们点击框体时，内部会发生什么呢？我们在这里没有实现OnClick处理程序，但是这些框体仍像普通的单位框体一样运行。让我们看看我们的框体拥有的一个预定义处理程序：OnClick处理程序，它调用函数SecureUnitButton\_OnClick。

[b]·OnClick处理程序背后的代码(The Code Behind the OnClick Handler)[/b]

与其他预定义的脚本处理程序和辅助函数一样，SecureUnitButton函数在文件FrameXML\SecureTemplates.lua中定义。

[code=lua]

function SecureUnitButton\_OnClick(self, button)

local type = SecureButton\_GetModifiedAttribute(self, “type”, button);

if ( type == “menu” ) then

if ( SpellIsTargeting() ) then

SpellStopTargeting();

return;

end

end

SecureActionButton\_OnClick(self, button);

end

[/code]

不要忘记，这个函数是默认UII的一部分，因此被认为是安全代码。所以它允许调用所有我们可能无法调用的受保护函数。

辅助函数SecureButton\_GetModifiedAttribute被用于根据你按下按钮和键盘上转换键(如Ctrl)来获得你想要执行的操作。这个操作是在属性mod-typeX中定义的，其中X是按钮(1=左键，2=右键，3=中键)，mod可以是键盘上的修饰键，如alt、shift或ctrl。可以使用\*替代mod-来忽略修饰键。这个属性可能的值是\*type1 = ”target”和\*type2 = “menu”，这意味着左键单击以某个单位为目标，右键单击目标打开菜单。开头的星号表示修饰键被忽略。这些默认值适用于我们的目的，所以让我们保留它们。

之后这个函数会检查我们是否想打开菜单，如果想的话，还会检查我们当前是否有一个正在等待目标选择的法术施放。如果是这种情况，法术施放将被取消，函数此时返回。否则，它将调用转发给SecureActionButtonTemplate的OnClick处理程序，因为一个单位框体共享一个动作按钮的基本功能，也就是说，当你点击它时会发生一些事情。这个OnClick处理程序相当长，而且无聊，因为它涵盖了所有可能的按钮操作，比如基于特定条件施放法术，等等。稍后我们将回到动作按钮。

你可能想知道，我们是否不能通过手动调用OnClick处理程序来绕过这样的限制，执行受保护的函数总是需要用户输入。你可以执行下面的函数调用来模拟鼠标单击：

[code=lua]

protectedFrame:GetScript(“OnClick”)(protectedFrame, “LeftButton”)

[/code]

通常情况下，插件无法分辨这是真的点击还是伪造的。但是请记住，整个处理程序是由暴雪在模板中定义的，它是安全代码。执行路径开始至终是安全的，如果它直接进入这个预定义的处理程序，则始终是安全的。但是你的调用假装点击是从受污染的代码开始的，因为你的代码总是受污染的，所以整个执行路径都是受污染的。这会导致操作失败。

在创建一个完整的单位框体插件的下一步是实现队伍和团队框体，对于这些，我们需要从SecureGropHeaderTemplate继承的模板。

[b][size=130%]&#9675; 使用团长模板 (Using Group Header Templater)[/size][/b]

团长模板控制一组单位框体，如你的队伍或团队子组(raid subgroup)。它们自动管理一切，比如删除离开队伍的玩家，或根据它们的名字或其他条件对显示的框体进行排序。我不会在这里向你展示使用这些模板的完整示例，因为它是非常难以测试的，因为你需要一个愿意帮助你测试的团队。你也不太可能需要这些模板，因为它们只对团队和团队框体模块有用。现在已经有很多不同的团队框体插件了，而且它们都是高度可定制的。无论如何，主要的操作都受到模块的限制，所以不太可能需要编写一个新的模板，因为所有可能的特性都已经由现有的模板实现了。

但我们还是来看看它们在理论上是如何工作的。你只需从该模板创建一个新的模板，并添加一个简单的标题(如文本)，以显示该标题所示的团队子组或职业。然而可以使用表11-1设置的这个安全模板属性，以定义如何显示它。

[align=center][size=80%]表11-1 页眉模板的属性[/size][/align]

[table]

[tr]

[td25][align=center]属性[/align][/td]

[td25][align=center]类型[/align][/td]

[td][align=center]描述[/align][/td]

[/tr]

[tr]

[td][align=center]showRaid[/align][/td]

[td][align=center]boolean[/align][/td]

[td][align=center]在团队中时显示标题[/align][/td]

[/tr]

[tr]

[td][align=center]showParty[/align][/td]

[td][align=center]boolean[/align][/td]

[td][align=center]在队伍中时显示标题[/align][/td]

[/tr]

[tr]

[td][align=center]showPlayer[/align][/td]

[td][align=center]boolean[/align][/td]

[td][align=center]如果你不在团队中，为自己显示单位框体[/align][/td]

[/tr]

[tr]

[td][align=center]showSolo[/align][/td]

[td][align=center]boolean[/align][/td]

[td][align=center]显示你是否不在组中(要求设置showPlyer)[/align][/td]

[/tr][tr]

[td][align=center]nameList[/align][/td]

[td][align=center]string[/align][/td]

[td][align=center]以逗号分隔的玩家列表，将显示在此标题下。(这将导致模板忽略属性groupFilter)[/align][/td]

[/tr]

[tr]

[td][align=center]groupFilter[/align][/td]

[td][align=center]string[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]strictFiltering[/align][/td]

[td][align=center]boolean[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]point[/align][/td]

[td][align=center]string[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]x0ffset[/align][/td]

[td][align=center]number[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]y0ffset[/align][/td]

[td][align=center]number[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]sortDir[/align][/td]

[td][align=center]string[/align][/td]

[td][align=center][/align][/td]

[/tr][tr]

[td][align=center]template[/align][/td]

[td][align=center]string[/align][/td]

[td][align=center][/align][/td]

[/tr][tr]

[td][align=center]templateType[/align][/td]

[td][align=center]string[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]groupBy[/align][/td]

[td][align=center]string[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]groupingOrder[/align][/td]

[td][align=center]string[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]maxColumns[/align][/td]

[td][align=center]number[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]unitsPerColumn[/align][/td]

[td][align=center]number[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]startingIndex[/align][/td]

[td][align=center]number[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]columnSpacing[/align][/td]

[td][align=center]number[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]columnAnchorPoint[/align][/td]

[td][align=center]string[/align][/td]

[td][align=center][/align][/td]

[/tr]

[/table]

模板完成了所有艰苦的工作，你只需为单位按钮定义XML模板并将其设置为标题的template属性即可。标题管理单位按钮，它从你的模板创建它们并进行所有定位。你也不必担心诸如玩家加入或离开小队之类的事件，标题模板负责所有这些工作。模板会为你执行此操作，因为你将无法执行此操作。在战斗中，你无法显示、隐藏或移动单位框体，但是在战斗中，玩家可以加入并离开团队。整个模板是安全代码，可以执行战斗中不允许执行的所有这些操作。

表11-1只是列出了团队或队伍框体的属性，宠物框体具有两个额外属性：useOwnerUnit和filterOnPet。两者都是布尔属性。useOwnerUnit会为宠物的所有者显示一个单位框体，而不是为宠物本身显示一个单位框体，并且filterOnPet可以设置为使用宠物的名称进行过滤和排序，而不是使用所有者的名称。

安全框体的下一个重要类型是操作按钮，该按钮基本上是一个单机按钮即可执行受保护功能的按钮。

======

[color=blue][b][size=160%]&#9679; 使用安全动作按钮模板[/size]

[size=130%](Using Secure Action Button Templates)[/size][/b][/color]

动作按钮广泛使用属性来定义单击按钮时要执行的操作。属性类型控制单击按钮时执行哪些操作。表11-2显示了所有可能的类型值。所有这些操作都引用受保护的函数，这些函数通常不会从受污染的代码中调用，但是模板是安全代码。

[align=center][size=80%]表11-2 安全动作按钮类型[/size][/align]

[table]

[tr]

[td50][align=center]类型[/align][/td]

[td][align=center]描述[/align][/td]

[/tr]

[tr]

[td][align=center]actionbar[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]action[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]assist[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]attribute[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]cancelaura[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]click[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]focus[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]item[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]macro[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]mainassist[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]maintank[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]pet[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]spell[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]stop[/align][/td]

[td][align=center][/align][/td]

[/tr]

[tr]

[td][align=center]tatget[/align][/td]

[td][align=center][/align][/td]

[/tr]

[/table]

对于我们的按钮，有很多可能的选项，但是它都非常简单。当按钮被点击时，它们中的大多数只是调用一个受保护的功能。让我们看一个安全动作按钮的示例。

[b][size=130%]&#9675; 一个简单的安全动作按钮 (A Simple Secure Action Button)[/size][/b]

一个简单的安全动作按钮的例子是DBM中的战歌峡谷战场模块。它在分数显示中添加一个小框体，显示当前旗帜运送者的名称。这个框体是一个安全的动作按钮，当你点击它时，它会以当前旗帜运送者为目标。让我们看看如何在这个BDM模块中实现这一点。你可以在DBM-Battlegrounds\Warsong.lua文件中找到这个模块。

显然，模块必须做的一件事就是从安全模板创建一个框体。这是在方法CreateFlagCarrierButton中完成的，该方法在加入战歌峡谷时被调用。

[code=lua]

function Warsong:CreateFlagCarrierButton()

if not Warrson.Options.ShowFlagCarrier then return end

if not self.FlagCarrierFrame1Button then

self.FlagCarrierFrame1Button = CreateFrame(“Button”, nil, nil “SecureActionButtonTemplate”)

self.FlagCarrierFrame1Button:SetHeight(15)

self.FlagCarrierFrame1Button:SetWidth(150)

self.FlagCarrierFrame1Button:SetAttribute(“type”, “macro”)

self.FlagCarrierFrame1Button:SetPoint(“LEFT”, “AlwaysUpFrame1”, “RIGHT”, 28, 4)

end

if not self.FlagCarrierFrame2Button then

self.FlagCarrierFrame2Button = CreateFrame(“Button”, nil, nil, “SecureActionButtonTemplate”)

self.FlagCarrierFrame2Button:SetHeight(15)

self.FlagCarrierFrame2Button:SetWidth(150)

self.FlagCarrierFrame2Button:SetAttribute(“type”, “macro”)

self.FlagCarrierFrame2Button:SetPoint(“LEFT”, “AlwaysUpFrame2”, “RIGHT”, 28, 4)

end

self.FlagCarrierFrame1Button:Show()

self.FlagCarrierFrame2Button:Show()

end

[/code]

代码创建了两个button类型的框体，这些两题继承自SecureActionButtonTemplate。之后将其类型属性设置为macro。你可能期望这里有target类型，但target类型只能针对特地的单位ID，战场上没有固定的敌人ID。相比之下，marco可以执行任意的宏文本，例如/targetexact somePlayer，这意味着macro可以提供更大的灵活性。

之后，这些框体将锚定到AlwaysUpFrame1和AlwaysUpFrame2，他们是默认UI顶部的分数显示。实际的宏文本是通过方法CheckFlagCarrier设置的。当旗帜运送者在未参加战斗的情况下改变时，以及每次在战歌峡谷离开战斗时，都会调用次方法。局部变量FlagCarrier包含一个带有Alliance(1)和Horde(2)的旗帜运送者的表。

[code=lua]

function Warsong:CheckFlagCarrier()

if not UnitAffectingCombat(“player”) then

if FlagCarrier[1] and self.FlagCarrierFrame1 then

self.FlagCarrierFrame1Button:SetAttribute(“macrotext”, “/targetexact” ..FlagCarrier[1])

end

if FlagCarrier[2] and self.FlagCarrierFrame2 then

self.FlagarrierFrame2Button:SetAttribute(“macrotext”, “/targetexact”..FlagCarrier[2])

end

end

end

[/code]

此方法确保旗帜运送者按钮在可能的情况下以正确的玩家为表。在此功能之外更改框体的文本，因为在战斗中此策略可以正常工作，因此，如果你在战斗中被其他人捡起旗帜，则被当成目标的单位可能与显示的文本有所不同。但是由于受保护框体的限制，我们无法在战斗中更新目标。

我没有打印确定当前旗帜运送者并设置字符串的函数。该函数相当长，大约有70行，因为函数要解析系统聊天消息以确定有string.match的运送者，因为没有良好的可用事件。设置框体文本的代码又长又复杂，因为它需要确定捡起旗帜的敌方玩家的等级。你不能只对超出范围的敌人调用UnitClass，所以该函数使用一种变通方法并访问战场计分板，其中包含所有敌人的类别。如果你想知道它的工作原理，可以阅读存储在文件末尾的局部变量updateflagcarrier中的函数。

但是，动作按钮不仅仅可以在你单击某个对象时将其作为目标。

[b][size=130%]&#9675; 高级安全动作按钮 (An Advanced Secure Action Button)[/size][/b]

让我们创建一个简单的动作按钮实例，来查看该函数的功能。使用适当的TOC文件创建一个新的插件，然后向其中添加一个新的XML文件。如果你不想为新的插件重启游戏，也可以重复使用我们在本章前面创建的单位框体模块实例。

[b]· 创建一个带有XML的安全按钮(Creating a Secure Button with XML)[/b]

我们将在这里创建一个简单的动作按钮来进行测试，这个按钮将只显示在屏幕的中央，并使用OptionsButtonTemplate的默认样式。然后我们可以使用TinyPad这样的游戏编辑器来执行简单的Lua脚本来设置属性来测试我们的按钮。

使用默认动作按钮的样式创建安全按钮非常容易，这得益于多重继承。我们可以将SecureActionButtonTte和OptionButtonTemplate添加到inherits属性。更一般的情况是，如果需要安全模板的功能，你总是可以将安全模板添加到你的继承属性中。XML文件看起来是这样的：

[code=xml]

<Ui xmlns="http://www.blizzard.com/wow/ui/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.blizzard.com/wow/ui/..\FrameXML\UI.xsd">

<Button name=”SecureTestButton” text=”Secure Button” parent=”UIParent” inherits=”SecureActionButtonTemplate,OptionsButtonTemplate”>

<Anchors>

<Anchor point=”CENTER”/>

</Anchors>

<Size>

<AbsDimension x=”128” y=”24”/>

</Size>

<Scripts>

<OnLoad>

self:RegisterForClicks(“AnyUp”)

</OnLoad>

</Scripts>

</Button>

</Ui>

[/code]

这段代码在你的屏幕中央显示一个小按钮，上面写着“Secure Button”。你可以随意把按钮放在任何你想放的地方。OnLoad处理程序中对RegisterForClicks的调用很重要，因为默认情况下按钮只侦听鼠标的左键点击，并且我们希望在本示例中的后面使用右键。

现在我们将使用高级属性对这个按钮做一些事情。我们可以在全局变量SecureTestButton下访问它，所以让我们向它添加一些更高级的属性。

[b]· 摆弄属性(Playing Around with Attributes)[/b]

在单位框体的例子中，你已经看到属性可以包含允许你根据所按下的鼠标按钮执行不同操作的修饰符。属性的修饰符由前缀和后缀组成，其中前缀保存键盘上的修饰键，而后缀定义鼠标按钮。前缀和后缀都是可选的，默认值匹配所有键位。

一个好的测试按钮类型总是macro，因为你可以指定一些像/say hello, world这样的东西作为宏文本属性。执行以下代码将按钮的类型设置为宏。

[code=lua]

SecureTestButton:SetAttribute(“type”, ”macro”)

[/code]

现在让我们根据所使用的鼠标按钮添加宏文本。之前我们在单位框体模板的默认值中看到前置\*，这实际上并不总是必要的。省略前置和后缀与使用星号具有相同的效果。然而，仅仅省略前置或后缀意味着不能按任何修饰键或鼠标按钮。显然，不适用鼠标按钮就不可能点击屏幕上的按钮(使用另一个类型为“click”的按钮来假装点击发送鼠标按钮，屏幕按钮被点击)，所以你不想在使用前缀时省略后缀。

下面的代码设置用鼠标左键单击按钮时使用的macrotext属性(回想一下，我们可以在属性名后面加1，以将其绑定到鼠标左键)，我们不需要在前面加上星号。

[code=lua]

SecureTestButton:SetAttribute(“macrotext1”, ”/say hello”)

[/code]

当你左键单击时，你的角色会说“hello”。一个真实的例子会说“hello, world”，说以让我们添加world作为右击文本。

[code=lua]

SecureTestButton:SetAttribute(“macrotext1”, “/say hello”)

[/code]

我们现在也可以添加键盘修饰键。我们需要在这里使用星号作为后缀，否则操作将不匹配鼠标按键。

[code=lua]

SecureTestButton:SetAttribute(“shift-macrotext\*”, ”/say hello world”)

[/code]

我们的角色现在说“hello world”，当我们移动-点击任何鼠标按钮。请注意，旧的功能我们没有破坏，我们仍然可以使用左或右按钮分别获得“hello”或“world”。

也可以通过将其值设置为变量ATTRIBUTE\_NOOP的值来删除特定的修饰符组合(noop是no operation的缩写，变量实际上之保存空字符串)。使用Shift键和鼠标中键可以得到“hello world”，让我们删除。

[code=lua]

SecureTestButton:SetAttribute(“shift-macrotext3”, ATTRIBUTE\_NOOP)

[/code]

可以对包括type属性在内的所有属性使用这些修饰符。一个按钮可以根据使用的鼠标按钮和修饰键执行完全不同的操作。当你用Alt键按下右键单击按钮时，让我们尝试将类型更改为target。

[code=lua]

SecureTestButton:SetAttribute(“alt-type2”, ”target”)

[/code]

测试的一个很好目标单位是玩家，所以让我们将其设置为单位属性。

[code=lua]

SecureTestButton:SetAttribute(“unit”, “player”)

[/code]

你现在可以通过alt-右击按钮来定位自己。请注意，安全模板总是在获取所需的额外模板之前，对type属性及其所有修饰键进行匹配。这意味着添加属性alt-macrotext2，是没有意义的，因为它永远不会被使用，因为类型将永远是target，当你用Alt键右击按钮。

整个实例非常人为，因为你肯定不需要一个基于所使用的鼠标按钮发送文本的按钮。发送聊天消息也可以在完全不适用安全框体的情况下实现。在下一章讨论宏时，我们将看到与安全动作相关的更有用、更强大的命令。宏可以用来实现非常类似于安全按钮的功能，它们甚至可以执行现有安全按钮的OnClick处理程序。

======

[color=blue][b][size=160%]&#9679; 总结[/size]

[size=130%](Summary)[/size][/b][/color]

本章首先介绍了游戏如何通过代码污染系统防止插件使游戏自动化。此污染系统可确保默认UI的未修改代码比用户提供的代码具有更多的权限。我们了解了受保护的函数，我们的代码无法调用这些函数。但是我们也看到了插件在特定情况下如何通过使用安全模板来规避这些限制。

本章的最后一部分是关于通过在属性中使用修饰键来实现高级安全按钮操作。你看到了如何创建一个按钮，该按钮根据使用的鼠标按钮和键盘修饰键来执行完全不同的操作。注意，安全模板背后的所有内容，包括属性中的修饰键，都是在Lua中实现的。如果你想知道这些模板内部是如何工作的，可以阅读默认UI的文件FrameXML\SecureTemplates.lua。这个文件是默认UI中比较复杂的文件之一，但它也是注释最好的文件之一，所以应该是可以理解的。