目录

什么是 Burp Suite

Burp Suite 是用于攻击 web 应用程序的集成平台。它包含了许多工具，并为这些工具设计了许多接口，以促进加快攻击应用程序的过程。所有的工具都共享一个能处理并显示 HTTP 消息，持久性，认证，代理，日志，警报的一个强大的可扩展的框架。

Burp Suite 能高效率地与单个工具一起工作，例如：

一个中心站点地图是用于汇总收集到的目标应用程序信息，并通过确定的范围来指导单个程序工作。

在一个工具处理 HTTP 请求和响应时，它可以选择调用其他任意的 Burp 工具。例如，代理记录的请求可被 Intruder 用来构造一个自定义的自动攻击的准则，也可被 Repeater 用来手动攻击，也可被 Scanner 用来分析漏洞，或者被 Spider(网络爬虫)用来自动搜索内容。

应用程序可以是“被动地”运行，而不是产生大量的自动请求。Burp Proxy 把所有通过的请求和响应解析为连接和形式，同时站点地图也相应地更新。由于完全的控制了每一个请求，你就可以以一种非入侵的方式来探测敏感的应用程序。

当你浏览网页(这取决于定义的目标范围)时，通过自动扫描经过代理的请求就能发现安全漏洞。

IburpExtender 是用来扩展 Burp Suite 和单个工具的功能。一个工具处理的数据结果，可以被其他工具随意的使用，并产生相应的结果。

Burp Suite 工具箱

Proxy——是一个拦截 HTTP /S 的代理服务器，作为一个在浏览器和目标应用程序之间的中间人，允许你拦截，查看，修改在两个方向上的原始数据流。

Spider——是一个应用智能感应的网络爬虫，它能完整的枚举应用程序的内容和功能。

Scanner[仅限专业版]——是一个高级的工具，执行后，它能自动地发现 web 应用程序的安全漏洞。

Intruder——是一个定制的高度可配置的工具，对 web 应用程序进行自动化攻击，如：枚举标识符，收集有用的数据，以及使用 fuzzing 技术探测常规漏洞。

Repeater——是一个靠手动操作来补发单独的 HTTP 请求，并分析应用程序响应的工具。

Sequencer——是一个用来分析那些不可预知的应用程序会话令牌和重要数据项的随机性的工具。

Decoder——是一个进行手动执行或对应用程序数据者智能解码编码的工具。

Comparer——是一个实用的工具，通常是通过一些相关的请求和响应得到两项数据的一个可视化的“差异”。

Burp Suite 的使用

当 Burp Suite 运行后，Burp Proxy 开起默认的 8080 端口作为本地代理接口。通过置一个 web 浏览器使用其代理服务器，所有的网站流量可以被拦截，查看和修改。默认情况下，对非媒体资源的请求将被拦截并显示(可以通过 Burp Proxy 选项里的 options 选项修改默认值)。对所有通过 Burp Proxy 网站流量使用预设的方案进行分析，然后纳入到目标站点地图中，来勾勒出一张包含访问的应用程序的内容和功能的画面。在 Burp Suite 专业版中，默认情况下，Burp Scanner 是被动地分析所有的请求来确定一系列的安全漏洞。

在你开始认真的工作之前，你最好为指定工作范围。最简单的方法就是浏览访问目标应

用程序，然后找到相关主机或目录的站点地图，并使用上下菜单添加 URL 路径范围。通过配置的这个中心范围，能以任意方式控制单个 Burp 工具的运行。

当你浏览目标应用程序时，你可以手动编辑代理截获的请求和响应，或者把拦截完全关闭。在拦截关闭后，每一个请求，响应和内容的历史记录仍能再站点地图中积累下来。

和修改代理内截获的消息一样，你可以把这些消息发送到其他 Burp 工具执行一些操作：

你可以把请求发送到 Repeater,手动微调这些对应用程序的攻击，并重新发送多次的单独请求。

[专业版]你可以把请求发送到 Scanner,执行主动或被动的漏洞扫描。

你可以把请求发送到 Intruer，加载一个自定义的自动攻击方案，进行确定一些常规漏洞。

如果你看到一个响应，包含不可预知内容的会话令牌或其他标识符，你可以把它发送到 Sequencer 来测试它的随机性。

当请求或响应中包含不透明数据时，可以把它发送到 Decoder 进行智能解码和识别一些隐藏的信息。

[专业版]你可使用一些 engagement 工具使你的工作更快更有效。

你在代理历史记录的项目，单个主机，站点地图里目录和文件，或者请求响应上显示可以使用工具的任意地方上执行任意以上的操作。

可以通过一个中央日志记录的功能，来记录所单个工具或整个套件发出的请求和响应。这些工具可以运行在一个单一的选项卡窗口或者一个被分离的单个窗口。所有的工具和套件的配置信息是可选为通过程序持久性的加载。在 Burp Suite 专业版中，你可以保存整个组件工具的设置状态，在下次加载过来恢复你的工具。

Burp 菜单

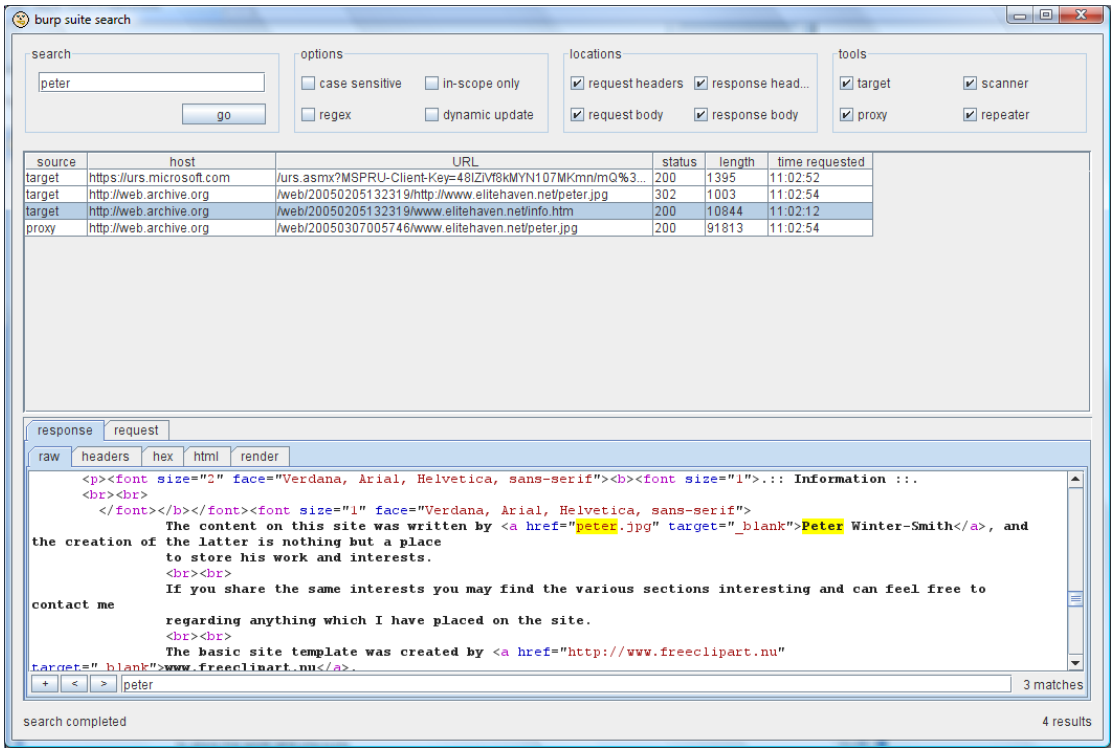
这个菜单包含了以下描述的一系列关键功能和配置选项。

搜索

[专业版]在 Burp 菜单上选择“search”后，会打开一个用起来非常简单的搜索对话框。你可以设定以下搜索参数：

搜索表达式
搜索是否区分大小写
搜索的是简单的文本还是正则表达式
搜索范围是否有限制范围
搜索的结果是否随着 HTTP 消息的处理而动态更新
是在那种 HTTP 消息的位置上搜索(请求 vs 响应，消息头 vs 消息主体)
使用那种工具进行搜索

当你单击"go"时，搜索开始，在一个有序的表格里，将显示出每个与搜索相关的关键内容细节，在一个预览窗格里，你能看到所有的请求和响应，也包括匹配你强调的搜索项。通常上下文菜单可用于指定发起攻击的具体项目，也可将他们发送到其他工具里做进一步的分析：



请注意，如果你通过上下文菜单进行对目标站点地图的搜索，那么这次搜索将被具体化到一些选定的分支上。

保存和恢复状态

[专业版]下面的帮助说明了保存和恢复状态的过程，和一些通常使用的情况。

保存状态

能保存的项目有：

包含通过 Proxy 和 Spider 的所有内容的目标站点地图

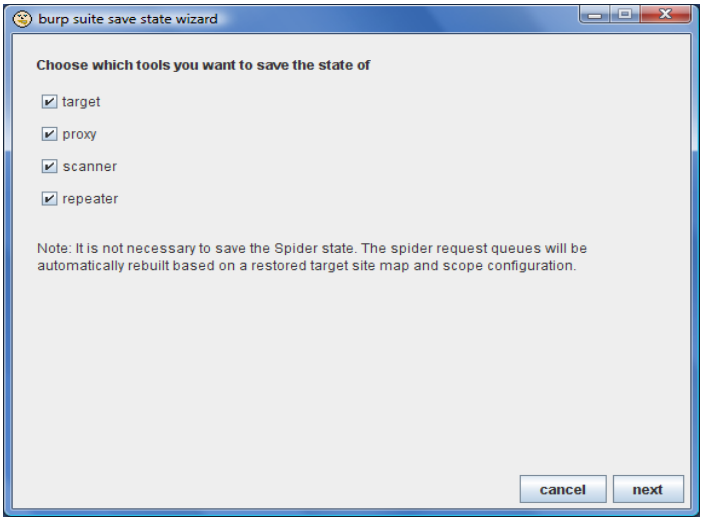
Proxy 历史记录

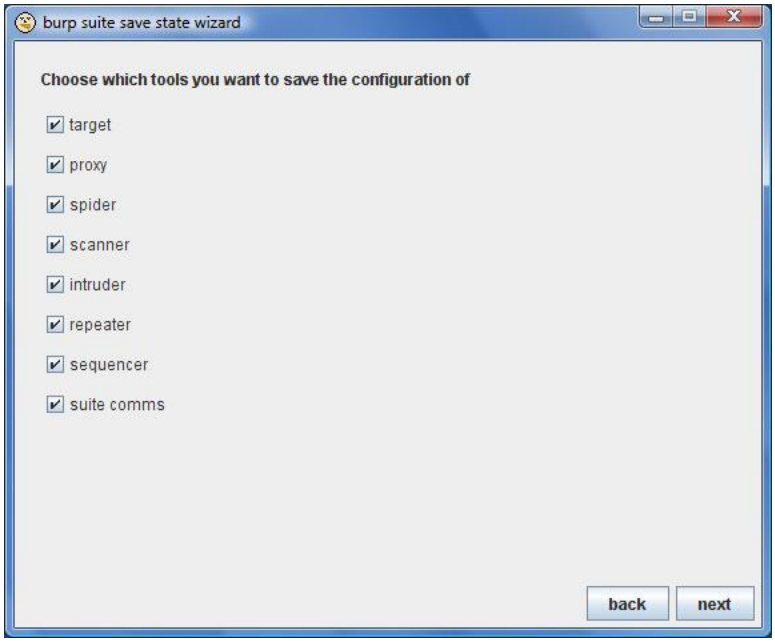
Scanner 发现的问题

Repeater 选项卡的内容和历史记录

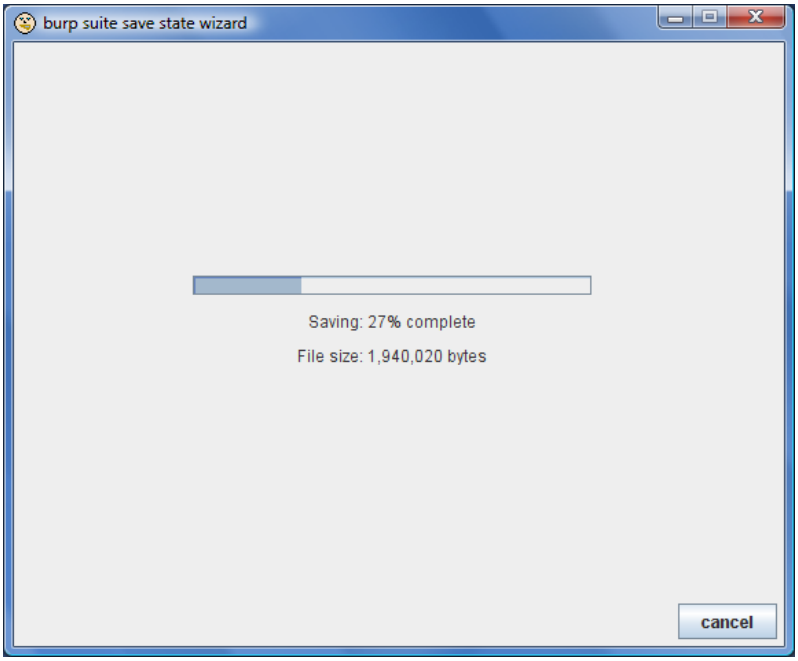
所有套件工具的配置信息

在 Burp 菜单选择“ save state”，会启动一个向导，你可以定义要保存的状态和配置信息：





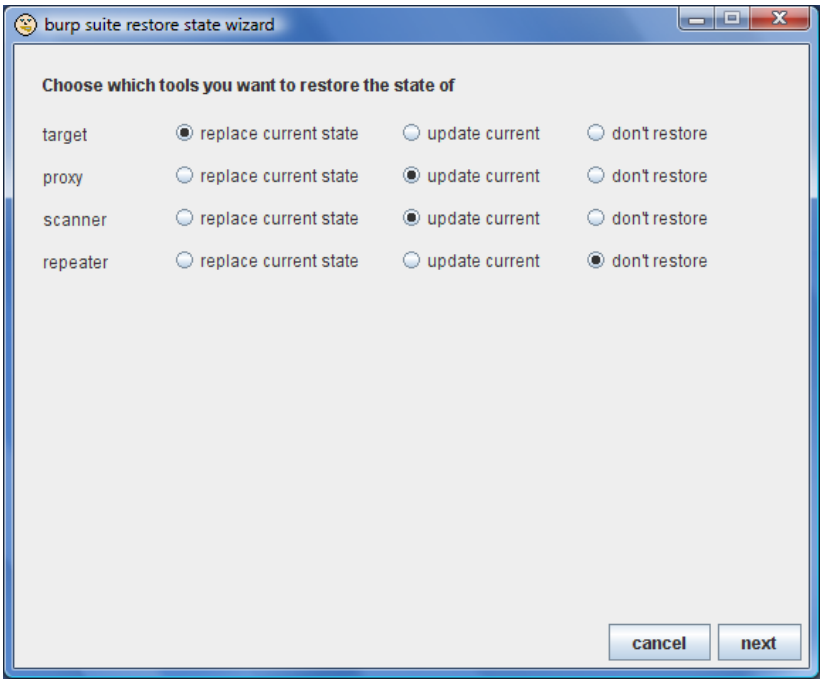
这时你来选择要输出的文件，剩下的由 Burp 来完成。当 Burp 正在保存状态时，你可以继续使用它，如果你尝试对 Burp 正在保存的数据执行操作，可能会遇到一些短暂的延时，这样做的目的是为了防止数据损坏。

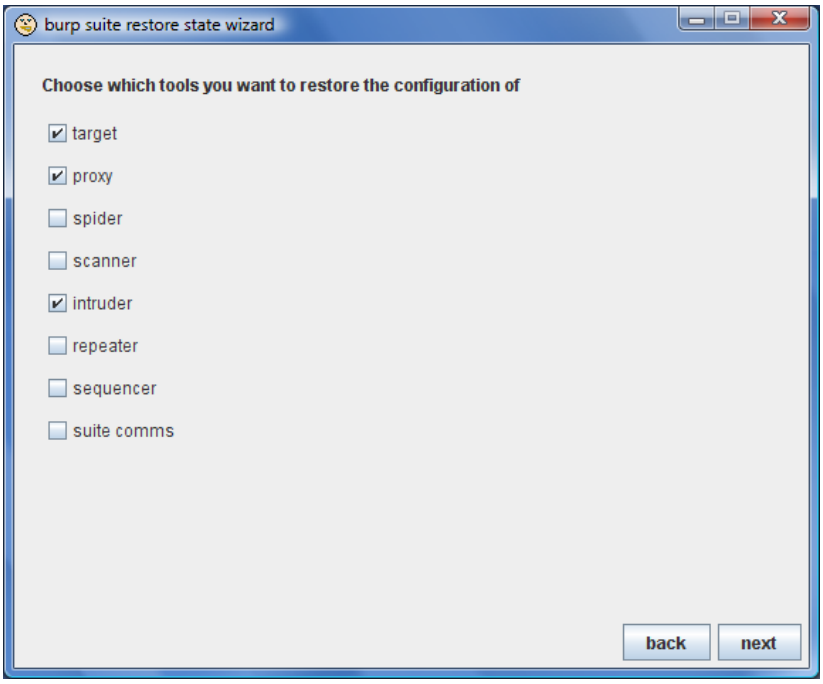


显而易见，由于保存的文件包含在你正在保存的工具内不断积累的请求和响应，所以这个文件会越来越大。实际上，保存或恢复几个小时的测试通常会再 1 到 2 分钟完成。在执行保存之前，你可以通过删除不必要的站点地图和代理历史记录的方式，来加快和精简这个过程。

恢复状态

在 Burp 菜单选择“restore state”，会启动一个向导，在这里你可以自定义要恢复的状态和配置的选项。首先，你要选择先前保存的文件。Burp 会分析这个文件识别其中的内容(注意每个保存的文件都能包含任意组合工具的状态和配置信息)。对于每种保存的状态和配置信息，Burp 会让你选择是否想要恢复，以及是否想要增加或者替换工具耳朵现有状态。





Burp 会恢复你的选择，然后继续工作。当 Burp 正在恢复状态时，你可以继续使用它，如果你尝试对 Burp 正在恢复的数据执行操作，可能会遇到一些短暂的延时，这样做的目的是为了防止数据损坏。

使用场合

对渗透测试人员来说，保存恢复工具状态和配置信息有巨大的好处：

你可以在每天结束的时候保存你的工作，然后在第二天早上无缝地恢复恢复你的工作。

你可以备份整个工作的关键测试信息，以防系统崩溃。

定制方案完成后，你可以存储一个完整的归档文件的所有积累信息，使你能在稍后重新打开继续你的工作，回答客户的问题或者重新测试同一个问题。

映射出应用程序内容的任务可以由顾问们来一起分担，这些的小的网站地图可以逐渐合并到

一个中去，这些都可以被所有的顾问共享。
团队的领导可以优化一个特定的定制 Burp 配置，包括细小的目标范围的界定，以及把这种配置直接传递给其他团队成员开始测试。
你可以为不同的任务设计相应的配置模板，以备将来使用，这样可以在他们之间简单地切换。

记录设置

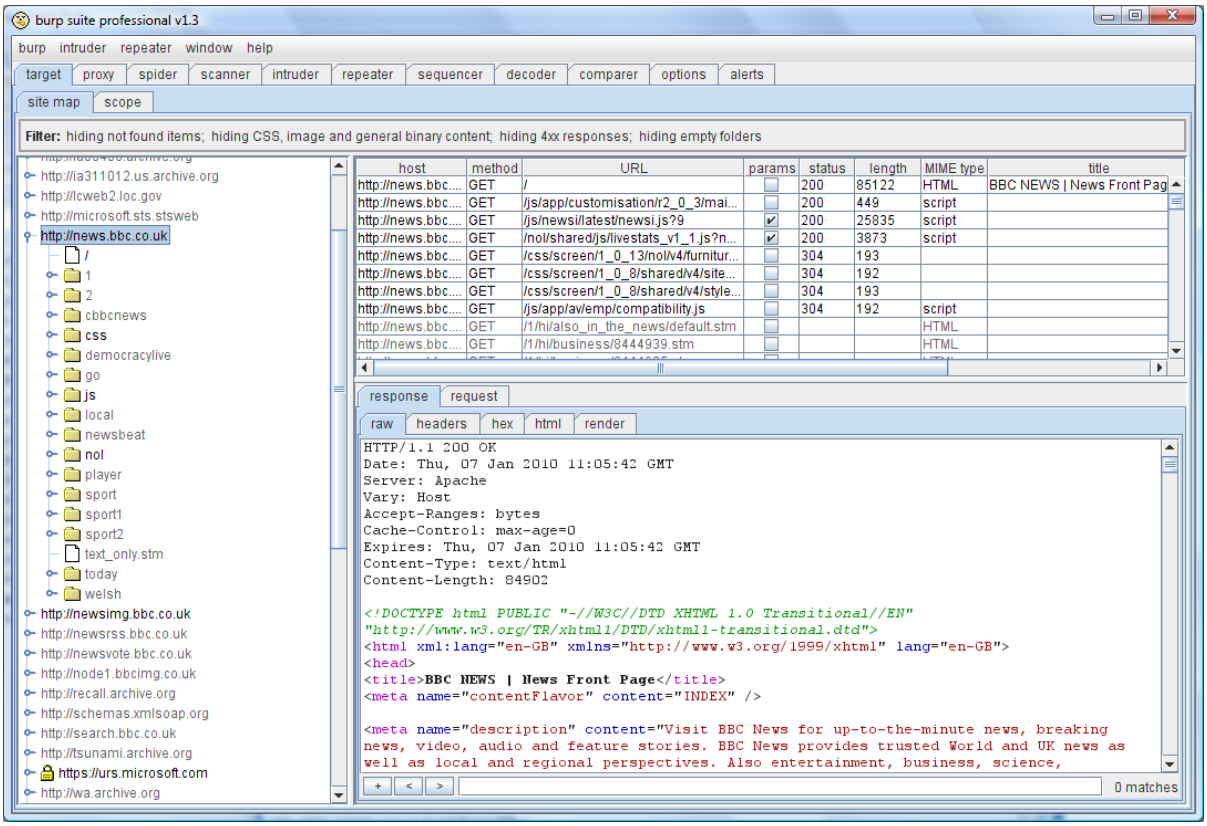
在加载不同的软件时，“remember setting”选项决定了是否让 Burp 记住配置信息的设置。你可以让 Burp 对所有软件都记住设置，也可以对选择的单个软件。
“restore defaults”选项把 Burp Suite 或者单个工具的配置设置重置到它们的默认值。

精简模式

如果这个选项被选中，接下来 Burp 将以精简模式运行，这时可用的工具只有：Burp Proxy,Intruder,和 Repeater。在这个模式下运行，对系统资源消耗很低，这是为了一些喜欢简单轻便工具的用户而设计的。

目标站点地图

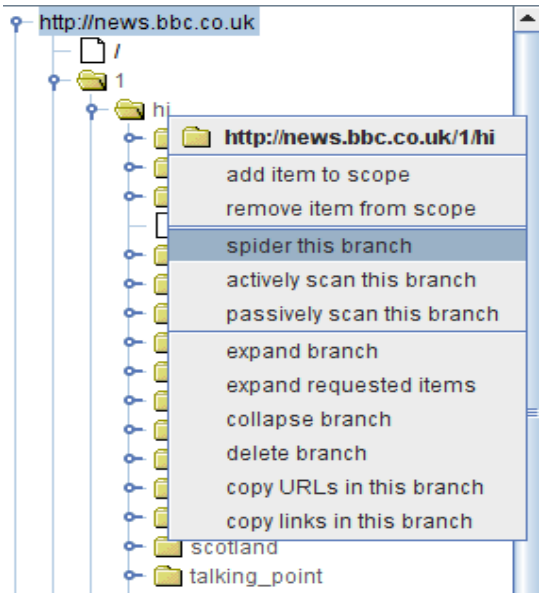
中央站点地图汇聚了所有 Burp 收集的关于你正在攻击的应用程序的信息。这包括了直接通过 Proxy 的请求，以及根据这些请求得到的响应的分析结果，和 Spider 发现的所有内容。当你浏览一个应用程序时，会为你映射出大量的内容，甚至远远超过你的要求。例如：



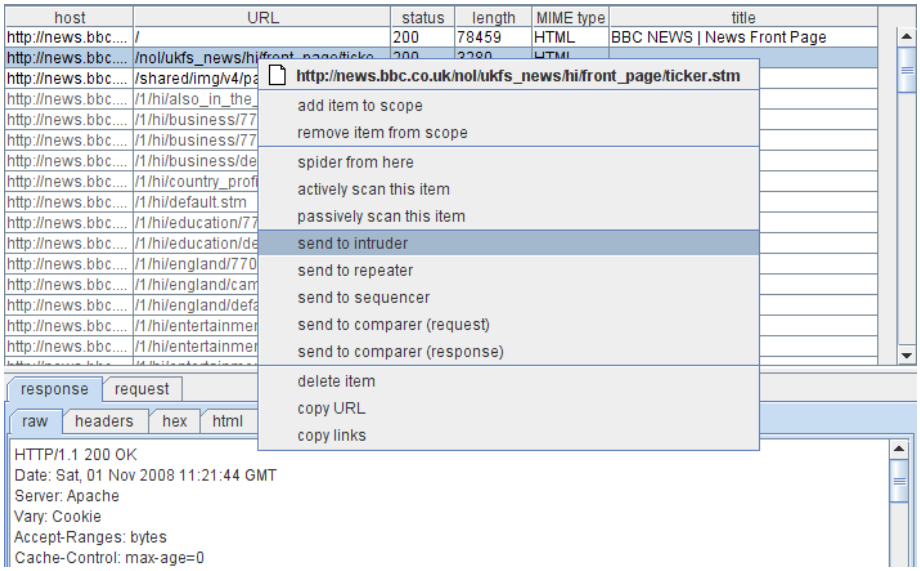
已请求的项目会以黑色显示；那些也被 Burp 推断出，但还未被请求过的以灰色显示。在默认情况下，那些让渗透测试人员不感兴趣的项目是被过滤掉而不被显示的，但种行为是可以被修改的(这将在下面介绍)。

站点地图的界面基本上像一个图形化的电子邮件客户端。一个主机和目录树显示在左侧。在数视图中选择一个或多个节点会导致下面的这些节点的所有项目以表格的形式显示在右上方。表格里面包含了项目的关键细节(URL，编码方式，页面标题等等)和能按任意列排序的项目(单击任意列的标题按降序排列，按住 shift 单击是升序排列)。选中表格中的一个项目，相关的请求和响应就会显示在右下角的预览窗格里。这个预览窗格里包含所有和 Burp 里相似的功能—消息头和参数的分析，文本搜索，媒体渲染等等。

当显示出你的目标的信息全部收集完毕时，站点地图通过随时出现的上下文菜单使你能控制发起特定的攻击。例如，你可以选择一个主机或者树视图的文件夹，以及执行树视图的整个分支上的指令，如爬虫搜索和扫描：

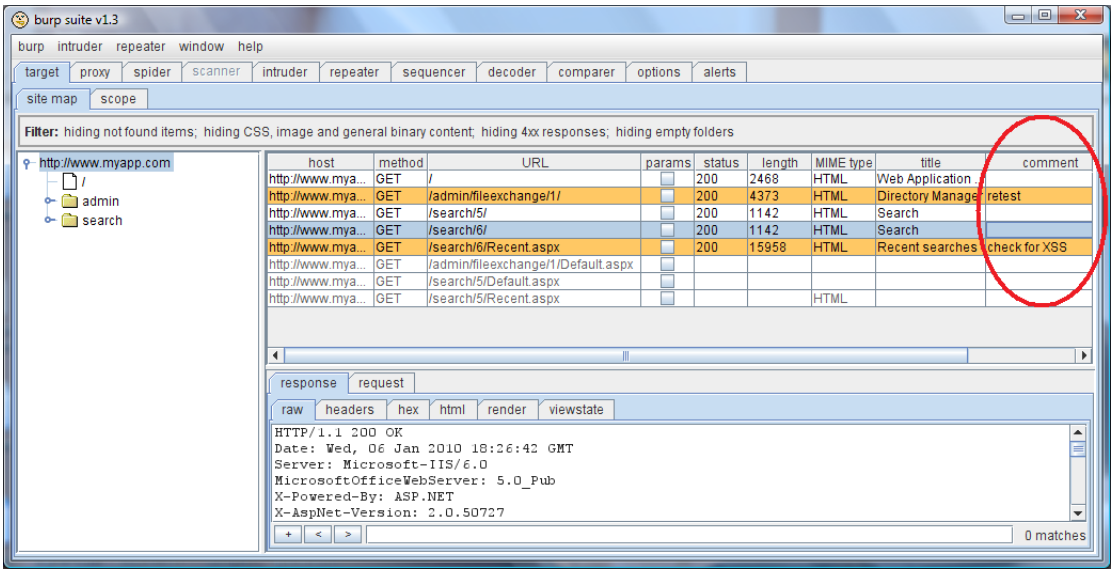


同样地，你可选择视图树或表格中的单个文件，并把相关的请求发送到其它工具，如 Intruder 或者 Repeater。如果浏览器没有请求过这一项，Burp 就会根据 URL 和从目标域接受到的 cookies 为这一项构造出一个默认请求：

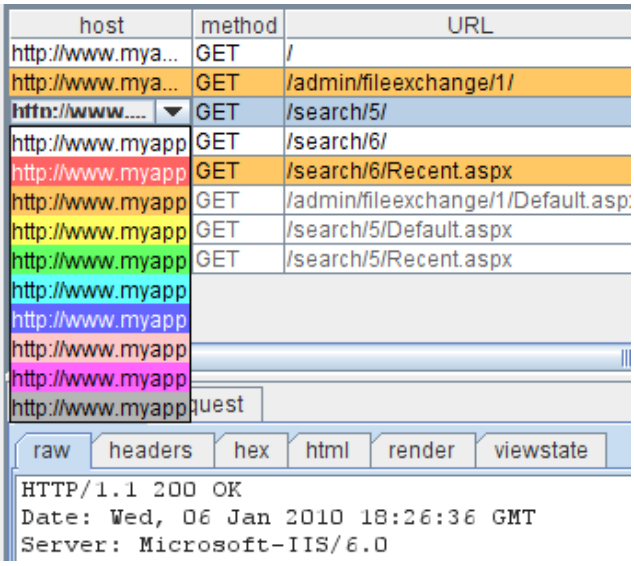


[专业版]你可以通过上下文菜单访问各种定制工具，如搜索注释和脚本，分析你的目标 web 站点，调度任务等等。

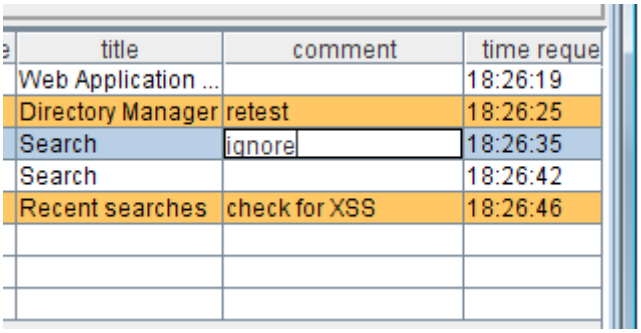
在视图表格里，你可以对单个或多个的项目进行注释，对他们进行添加注释和强调：



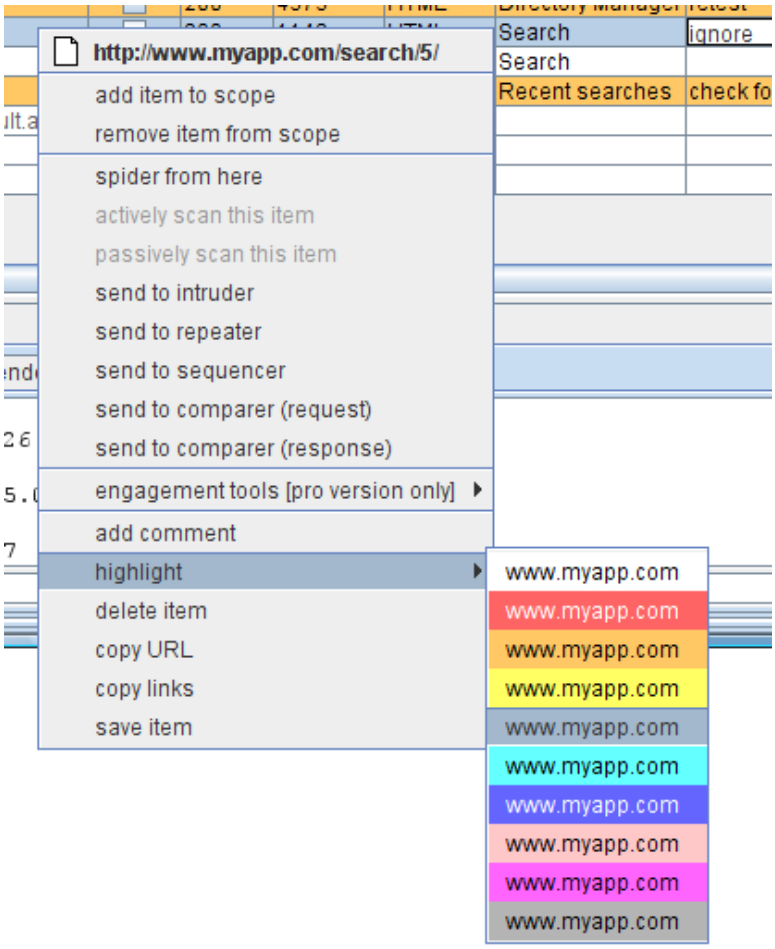
你可以使用表格最左列的下拉菜单设置单个项目醒目：



你可以通过双击来编辑单元格的方式就地对单个项目进行注释：

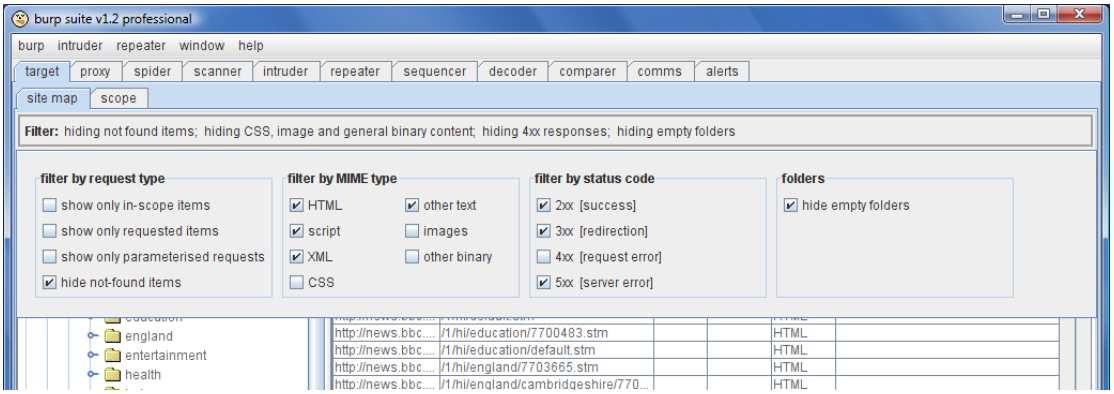


另外，如果你想一次对多个项目进行注释，你选择相关的项目，并使用上文菜单来添加注释和申请加亮：



当你已经注释出了自己感兴趣的项目时，可以通过列排序并和显示过滤来快速找到这些项目。

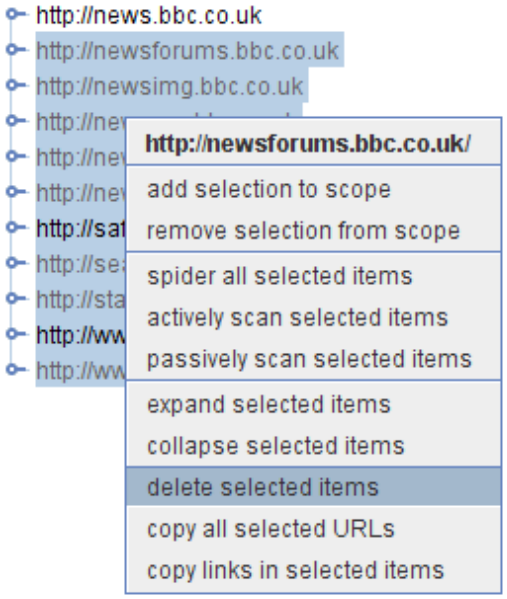
站点地图显示的内容是有效地进入基础数据库的一个视图，并且你可以配置过滤器以决定在地图上显示那些项的基础数据。一些应用程序包含大量的内容，如：图片，CSS 等，这些通常在视图上是隐藏的。在站点地图的上方有一个过滤器栏。单击这里，会有一个弹出栏，让你能精确地设置在地图上显示那些内容：



你可以选择只显示参数有请求，或者在当前的目标范围内。你可以通过 MIME 类型，HTTP 的状态码和文件扩展名进行过滤。如果你设置一个过滤器来隐藏一些项，这些项不会被删除，仅仅是隐藏了，如果你解除相关的过滤器，它们会重新出现。这就意味着你可以使用过滤器来帮助你系统地研究一个复杂的网站地图，了解各种不同兴趣的内容。

[专业版]你可以对指定的搜索词进行过滤，在请求和响应以及可用的用户注释中包含的这些

表达式的项都将被显示出来。
除了视图里的过滤内容，你有时可能会想彻底删除它。例如，如果你浏览到目标域外，你将会在 Burp 里积累了一些你不需要的数据。在这种情况下，你可以使用站点地图的上下文菜单永久地删除多余的项目。例如，你可以选择多个主机或者视图树里的文件夹或者表格视图，并彻底删除他们：

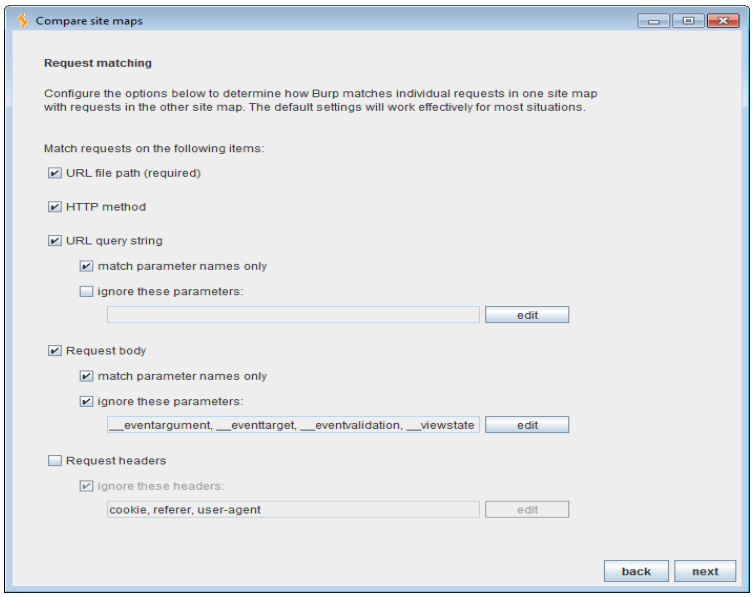


比较站点地图

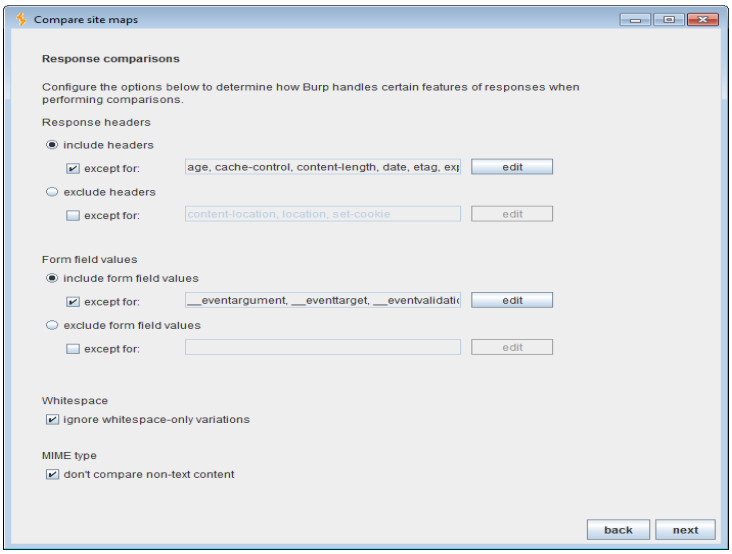
你可以使用 Burp 来对比两个站点地图，并突出其差异。这个功能可以通过各种方式来帮助寻找不同种类的访问控制漏洞，并确定一个大型的应用程序的哪一块需要进行人工密切检查。此功能的一些典型使用情况如下：
你可以使用不同权限级别的账号来映射应用程序，并比较得到的结果来确定一个功能是对其中的一个用户而不是其他的用户是可见的。
你可以使用一个高权限的账号来映射应用程序，然后使用一个低权限的账号来重新请求整个网站地图，以确定是否有访问权限的限制。
你可以使用两个相同权限的不同账号来映射应用程序，以确定哪些是需要特定的用户标识符来进行访问的敏感资源的事件，并确定是否正确地划分了每个用户的数据。
你可以通过在主站点地图上的上下文菜单来使用“compare site maps”功能。打开一个向导，需要你配置要比较的站点地图的细节，以及比较的方法。当你选择了需要比较的站点地图，可以选择下面的这些选项：
显示在 Burp 的目标选项中的当前站点地图。
从一个以前保存 Burp 文件加载过来的站点地图。
以不同的 session context，对上面的两者进行重新请求。

你可以选择包含所有站点地图的内容，也可以限制选定范围内的项目。如果你选择以一个不同的 session context，特别重要的是不要包括会破坏上下文的请求—例如登陆，注销，用户模拟功能等等。在执行比较时，Burp 通过对第一个站点地图的每一个请求与第二个站点地图的请求进行对比，反之亦然。通过对匹配请求的响应进行对比，来查找出所有的差异。对站点地图内不匹配的项分别进行标记为删除或添加。这样高度可配置的确切过程，允许你定

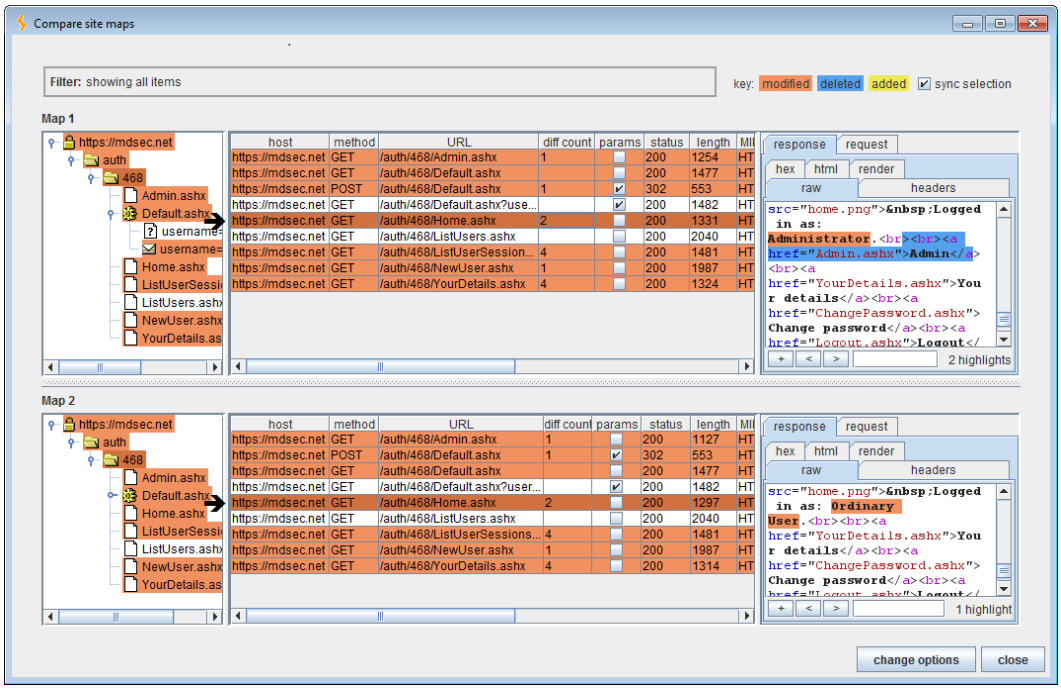
制出针对目标应用程序的比较功能。下面列出了 Burp 怎样比较两个站点地图的配置选项：



在大多数情况下，默认的选项能很好的工作，并对比在查询的字符串和消息里面的 URL 文件路径，HTTP 方法和参数的名称的请求。对于一些应用程序，你需要修改这些默认选项，以确认能准确地对请求进行匹配。例如，一个应用程序在同一个 URL 上进行不同的操作，通过查询的字符串值来指定操作，你需要按他们的名字和参数的值匹配他们的请求。
下面显示的是怎样是 Burp 请求与响应相匹配的设置选项：

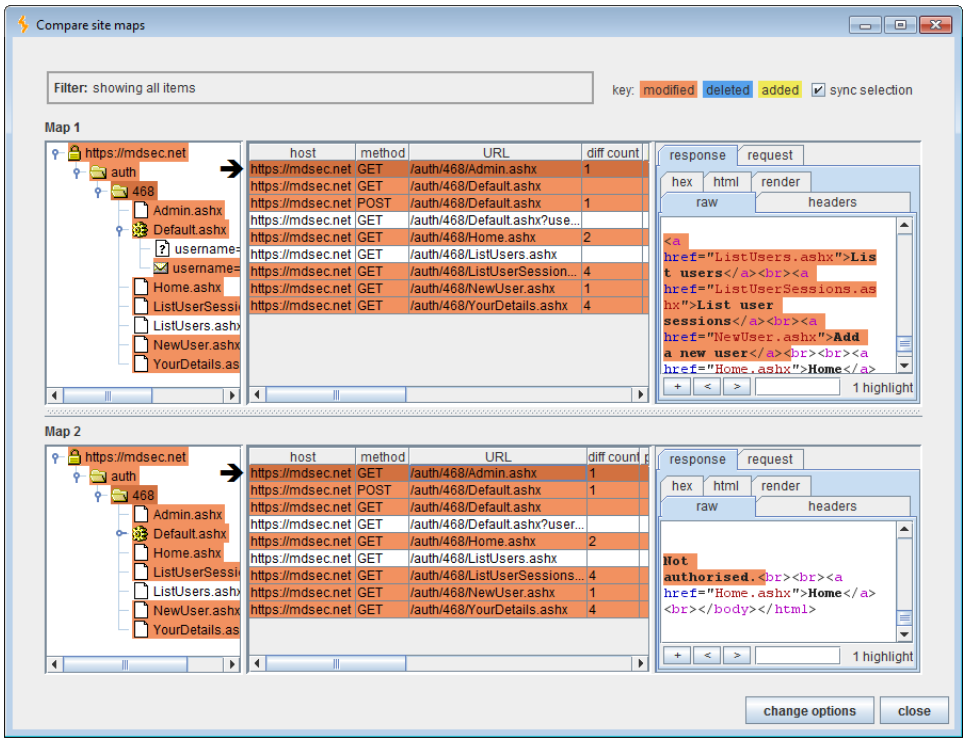


同样，在大多数情况下可以使用默认设置。这些选项忽略了没有太大价值的 HTTP 消息头和表单域，也忽略了空白响应。默认选项是减少无关噪声响应带来影响而设计的，让你能更容易地集中精力在差异上。
下面显示了一个简单的站点地图对比结果。显示出在管理员权限下和用户权限下发出的请求映射出的站点地图。经过分析后，两个站点地图的不同点，用不同颜色标出，并显示出两者之间的添加，删除，修改(如果请求过整个第一个站点地图，在地图里将不会有添加和删除项)。对于修改项，表格里有若干不同列，来表示由第一个站点地图内容修改为第二个站点地图内容。当你选择其中的一项时，另一个站点地图里的相关的项也被选中，响应里面的修改点会被加亮：



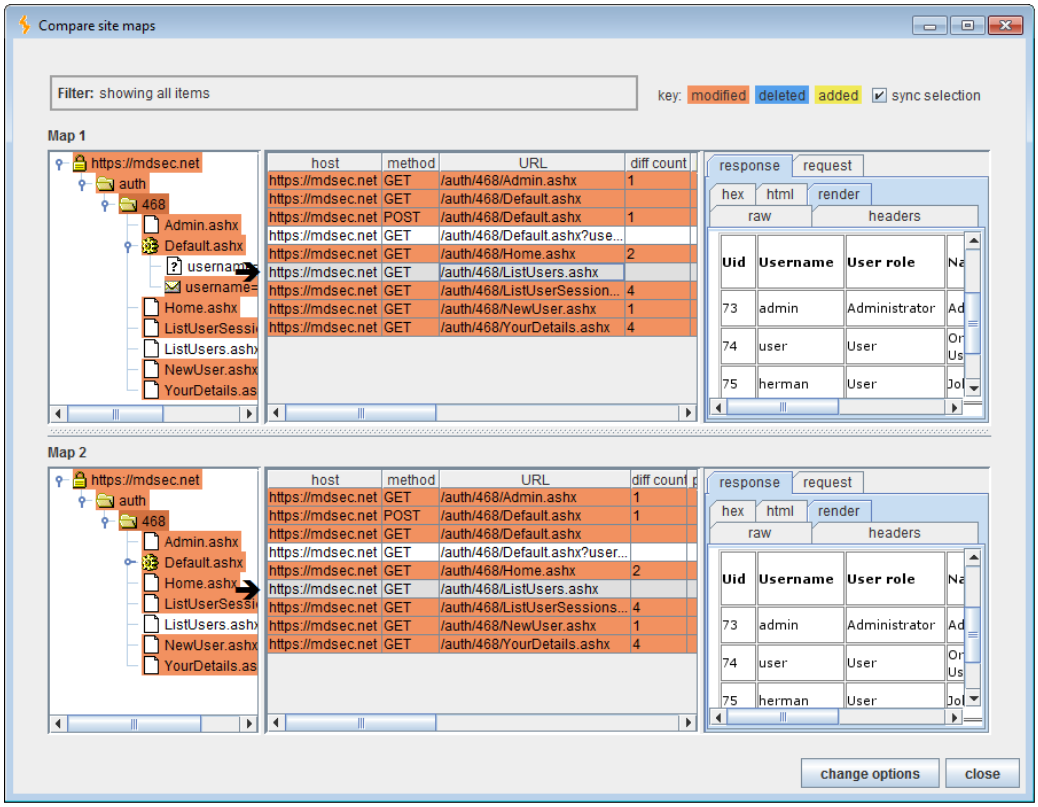
要想分析出站点地图的对比结果，就需要明白指定应用程序的上下文和意义，以及人的理解能力。例如，上面的截图显示了不同用户访问主页时得到的不同响应。这两个响应显示出了登陆用户的不同描述，管理员权限的用户有一个附加的菜单项。这些不同都可以预见的，他们对应用程序访问控制的有效性没有影响的，因为他们仅仅只在用户界面上。

下面的截图显示了两种用户请求顶级管理页面时得到的响应。这里管理员用户可以看到可用的菜单选项，而普通用户只能看到“not authorised”消息。这些不同显示了访问控制得到了正确的配置：



下面的截图显示了每个用户在请求具有管理员功能的“list users”得到的响应。这里响应相

同说明应用程序是有漏洞的，因为普通用户是不应该能使用这种功能，在他们界面里没有任何指向它的连接：



通过这个示例显示出，简单地对站点视图树的探索和查看项目的不同点，不足以评估应用程序的访问控制的有效性。两个想同的响应表明可能会有一个漏洞(例如，管理功能暴露的敏感信息)，也许会是安全的(例如，在显示登录用户个人资料的页面上)，所有的这些情况可以共存的，甚至在同一个应用程序上。这就是为什么全自动化工具在探测访问控制漏洞过程中效率是如此的低下。

因此，Burp 需要你密切地检查应用程序的功能，这不会减轻你的任务，对每一种情况都作出评估，看看访问控制是否正确的使用。站点地图的比较功能是一个尽可能自动化的过程，在一个清晰的表格里会给出你所需要的信息，并让你通过自己对应用程序的了解来发现任何的实际漏洞。

目标范围：

这个“scope”选项是让你告诉 Burp，以什么样扫描水准，以及哪些主机和 URL 来进行你的工作。你可以把目标范围大致想象成你有兴趣的并准备攻击的选项。
目标范围以许多方式影响单个 Burp 工具的行为，例如：
你可以设置过滤器只显示范围内的项。
你可以告诉代理只拦截范围内的请求和响应。
网络爬虫只会跟踪范围内的链接。
在 Burp Suite 专业版中，你可以启动自动扫描范围项内的漏洞。
你可以配置 Intruder 和 Repeater 重定向跟踪任何范围内的 URL。
通过告诉 Burp 你的目标是什么，可以确保 Burp 在你指定有兴趣的攻击目标后，以适当的方

式进行一系列此类的举动。在任何情况下，你都可以随时微调目标范围和单个相关工具的工作方式，如果需要，你可以精准地控制 Burp 的一切。然而 Burp-wide 宽度设定提供了一种既快又简单的方式，在你开始工作之前，来告诉 Burp，游戏的规则是什么和什么是禁区，以及什么有价值的。

目标范围的配置功能非常强大，却很简单。在“scope”选项卡里的用户界面里，你可以自定义目标范围包括哪些或者不包括哪些的规则。对于每条规则，你都可以定义以下字段：

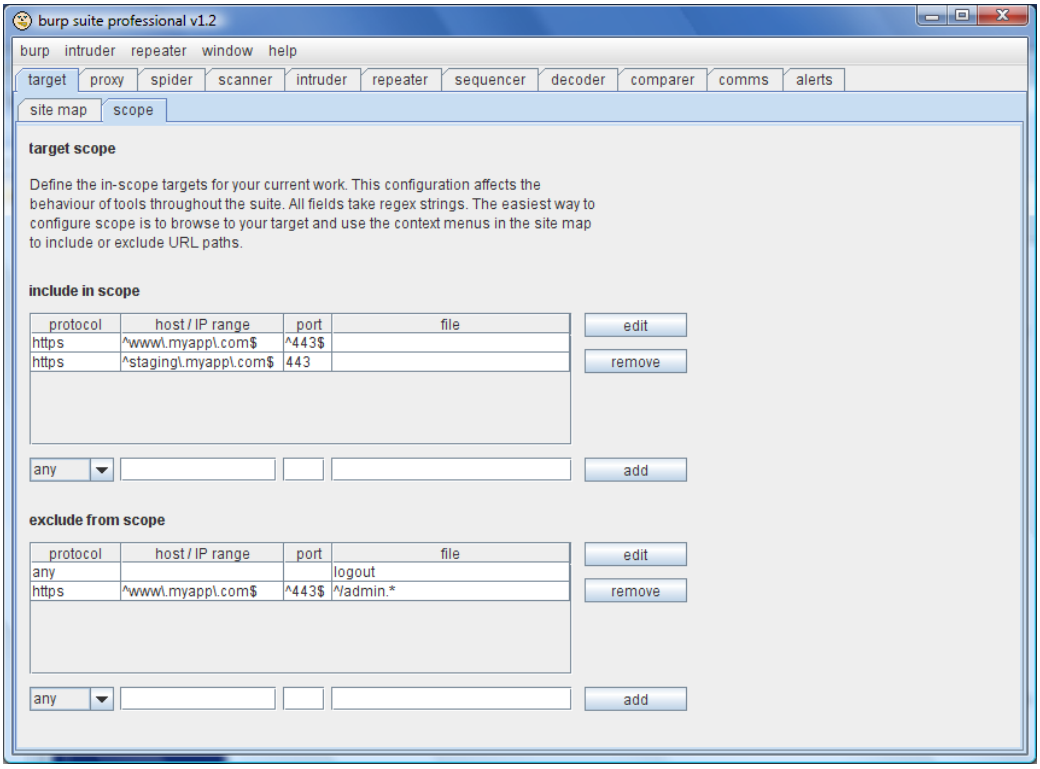
协议—HTTP，HTTPS，或者其他的

主机—这可以是一个正常的主机名正则表达式，也可以是标准格式的 IP 区间段，例如，10.1.1.1、24 或者 10.1.1-20.1-127。如果主机字段是空的，规则就会认为是任意主机。

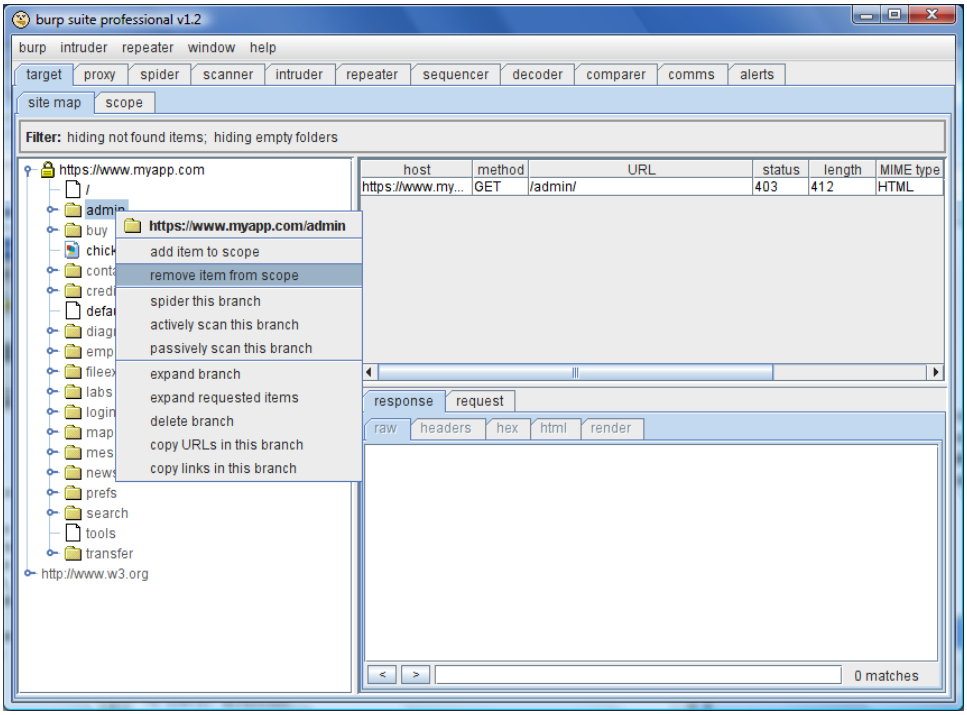
端口—这是一个匹配正常端口号的正则表达式，如果为空，则认为是任意端口。

文件—这是一个匹配文件部分 URL 的正则表达式，如果为空，则认为是任意文件。

当 Burp 决定评估一个在目标范围内的 URL 时，如果 URL 匹配包含规则其中一条并不符合任意一条不包含规则，这个 URL 将被视为在目标范围内。这使你可以自定义一般范围的内的特定主机和目录，但排除该范围内特定的子目录和文件。例如，下面这样定义的目标范围会匹配 http://www.myapp.com 和 https://staging.myapp.com 以及在 https://www/myadd.com/admin 目录下的内容的表达式，还有任何包含 “logout”表达式的 URL。



像上面描述那样直接设置范围规则对许多用户来说，显得不太友好。有一个更简单的方法，就是让 Burp 根据你在站点地图或其它地方使用上下文菜单给出的直观说明来为自己定义规则。在你开始探测应用程序前，你需要简单地浏览目标页面，这样才能让它显示在站点地图上。然后，你可以选择一个或多个主机和目录，并通过上下文菜单把他们在范围内包含或排除。这个过程非常简单，大不多情况下，这会让你迅速地定义出满足你测试的所有规则：

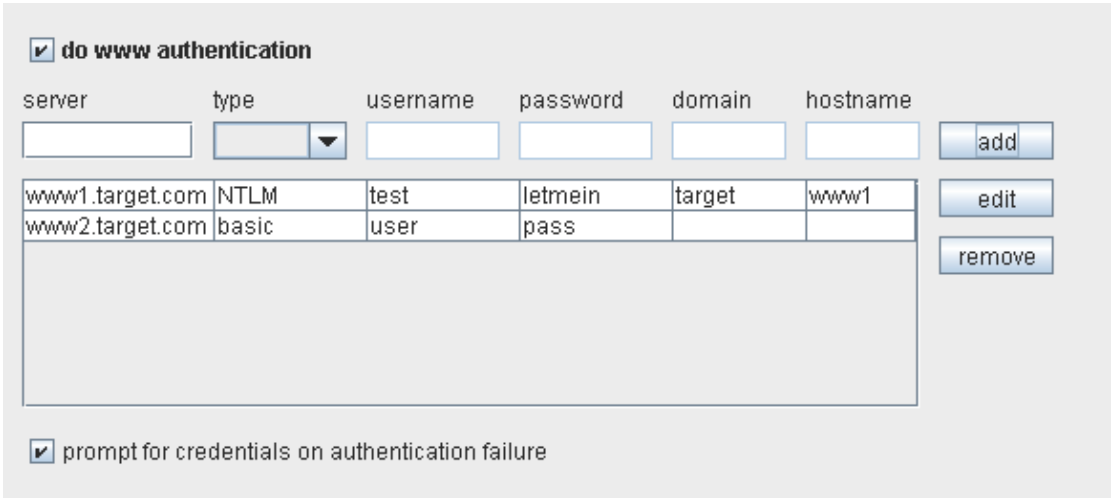


套件选项

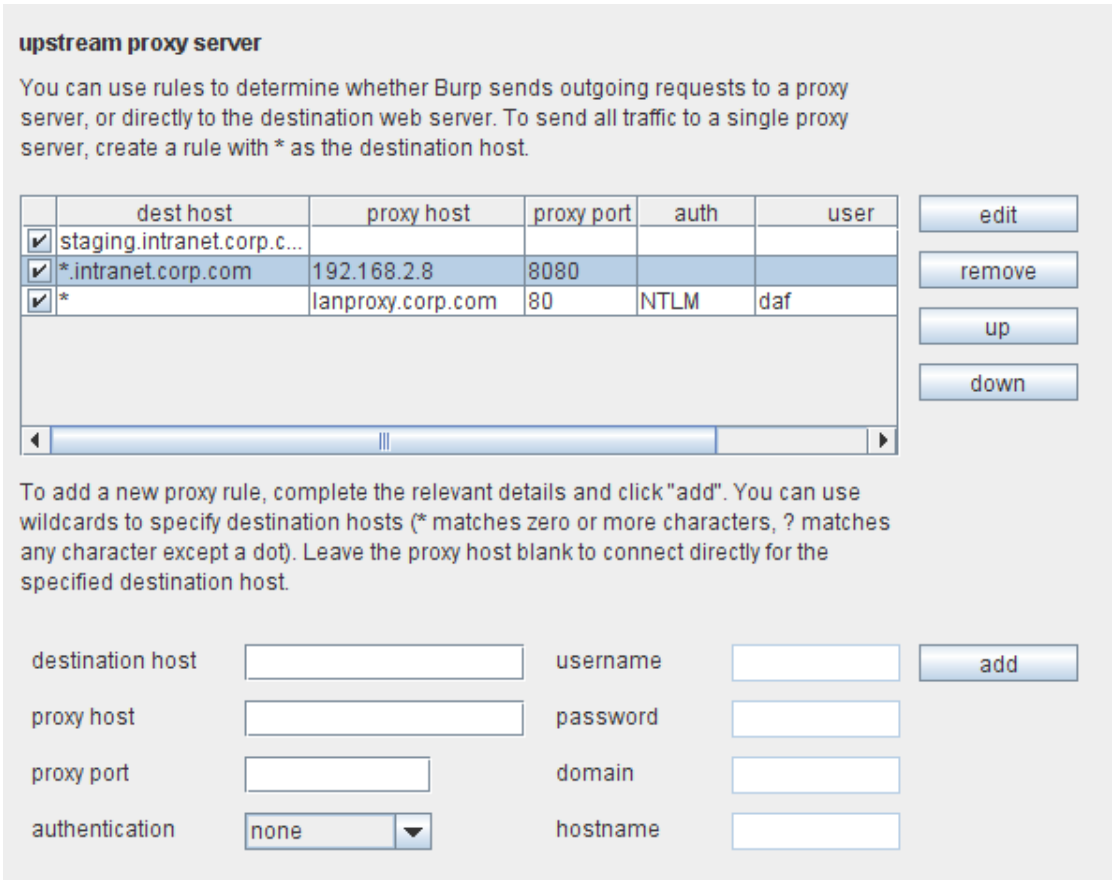
这个选项卡里包含了不针对任何单一工具的 Suite-wide 设置，他们分为包含许多模块的子标签。

连接选项

这个标签里包含了控制 Burp 怎样处理网络连接的选项，认证，代理服务器，重定向，超时，主机名解析。



这些控制 Burp Suite 在连接目标 web 服务器时是否使用认证。可以为不同的主机配置不同的认证类型和验证。支持的类型有：basic，NTLMv1，NTLMv2 和摘要式身份认证。域字段和主机名字段仅用在 NTLM 认证。在遇到认证错误时，“prompt for credentials”选项会弹出一个互动窗口来提示错误。

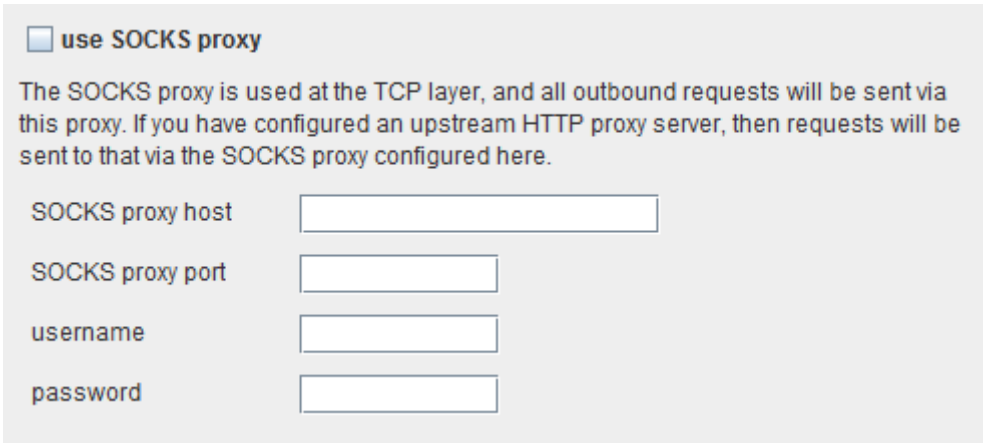


这个设置允许你配置规则来为不同域的主机指定不同的代理。

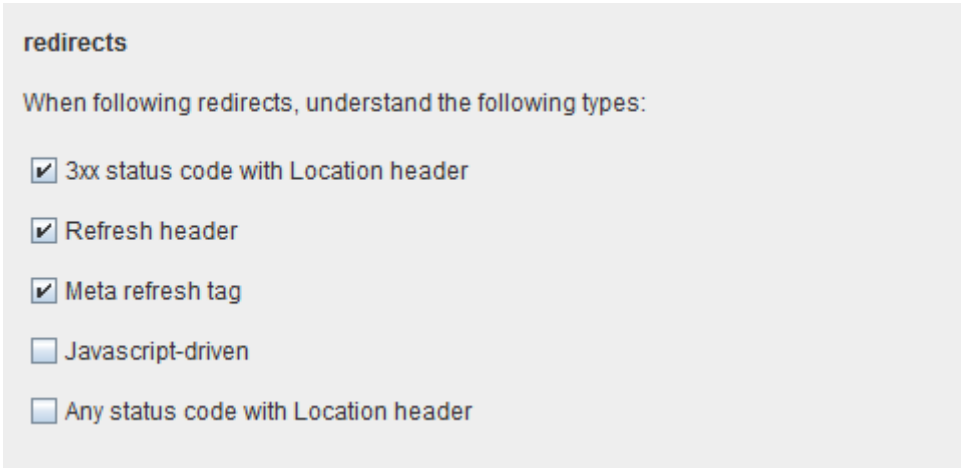
上面的这个设置会使 Burp 直接地与 staging.intranet.corp.com 进行会话。使用一个没有任何验证的内部代理服务器访问*.intranet.crop.com，对所有的访问，包括公共互联网，都使用一个有网关认证的 web 代理。

你可以在特定的目标主机上使用标准通配符。规则是按顺序使用的，第一条符合你连接的 web 服务器的规则将被使用。如果没有找到匹配规则，Burp 将默认不使用代理进行连接。

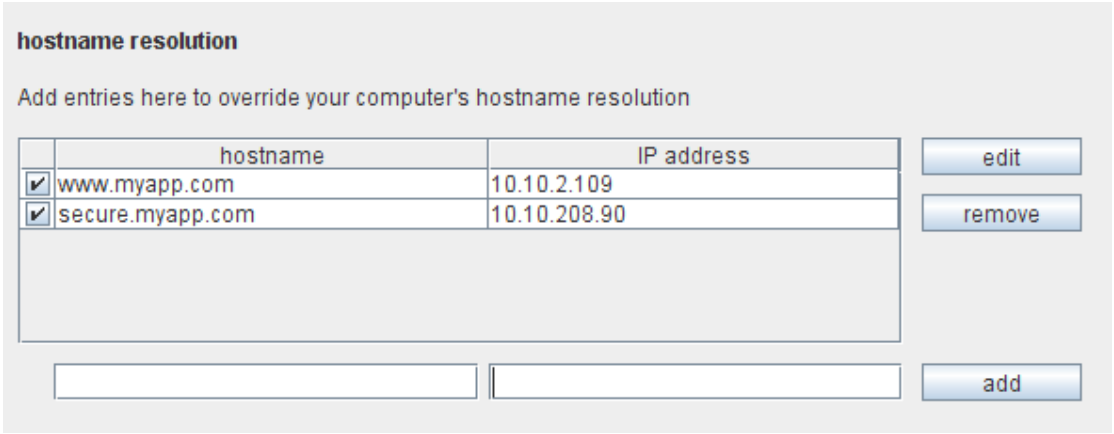
如果需要，你可以为每一个上游代理指定一个认证类型和验证。支持的认证类型：basic,NTLMv1,NTLMv2 和摘要式身份验证。域字段和主机名字段只对 NTML 认证有效。



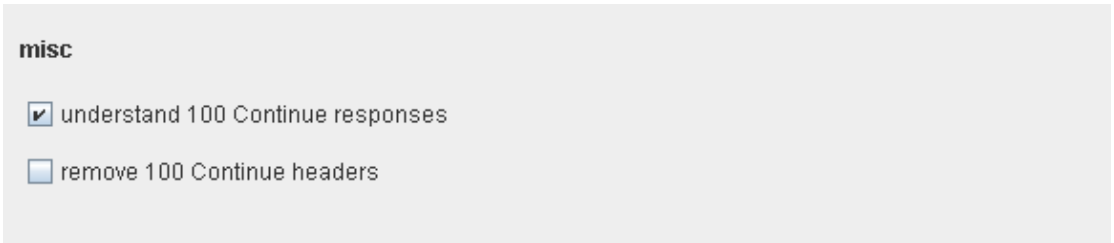
这些设置是让你配置 Burp 为所有代发的连接指定一个 SOCKS 代理。



这些设置决定了各种网络任务的最大延时。“normal”项是最多网络连接有效的设置，并决定了 Burp Suite 在放弃一个请求并记录超时前等待的最长时间。“read til close”设置是用在处理响应过程的，并且这个响应包含一个没有 Conntent-Length 或者 Transfer-Encoding 的 HTTP 消息头。在这种情况下，在确定传输完成前，Burp Suite 会等待一个特殊间隔。“domain name resolution”设置 Burp Suite 成功解析域名的频率。如果目标主机地址频繁地变化，就需要设置一个适当低的值。“failed domain name resolution”设置决定了 Burp Suite 重新尝试解析没有成功解析的域名的频率。



这些设置让你指定特殊的主机名到 IP 的映射，并覆盖由操作系统提供的 DNS 解析。当主机文件被修改，使用无代理感知后客户端组件为传输提供不可见的代理时，这个功能能确保正确地向前转发请求。



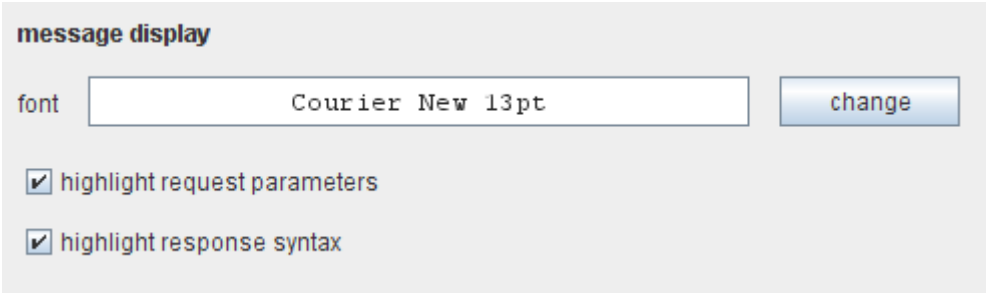
这些设置是控制处理从服务器接收到的 HTTP 100 的连续响应。这会发生在这种情况下，当你向服务器提交一个 POST 请求后，在请求完全传送完之前，服务器会返回一个响应。如果选择“understand 100 Continue responses”，Burp Suite 会跳过中间响应，并为响应信息解析真正的消息头像状态编码和内容类型。如果选择“remove 100 Continue headers”，在这些服务器的响应经过单个工具前，Burp Suite 会清除任意的一个响应的消息头。

Sessions 选项

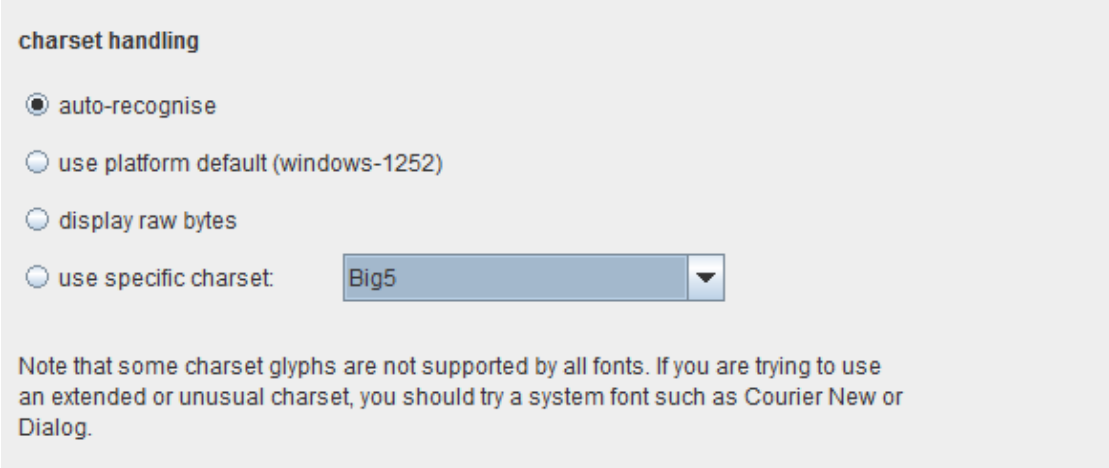
这个选项是让你配置 Burp 的会话处理和宏功能的。要想了解这些功能的更多信息，请查看会话处理帮助。

显示选项

这个选项卡里包含了控制怎样显示 HTTP 请求和响应的选项。

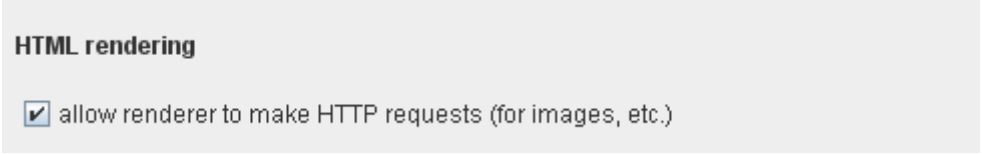


这些设置控制着用于显示 HTTP 消息的字体，以及在请求和响应中，是否执行加亮语法。



在显示 HTTP 请求和响应时，这些设置控制 Burp 怎样处理字体设置。默认情况下，在每条响应中，能自动识别字体并正确显示出来的。在开起 Burp 时，就避免了需要在命令行上设置一个特定的字体，在 Burp 里的同一个实例里，允许的你的工作中有多种不同字体的内容。

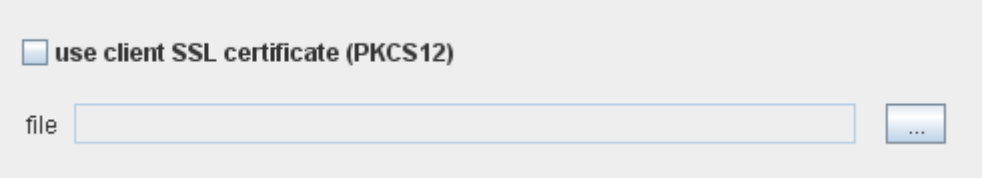
在上面的选项里，你可以设置一个特殊的字体覆盖默认的值，或者告诉 Burp 显示没有字体处理的原始字节。
注意有些字符不是被所有字体支持的。如果你想使用一个非拉丁字形的字符，你需要首先去尝试一个系统字体，如：Courier New 或者 Dialog。



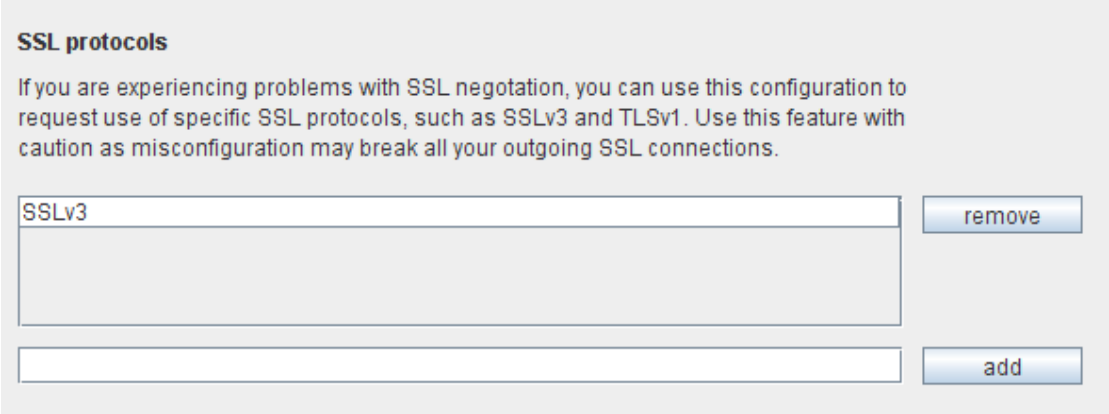
在 Burp Suite 里的显示 HTTP 响应的任意位置都有可能呈现出 HTML 内容，因为他会出现你的浏览器里。这些选项控制 Burp Suite 是否要添加一些 HTTP 来完全呈现出 HTML 内容(如嵌入的图像)。使用这个选项涉及到 Burp Suite 执行速度和 HTML 内容质量的权衡。

SSL 选项

这个标签里包含了怎样使用 SSL 的选项，以及服务器提交的 SSL 认证的信息。

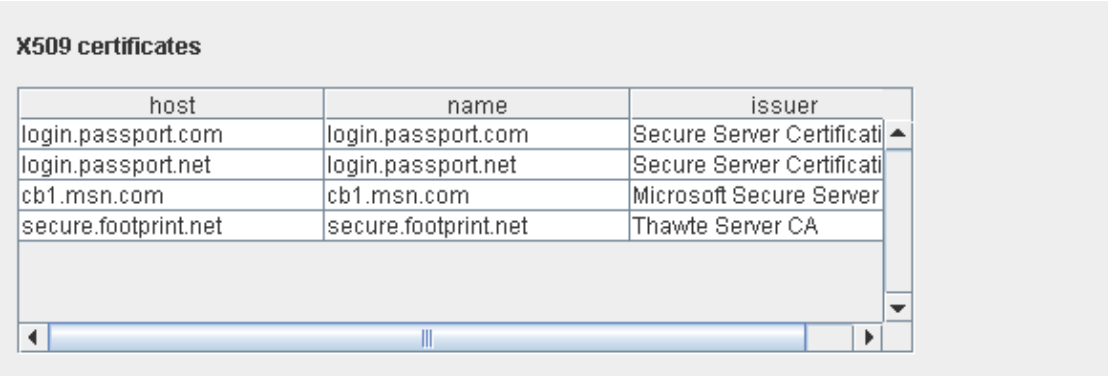


这个选项使你你能够配置一个客户端 SSL 认证(PKCS12 格式)，这被用在目标 HTTP 服务器要求客户端提供证书认证时候。当没必要使用一些客户端认证时，你可以配置 Burp 允许不太安全的重复协商。



有时，你在与某些 web 服务器进行 SSL 连接协商时运到困难。Java 的 SSL 栈里有一些捣蛋鬼，不会与其他不明服务器设置兼容的。为了帮助你排除这个故障，在进行 SSL 协商时，Burp 允许你指定服务器提供的是哪一种协议。
这里要注意 Burp 本身采用了一些 SSL 问题的解决方法，如果用你配置这些协议协商失败，Burp 会尝试采用其他能用的组合协议。因此你不会用到测试服务器支持哪种协议的功能。

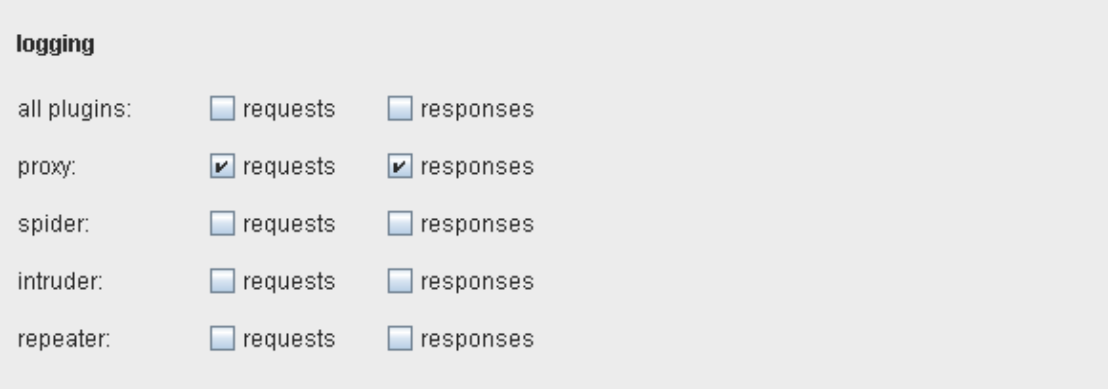
在 SSL 协商时，你可以配置 Burp 使用所有的密码套件。这个选项一般不会用到，但当尝试连接一些不常见配置的 SSL 栈时，它会非常有用。



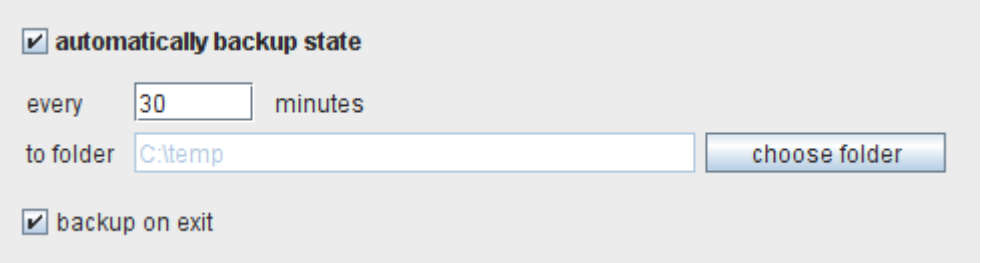
这个信息面板包含了从目标 web 服务器接收到的所有 X509 证书的细节。双击表中的一个项就可以显示出整个证书的细节。

杂项

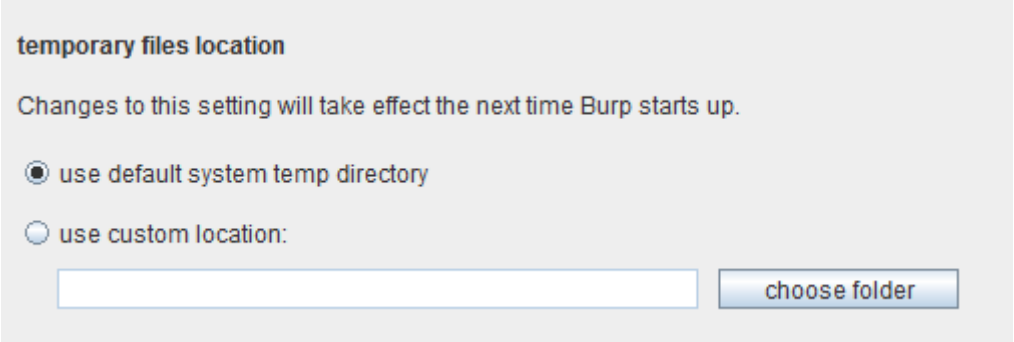
这个选项包含了一些杂项设置：日志记录，备份，临时文件存放位置，预定任务。



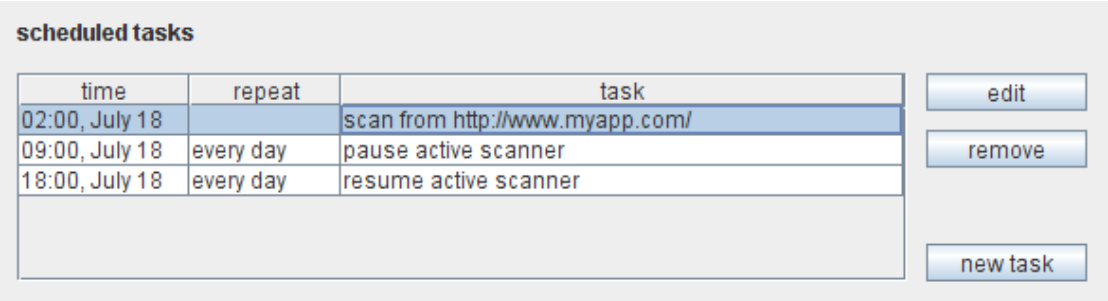
这些设置控制网络请求和响应的日志。可以针对一个工具或者整个 Burp Suite 进行配置日志。



[专业版]这些设置让你通过配置 Burp 来保存所有工具状态的备份，和在一个可配置的时间间隔内进行后台配置，以及选择性退出。这些设置仍然能够加载到 Burp，所以你可以把 Burp 设置成实时保存到一个临时目录下，这样在你使用 Burp 的每一时刻，都会有你工作的备份副本。

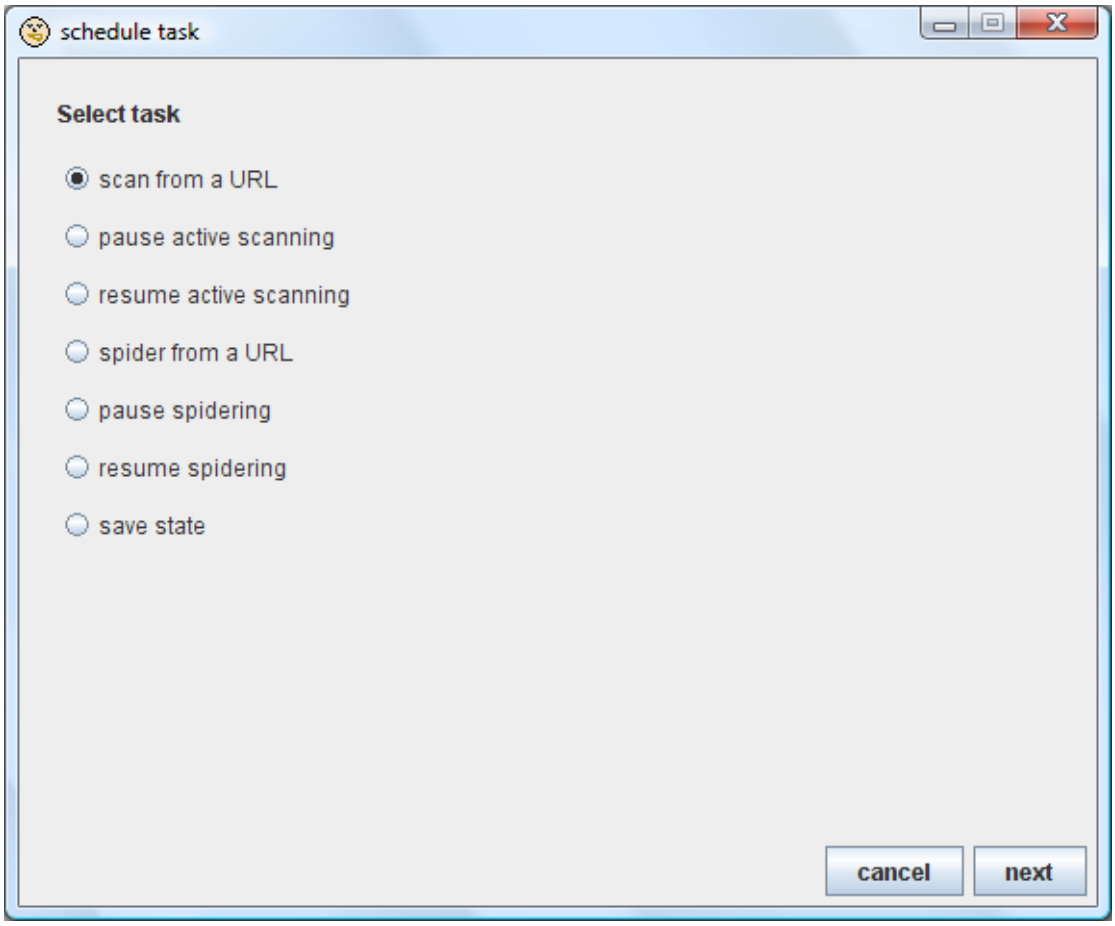


这些选项是让你设置 Burp 临时文件保存的目录路径。这允许你指定一个不同的路径，如果需要，可以指定为受限访问。在下次 Burp 启动的时候，这些设置才会生效。



[专业版]在规定的时间间隔内，你可以使用任务调度来自动地开始和停止某些任务。例如，上面的设置显示：在凌晨 2 点开始扫描目标，并在每天的工作时间段内暂停。

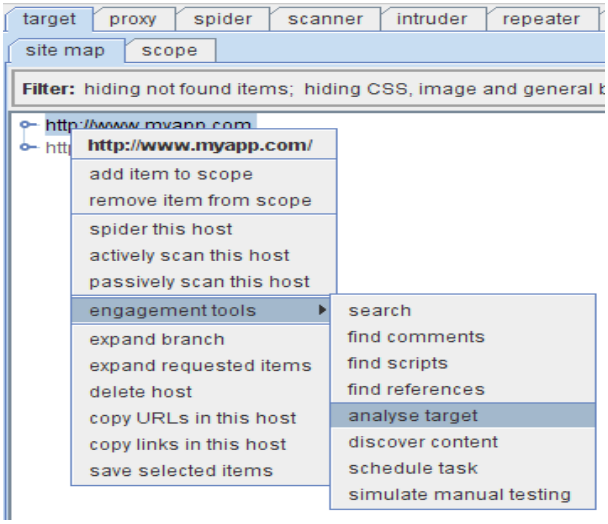
你可以通过出现在整个 Burp 范围内的上下文菜单来创建任务，也可以通过面板上的“new task”按钮来创建。开始会有一个向导出现，让你来配置任务的细节和时序。提供了各种不同类型的任务：



你可以配置每一项任务是一次性的还是过一段时间就重复执行。

定制工具

[专业版]一些工具就是为了帮助使你的工作更快更效率的完成而存在的。可以通过显示在 Burp 整个范围内的上下文菜单来找到他们：

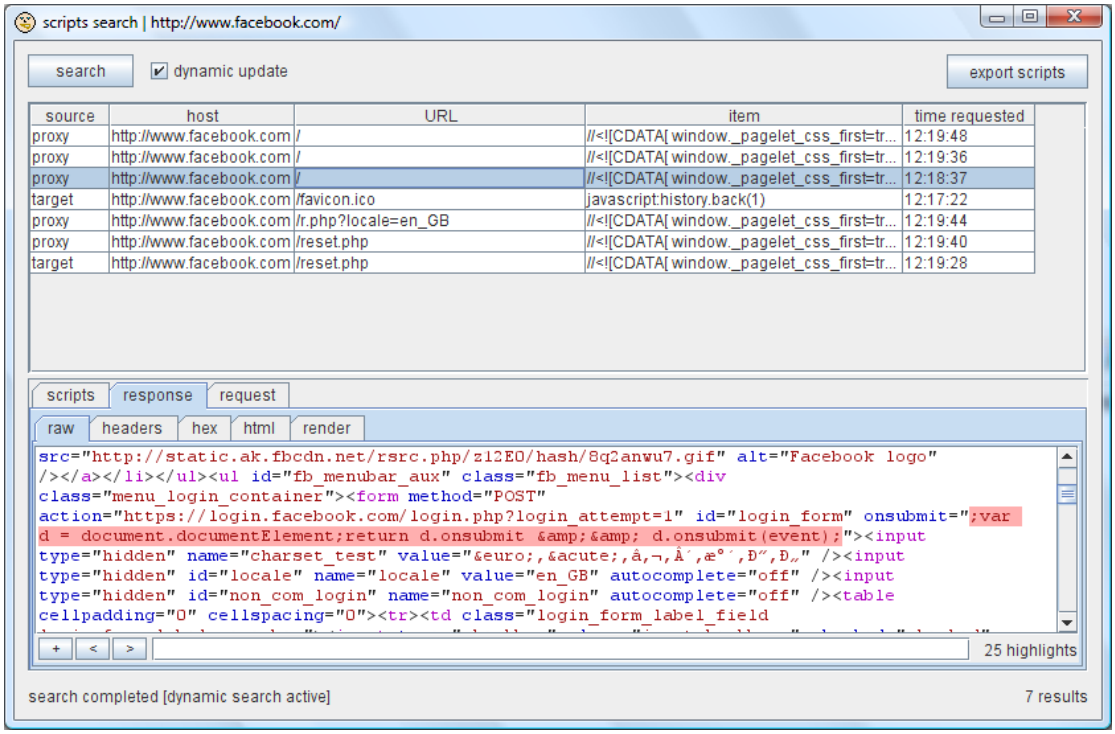


搜索

具体查看搜索帮助。
注意如果你在目标站点地图上开始一个搜索(和 Burp 菜单不一样)，那么将指定在选择中的站点地图分支上进行搜索。

查找注释和脚本

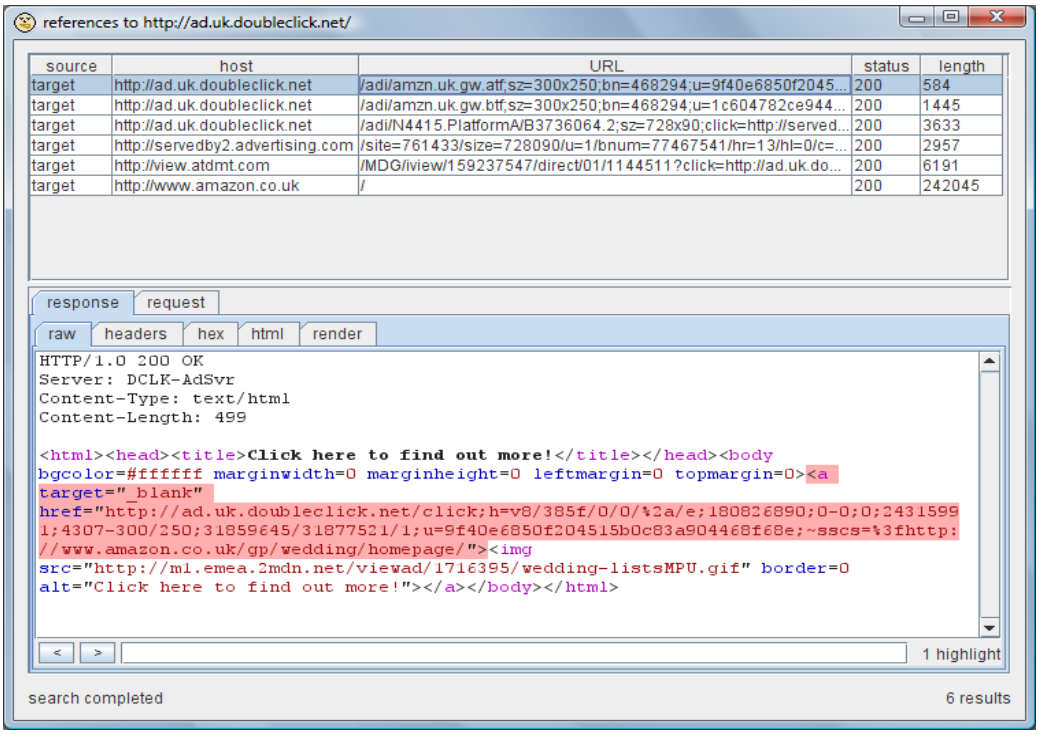
你可以使用这个功能来搜索部分或整个站点地图内的注释和脚本。搜索结果窗口内会显示出所有包含脚本和注释的 Burp 工具得到的响应。在预览表格内会显示出选中项的完整的请求和响应，相关的项会被自动加亮，并同时提取到他们的标签：



你可以使用“export”按钮把所有的脚本或注释保存的文件或粘贴板上。可选择性地加强重复出现的项。

查找引用

任何地方你都能看到 HTTP 请求，URL，域名等等。你可以在所有的 Burp 工具使用“find references”来搜索链接到这些项的 HTTP 响应。搜索结果的窗口显示链接到选中项的所有 Burp 工具的响应。当你查看其中的一个搜索结果时，响应会自动加亮显示出链接引用发生的地方：



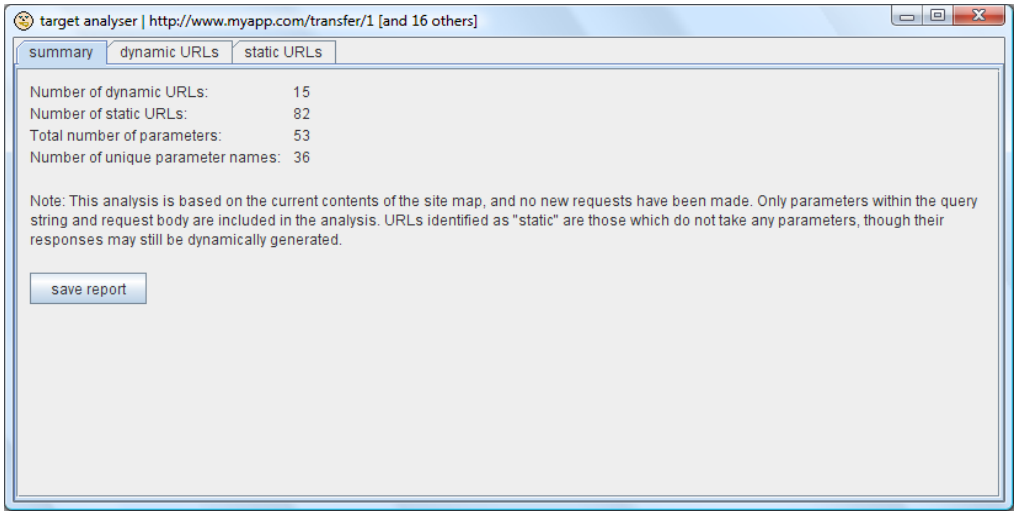
当搜索链接时，注意这个功能会把本地 URL 当做一个前缀来处理。因此如果你选择一个主机，你找到所有对这个主机的引用；如果你选择一个文件夹，你会找到所有对这个文件夹或更深目录的引用。

新的“find references”和老版本的 Burp Spider 中的“linked from”列表的服务目的是一样的，但它更强大一些。

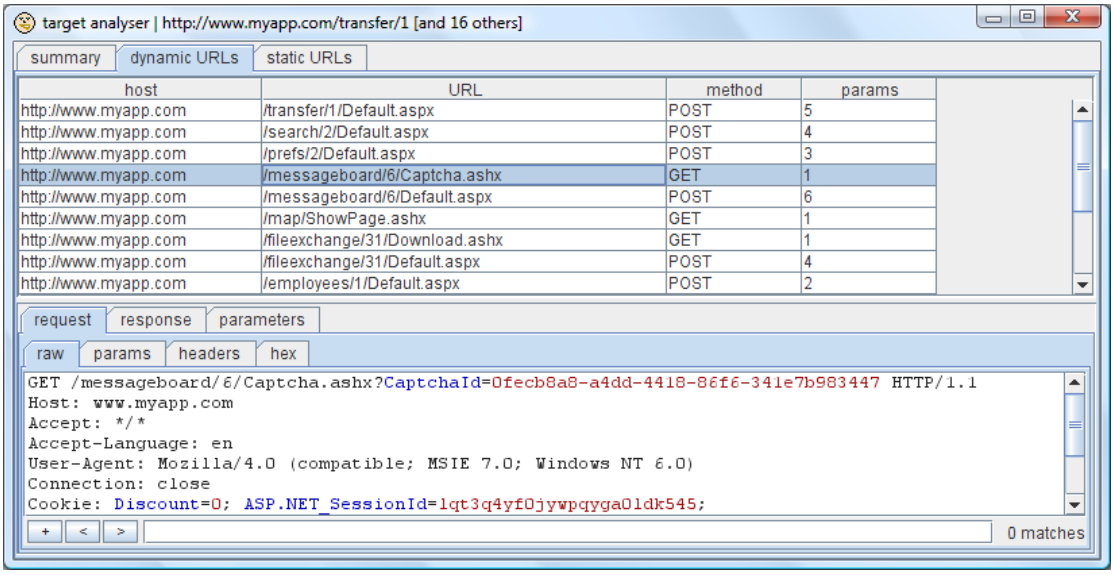
分析目标

这个功能是用来分析一个目标 web 应用程序，并告诉你他有多少静态和动态的 URL，以及每个 RUL 有多少参数。这能帮助你评估出需要花多少精力来完成相关的渗透测试，还帮助你决定把注意力放在测试的那一块。

要想使用这个功能，要选择一个或多个主机或者站点地图的分支，然后使用上下文菜单运行它。汇总后的信息就是这样的：



并且你可以向下挖掘出更多的 URL 细节：



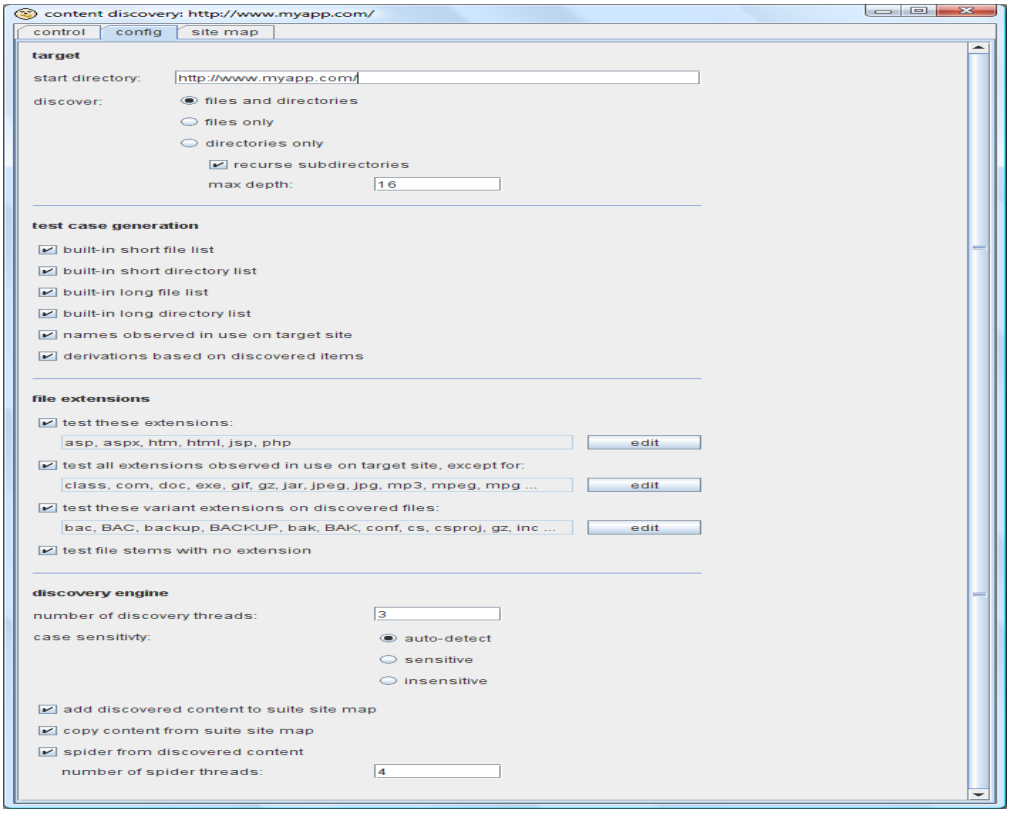
你可以把所有的信息以 HTTP 报告的形式输出，你可以把这些信息附加到客户端的建议和报告中来显示你覆盖的攻击面。

注意：(1)这个功能只能用来分析站点地图已经捕获的内容，所以在你运行它之前，必须保证你已经浏览或追踪到了应用程序的所有功能和内容。

(2)如果 URL 或消息体内没有任何参数的话，URL 将被认为是“静态”；然而应用程序产生的 URL 响应可能会是动态的。

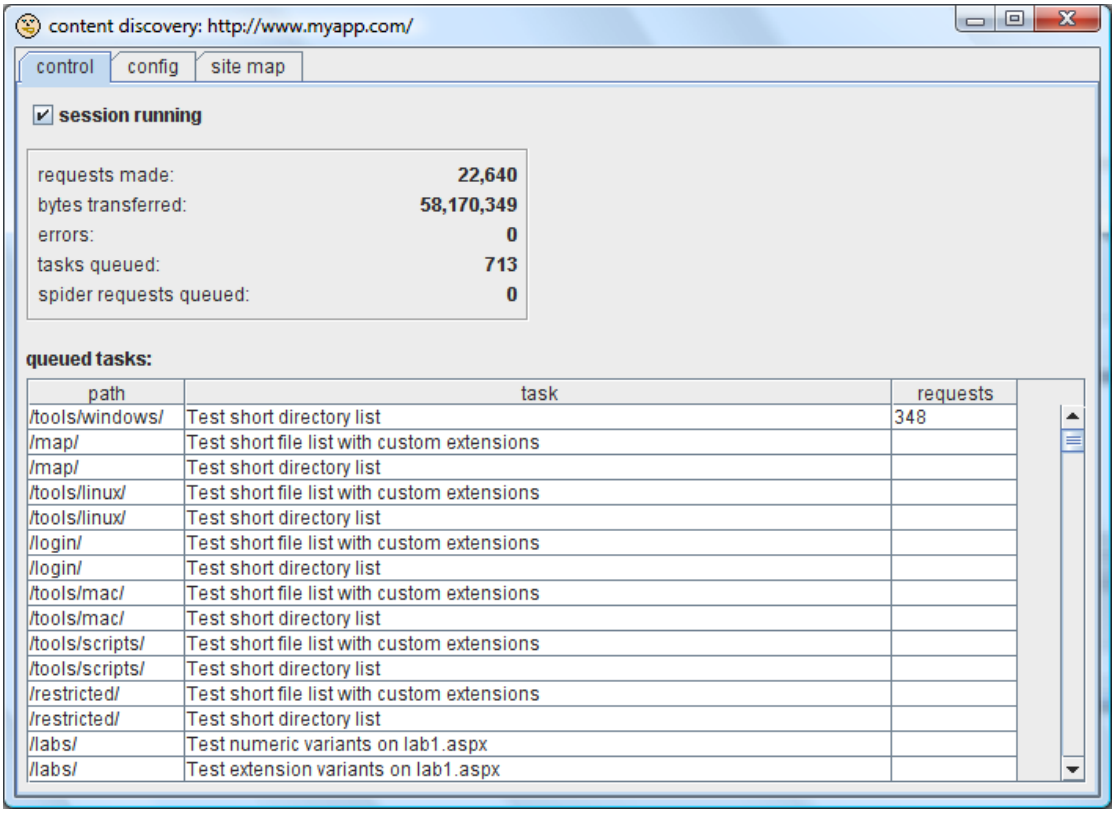
探索内容

你可以使用这个功能来发现一些和你浏览或追踪到的可见内容无关的一些内容和功能。Burp 使用许多技术来发现内容，有猜测名字，网络爬虫，以及在应用程序中使用约定命名来推断。这些功能是高度可配置的，下面显示出了一些可用的选项卡：



target：这个选项控制着开始探索的目录。在会话期间只有这个路径和它的子目录才会被请求。你可以选择探索文件或目录或两者，以及探索子目录的深度。
test case generation：当要向探索内容发送请求时，Burp 就会使用由这些选项控制的文件和目录名字。和内置列表一样，Burp 能获得在应用程序任何地方使用的名字，并且在其他地方重复它们，还可以通过探索项来构造名字，通过循环值得到文件名中包含的数字。
file extensions：你可以指定一个用来测试可能文件名字的文件扩展名列表。Burp 能通过观察应用程序使用的扩展名来获得扩展名，并用每一文件名来测试这些扩展名。当有一个文件符合时，Burp 就会尝试一个指定的有文件名组成的变种扩展名列表，例如检查老的或备份版本里的同一文件。
Discovery engine：你可以控制使用多少线程来进行内容探索和追踪，以及处理的文件名字是否敏感，以及在 Burp 的主站点地图怎样进行探索互动会话的(在套件的目标选项卡里)。

当你配置完探索会话时，你可以在控制选项上启动它了，这里提供了执行动作的运行时间信息。这项工作被分成了无数个分散的小任务，他们的优先权取决于快速发现新内容的可能性，随着内容的确定，新任务会递归地产生：



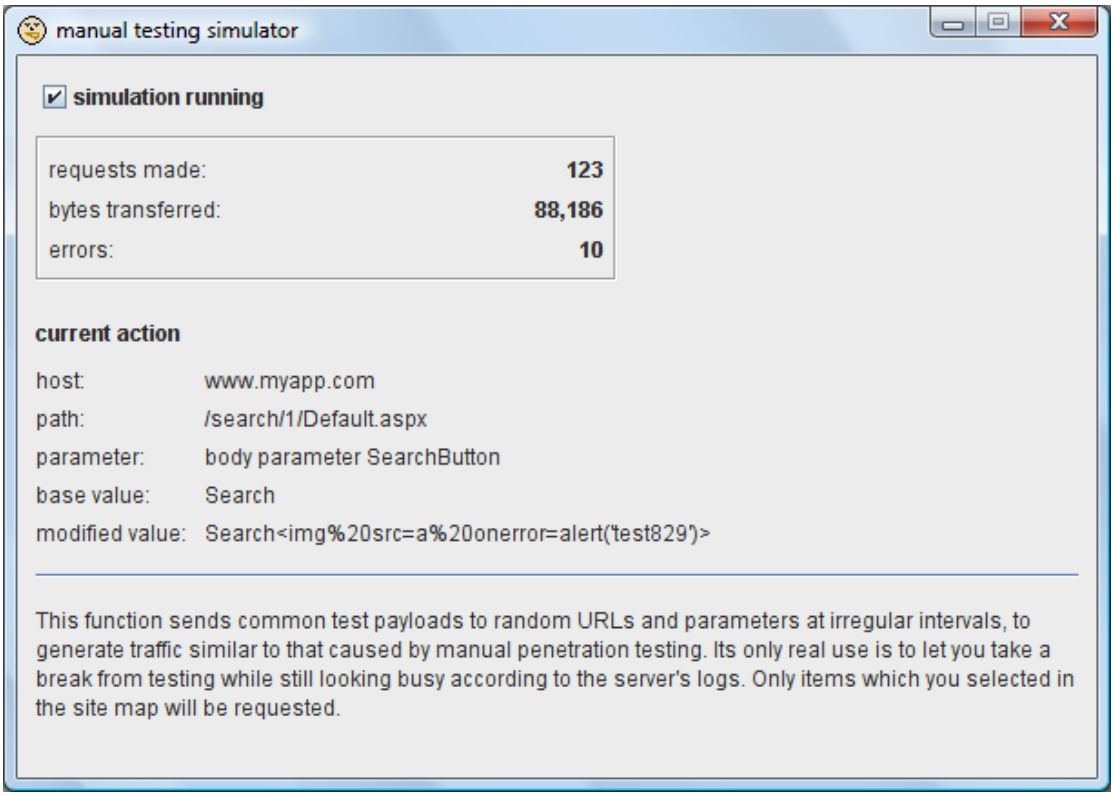
这个探索会话使用的是它自己的站点地图，显示出所有的发现的在定义范围内的内容。如果你是通过配置 Burp 这样做的，接下来新发现的项会被自动地添加到主站点地图里。

任务调度

可以查看任务调度帮助。

手动模拟测试

这个功能不会明显地提高你的效率，但有时你会发现它还是有用的。你可以使用它让 Burp 进行人工模拟测试活动，通过向目标应用程序里的随机 URL 和参数发送常规的测试有效载荷。Burp 不会处理响应，所以在这情况下你不会发现漏洞。但是如果你认为有些人可能会来查看应用程序的日志来确定你在做的事，当你在进行一个长时间的午餐，健身房会议，或者其他一些使你不能分身的情况下，你可以使用这个功能。



复活节彩蛋，有人吗？

消息编辑器

(略)

可扩展性

(略)

Burp Proxy Help

什么是 Burp 代理

Burp 代理是一个用来攻击和调试 web 应用程序的交互式的 HTTP /S 代理服务器。他就像一个在浏览器和目标服务器之间的中介人，并允许用户拦截，查看和修改两个方向上的原始流量。

Burp 代理允许你通过监视和操纵应用程序传输的关键参数和其他数据来查找和探索应用程序的漏洞。通过以恶意的方式修改浏览器的请求，Burp 代理可以用来进行攻击，如：SQL 注入，cookie 欺骗，提升权限，会话劫持，目录遍历，缓冲区溢出。拦截的传输可以被修改成原始文本，也可以是包含参数或者消息头的表格，也可以十六进制形式，甚至可以操纵二进制形式的数据。在 Burp 代理可以呈现出包含 HTML 或者图像数据的响应消息。

除了每一个请求的操纵外，Burp 代理保持着一个完整的历史记录，包括浏览器发送的每一个请求，所有的操作，以及接收到的所有响应。你可以复查较早的请求，并修改后补发任何请求，还能以原始的形式或者 web 页面的形式来查看保存的响应。可以把两个方向上的整个会话记录到一个文件中，用来作进一步分析或者提供审查线索。

在 Burp Suite 里，Burp 代理是何其他工具紧密地集成在一起的。并允许把任意的请求和响应发送到其他工具里作进一步处理。通过一次单击，你可以把一个感兴趣的请求作为分析会话令牌的基础，或者经过人工修改后再发送，漏洞分析，或者使用 Burp Intruder 进行一次自定义的自动攻击。

通过基于域名，IP 地址，协议，HTTP 方法，URL，资源类型，参数，cookies，消息头/消息体内容，响应代码，内容类型和 HTML 页面标题的请求和响应的细微规则来控制 Burp 代理的交互式行为。可以悄悄地执行配置操作，而不会影响任何请求。你可以晚一点查看历史记录以确认请求经过更严格检查。虽然使用了正则表达式匹配规则和替换规则，Burp 代理还可以用来自动修改 HTTP 请求和响应消息。

除了用户的主界面，在浏览上也可以控制 Burp 代理用来查看请求历史记录和重发请求。

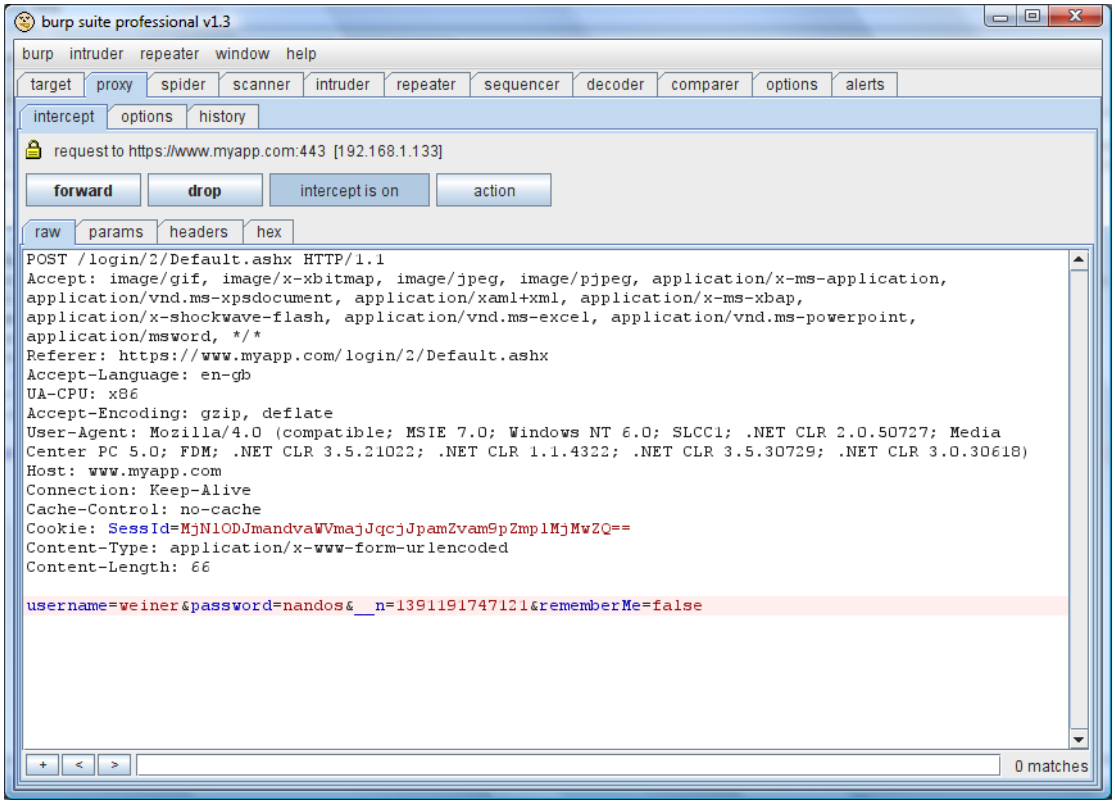
Burp 代理可以与上游代理结合起来使用。它可以处理上游代理和 web 服务器的 basic，NTML 和摘要式身份认证，同样也可以用在大多数局域网环境中。它支持 SSL(可以自定义服务器或客户端证书)，可以查看 HTTPS 传输，还可以把它修改成明文。另外，它还能自动处理许多种编码的服务器响应，包括分块传输编码和压缩内容编码。

使用 Burp 代理

当 Burp 代理运行后，HTTP/S 代理服务器自动开启 8080 端口仅作为回环接口。要开始使用 Burp 代理，需要简单地把你的浏览器的代理服务器设为 127.0.0.1 :8080，开始浏览。

默认情况下，Burp 代理配置的是自动拦截对非媒体资源的请求。显示出来后，可以查看和修改。其他的请求(图片和样式表)和所有服务器的响应自动地转发。这些默认动作是可以修改的。

拦截选项卡(Intercept tab)



你可以使用快捷键 ALT+F 和 ALT+D 来代替“forwar”“drop”。

通过单击其中一个可以使用的选项卡，能以几种形式来显示和分析每个请求和响应。随着适当的消息类型的显示，可用选项卡也会随之显示或消失：

raw：这里显示的是纯文本形式的消息。在文本窗口的底部提供了一个搜索和加亮功能，可以用它来快速地定位出消息中的感兴趣的字符串，如错误消息。在搜索的左边有一个弹出项，让你来处理大小写问题，以及是使用简单的文本搜索还是正则表达搜索。

params：对包含参数(URL 查询字符串，cookies 消息头，或消息体)的请求，这个选项可以把参数分析成名称/值的组合，并且允许你能简单地查看和修改。

headers：这里以名称/值的组合来显示 HTTP 的消息头，并且还以原始的形式显示消息体。

hex：这里允许你直接编辑消息的原始二进制数据。如果在文本编辑器里修改，某些传输类型(例如，使用 MIME 编码的浏览器请求的部分)包含的二进制数据可能被损坏。为了修改这些类型的消息，应使用十六进制。

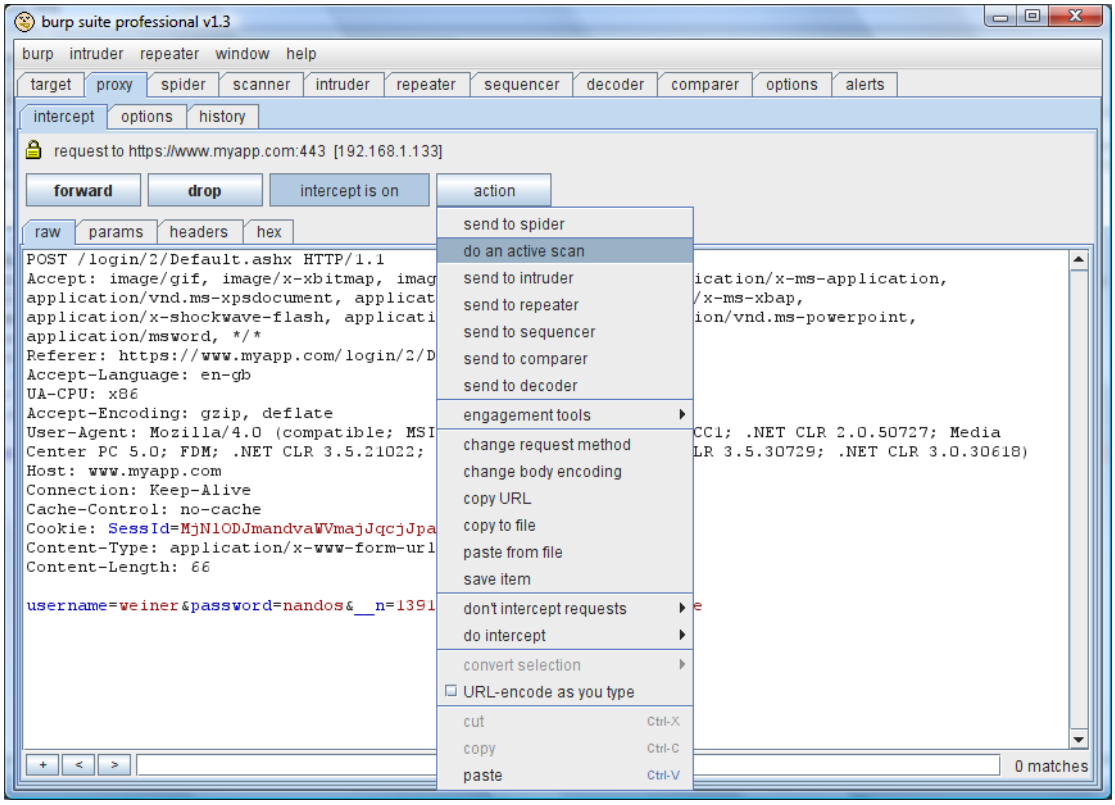
HTML/XML：对于响应中的这种格式的内容，为消息体提供了一种 syntax-colourised 视图。

render：对于响应中的包含 HTML 或者图像的内容，以视觉形式显示这些内容，最后会显示在浏览器里。

AMF：对于 Action Message Format 格式的响应或请求，这显示了一个解码消息的视图树。如果需要编辑，你可以双击树上的节点来修改它们的值。

viewstate：对于包含一个 ASP.NET 视图状态的请求和响应，会把视图状态的内容非序列化，这允许你查看任意敏感内容的数据。也可以表明视图状态 MAC 是否可用(也就是视图状态是否可被修改)。

在任意的显示的项上右击打开上下文菜单可执行多种操作。也可以通过主界面上的“action”按钮来打开同样的菜单：



send to:你可以把任意的消息，或者选中的部分消息，发送到 Burp Suite 中的其他工具里，来作进一步分析或攻击。

find reference[专业版]你可以在所有的 Burp 工具里使用这个功能来搜索和选中项有关的 HTTP 响应。

discover content[专业版]你可以使用这个功能来探索和用浏览器和网络爬虫发现的内容不同的内容和潜在的功能。

schedule task[专业版]你可以使用这个功能来创建一些设定好运行次数和时长的自动执行的任务。

change request method 对所有的请求，经过把所有相关的请求参数适当地搬迁到这个请求里来，你就可以自动地把请求的方法在 POST 和 GET 中间切换。通过发送恶意的请求使用这个选项来快速测试应用程序的极限参数是多少。

change body encoding 对于所有的请求，你可以在应用程序/X-WWW 格式的 URL 编码和多重表单/数据之间切换消息体的编码方式。

copy URL 这个功能是把当前的 URL 完整地复制到粘贴板上。

copy to file 这个功能允许你把选择一个文件，并把消息的内容复制到这个文件里。这个对二进制数据来说是很方便的，要是通过粘贴板来复制会带来一些问题。复制操作是在选择的文本上进行的，如果没有被选中的内容，则是针对整个消息了。

paste from file 这个功能允许你选择一个文件，并把文件里的内容粘贴到消息里。这个对二进制数据来说是很方便的，要是通过粘贴板来复制会带来一些问题。粘贴操作会替换掉被选中的内容，如果没有内容被选中，则在光标位置插入这些内容。

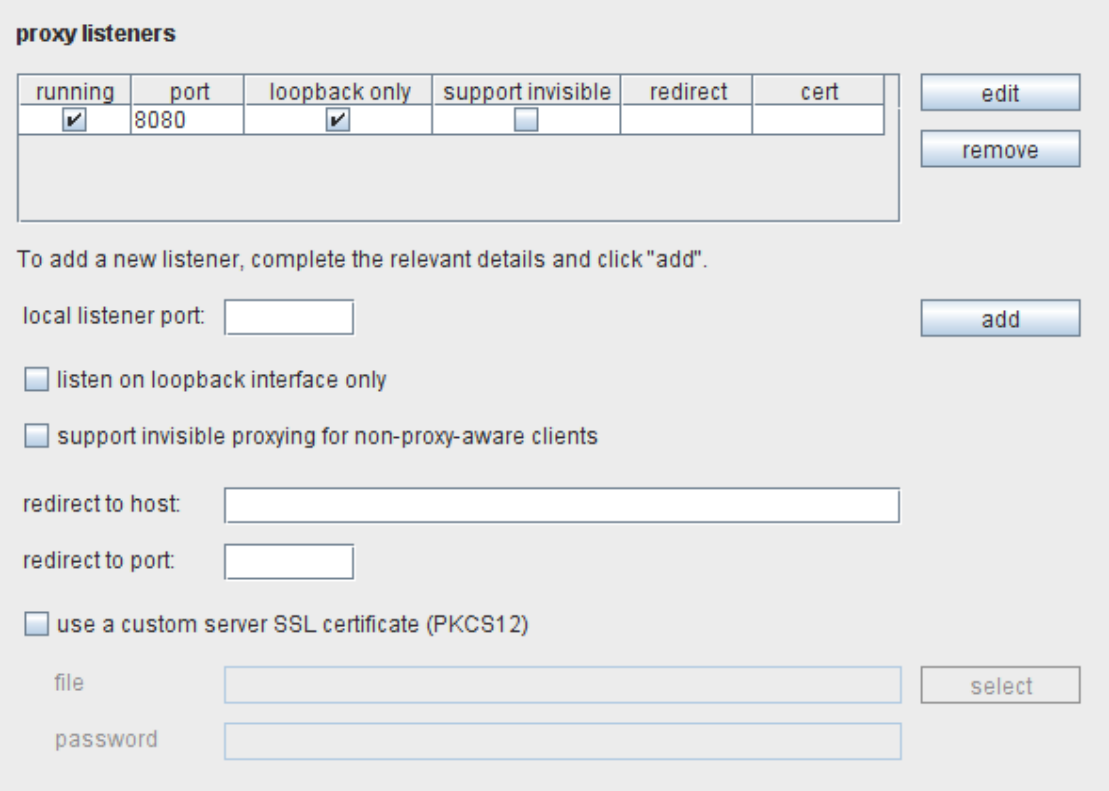
save item 这个功能让你指定一个文件，把选中的请求和响应以 XML 的格式保存到这个文件，这里面包括所有的元数据如：响应的长度，HTTP 的状态码以及 MIME 类型。

don't intercept 通过这些命令可以快速地添加拦截动作的规则来阻止拦截到的消息，这些消息和当前的消息有着相同的特征(如远程主机，资源类型，响应编码)。

do intercept 仅对请求有效，这允许你可以对当请求和响应的进行强制拦截。
convert selection 这些功能让你能够以多种方案对选择的文本进行快速的编码和解码。
URL-encode as you type 如果这个选项被打开，你输入的像&和=这样的符号会被等价的 URL 编码代替。
拦截选项卡里包含了一个可以用来快速切换拦截开关状态的切换按钮。如果显示“intercept is on”，则消息会被自动拦截或者按照在选择卡里配置的拦截规则进行转发。如果显示“intercept is off”，消息不会被拦截。

项目选项卡(Options tab)

这个选项里包含了多种控制 Burp 代理行为的配置选项，就像下面描述的那样。



Burp 代理允许你定义多个监听点。每个监听点在你电脑上开启一个端口并等待浏览器来连接。默认情况下，Burp 代理在端口 8080 上开启一个监听点，但是你可以修改这个监听点，并且可以添加多个你需要的其他监听点。对于每个监听点，你都可以上面那样来配置他们的一些属性。
local listener port 这是一个本地计算机上的一个端口，开启它是用来监听进入的连接。你应该把浏览器的代理服务器设置为主机 127.0.0.1 和这个端口。
listen on loopback interface only 这里控制着监听点是只绑定在 loopback 接口上还是整个网络接口上。注意：如果这个选项没被选中，其他的电脑就也可以连接这个监听点了。这可能会让他们通过你的 IP 地址进行出站连接，以及访问代理历史记录内容的敏感数据，如登陆认证。你只有处在可信任的网络的时候才能取消这个选项。
support invisible proxying for non-proxy-aware clients 如果你使用的是标准浏览器，你就不应该选上这个项。如果你的目标应用程序使用了一个在浏览器外的运行厚客户端组件或者一个使它自己的 HTTP 请求独立于浏览器之外的框架，这时你会发现这个选项是有用的了。在这

种情况下，通过重定向客户端的请求降低网络堆栈(如给主机文件添加一个条目，或者改变路由器配置)，这样你会有效地强制客户端连接 Burp。然而，这些客户端处理的请求可能和 web 代理上通常是的类型不同。

代理上的请求类型是像这样的：

GET http://myapp.com/foo.php HTTP/1.1

Host: myapp.com

然而无代理的请求是这样的：

GET /foo.php HTTP/1.1

Host: myapp.com

通常情况下，web 代理需要接收请求的第一行的完整 URL，这样用于确定请求是发送到哪个域的主机(他们不会，如果他们遵守规范，就会把主机消息头作为发送的目标域)。为了让 Burp 代理兼容客户端发送的无代理类型的请求，你需要选上“support invisible proxying”。当你这样做了后，如果 Burp 接收到无代理类型的请求，它会解析出主机的消息头，并把这个作为请求的发送的目标域主机。

redirect to host/port 通常你应该把这些选项留为空白。如果配置了他们，Burp 代理就会把所有的请求转发到这个指定的主机端口，忽视浏览器发出这个请求的目标。注意如果你使用这个选项，你可能也需要配置 match and replace 规则为请求重新指定主机消息头，如果你把请求重定向到的服务器期待一个和浏览器发出的不同的消息头。

server SSL certificate 这个选项让你配置一个提交给浏览器的服务器 SSL 证书。正确地使用这个选项能解决一些使用拦截代理产生的一些 SSL 问题。通过查看服务器 SSL 证书帮助的所有细节来了解怎样来使用这个选项。

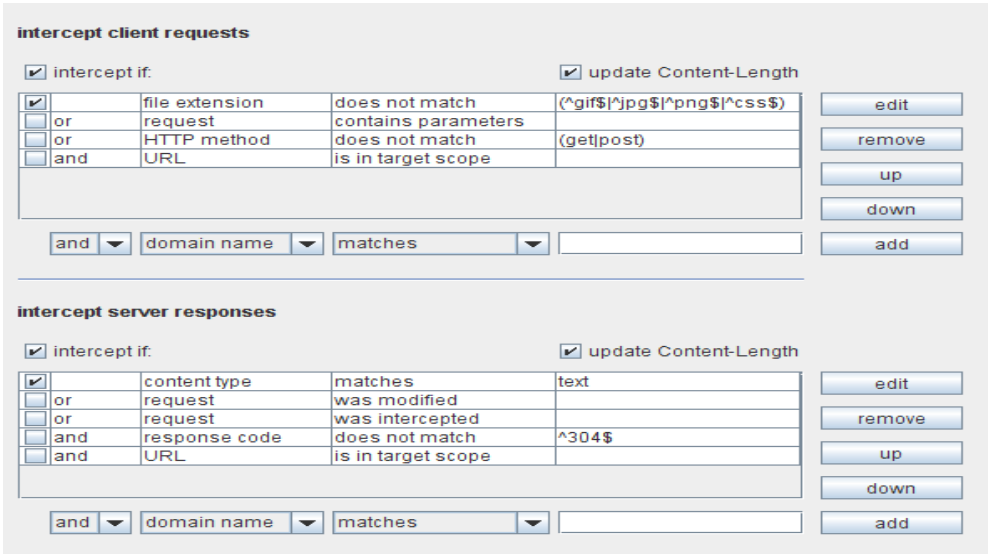
Note 默认情况下，安装时，Burp 会创建一个特殊的自签名式的 CA 证书，把它保存在你的电脑上，当 Burp 启动时使用它。每当你连接一个 SSL 保护的站点，Burp 就会搜集这个主机上的服务证书，并用 CA 证书来签署。为了充分地利用这个功能，你可以以信任的方式来安装 Burp 的 CA 证书，于是在接收每个主机的证书时，不会出现警告。

有时，你希望能在 Burp 上使用一个自定义的 SSL 证书。你可以在 OpenSSL 中使用以下命令来创建一个自定义的证书(名字“foo.crt”)，这个名字是自己选的：

openssl genrsa 1024 > foo.key

openssl req -new -x509 -nodes -sha1 -days 7300 -key foo.key > foo.crt

openssl pkcs12 -export -out foo.p12 -in foo.crt -inkey foo.key -name "your name"



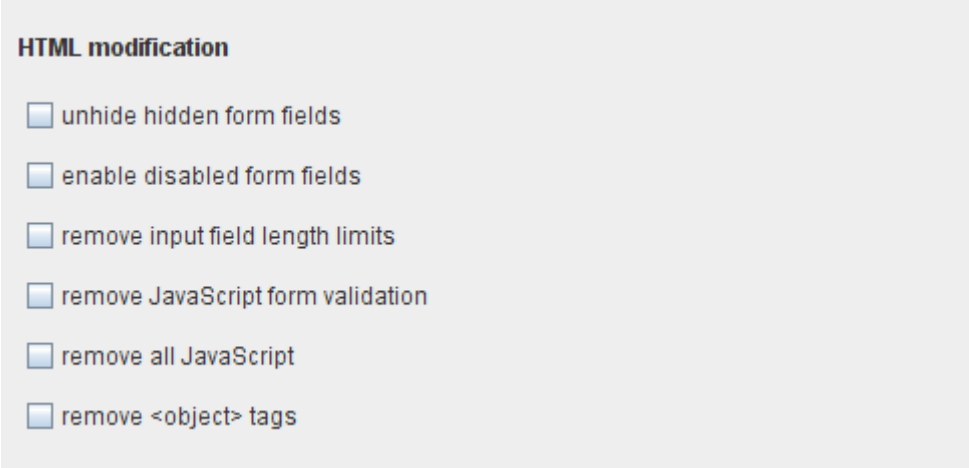
这个面板里允许你使用细微拦截规则来配置这个管理请求和响应的拦截。复选框“intercept if”控制着是否一些清求和响应该被拦截。如果选择了 1 到 2 个框，按照表格里的活动规则相关的信息就会被拦截。根据每条规则左边的复选框，决定这些规则是有效还是停用。通过右边的按钮，可以对这些规则进行编辑，删除，添加和加载。

几乎能对任何消息属性的规则进行配置，包括域名，IP 地址，协议，HTTP 方法，URL，资源类型，参数，cookies，消息体/消息体内容，响应编码，内容类型和 HTML 页面标题。你配置的这个规则仅仅是用来拦截目标范围的 URL 项。可以使用正则表达式来为每个属性设置复杂的匹配条件。通过使用布尔运算符 AND 和 OR 来融合规则。按从左到右的逻辑顺序处理规则表达式，在范围内的每个运算符就是这样的：

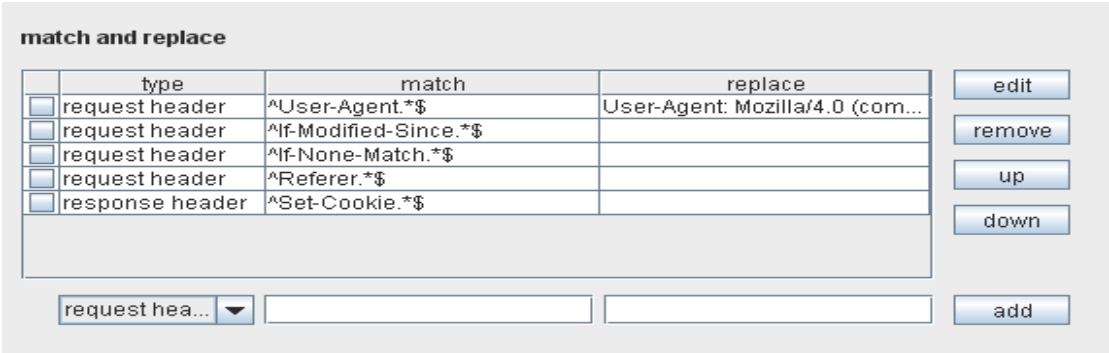
(cumulative result of all prior rules) AND/OR (result of current rule)

所有生效的规则会做到每条消息上，在使用过最后一个生效的规则的结果在后台决定了这个消息是被拦截还是转发。

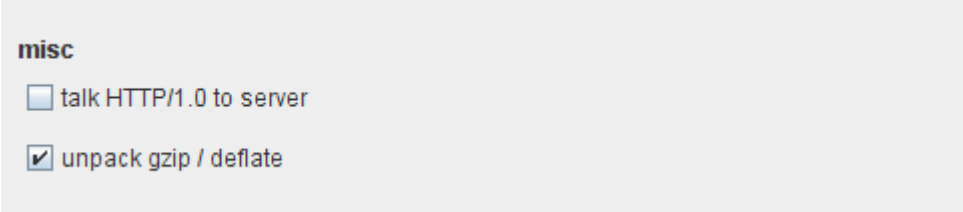
当用户修改请求和响应后，“update Content-Length”复选框控制着 Burp 代理是否更新他们消息头的长度。如果选中，Burp 代理会自动重新计算被修改的 HTTP 消息体的长度，并为 HTTP 消息头设置正确的值。当 HTTP 消息体被修改时，这个功能显的非常必要。HTTP 规范和大多数的 web 服务器一样，需要在消息头内容长度上提交的 HTTP 消息体长度的正确值。如果没有指定正确的值，服务器或浏览器收到消息会产生错误，或者收到不完整的消息，或者会无限期地等待接收下面的数据。



通过自动地回写到应用程序响应的 HTML，你可以使用这些选项来完成许多任务。Unhiding hidden fields 使你能够直接编辑浏览器的值，而不是通过拦截后面的请求。和 enabling disabled fields，removing length limitations 一样。Disabling JavaScript and OBJECT tags 在测试中提供一种快速的方式式客户端逻辑失效(注意这个功能并不是设计用来进行 NoScript 方式的安全防御的)。



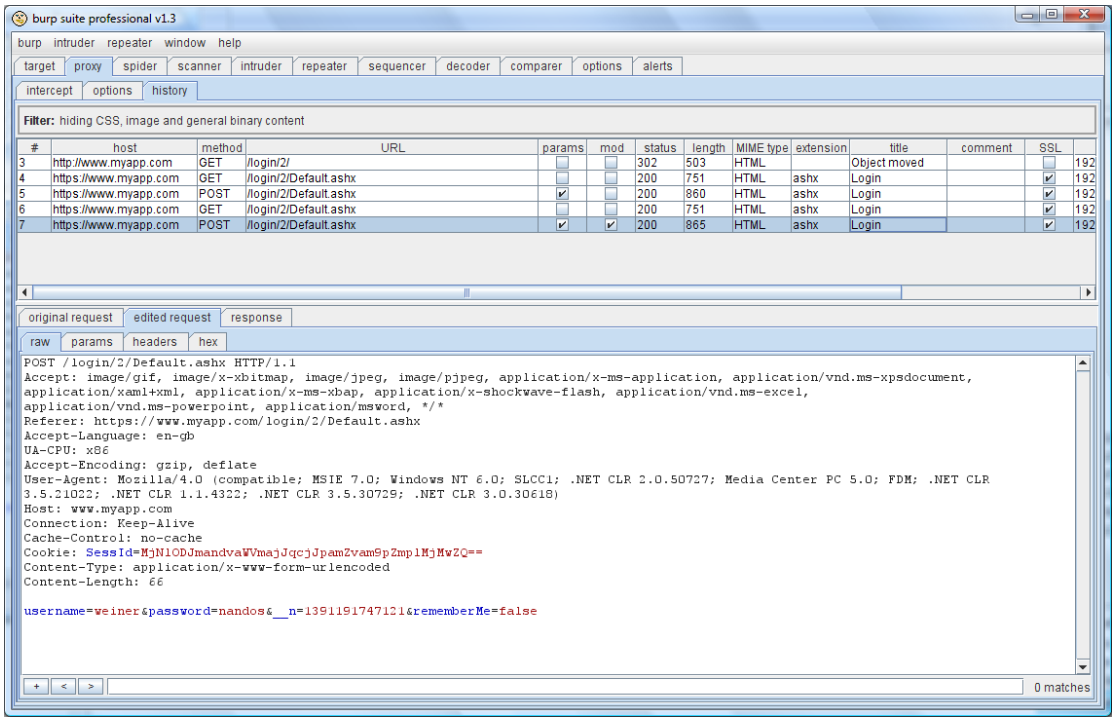
match and replace 这些选项配置 Burp 代理执行基于正则表达式的匹配准则和 HTTP 请求和响应的消息头和消息体的替换。对于选中的每个规则，都会使用正则表达式来测试消息头或体是否匹配，并用指定的字符串替换掉匹配的部分。
对于整个消息头来说，如果整个消息头都匹配，并且替换的字符串为空，则删除这个消息头。如果没有匹配这些指定的表达式，则替换字符串会作为一个新消息头添加进来。这个功能在自动进行某种应用程序攻击时会非常有用，如操纵 cookies 和 URL 字符段查询。



talk HTTP/1.0 to server 这里控制着 Burp 代理是否要使用 1.0 版本的 HTTP 来强行连接目标服务器。默认设置是使用浏览器的 HTTP 版本。对于使用 1.0 和 1.1 版本的大多数浏览器，Burp 代理都能成功地通过测试。然而对于一些传统的服务器或应用程序使用 1.0 版本才能正常，因此需要在这里来指定它。
unpack gzip / deflate 一些浏览器会从服务器上接收到 gzip 和 deflate 的压缩内容。为了能够查看和修改这些内容，你需要对他们进行解压缩。这个选项就是控制 Burp 代理是否自动进行解压缩。

历史记录选项(History tab)

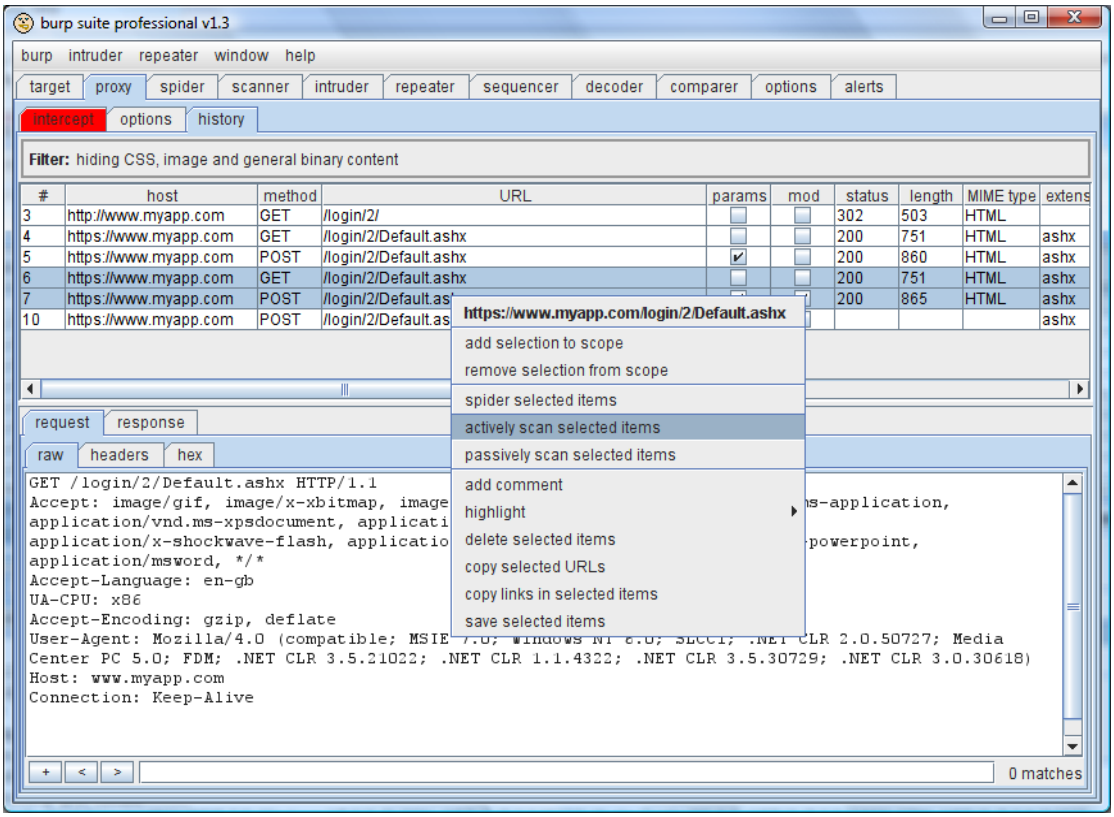
这个选项是来显示所有请求产生的细节，显示的有目标服务器和端口，HTTP 方法，URL，以及请求中是否包含参数或被人工修改，HTTP 的响应状态码，响应字节大小，响应的 MIME 类型，请求资源的文件类型，HTML 页面的标题，是否使用 SSL，远程 IP 地址，服务器设置的 cookies，请求的时间。当你把拦截关闭时，这个选项是非常有用的。它允许你在不适用拦截的情况下进行浏览时，仍能够监视应用程序传输的关键细节。



你可以通过单击历史记录表里的列标题对表格里的列里的内容进行排序(或者用 shift+单击进行倒置排序)。例如，你想让历史记录表向上增长，即最近产生的项在表的顶端，则你可以使用 shift+点击最左边的列来显示请求的序号。另外，如果你想按照内容类型列出请求，你就点击“MIME type”这一列。

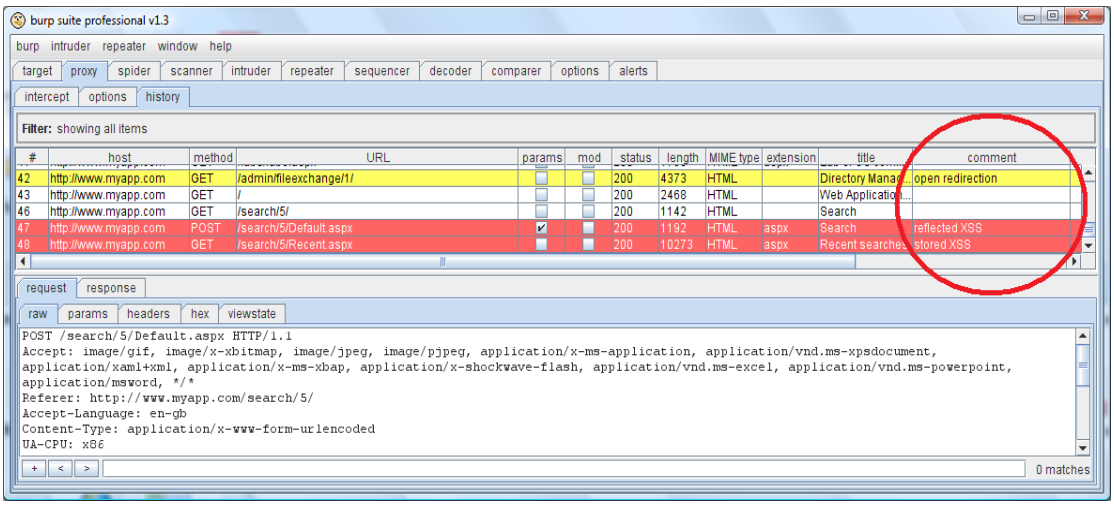
下面的这个历史表格是一个预览表格。如果你选择了历史记录里的一个项，相关的请求和响应就会显示在窗格的下部。如果这些请求或响应时通过手动或者你配置的规则修改的，修改后的项会显示在原来项的旁边。

在历史记录表里，右击一个或多个选项，就会显示一个上下文菜单让你执行一些操作，包括修改目标范围，把这些选项发送到其他 Burp 工具，或者删除这些项：

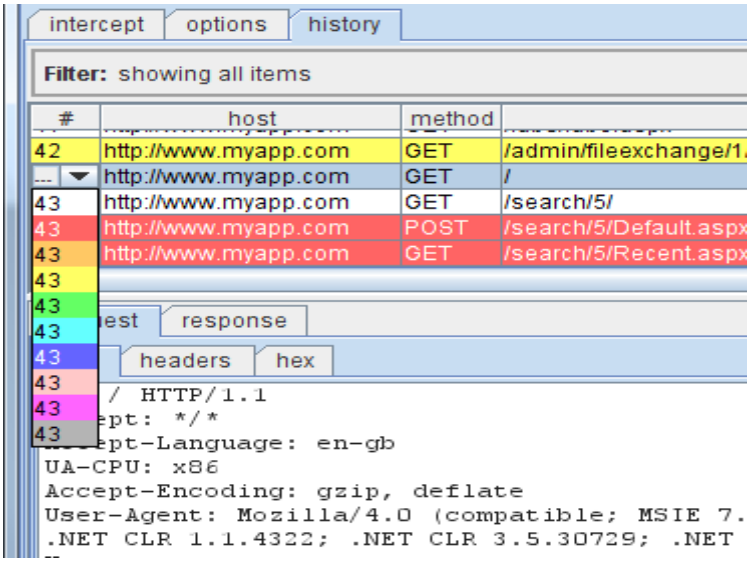


[专业版]你可以通过上下文菜单使用许多定制工具，如搜索脚本与注释，分析目标 web 站点，调度任务等等。

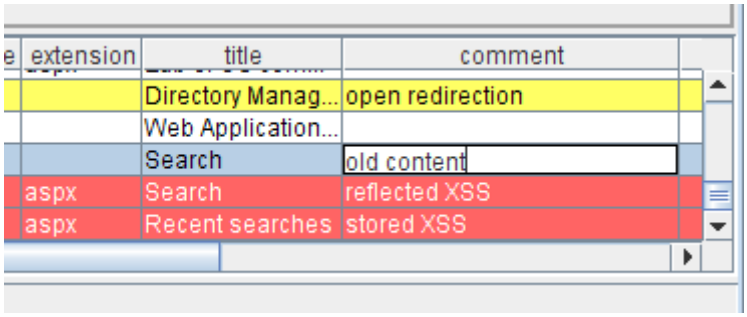
你可以通过添加注释或加亮，来注释一个或多个选项：



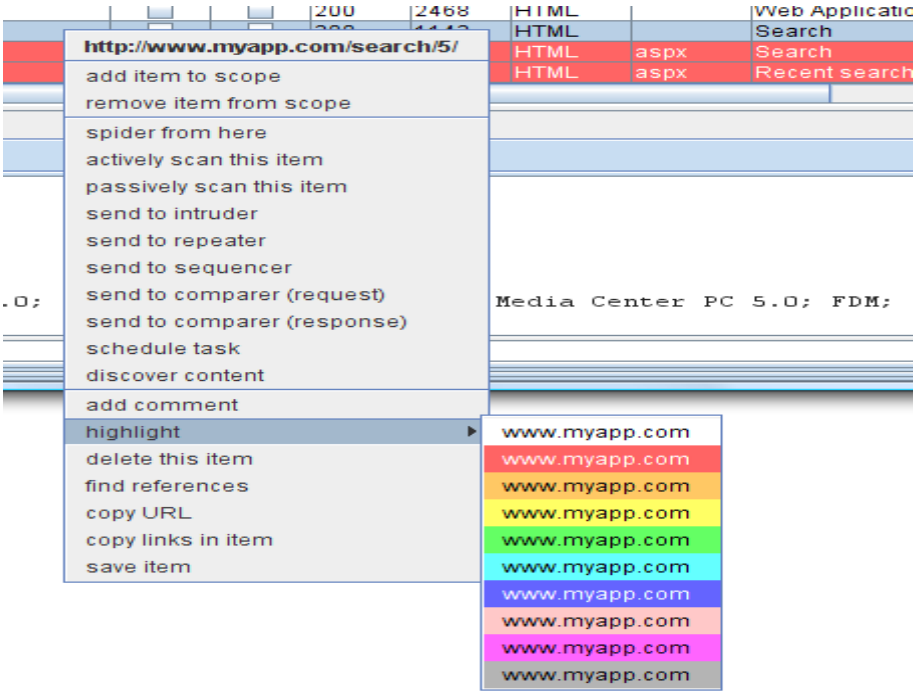
你可以通过最左边的列里的下拉菜单来加亮单个选项：



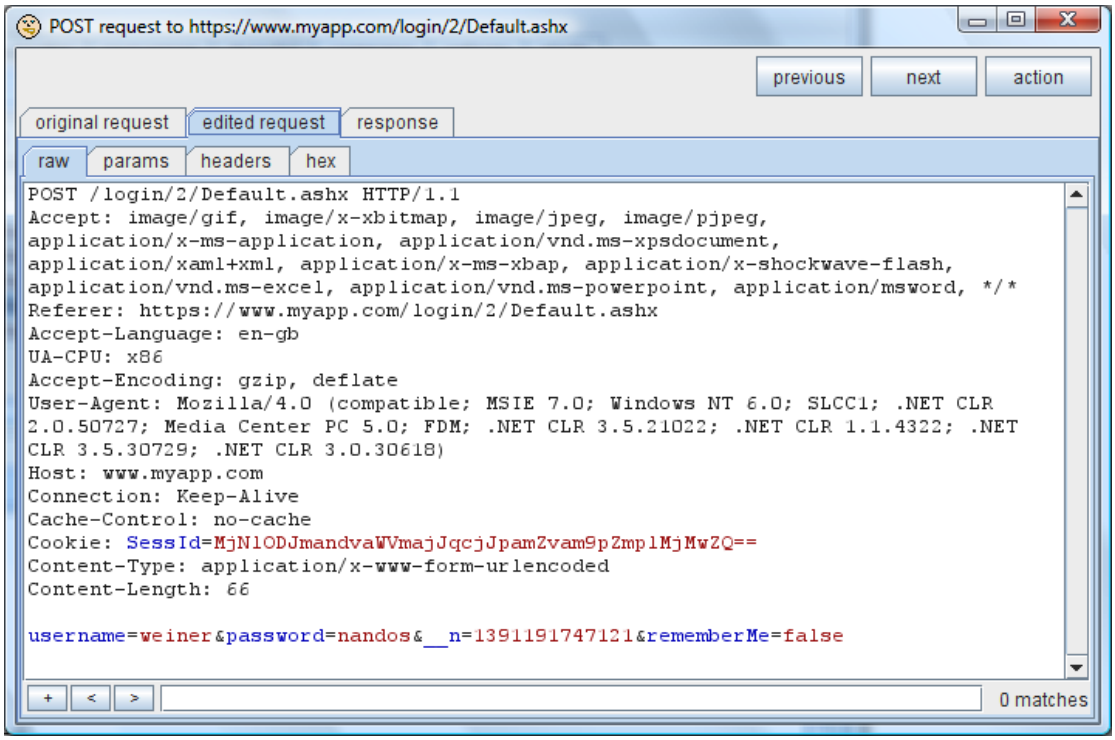
通过双击来就地注释单个选项，并编辑单元格：



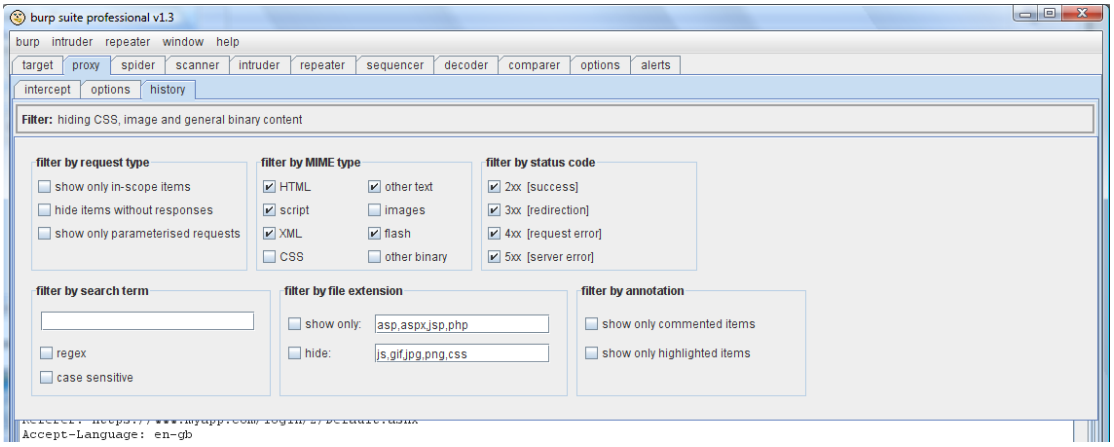
另外，如果你想一次注释多个选项，可以选中相关的项，然后使用上下文菜单进行注释或加亮：



当你把感兴趣的项进行了注释后，可以使用列排序和过滤显示来快速地找到这些项。
除了使用预览表格里的视图请求，在弹出窗口里，可以通过对表格里的任意项进行双击来显示请求和响应。



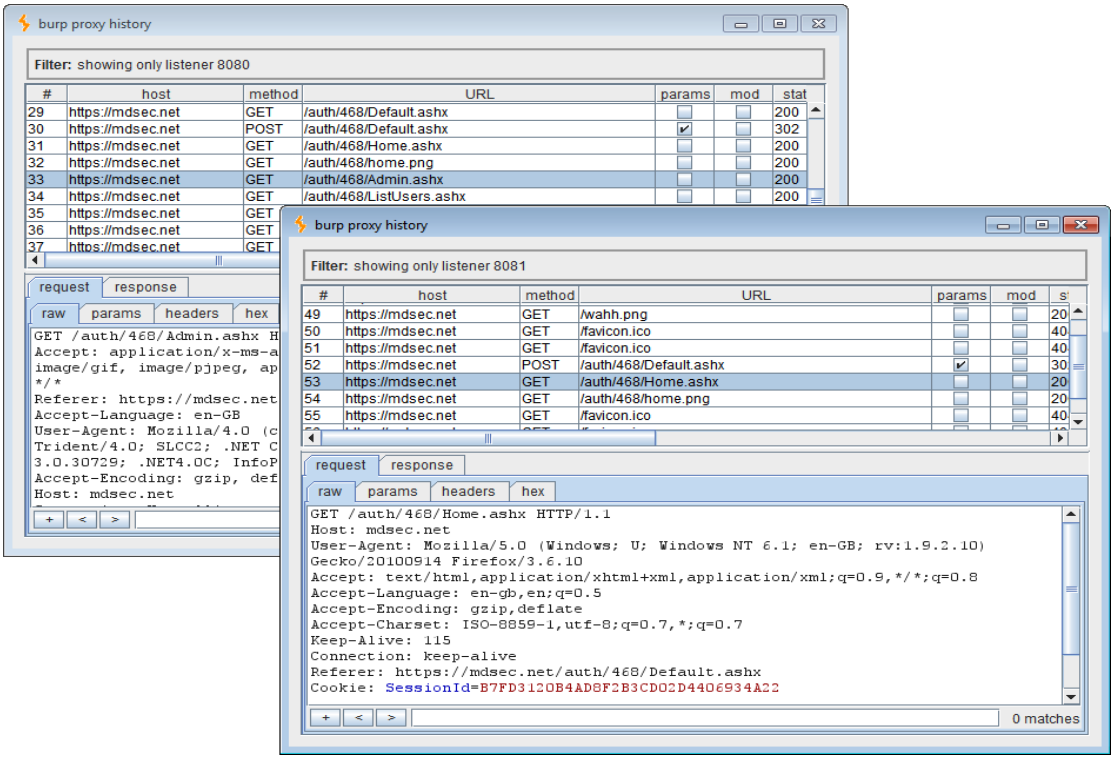
显示在历史记录表里的内容实际上是一个进入底层数据库的视图，你可以通过配置过滤器来确定哪些顶层的数据项显示在表格里。有效应用程序包含了大量的内容，如图像，CSS 等，这些有利于从视图上隐藏的。AJAX 应用程序产生大量相似的异步请求，你可能会想把他们从视图上过滤出来来查看一些感兴趣的项。在这个历史记录表的顶部有一个过滤栏。单击会有一个弹出窗口，让你来精准地配置显示哪些内容在表格里：



你可以选择只显示那些有参数的请求，或者在当前目标范围内的项，或者接收到的响应。你可以通过 MIME 类型，HTTP 状态码，文件扩展名进行过滤。如果你设置过滤隐藏了一些项，他们不会被删除，只是隐藏了，通过解除相关的过滤就可以重新显示出来了。
[专业版]你可以为过滤器指定一个搜索项，让它显示那些只包含请求或响应的项，或者显示需要的用户添加的注释。
和过滤器一样，你可以在历史记录表里通过选中一个或多个项，然后在上下文菜单里选择“delete”来永久地删除这些项。

在一些情况下，这对显示更多进入底层历史记录数据的视图很有用，并对每个视图采用不同的过滤器。例如，在测试访问控制时，你可以以不同用户的身份登录到应用程序，以及想单独地查看在用户上下文产生的请求序列。你可以通过代理记录的上下文菜单上的“show new history window”选项来打开添加的代理记录视图。然后你可以为每个记录窗口配置过滤器来显示你想看到的请求。

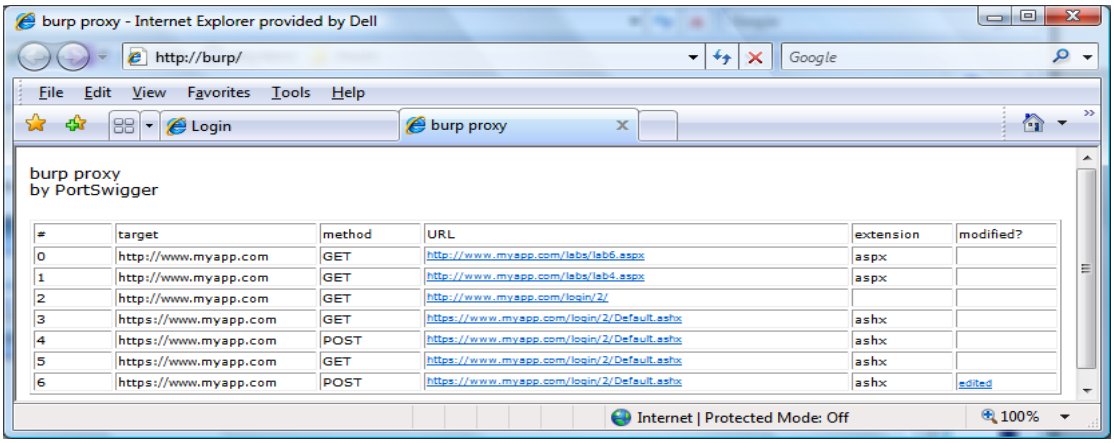
为了使用这个功能进行测试访问控制，你需要对测试的每个用户上下文使用单独的浏览器，为每个浏览器在 Burp 上创建单独的代理监听点(你需要更新每个浏览器里的代理设置并把它指向相关的监听点)。然后对于每个浏览器，在 Burp 上你打开一个单独的代理记录窗口，并设置过滤器来显示和代理监听点的端口相关的请求。当你用每个浏览器访问应用程序时，每个记录窗口里只显示和用户上下文相关的项：



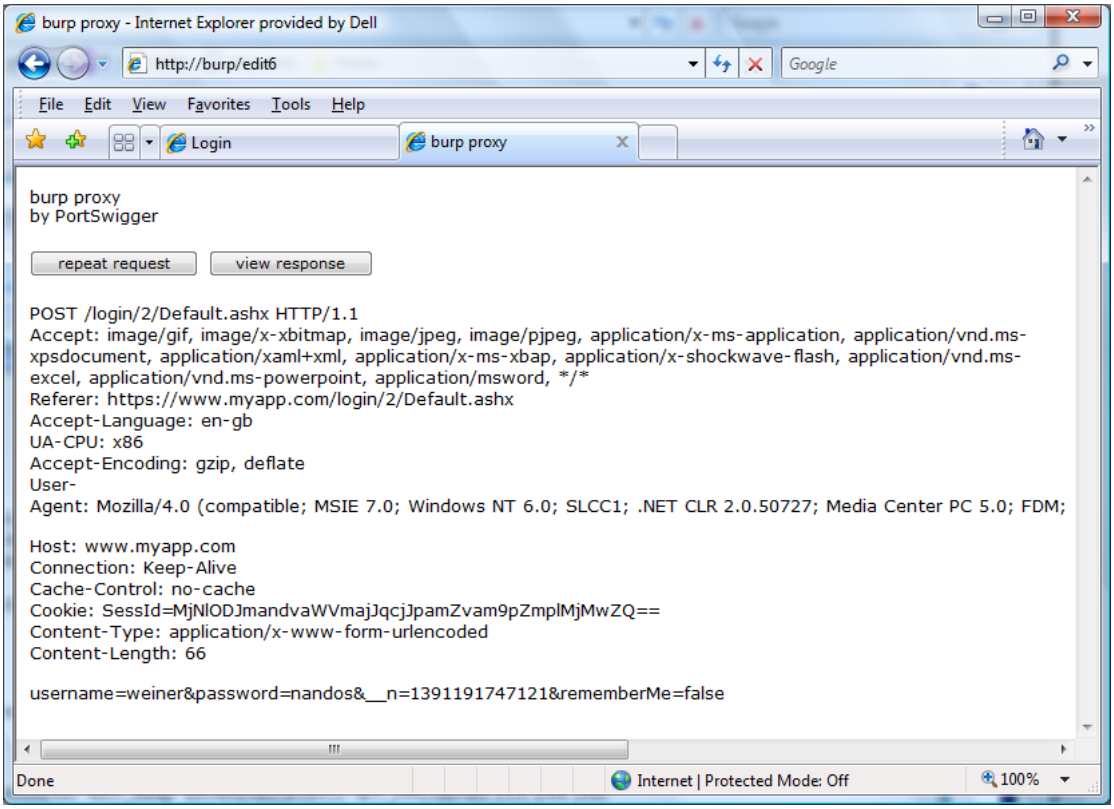
浏览器控制(In-browser controls)

除了使用主界面，你还可以直接使用浏览器来控制 Burp 代理。

在浏览器里输入 Http://burp 就可以访问所有的代理记录。这个记录会以一个表格的形式显示出来。内容有：目标服务器和端口号，HTTP 方法，URL，文件扩展名，请求是否被修改过：



单击”URL”列的条目显示出本地的请求，单击”modified?”列的条目显示出相关修改过的请求。



当完全地显示出一个单独的请求时，可以通过” repeat request”按钮重发这个请求。依据当前配置的拦截规则，可以通过 Burp 代理的修改来显示这个请求。当浏览器接收到服务器对重发请求作出的响应时，前进浏览一下，就可以像平常一样继续了。

如果可用，你可以在浏览器里单击” view response”按钮来查看本地响应。这会让 Burp 代理退回到从服务器里接收到的那个响应，Burpd 代理既不会显示请求也不会显示响应来修改。注意当浏览器从 Burp 代理接收到保存过的响应时，这会导致浏览器加载添加的请求。Burp 代理会以平常的方式处理这些请求，将不会返回以前保存的数据。

Burp Spider Help

Burp 网络爬虫是什么？(What is Burp Spider?)

Burp Spider 是一个映射 web 应用程序的工具。它使用多种智能技术对一个应用程序的内容和功能进行全面的清查。

Burp Spider 通过跟踪 HTML 和 JavaScript 以及提交的表单中的超链接来映射目标应用程序，它还使用了一些其他的线索，如目录列表，资源类型的注释，以及 robots.txt 文件。结果会在站点地图中以树和表的形式显示出来，提供了一个清楚并非常详细的目标应用程序视图。

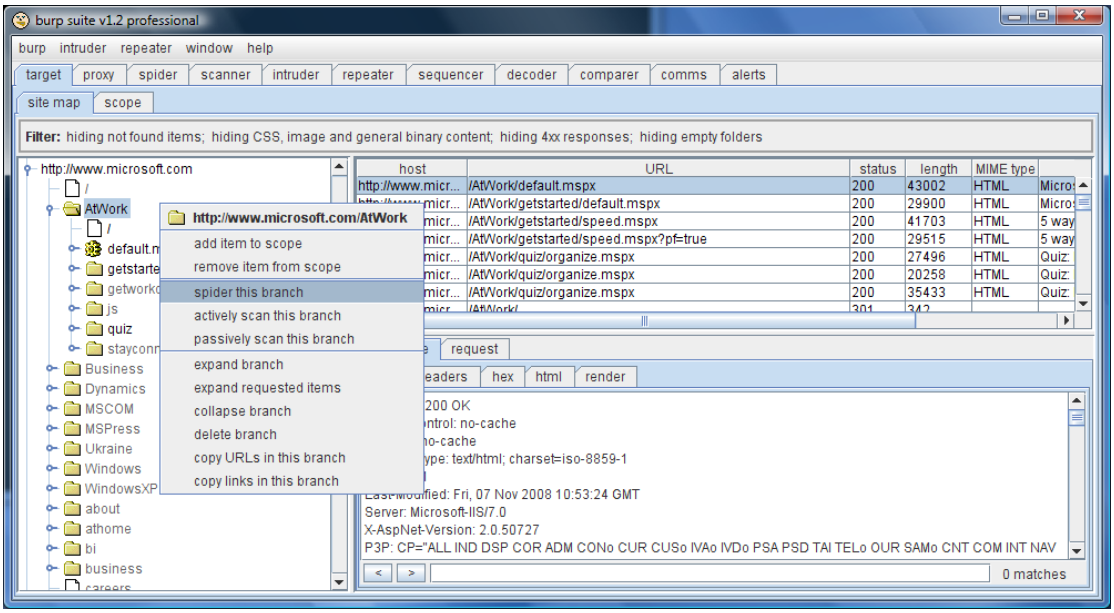
Burp Spider 能使你清楚地了解到一个 web 应用程序是怎样工作的，让你避免进行大量的手动任务而浪费时间，在跟踪链接，提交表单，精简 HTNL 源代码。可以快速地确人应用程序的潜在的脆弱功能，还允许你指定特定的漏洞，如 SQL 注入，路径遍历。

使用 Burp Spider(Using Burp Spider)

要对应用程序使用 Burp Spider 需要两个简单的步骤：

1.使用 Burp Proxy 配置为你浏览器的代理服务器，浏览目标应用程序(为了节省时间，你可以关闭代理拦截)。

2.到站点地图的"target"选项上，选中目标应用程序驻留的主机和目录。选择上下文菜单的"spider this host/branch"选项。



你也可以在任何 Burp 工具的任意请求或响应上使用上下文菜单上选择" spider this item".

当你发送一个站点地图的分支来 spidering，Spider 会首先检查这个分支是否在定义好的 spidering 的范围内。如果不是，Burp 会提示你是否把相关的 URL 添加到范围里。然后，Burp 开始 spidering，并执行下面的操作：

在分支上，请求那些已被发现的还没被请求过的 URL。

在分支上，提交那些已被发现但提交 URL 错误的表单。

重复请求分支上的先前收到的状态码为 304 的项，为检索到一个应用程序的新(未进入缓存)

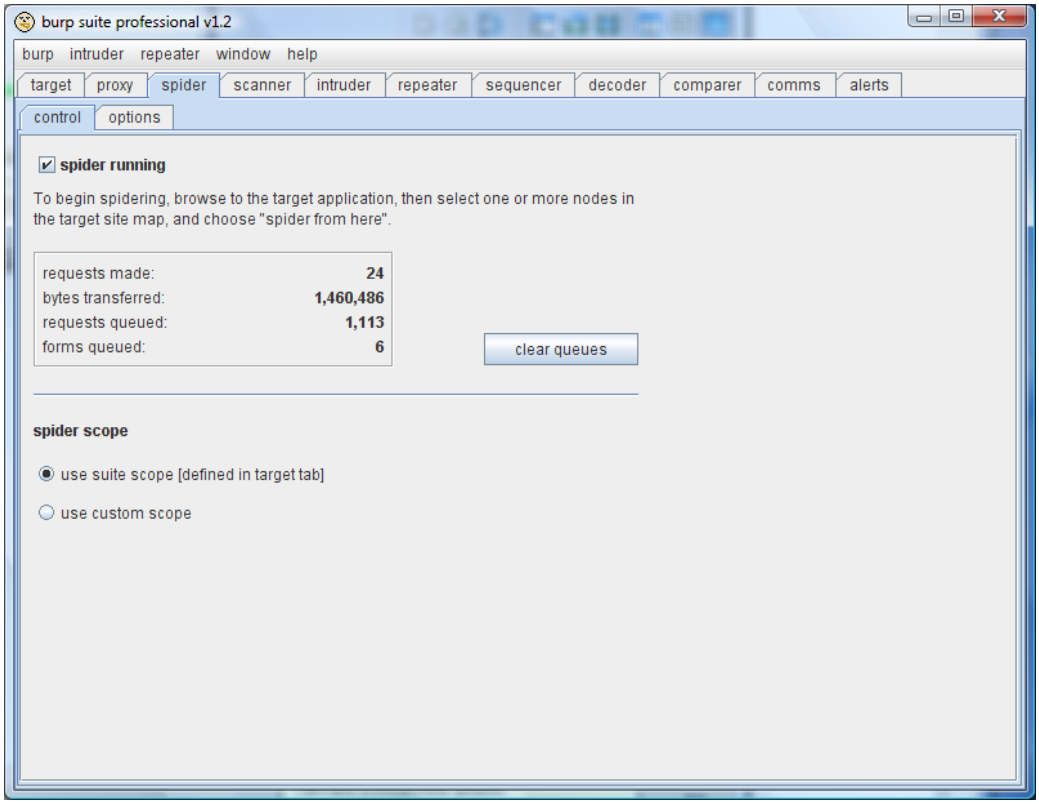
副本。
对所有的检索到内容进行解析以确认新的 URL 和表单。
只有发现新内容就递归地重复这些步骤。
继续在所有的范围区域内 spidering，直到没有新内容为止。

注意 Spider 会跟踪任何在当前定义的 spidering 范围内的 URL 链接。如果你定义了一个比较宽的目标范围，并且你只选择了其中的一个分支来 spidering，这时 Spider 会跟踪所有进入到这个比较宽的范围内的链接，于是也就不在原来的分支上 spider。为了确保 Spider 只在指定分支内的请求上，你应该在开始时，就把 spidering 范围配置为只在这个分支上。

你应该小心地使用 Burp Spider。在它的默认设置上，Spider 会在 spidering 范围内使用默认输入值，自动地提交任意表格，并且会请求许多平常用户在只使用一个浏览器不会发出的请求。如果在你定义范围的 URL 是用来执行敏感操作的，这些操作都会被带到应用程序上。在你完全地开始自动探索内容之前，使用浏览器对应用程序进行一些手动的映射，是非常可取的。

控制选项(Control tab)

这个选项是用来开始和停止 Burp Spider，监视它的进度，以及定义 spidering 的范围。

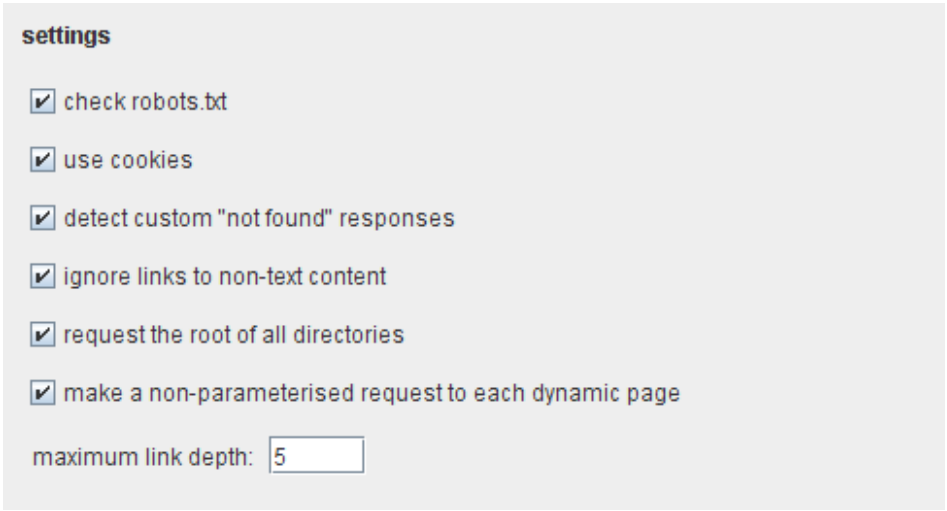


Spider running 这个是用来开始和停止 Spider。Spider 停止后，它自己不会产生请求，但它会继续处理通过 Burp Proxy 的响应，并且在 spidering 范围内的新发现的项都会送入请求队列里，当 Spider 重新启动时，再来请求。
这里显示的一些 Spider 进度的指标，让你能看到剩余的内容和工作量的大小。
Clear queues 如果你想改变你工作的优先权，你可以完全地清除当前队列的项目，来让其他的项目加入队列。注意如果被清除的项目如果还在范围内并且 Spider 的分析器发现有新的链接到这个项目，那么它们还会加入队列

Spider scope 在这个面板里，你能精确地定义 Spider 的请求范围。如果你已经根据当前目标细节配置好了 Suit-wide 的目标范围，你可以丢弃 Spider 默认设置的活动范围。如果你需要使用一个定义的不同的 Spider 范围，则选择” use custom scope”。一个进一步设置面板会出现，和 Suite-wide 范围面板的功能相似。如果你使用自定义范围并向 Spider 发送不在范围内的项，则 Burp 会自动更新这个自定义的范围而不是 Suite 范围。

项目选项卡(Options tab)

这个选项里包含了许多控制 Burp Spider 动作的选项，如下描述。这些设置在 spider 启动后还可以修改的，并且这修改对先前的结果也是有效的。例如，如果增加了最大链接深度，在以前的最大链接深度外的链接如果满足现在的条件，也会加入到请求队列里。



check robots.txt 如果这个被选中，Burp Spider 会在所有范围内的域名上请求和处理 robots.txt 文件。通过 robots 排除协议来控制互联网上的 spider-like 代理的行为。注意 Burp Spider 不会确认 robots 排除协议。Burp Spider 会列举出目标应用程序的所有内容，请求所有在范围内的 robots.txt 条目。

use cookies 如果这个被选中，Burp Spider 会处理服务器响应中的 Set-Cookie 指令，会在后面的请求中向同一域名提交任意接收到的 cookie。当 spidering 一个 web 应用程序时，并且这个应用程序保持着服务器端的状态时，这个选项就显得非常有必要了。

detect custom "not found" responses 如果没有找到请求的资源，HTTP 协议要求 web 服务器返回一个 404 的状态码。然而许多 web 应用程序会返回自定义的”not found”页面，用来代替不同的状态码。使用这个选项就能阻止在映射站点内容时产生的这种误报。Burp Spider 通过请求每个域名里的一些不存在的资源就能探测到这种自定义的”not found”响应。通过编译出的这个指纹来识别其他请求的”not found”响应。

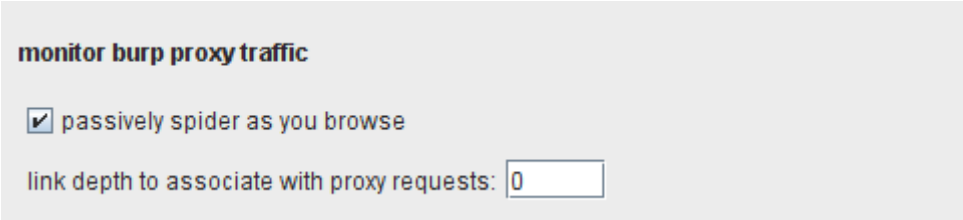
ignore links to non-text content 常常需要推断出在 HTML 上下文里链接到特殊资源的 MIME 类型。例如，带有 IMG 标记的 URL 会返回图像；那些带有 SCRIPT 标记的会返回 JavaScript。如果这个选项被选中，Spider 不会请求在这个上下文出现的出现的非文本资源。使用这个选项，会减少 spidering 时间，降低忽略掉感兴趣内容的风险。

request the root of all directories 如果这个选项被选中，Burp Spider 会请求所有已确认的目标范围内的 web 目录，除了那些目录里的文件。如果在这个目标站点上目录索引是可用的，这选项将是非常的有用。

make a non-parameterised request to each dynamic page 如果这个选项被选中，Burp Spider 会

对在范围内的所有执行动作的 URL 进行无参数的 GET 请求。如果期待的参数没有被接收，动态页面会有不同的响应，这个选项就能成功地探测出添加的站点内容和功能。

maximum link depth 这是 Burp Suite 在种子 URL 里的巡览"hops"的最大数。0 表示让 Burp Suite 只请求种子 URL。如果指定的数值非常大，将会对范围内的链接进行无限期的有效跟踪。

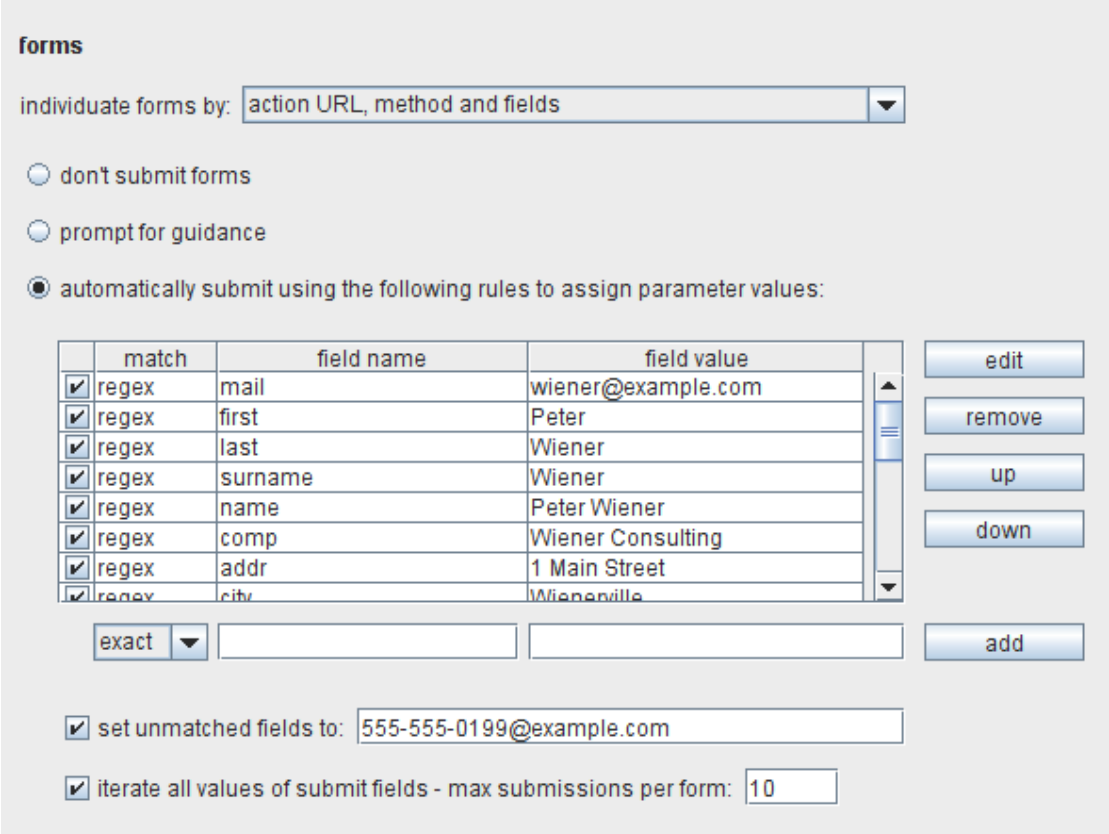


这些选项控制着 Burp Proxy 与 Burp Spider 之间的接口，它允许"被动"的 spidering web 应用程序，控制浏览器。

passively spider as you browse 如果这个选项被选中，Burp Suite 会被动地处理所有通过 Burp Proxy 的 HTTP 请求，来确认访问页面上的链接和表格。使用这个选项能让 Burp Spider 建立一个包含应用程序内容的详细画面，甚至此时你仅仅使用浏览器浏览了内容的一个子集，因为所有被访问内容链接到内容都会自动地添加到 Suite 的站点地图上。

link depth to associate with proxy requests 这个选项控制着与通过 Burp Proxy 访问的 web 页面有关的" link depth"。为了防止 Burp Spider 跟踪这个页面里的所有链接，要设置一个比上面选项卡里的" maximum link depth"值还高的一个值。

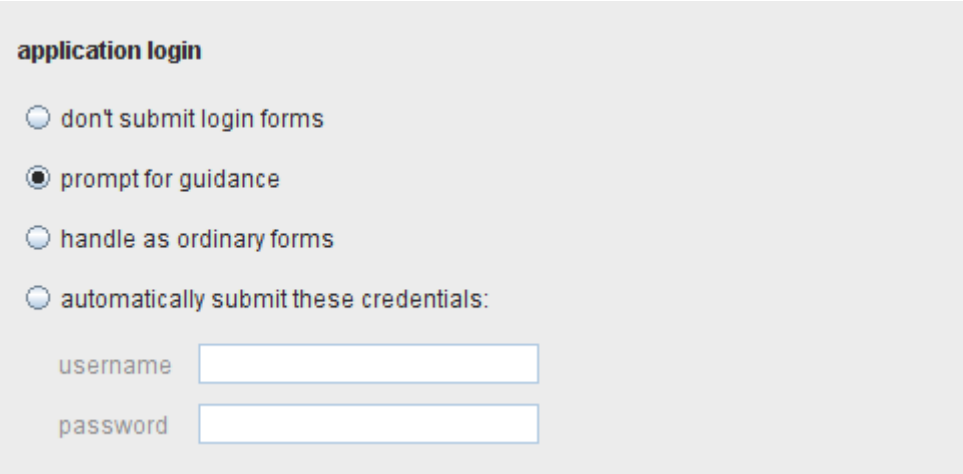
注意：在早期的 Burp Spider 版本里，这里包含的选项是控制在 Proxy 里的请求和响应的 cookies 上，怎样更新 spider cookie jar。现在这些配置已被删除了，你可以使用 suite-wide 里的会话处理来代替。



individuate forms 这个选项是配置个性化的标准(执行 URL，方法，区域，值)。当 Burp Spider 处理这些表格时，它会检查这些标准以确认表格是否是新的。旧的表格不会加入到提交序列。
do not submit 如果选中这个，Burp Spider 不会提交任何表单。
prompt for guidance 如果选中这个，在你提交每一个确认的表单前，Burp Suite 都会为你指示引导。这允许你根据需要在输入域中填写自定义的数据，以及选项提交到服务器的哪一个区域，以及是否遍历整个区域。
automatically submit 如果选中这个，Burp Spider 通过使用定义的规则来填写输入域的文本值来自动地提交范围内的表单。每一条规则让你指定一个简单的文本或者正则表达式来匹配表单字段名，并提交那些表单名匹配的字段值。可以为任意不匹配的字段指定默认值。

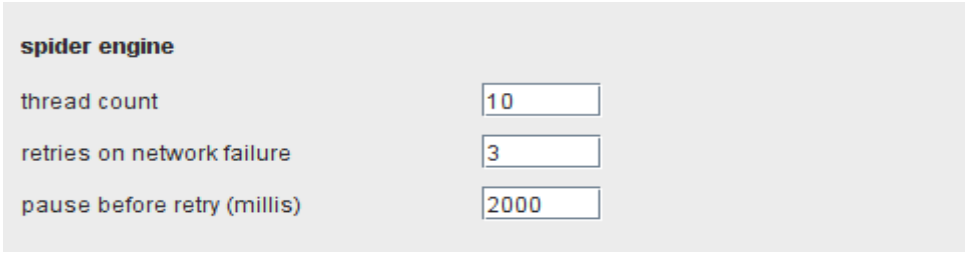
在应用程序通常需要对所有输入域都是有效格式的数据的地方，如果你想通过登记表单和相似功能自动地 spider，则这个选项会非常有用。在自动地把表单数据提交到广阔范围内的应用程序时，Burp 使用一组非常成功的规则。当然，如果你遇到有自己需要提交的特定值的表单字段名时，你可以修改这些或者添加自己的规则。你要小心地使用这个选项，因为提交了表单里的虚假值有时会导致一些不希望看到操作。

许多表单包含了多个提交元素，这些会对应用程序进行不同的操作，和发现不同的内容。你可以配置 Spider 重复通过表单里提交元素的值，向每个表单提交多次，次数低于配置的最大值。

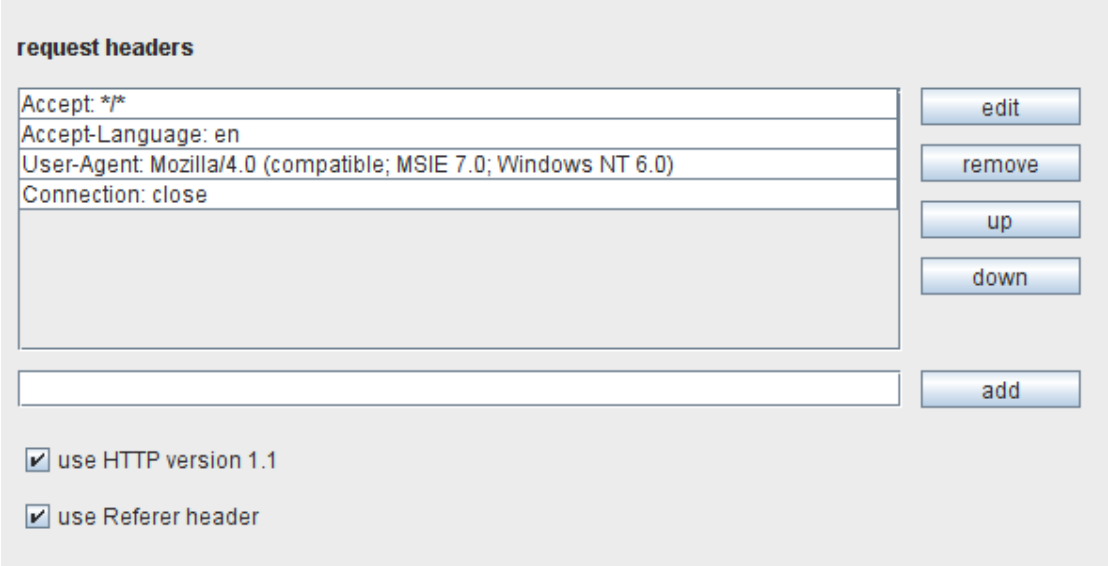


登陆表单在应用程序中扮演一个特殊角色，并且你常常会让 Burp 用和处理平常表单不一样的方式来处理这个表单。使用这个配置，你可以告诉 Spider 在遇到一个表单执行下面 4 种不同操作的一种：
1.如果你没有证书，或者关注 spidering 的敏感保护功能，Burp 可以忽略登陆表单。
2.Burp 能交互地为你提示引导，使你能够指定证书。这时默认设置项。
3.Burp 通过你配置的信息和自动填充规则，用处理其他表单的方式来处理登陆表单。
4.在遇到的每个登陆表单时，Burp 能自动地提交特定的证书。

在最后一种情况下，任何时间 Burp 遇到一个包含密码域的表单，会提交你配置的密码到密码域，提交你配置用户名到最像用户名的字段域。如果你有应用程序的证书，想让 Spider 为你处理登陆，通常情况下这是最好的选项。



这些选项让你来对 Spider 引擎进行微调，这取决于应用程序的性能的影响和你的处理能力和带宽。如果你发现 Spider 运行缓慢，但应用程序运行良好,CPU 的利用率也很低时，你可以增加扫描线程数，来使扫描过程更快。如果你发现有连接错误产生，应用程序开始变慢，或者电脑特别卡，这时你应该降低线程数，或者增加下面的那两个参数。



Request headers 这里允许你自定义 HTTP 消息头，通过配置把它应用到所有的请求中。这对满足个别应用程序的特别需求很有用。例如，当测试一个为移动设备设计的应用程序时，来模拟一个被期望的用户。

use Referer header 如果这个选项被选中，当 Burp Spider 请求的项是从其他页面的链接过来时，就会提交相关的 Referher heards。

Spider 的结果(Spider results)

所有的在 spidering 过程里发现的内容都会被添加到目标站点地图里，其他组件工具也共享这个地图信息。这个地图显示了通过 Spider 和 Proxy 发现的内容的视图树和表格。它让你可以过滤掉视图树里任何不感兴趣的项，并且执行一些其他的操作，如开始扫描漏洞，进一步 spidering，把一些请求发送到其他 Burp 工具上执行自定义攻击。请查询站点地图帮助来得到进一步的信息。

Burp Scanner Help

Burp Scanner 是什么？(What is Burp Scanner?)

Burp Scanner 是一个进行自动发现 web 应用程序的安全漏洞的工具。它是为渗透测试人员设计的，并且它和你现有的手动执行进行的 web 应用程序半自动渗透测试的技术方法很相似。

使用的大多数的 web 扫描器都是单独运行的：你提供了一个开始 URL，单击”go”,然后注视着进度条的更新直到扫描结束，最后产生一个报告。Burp Scanner 和这完全不同，在攻击一个应用程序时它和你执行的操作紧紧的结合在一起。让你细微控制着每一个扫描的请求，并直接反馈回结果。

Burp Scanner 可以执行两种扫描类型：

1. Active scanning 扫描器向应用程序发送大量的伪造请求，这些请求都是有一个基础请求衍生出来的，然后通过分析响应结果来查找漏洞特征。

2.Passive scanning 扫描器不发送他自己的任何新请求，只分析现有的请求和响应的内容，从这些信息中推断出漏洞。

你可以在目标应用程序使用两种不同方式：

1.Manual scanning 你可以发送其他 Burp 工具的一个或多个请求，来对这些特定的请求执行主动或被动的扫描。

2. Live scanning as you browse 你可以配置扫描器来自动执行主动或被动的扫描那些你浏览应用程序时经过代理的请求。

这种自动探测漏洞的方法给渗透测试人员带来了几点好处：

1.通过逐个的请求，能快速可靠地对常规的漏洞进行扫描，这很大程度地减少你的测试精力，还能使你对那些不能进行自动可靠地探测的漏洞直接使用个人经验来判断。

2.每种扫描的结果会被立即显示出来，并通报出在这个请求中包含的其他的测试操作。

3.Burp 避免了其他扫描器的令人沮丧的问题，进行一次自动扫描需要 1 年的时间，并还不能保证扫描是否有效，或者是否遇到了影响扫描效率的问题。

Burp 精准地控制着要扫描的内容，并对扫描结果和应用程序上的广范围的影响进行实时监控，Burp Spider 让你把可靠自动化的优点和人类直观智慧结合起来，常常会得到压倒性的结果。

主动扫描(Active scanning)

在这种扫描模式下，Burp 使用应用程序的一个叫做” base request”单个请求，通过一些方法修改后，来触发一些漏洞存在的迹象。这些被修改过的请求被发送到应用程序，然后分析响应的结果。在许多情况下，根据初步探测的结果，会发送进一步的请求。

这种操作模式会产生大量的恶意的请求，并导致应用程序妥协。你要谨慎地使用这种扫描模式，仅当得到应用程序的所有者允许时，并且警告过他们自动扫描会给他们的应用程序和数据带来影响。如果可能，扫描不用的系统，并在扫描前进行备份。

对应用程序中的已知缺陷的漏洞的自动探测是很可靠的。Burp 的主动扫描能力的是为扫描器能可靠地查找到基于输入的漏洞而设计的。为了避免在其他地方产生的误报，Burp 在他的输出上给了你自信，让你集中精力到那些需要提供人类经验和智慧的工作上。

Burp 主动扫描能确认的问题大体上可分为下面 2 类：

1.在客户端上的输入漏洞，如:跨站点脚本，HTTP 消息头注入，开放重定向。
2.在服务端的输入漏洞，如:SQL 注入，操作系统命令注入，文件路径遍历。

可以以非常高的可靠度探测到第 1 类的问题。在大多数情况下，在客户端上，和查找漏洞相关的任何事情都是可见的。例如，为了探测反射型 XSS，Burp Scanner 会在应用程序的每个入口点提交一些良性的输入，并查看回复的响应。如果有回复，Burp 会解析出响应的内容来确定回复显示的上下文。然后通过许多修改的输入来确定在上下文里组成一个攻击的字符串是否被回复了。Burp Scanner 有打破输入过滤的广泛能力，并且绕过 web 服务器的相关设置，来检查所有使用的上下文。通过先前检查的反馈对决策树进行全面检查，Burp 能高效地模拟一个富有经验并且有条不紊的测试人员的检测行为，如一个输入封装计划。

第 2 类问题本来就不适合使用自动化手段探测，因为大多数情况下和漏洞有关的行为只能发生在服务端，客户端能看到的现象太少。例如，SQL 注入漏洞会在响应中返回详细的数据库错误，或者完全地被遮盖。Burp Scanner 使用许多技术来确认盲服务端注入问题，有延时，改变布尔条件，以及执行模糊的响应比较，等等。此时这些技术会比用在第 1 类问题时产生更多的错误。然而，Burp Sanner 在这里却实现了一个高的成功率，能可靠地发现许多问题，这对测试人员来说是费力的或者困难的来进行诊断。

被动扫描(Passive scanning)

在这种模式下，Burp 不会向服务器发送任何新的请求。它只分析现存的请求和响应，并从中推断出漏洞。在你访问的任何授权应用程序，使用这种操作模式是安全和合法的。

只使用这种被动的技术，Burp Scanner 能发现几种漏洞，有：

1.明文提交的密码。
2.不安全的 cookie 属性，如丢失 HttpOnly 和安全标志。
3.开放的 cookie 范围。
4.跨站点脚本泄露 Referer 信息。
5.自动填充的表单。
6.SSL 保护的缓冲区内容。
7.目录遍历。
8.提交的密码会在后面返回的响应中。
9.不安全的会话令牌传输。
10.信息泄露，如互联网 IP 地址，电子邮件地址，堆栈跟踪，等。
11.不安全的视图配置。
12.不清楚的，没完成的，不正确或不标准的内容类型指示。

这些问题都是相对平淡的，对我们来说记录它们显得无聊和重复。但是作为一个渗透测试人员，有人责任指出它们。在你浏览应用程序一次时，Burp Scanner 既然可靠地扫描出这些问题，你就应该理智清楚它们。

在下面的一些情况，只进行被动的扫描是有好处的：

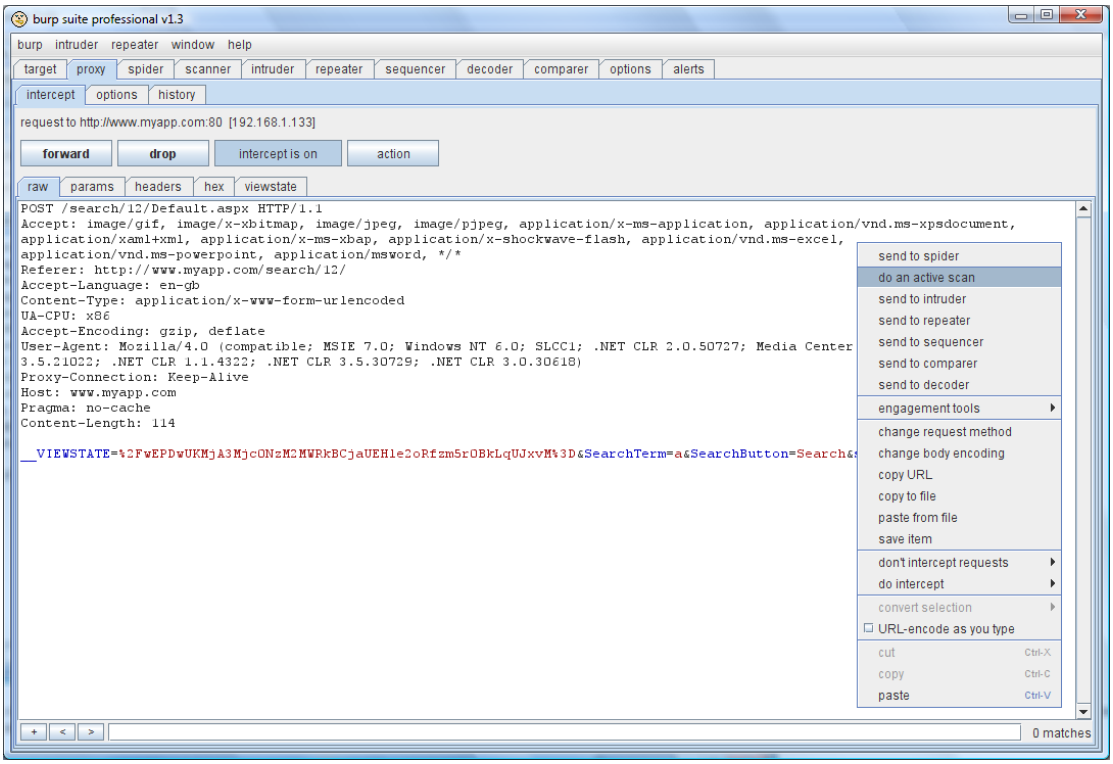
1.因为被动扫描不必向应用程序发送任何新的请求，你就可以安全地用在重要的应用程序上，你可以完全地控制发送的每一个请求。

2.有些应用程序对攻击的反应是很气愤的，当收到一个恶意的请求时，会终止你的会话或者锁定你的账号。在这种情况下，只能一次次地进行手工测试，这时你可以通过被动扫描来确认许多问题，而不会出现麻烦。

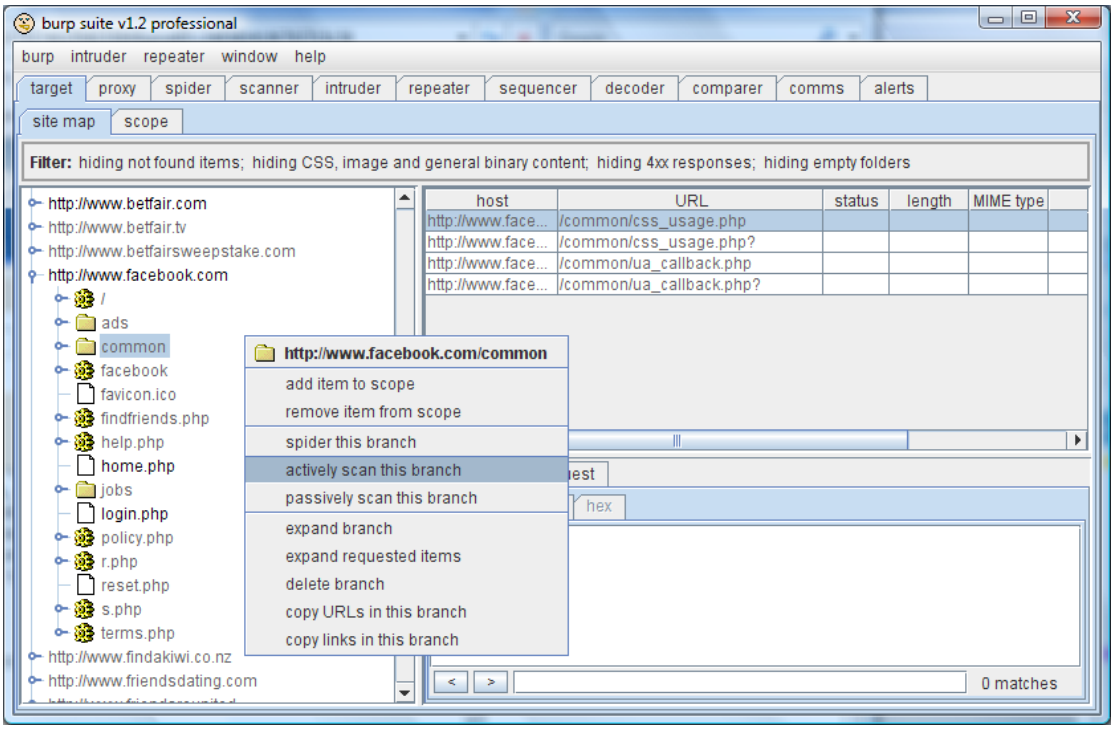
3.如果你没被授权进行攻击一个目标，你可以像普通用户那样浏览应用程序使用被动扫描来确认漏洞。如果你提出了一个新的渗透测试方案，你可以通过被动扫描你的目标来获得对安全状态的了解，甚至在你进行测试之前，有希望发现一些公布的问题。

开始扫描(Initiating scans)

Manual scanning 在 Burp Suite 的任何地方的都可以把一个或多个 HTTP 请求发送到 Scanner 执行主动或被动的扫描。例如，你使用 Burp Proxy 拦截下一个感兴趣的请求，你可以通过使用上下文菜单，只对这一个请求进行扫描：

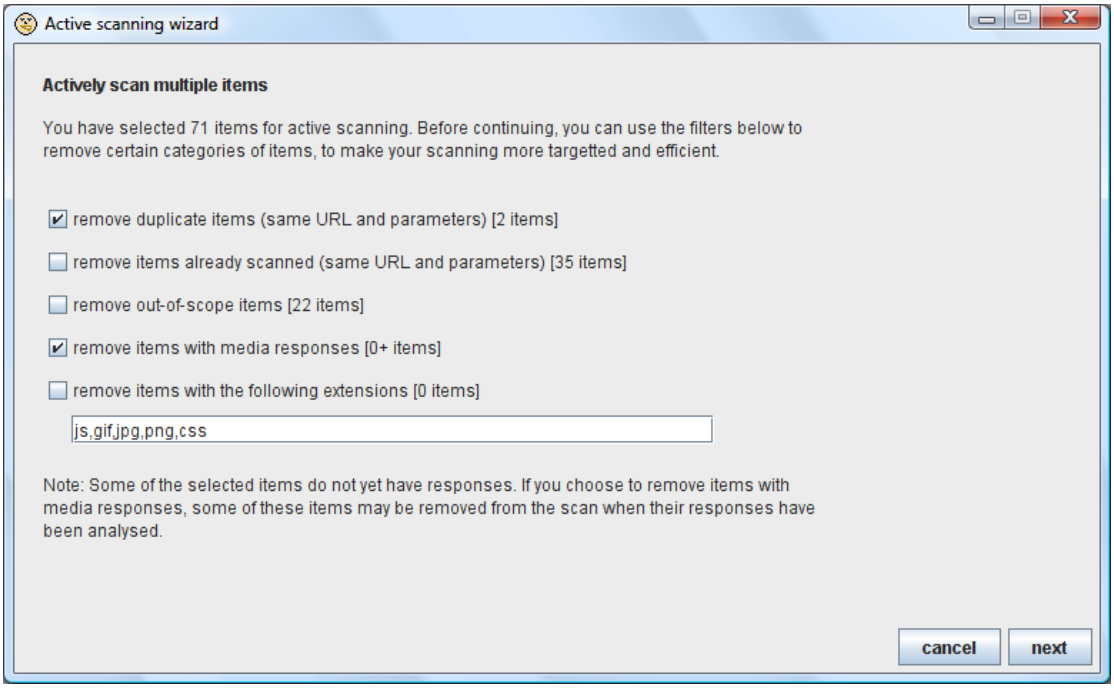


同样地，你可以选择目标站点地图上或者历史记录里的一系列的请求，把它们发送到 Scanner 上。因此，在浏览完应用程序一圈并建立起它内容的全面地图后，你就可以告诉 Burp 扫描应用程序功能的特定区域：

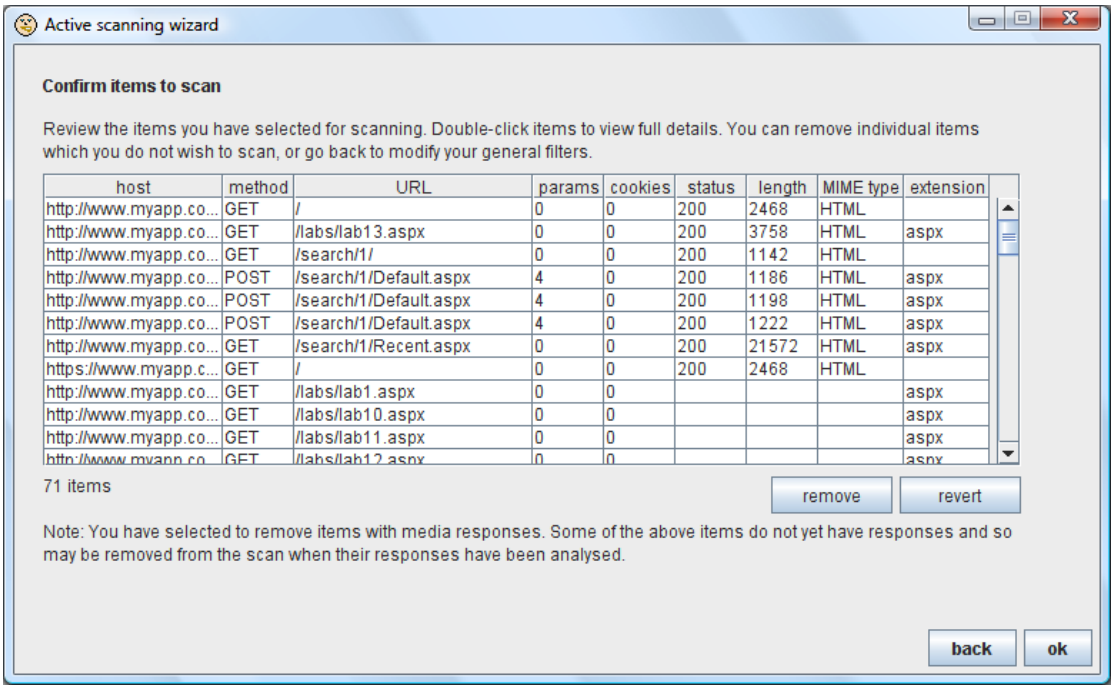


如果你选择了多个项并把它们发送到主动扫描，Burp 会启动一个简洁的向导，让你微

调你的选择。向导的第一个画面为你提供了许多直观的过滤器来删除那些需要的潜在项(重复的，已经扫描过的，媒体内容，等等)，以及显示过滤器影响到的项有多少：



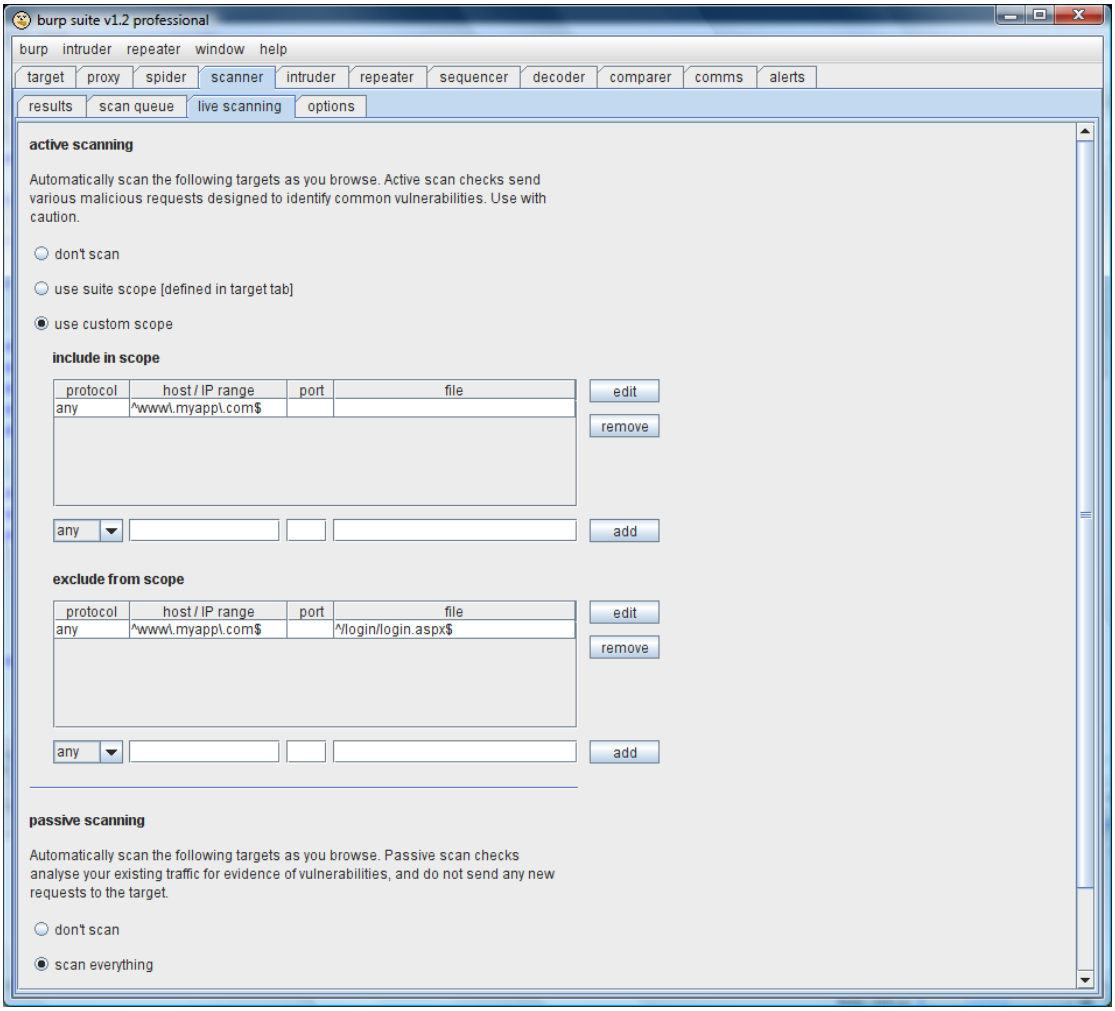
第二个向导画面向显示了剩下项的表单，你可以使用多种相关的属性对表格进行排序，查看所有的请求和响应，以及删除单个项：



这时向导设置已经完成，以平常方式来对这些选中进行扫描。

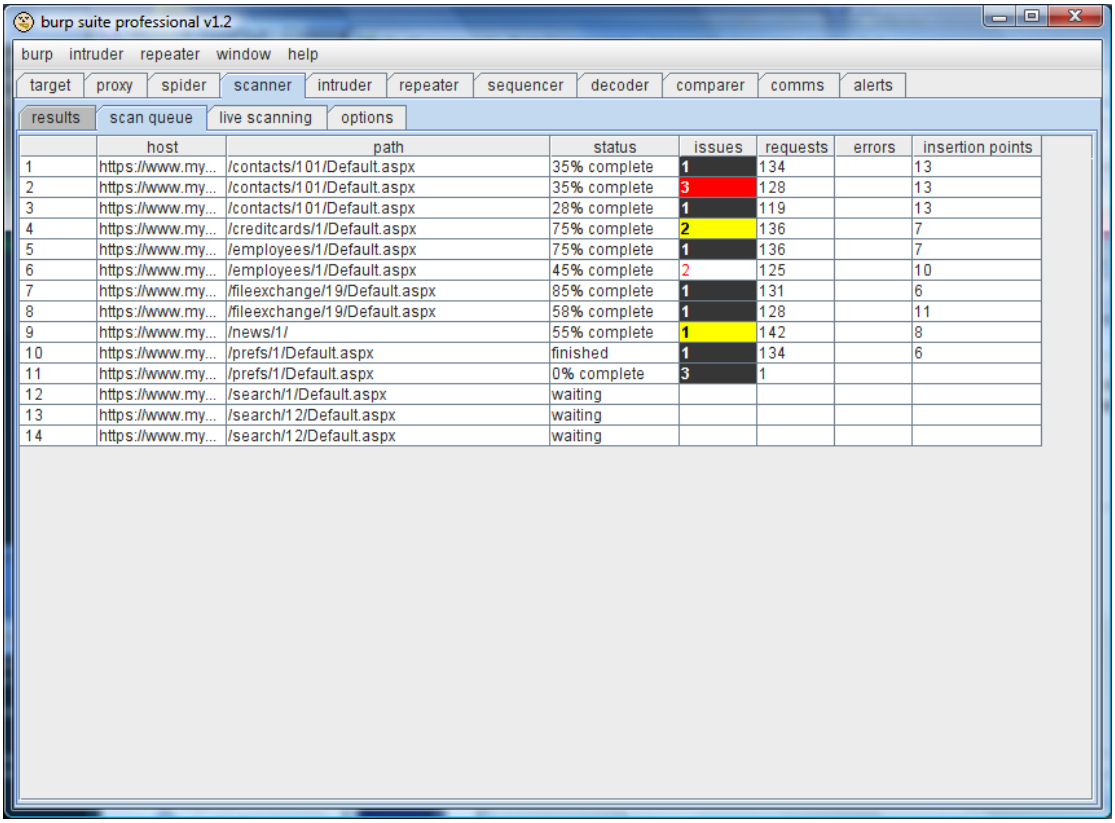
Live scanning 可以使用另一种方式—"live scanning"来执行扫描。在这种模式下，你告诉 Burp 你使用主动或被动扫描的目标范围，然后它会自动启动扫描，对你使用应用程序产生的相关请求进行主动或被动的扫描。当使用这种模式操作时，你需要作为一个普通用户简单地把应用程序访问一圈，向 Burp 显示出应用程序内容和功能的位置，然后它就会在后台进行查找漏洞。

当使用现场扫描时，你细微地控制着 Burp 将要自动地扫描的请求。如果你已经为你的工作配置一个目标范围，然后你可以简单地告诉 Burp 扫描属于范围内的每一个请求。另外，你可以自定义一个用来主动和被动的扫描范围。在下面的这个例子里，Burp 配置为使用主动扫描发送到 www.myapp.com 的每一个请求，除了登陆请求，使用被动扫描发送到其他目的地的任何请求：



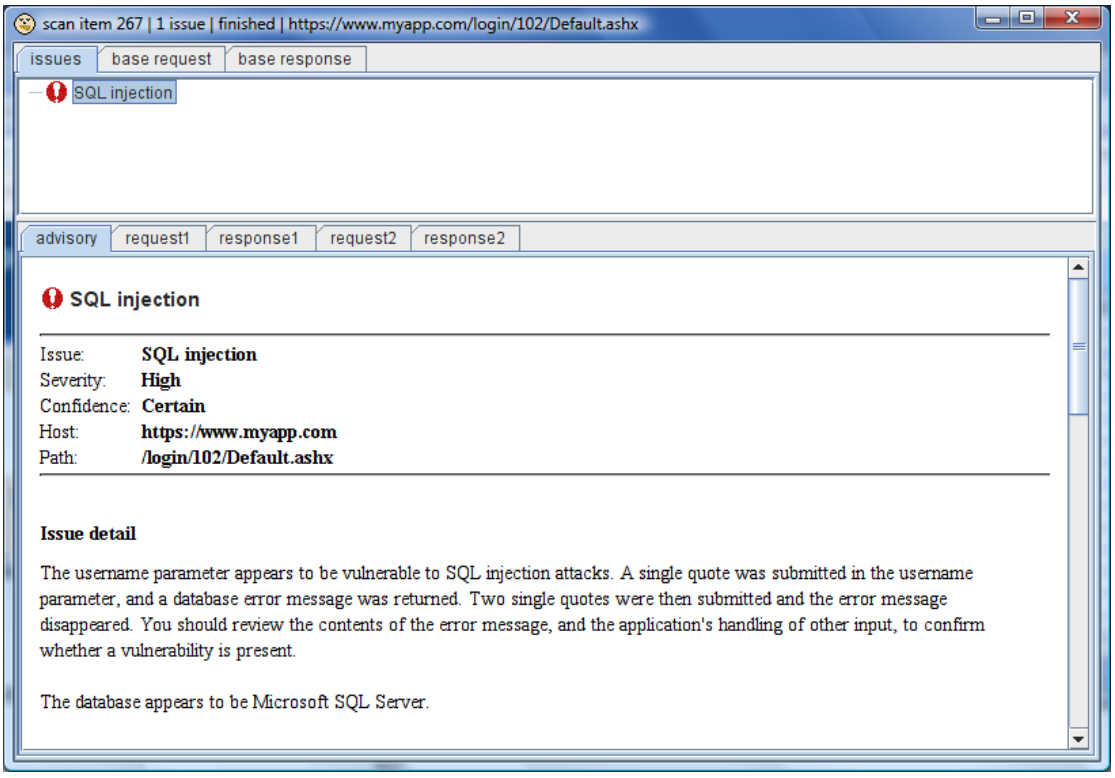
注意现场扫描会忽略对媒体(图像，等)资源的请求，在这些资源里请求不包含非 cookie 参数。这些请求的静态资源几乎不会有任何安全问题，所以扫描器会忽略它们。(这对手动扫描无效—如果你手动地选择了这样的一些项，并把它们发送到主动扫描，这时它们当然会通过正常的发送被扫描了)。

Active scan queue 当你发送请求到主动扫描时，它们会被立即处理。因为主动扫描会像服务器发送大量的请求，发送的请求会被添加到一个队列里。一个有许多参数的经典请求会在 1 到 2 分钟内扫描完，扫描队列通过配置的线程池来扫描，所以等待扫描项的数目会变得非常大。当每一项都被扫描完时，扫描队列表会显示出它的进展——发出的请求数，完成百分比，确认的漏洞。根据接近最严重问题的可信度和严重性，把项的最后的值图成了彩色：



	host	path	status	issues	requests	errors	insertion points
1	https://www.my...	/contacts/101/Default.aspx	35% complete	1	134		13
2	https://www.my...	/contacts/101/Default.aspx	35% complete	3	128		13
3	https://www.my...	/contacts/101/Default.aspx	28% complete	1	119		13
4	https://www.my...	/creditcards/1/Default.aspx	75% complete	2	136		7
5	https://www.my...	/employees/1/Default.aspx	75% complete	1	136		7
6	https://www.my...	/employees/1/Default.aspx	45% complete	2	125		10
7	https://www.my...	/fileexchange/19/Default.aspx	85% complete	1	131		6
8	https://www.my...	/fileexchange/19/Default.aspx	58% complete	1	128		11
9	https://www.my...	/news/1/	55% complete	1	142		8
10	https://www.my...	/prefs/1/Default.aspx	finished	1	134		6
11	https://www.my...	/prefs/1/Default.aspx	0% complete	3	1		
12	https://www.my...	/search/1/Default.aspx	waiting				
13	https://www.my...	/search/12/Default.aspx	waiting				
14	https://www.my...	/search/12/Default.aspx	waiting				

你可以双击扫描队列里那些显示已确认有问题的任何项，查看项的基础请求和响应：



你可以使用上下文菜单在扫描队列上进行许多操作：

1.显示选中项的细节。

2.取消选中的项。

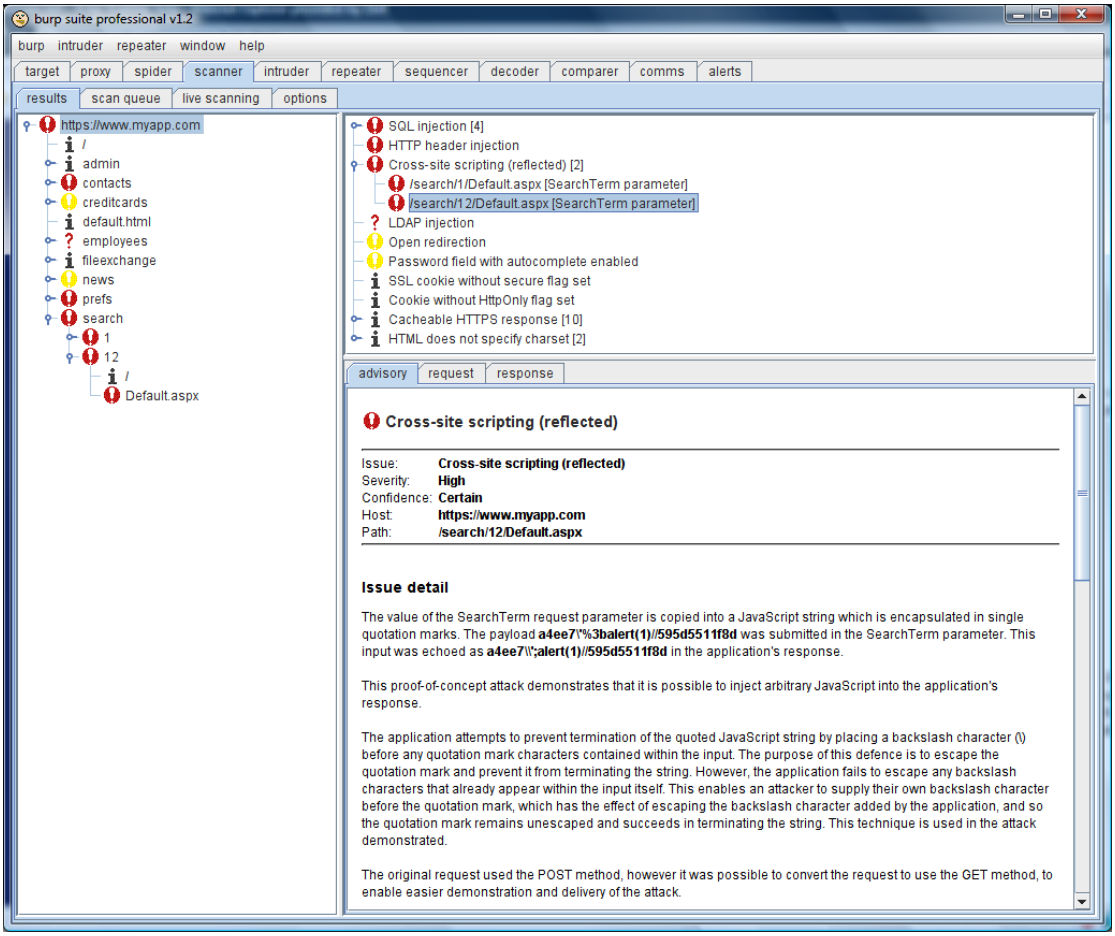
3.再次扫描选中的项。

4.暂停或继续扫描器。

使用这些方法，Burp Scanner 让你对它做的每事情都细微地控制着，并和其他测试操作紧紧地结合在一起。它让你设置自己感兴趣的应用程序的区域的优先权，使用现场扫描浏览它们，或者从站点地图选中它们进行扫描。它立即提供关于这些区域的反馈，来通知你的手动测试结果。

审查结果(Reviewing results)

除了上面单个请求视图里发现的问题，Burp Scanner 保存了一份对所有发现的问题的汇总记录，在一个目标应用程序的站点地图的视图树上显示出来。选中视图树上的一个主机或者文件夹显示出在站点上这个分支列举出的所有已确认的问题：



当发现同类型的多个问题时，这些问题会在面板的右上部汇总成一个问题。你可以展开这个汇总项来查看每个有问题的实例。在面板右上部里选中一个问题，在面板右下部里显示出这个问题的完整细节。这包含了自定义的漏洞咨询，所有相关的请求和响应，来理解和再现这个问题。

咨询包含了对了问题的标准描述和它的应对，以及这个问题特征和整治手段的描述。在上面的例子中，跨站点脚本咨询告诉我们：

1.提供了攻击输入的请求参数(搜索关键词)。

2.响应中的输入返回的上下文语法(在一个 JavaScript 的片段里的一个单引号分割的字符串)。

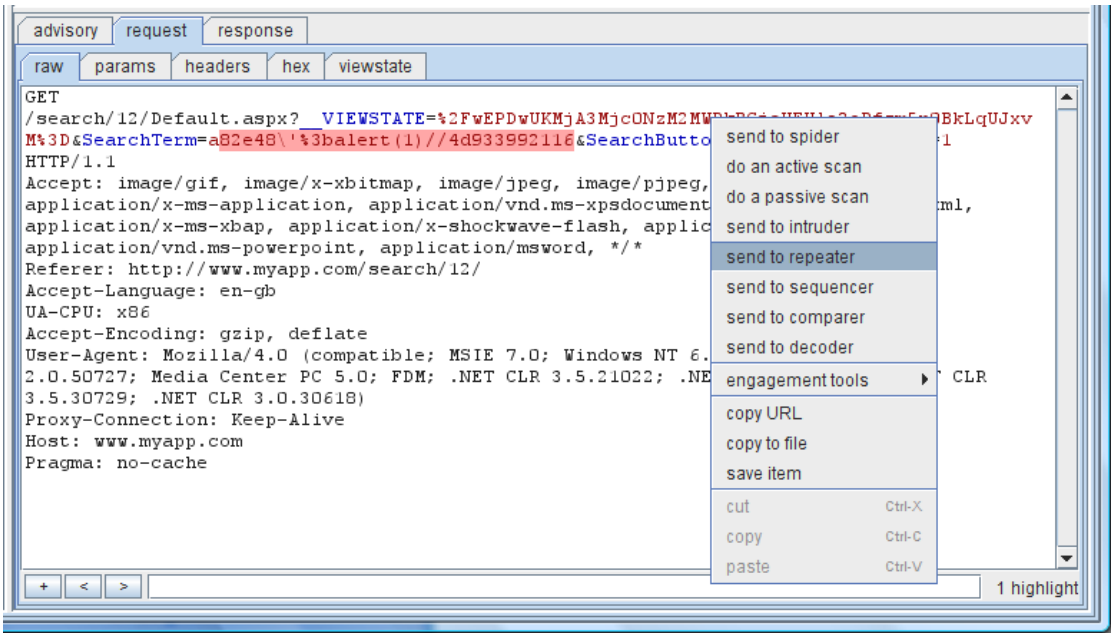
3.这个应用程序在我们的输入中过滤了单引号字符，但是却没过滤反斜杠，这就让我们回避了过滤器。

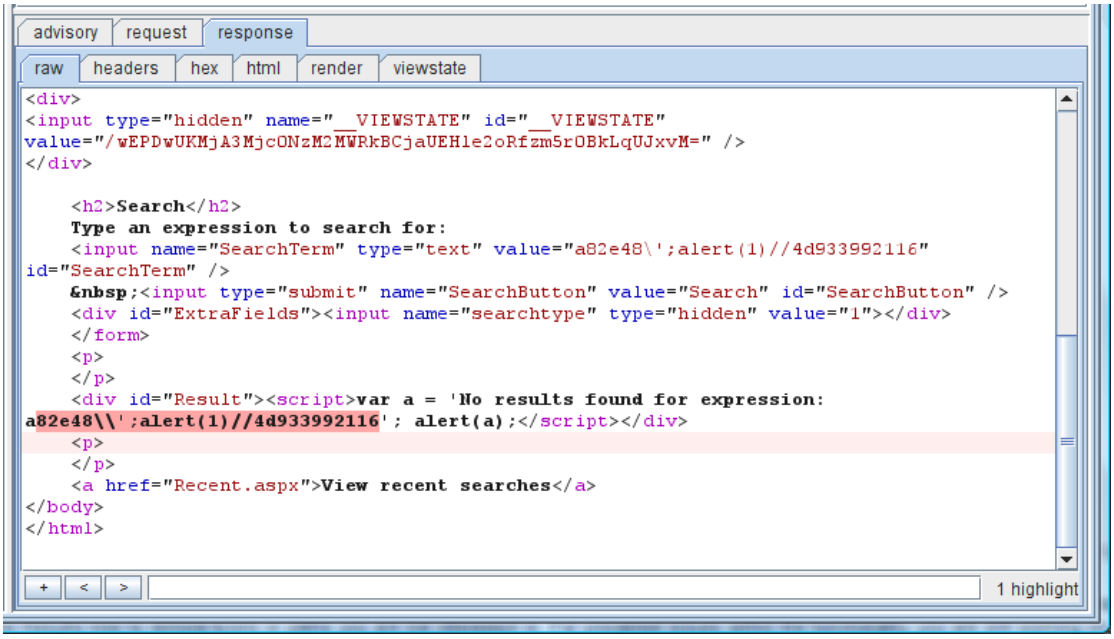
4.Burp 向应用程序提交了理论上的有效载荷，这个有效载荷的格式被返回了。

5.这个原本的请求使用的是 POST 方法，Burp 能把它转换成 GET 方法的请求来方便地证实和发现这个问题。

Burp Scann 对每一个发现的问题都给出了一个安全等级(高，中，低，信息性的)和可信度(一定，坚定，暂定)。当使用一个不太可靠的技术确认一个问题时，Burp 通过降低可信度来让你注意。

在咨询的旁边，Burp 显示了那些用来确认问题的请求和响应，并对相关部分进行加亮。你可以查看 Burp 是怎样来确认这个问题的，并能迅速地明白这个漏洞的本质。你也可以把这些请求发送到其他工具上进行手动验证这个问题，或者微调 Burp 搜集的攻击理论。





在扫描到的问题的列表里，你可以对单个或多个的安全级别和可信度进行修改(通过上下菜单)，或者集中删除问题(通过上下文菜单或使用”del”键)。

注意如果你删除一个问题，Burp 重新发现同一问题(例如，如果你重新扫描同样的请求)，这个问题会再次被报出。相反，如果你把这个问题标记为误判，就不会出现相同的问题了。

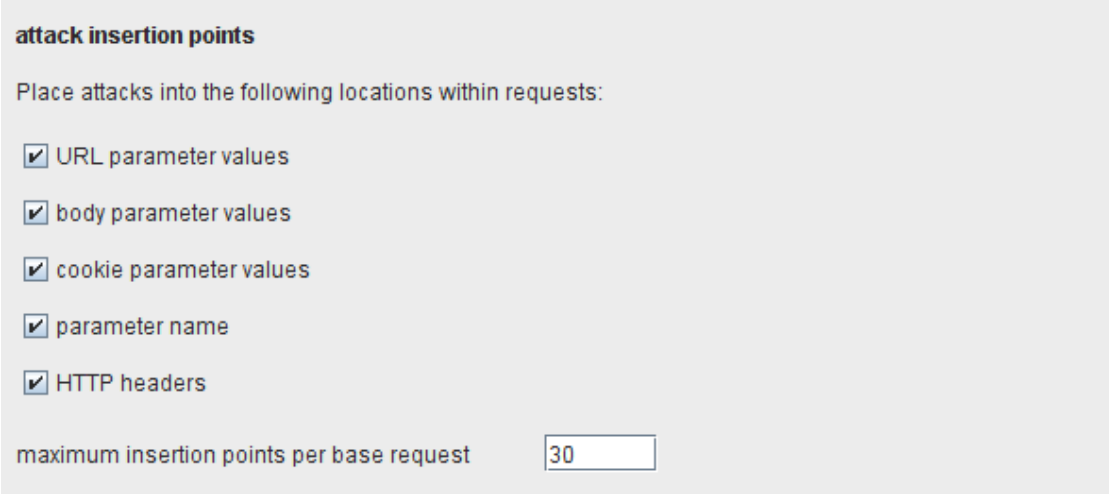
因此，删除问题的最好办法是再结果视图树上清除那些你不感兴趣的主机或路径。对于那些你不想要问题，当你正在使用它的功能时，可以使用误判标记。

扫描优化(Scan optimisation)

Burp Scann 会实时给提供正在执行的操作的细节信息。在扫描队列里，你可以监视每一个基础请求的扫描过程。这个表格向你显示了 Burp 放置有效负荷的“插入点”的数量，以及产生的攻击请求的数量。(后者不是前者的一个线性函数，通过观察应用程序的行为来反馈到后面的请求，就和一个测试人员一样)。

这个信息让你可以快速地查看扫描进度是否太慢，以及了解其原因。通过这个信息，你可以采取一些手段来优化你的扫描。在扫描队列里一个上下文菜单，你可以使用它对单个项进行取消或者重新设置优先权。通过你对应用程序的了解，你可以使用这个选项来对扫描器进行优化。

Attack insertion points 扫描速度和效率的关键因素是攻击插入点的选择。Burp 让你能细微地控制着进行有效负荷攻击的基础请求的位置，这时你可以下面的配置项：



复选框让你能够定义使用 HTTP 请求进行攻击的位置：

1.URL，消息体，以及 cookie 参数的值。

2. Parameter name – 如果被选中，Burp 会添加一个参数到请求中，并且攻击参数名字的位置，如果仅仅测试过参数值，常常能探测丢失的不寻常的错误。

3.HTTP headers – 如果被选中，Burp 会对 User-Agent 和 Referer headers 的位置进行攻击，常常使用日志功能来探测 SQL 注入或保存型 XSS 的问题。

4. AMF string parameters – 对动作消息格式的请求，Burp 攻击那些基于字符串数据类型的消息。

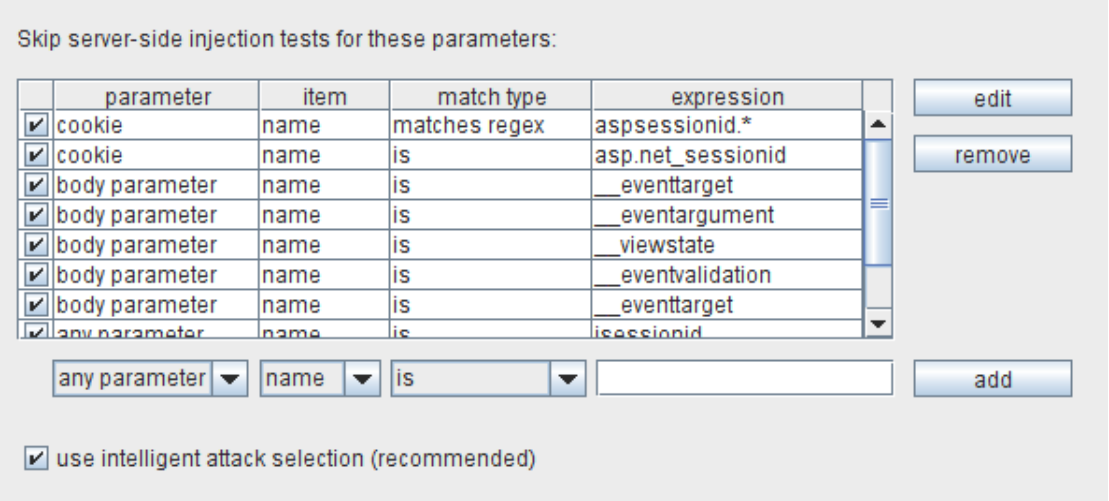
5. REST-style URL parameters – 如果被选中，Burp 会攻击那些有部分 URL 路径的每个目录和文件名。

你可以对 Burp 在个基础请求中的攻击插入点数量设置一个限制。偶尔，HTML 表单里会包含大量的区块(几百，或者更多)。如果 Burp 对每个模块进行完全的漏洞扫描，这次扫描会花费大量的时间才能完成。如果你遇到表单的参数非常多时，设置一个插入点的限制数量，防止扫描止步不前。使用限制后，进入扫描队列的项就会指示出跳过的插入点的数量，让你能手动查看这些基础请求，然后决定是否值得对它的所有可能的进入点进行完整的漏洞扫描。

你可以告诉 Burp 使用” intelligent attack selection”。这个选项让 Burp 执行或者忽略基于每个攻击插入点基值的服务端检查。例如，如果参数的值包含有不能在文件名中正常显示的

字符时，Burp 会跳过这个参数的文件路径遍历检查。使用这个选项可以大大加快扫描，而且这样存在丢失实际存在的漏洞的风险也最小。

插入点配置让你指出 Burp 在服务端注入检查跳过的参数。这些检查相对比较消耗时间，因为 Burp 发送大量的盲目探测多种漏洞的请求。如果你认为请求里的某个参数不会产生漏洞(例如，仅仅由平台或者 web 服务器使用的内置参数)，你可以告诉 Burp 不要测试这些。(注意客户端会检查像执行的跨站点脚本，因为如果一个参数没有问题，每个请求参数的测试会花费扫描过程中最小开销时间)

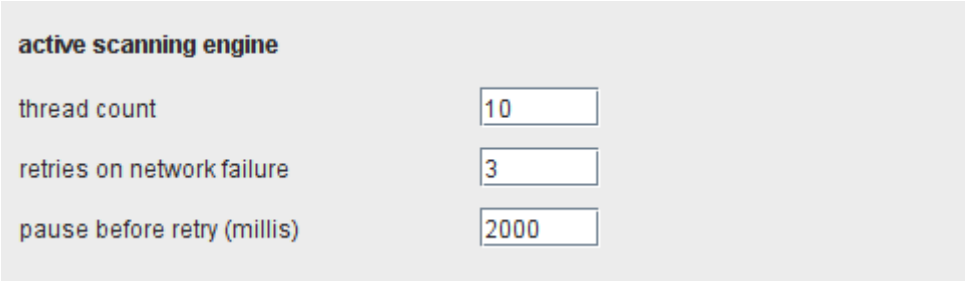


和其他值一样，你可以通过 URL 路径的位置(用斜杠分开)来确认 REST 参数。要这样做，在参数下拉菜单里选中”REST parameter”，在项下拉菜单选中”name”，然后指定你想排除测试的 URL 路径里的位置序列号。

你可以设置任意参数，Burp 不会任何检查的。

为主动扫描指定完整的自定义攻击插入点是非常有可能的，所以你可以指定基础请求的任意位置来放置攻击字符串。要使用这个功能，就需要把这个相关基础请求发送到 Intruder，通过常规方式使用有效负荷位置界面来定义每个开始/结束的插入点，并选中 Intruder 菜单选项里的” actively scan defined insertion points”.

Active scanning engine

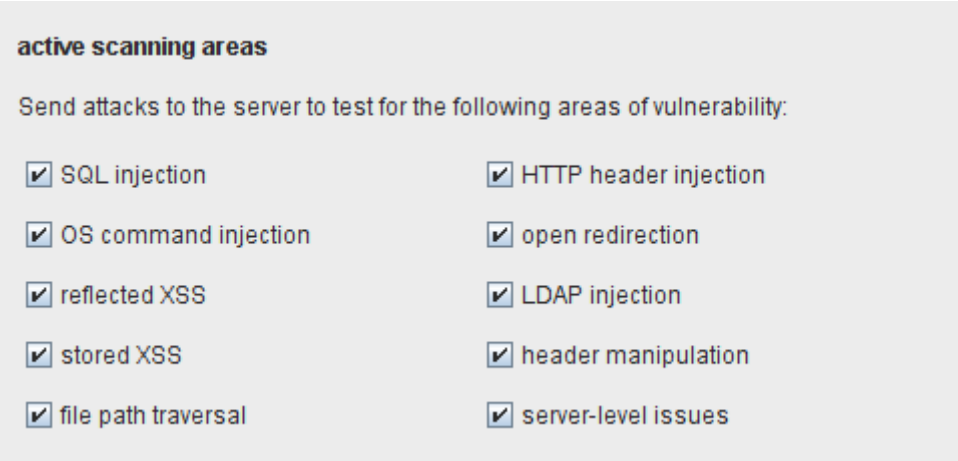


这个选项让你微调 Burp 的扫描引擎，这依据应用程序的反应和自己的处理能力以及带宽。如果你发现你的扫描进度很慢，但应用程序正常，CPU 使用率很低，这时你可以加大扫描线程的数量来使你的扫描更快。如果你发现有连接错误产生，应用程序开始慢下来，或者你的电脑很卡，你就应该降低线程数量，并且增加处理网络错误数量和在处理暂停的时间。如果应用程序的功能是处理一个基础请求接口和其他请求返回的响应直接的操作，你需要把线程数量降到 1，以确保每次只扫描一个基础请求。

如果你想避免应用程序超载，或者保持网络上隐身，你可以使用这个阀门设置来添加固定或随机的请求间隔。

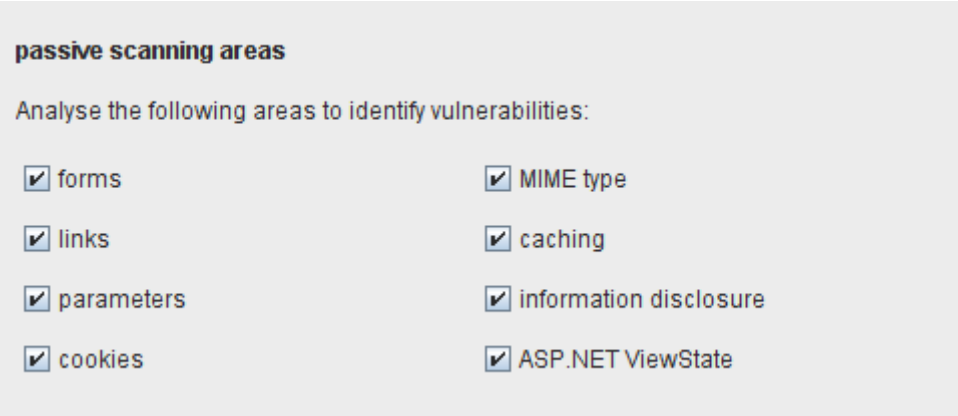
由于一些应用程序重定向到包含你提交的值的第三方 URL。Burp 通过不跟踪任何接收到的重定向，保护你在不经意间会攻击第三方应用程序。如果扫描的请求是在定义的目标范围内(例如，使用目标范围来控制扫描的内容)，Burp 会只跟着范围内的重定向。如果请求不在范围内(例如，你可以手动执行一次对范围外请求的扫描)，Burp 只跟踪和已扫描的请求的主机端口一致的重定向，并且不能满足范围排除规则(如，”logout.aspx”)。

Active scanning areas



这些选项让你在主动扫描时定义执行哪些选择。每个执行的选择都会增加请求的次数和整体扫描的时间。你可以根据自己对应用程序技术的了解，打开或关闭单个选择，或者对扫描要求有多严。例如，如果你知道应用程序不使用任何 LDAP，你就可以关闭 LDAP 注入测试。在对每一个插入点进行全面的漏洞检查之前，或者你可以配置 Burp 在应用程序上进行一次快速的检查，只查找 URL 和消息体里参数里的 SQL 注入和 XSS 漏洞。

Passive scanning areas



被动扫描不会发生它自己的任何请求，并且每个被动检测在电脑上的执行过程都是可以忽略不计的。不过，如果你对其中的一些项不感兴趣或者不想看到它们的扫描结果，可以使单个的选项设为不可用。

报告(Reporting)

当你完成测试后，可以把所有的或者选中的问题以 HTML 格式导出。要这样，就选中显示出的汇总结果里需要的问题(可以选中多个主机，文件夹，问题等等)，然后在上下文菜单中选” report issues”。报告向导会让你为自己的报告选择许多项：

1.报告的格式(屏幕，打印)。

2.问题的描述级别和包含的建议。

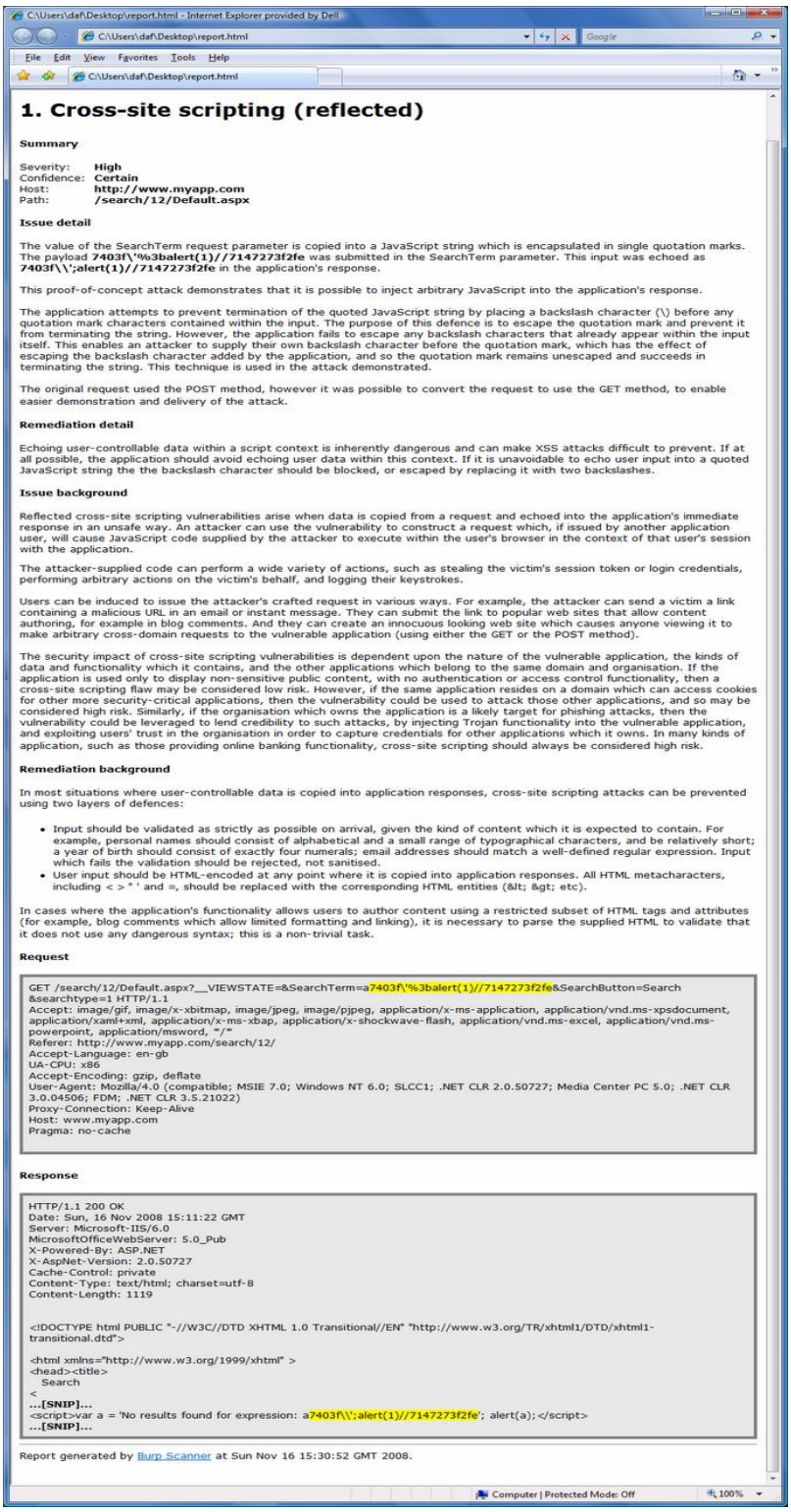
3.是否显示请求和响应的细节，或者像提取的，或者不是。

4.发现的问题类别是包含的还是排除的。

5.是否通过类型，安全或者 URL 来组织问题。

6.报告的标题，及其大小，等等。

跨站点脚本漏洞的报告会被优先显示出来，所有的细节都列出来了，以友好的打印格式显示出提取的应用程序响应的提取，像这样：



C:\Users\daf\Desktop\report.html - Internet Explorer provided by Dell

C:\Users\daf\Desktop\report.html

File Edit View Favorites Tools Help

C:\Users\daf\Desktop\report.html

1. Cross-site scripting (reflected)

Summary

Severity: **High**
Confidence: **Certain**
Host: **http://www.myapp.com**
Path: **/search/12/Default.aspx**

Issue detail

The value of the SearchTerm request parameter is copied into a JavaScript string which is encapsulated in single quotation marks. The payload **7403f\'%3balert(1)//7147273f2fe** was submitted in the SearchTerm parameter. This input was echoed as **7403f\\';alert(1)//7147273f2fe** in the application's response.

This proof-of-concept attack demonstrates that it is possible to inject arbitrary JavaScript into the application's response.

The application attempts to prevent termination of the quoted JavaScript string by placing a backslash character (\) before any quotation mark characters contained within the input. The purpose of this defence is to escape the quotation mark and prevent it from terminating the string. However, the application fails to escape any backslash characters that already appear within the input itself. This enables an attacker to supply their own backslash character before the quotation mark, which has the effect of escaping the backslash character added by the application, and so the quotation mark remains unescaped and succeeds in terminating the string. This technique is used in the attack demonstrated.

The original request used the POST method, however it was possible to convert the request to use the GET method, to enable easier demonstration and delivery of the attack.

Remediation detail

Echoing user-controllable data within a script context is inherently dangerous and can make XSS attacks difficult to prevent. If at all possible, the application should avoid echoing user data within this context. If it is unavoidable to echo user input into a quoted JavaScript string the the backslash character should be blocked, or escaped by replacing it with two backslashes.

Issue background

Reflected cross-site scripting vulnerabilities arise when data is copied from a request and echoed into the application's immediate response in an unsafe way. An attacker can use the vulnerability to construct a request which, if issued by another application user, will cause JavaScript code supplied by the attacker to execute within the user's browser in the context of that user's session with the application.

The attacker-supplied code can perform a wide variety of actions, such as stealing the victim's session token or login credentials, performing arbitrary actions on the victim's behalf, and logging their keystrokes.

Users can be induced to issue the attacker's crafted request in various ways. For example, the attacker can send a victim a link containing a malicious URL in an email or instant message. They can submit the link to popular web sites that allow content authoring, for example in blog comments. And they can create an innocuous looking web site which causes anyone viewing it to make arbitrary cross-domain requests to the vulnerable application (using either the GET or the POST method).

The security impact of cross-site scripting vulnerabilities is dependent upon the nature of the vulnerable application, the kinds of data and functionality which it contains, and the other applications which belong to the same domain and organisation. If the application is used only to display non-sensitive public content, with no authentication or access control functionality, then a cross-site scripting flaw may be considered low risk. However, if the same application resides on a domain which can access cookies for other more security-critical applications, then the vulnerability could be used to attack those other applications, and so may be considered high risk. Similarly, if the organisation which owns the application is a likely target for phishing attacks, then the vulnerability could be leveraged to lend credibility to such attacks, by injecting Trojan functionality into the vulnerable application, and exploiting users' trust in the organisation in order to capture credentials for other applications which it owns. In many kinds of application, such as those providing online banking functionality, cross-site scripting should always be considered high risk.

Remediation background

In most situations where user-controllable data is copied into application responses, cross-site scripting attacks can be prevented using two layers of defences:

- Input should be validated as strictly as possible on arrival, given the kind of content which it is expected to contain. For example, personal names should consist of alphabetical and a small range of typographical characters, and be relatively short; a year of birth should consist of exactly four numerals; email addresses should match a well-defined regular expression. Input which fails the validation should be rejected, not sanitised.
- User input should be HTML-encoded at any point where it is copied into application responses. All HTML metacharacters, including < > " ' and =, should be replaced with the corresponding HTML entities (< > etc).

In cases where the application's functionality allows users to author content using a restricted subset of HTML tags and attributes (for example, blog comments which allow limited formatting and linking), it is necessary to parse the supplied HTML to validate that it does not use any dangerous syntax; this is a non-trivial task.

Request

```
GET /search/12/Default.aspx?__VIEWSTATE=&SearchTerm=a7403f\'%3balert(1)//7147273f2fe&SearchButton=Search
&searchtype=1 HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-ms-application, application/vnd.ms-xpsdocument,
application/xaml+xml, application/x-ms-xbap, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-
powerpoint, application/msword, */*
Referer: http://www.myapp.com/search/12/
Accept-Language: en-gb
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0; SLCC1; .NET CLR 2.0.50727; Media Center PC 5.0; .NET CLR
3.0.04506; FDM; .NET CLR 3.5.21022)
Proxy-Connection: Keep-Alive
Host: www.myapp.com
Pragma: no-cache
```

Response

```
HTTP/1.1 200 OK
Date: Sun, 16 Nov 2008 15:11:22 GMT
Server: Microsoft-IIS/6.0
MicrosoftOfficeWebServer: 5.0_Pub
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Cache-Control: private
Content-Type: text/html; charset=utf-8
Content-Length: 1119


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head><title>
   Search
<
...[SNIP]...
<script>var a = 'No results found for expression: a7403f\\';alert(1)//7147273f2fe'; alert(a);</script>
...[SNIP]...
```

Report generated by Burp Scanner at Sun Nov 16 15:30:52 GMT 2008.

Computer | Protected Mode: Off 100%

你也可以使用 XML 格式来报告这些问题，这样能很容易地和其他工具整合在一起。XML 有一个平整的结构，在每一个问题的报告中包含了问题的列表，通过 meta 信息来显示问题类型，URL，等等。看起来像这样的：

```
<!DOCTYPE issues [
<!ELEMENT issues (issue*)>
<!ATTLIST issues burpVersion CDATA "">
<!ATTLIST issues exportTime CDATA "">
<!ELEMENT issue (serialNumber, type, name, host, path, location, severity, confidence, issueBackground?, remediationBackground?, issueDetail?, remediationDetail?, requestresponse*)>
<!ELEMENT serialNumber (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT host (#PCDATA)>
<!ELEMENT path (#PCDATA)>
<!ELEMENT location (#PCDATA)>
<!ELEMENT severity (#PCDATA)>
<!ELEMENT confidence (#PCDATA)>
<!ELEMENT issueBackground (#PCDATA)>
<!ELEMENT remediationBackground (#PCDATA)>
<!ELEMENT issueDetail (#PCDATA)>
<!ELEMENT remediationDetail (#PCDATA)>
<!ELEMENT requestresponse (request?, response?)>
<!ELEMENT request (#PCDATA)>
<!ELEMENT response (#PCDATA)>
]>
```

序列号是一个对其他单个问题的不同的长整数。如果你想从同一个 Burp 实例上，多次导出问题，你可以通过序列号来确认增加的新问题。

类型包含了一个用来唯一地确认查找的内容类型(SQL 注入，XSS 等等)的整数，在不同的 Burp 实例间，这个值是稳定的。

名字包含了相应的问题类型的描述名字。

路径包含了问题的 URL(不包含查询字符串)。

位置包含了攻击进入点的 URL 和描述，以及相关项(一个特殊的 URL 参数，请求消息头，等等)。

其他的项，有些事可选的，用户可以通过报告向导来选择，能进行自我解释。

Burp Intruder Help

Burp Intruder 是什么？(What is Burp Intruder?)

Burp Intruser 是一个对 web 应用程序进行自动化的自定义攻击的工具。

Burp Intruder 不是一个点击工具。要想高效地使用它，你需要明白目标应用程序是怎样工作的，以及一些 HTTP 协议的知识。在你使用 Burp Intruder 进行攻击之前，你需要调查清楚目标应用程序的功能和结构，尤其是再浏览器和服务器直接传输的许多 HTTP 消息。你可以使用标准浏览器和 Burp Proxy 拦截和查看应用程序产生的请求和响应。当你确认了一些一些感兴趣的 HTTP 请求需要更严格的检查时，你就已经为使用 Burp Intruder 准备好。

Burp Intruder 是高度可配置的，并被用来在广范围内进行自动化攻击。你可以使用 Burp Intruder 方便地执行许多任务，包括枚举标识符，获取有用数据，漏洞模糊测试。合适的攻击类型取决于应用程序的情况，可能包括：缺陷测试：SQL 注入，跨站点脚本，缓冲区溢出，路径遍历；暴力攻击认证系统；枚举；操纵参数；拖出隐藏的内容和功能；会话令牌测序和会话劫持；数据挖掘；并发攻击；应用层的拒绝服务式攻击。要想知道关于使用 Burp Intruder 执行的这类攻击的讨论细节，可以查看 The Web Application Hacker's Handbook 的第 13 章。

Burp Intruder 有许多预设的攻击“有效负荷”(在探索发现常规漏洞中有用的字符串)列表。它包含了许多工具，这些工具动态地产生适合应用程序内的特定机制的攻击载体。外部的文件也可被加载并纳入到 Burp Intruder(如，枚举用户名的列表，新发型的漏洞的模糊字符串)。

核心动作就是通过这些 HTTP 请求重复地攻击。在调查阶段确认有基础请求派生的请求。Burp Intruder 以特殊方式操纵这些基础请求来确认或探测应用程序漏洞。它使用一个或多个有效载荷来替换基础请求中的一部分来实现这个过程。可以为每次攻击配置时间和执行方案。同时可以使用多线程来产生请求。限制请求可以防止入侵检测系统的探测。拒绝服务模型可以使用请求来轰炸服务器，这时会忽略所有接收到的响应。

当一次执行时，细节结果表格也就产生，显示出从服务器上接收到的每个请求的响应。结果里包含了所有的相关信息，可以使用它来查明一些感兴趣或成功的响应。除了常见的每次攻击的标准结果，在运行时可以对这些结果执行许多自定义的测试，这些结果同样被记录下来。例如，可以为 Burp Intruder 精确配置指定一些 HTML 页面上的信息(如，用户信息页面上的个人细节信息)，并且在每个结果里记录这些信息。可以为做进一步操作导出所有结果，或者把它们当做输入文件进行下一步攻击。

Burp Intruder 是一个 Java 应用程序，可以在任何有 Java Runtime 环境的平台上运行。它需要 1.5 版本或者更新的。JRE 可以免费从 java.sun.com 上获得。

配置 Burp Intruder(Configuring Burp Intruder)

Burp Intruder 控制面板让你在他们的数字选项里同时能配置一个或多个攻击。你可以使用 Intruder 菜单创建一个新的选项或者重命名现存的选项。

在一些子选项(target, positions, payloads, options)里进行每次攻击配置。创建一个新攻击的最简单的方法是通过其他 Burp 工具(如代理历史记录或站点地图)定位相关的基础请求，然后使用上下文菜单里的” send to intruder”。这将会用相关的细节来填充 target 和 positions 选项。当你创建一个攻击选项时，可以通过 Intruder 菜单来控制怎样设置 payloads 和 options 选项。通过这种方式，你可以在第一次攻击选项(如模糊所有参数和搜索错误消息)里设置一个标准的攻击配置，然后把这些标准复制到发送给 Intruder 的每一次新攻击里。你可以使用 Intruder 菜单在任意选项之间复制攻击的配置，或者保存加载攻击配置。

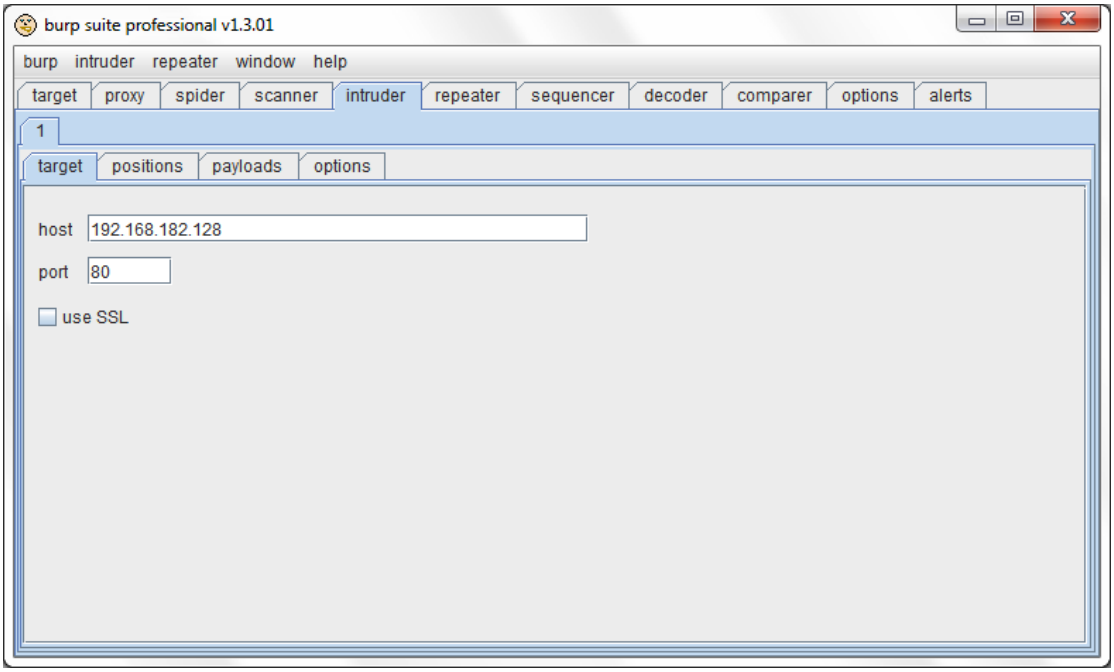
要开启一次攻击，需要设置攻击配置信息，然后在 Intruder 菜单上选择” start attack”。

下面的部分里描述了配置选项的细节。

要加载一个保存的攻击，需在 Intruder 菜单上选择” open saved attack”，然后选择需要的文件[专业版]。

目标选项(Target tab)

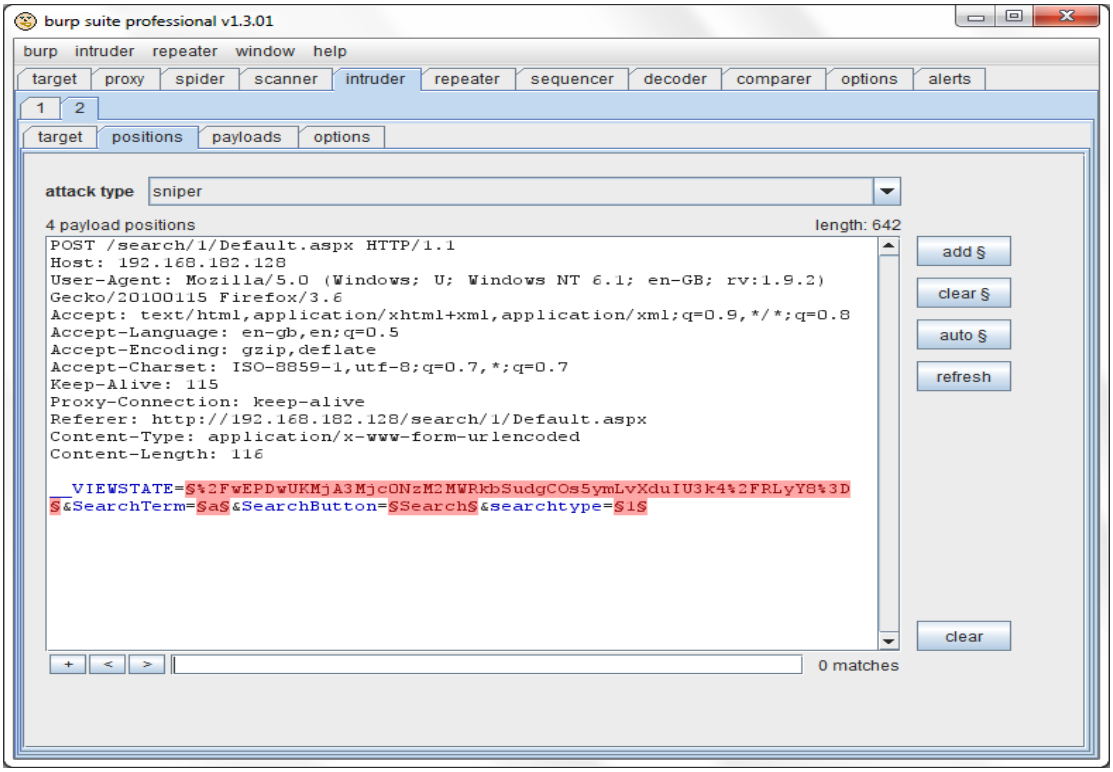
这个选项是用来配置目标服务器的细节：



“host”区域是用来指定目标服务器的 IP 地址或者主机名。”port”区域是用来指定 HTTP/S 服务的端口号。”use SSL”框是用来是否使用 SSL 连接。

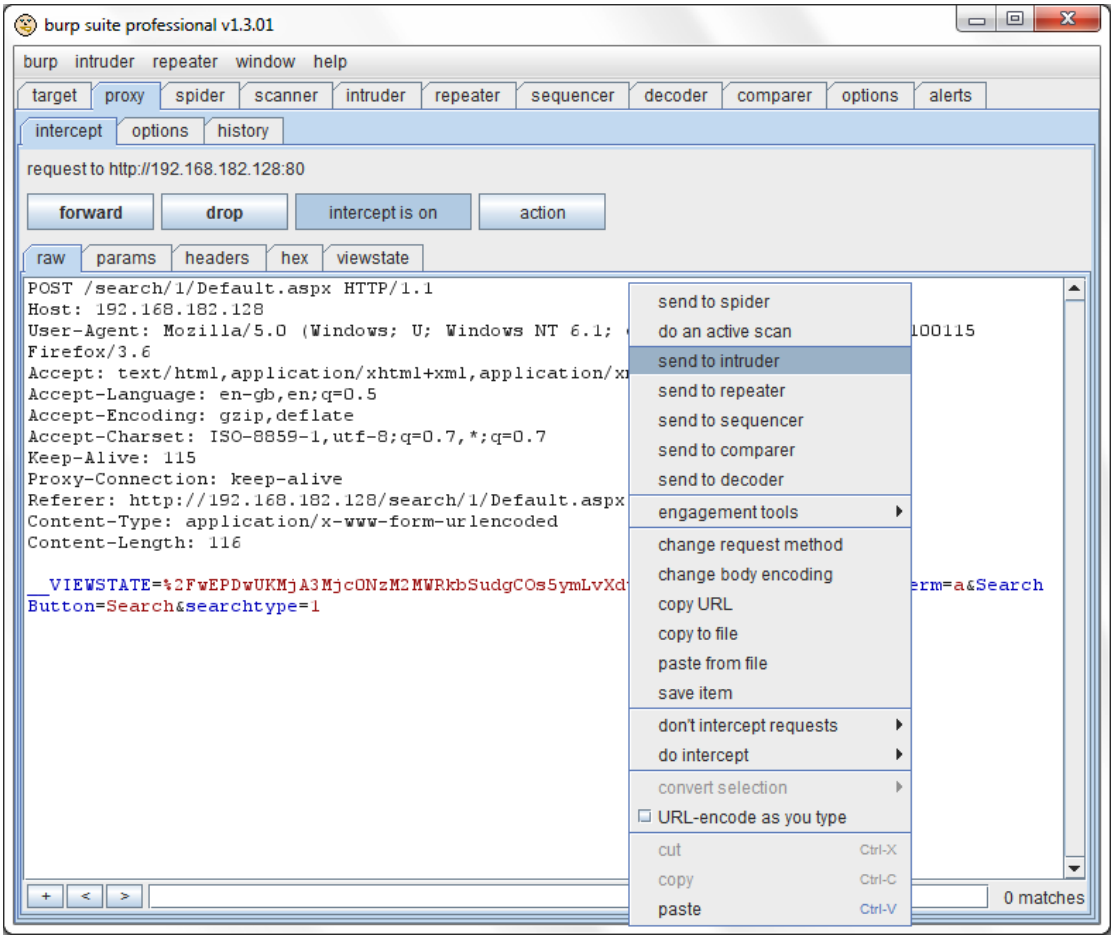
位置选项(Positions tab)

这个选项是用来配置在攻击里产生的所有 HTTP 请求的模板：



主文本编辑器是用来设置基础请求的内容，以及在攻击时，标记出有效负荷插入到单个 HTTP 请求的位置。这有一个有许多功能的上下文菜单。

设置攻击模板的最简单的方法是先定位出在一个其他 Burp 工具里的相关请求，然后选中” send to intruder”选项。你可以从 Burp Suite 里的任何显示出一个 HTTP 请求或响应的地方发送请求，也可以从 Burp Proxy 历史记录，站点地图里的视图树或者表格，以及从一个准备执行攻击的 Burp Intruder：



使用一对 § 字符来标记出有效负荷的位置，在这两个符号直接包含了模板文本的内容。当把一个有效负荷放置到一个给出的请求的特殊位置上时，就把这 § 符号放到这个位置，然后在两个符号之间的出现的文本都会被有效负荷替换。当有个特殊位置没有为一个给出的请求安排有效负荷时(这只适用”sniper”攻击类型—看下面)，那个位置的 § 字符会被删除，出现在它们之间的文本不会变化。

当使用 Burp Suite 发送一个其他地方的请求时，Burp Intruder 会对你最想放置有效负荷的位置做一个最好的猜测，并且它把这些放置在每个 URL 和主体参数的值里，以及每个 cookie 里。每个标记和它中间的文本都会被加亮以显得更清晰。你可以使用 Intruder 菜单上的选项标记的位置是要替换还是附加现有的参数值。在上面的请求编辑器里，指出了定义位置的数量和文本模板的大小。

你可以使用选项上的按钮来控制位置上的标记：

1. add § — 在当前光标位置插入一个位置标记。

2. clear § — 删除整个模板或选中的部分模板里的位置标记。

3. auto § — 这会对放置标记的位置做一个猜测，放哪里会有用，然后就把标记放到相应位置。这是一个为攻击常规漏洞(SQL 注入)快速标记出合适位置的有用的功能，然后人工

标记是为自定义攻击的。

4.refresh — 如果需要，可以刷新编辑器里有颜色的代码。

5.clear — 删除整个编辑器内容。

注意自动放置有效负荷位置需要识别出当前选中的请求模板内的 XML 格式的数据。一些应用程序会在一个请求主体里发送封装的 XML 格式的数据，如：

```
POST /function HTTP/1.0
Content-Type: multipart/form-data; boundary=weidhwiderfhwiuehwiuehfwerrf
Content-Length: 202

--weidhwiderfhwiuehwiuehfwerrf
Content-Disposition: form-data; name="data"

<data>
<param1>foo</param1>
<param2>bar</param2>
<param3>123</param3>
</data>

--weidhwiderfhwiuehwiuehfwerrf—
```

如果你在整个消息里都执行自动放置有效负荷的位置，Intruder 会用单个插入点标出所有的 XML 块，这些块未必都是你想要的。相反，如果你手动地选出那些 XML 块，自动放置功能是识别出包含 XML 的选择，并把单个 XML 参数值标记为插入点。

“attack type”下拉菜单是用来定义 Burp Intruder 行为的一个关键方面 — 为单个请求把有效负荷放置特定位置的方式。下面列出 4 个可能的攻击类型：

sinper — 这使用了单个有效负荷集合。它的目标是在每个位置上，并把每个有效负荷按顺序地插入到这些位置上。请求中的不是目标的位置不受影响 — 位置标记会被删除并且它们之间的模板里的文本不会变化。这类攻击类型对单独使用数据域来测试常规漏洞(如，跨站点脚本)非常有效。攻击产生的大量请求是位置数量和有效负荷数量的产品。

battering ram — 这使用了单个有效负荷集合。它是通过有效负荷迭代，并一次在所有定义的位置插入有效负荷。当一次攻击需要在 HTTP 请求(如，Cookie 消息头和消息体里的用户名)中的多个位置上插入相同的有效负荷时，这个攻击类型非常有用。攻击产生的所有请求数量就是有效载荷的数量。

pitchfork — 这个是用在多个有效负荷集合。在每个定义的位置有不同的有效负荷集合(最多 8 个)。攻击同时通过所有的有效负荷集合进行迭代，并在每一个位置上插入一个有效载荷。例如，第一个请求会把第一个有效负荷集合里第一个有效负荷插入到第一个位置，第二个有效载荷集合里的第一个有效负荷插入到第二个位置。第二个请求会把第一个集合里的第二个有效负荷插入到第一个位置，把第二个有效负荷里的第二个有效负荷插入到第二个位置，等等。当攻击需要将不同的但相关的输入插入到 HTTP 请求(如，一个数据域里的用户名，以及在其他数据域里的一个和用户名相关的 ID 号)的多个位置里时，这个攻击类型会有用的。攻击产生的所有的请求数量是最小有效负荷集合里的有效负荷数。

cluster bomb — 这个使用了多个有效负荷集合。每个定义的位置(最多 8 个)都有一个不同的有效负荷集合。攻击会按照每个有效负荷集合的顺序进行迭代，于是所有的有效负荷排列组合都会被测试。例如，如果有 2 个有效负荷位置，攻击会把第一个有效负荷集合里的第一个有效负荷放置在第一个位置，在位置 2 里会迭代第二个有效负荷里的所有有效负荷；然

后他会把第一个集合里第二个有效负荷放在第一个位置，然后在位置 2 上迭代集合 2 里的所有有效负荷。当攻击需要在 HTTP 请求(如，一个参数里的用户名和另一个参数里的密码)里插入不同的并且不相关的输入时，这个攻击类型会很有效。攻击产生的请求数量是在所有定义集合里的有效负荷数量 — 应该是最大值。

有效负荷选项(Payloads tab)

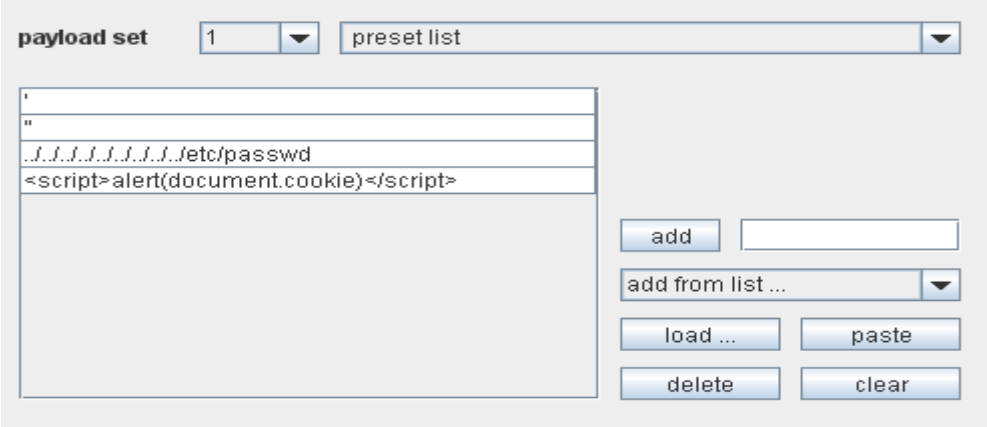
这个选项是用来配置一个或多个有效负荷的集合。如果定义了” cluster bomb”和” pitchfork”攻击类型，然后必须为每定义的有效负荷位置(最多 8 个)配置一个单独的有效负荷。使用” payload set”下拉菜单选择要配置的有效负荷。

对于每个有效负荷集合，都会定义一个有效负荷”源”(如，preset list, character blocks, brute forcer)来使用，并且在每个有效负荷上执行多种附加的处理。在 Burp Intruder 中有大量的可用的有效负荷源。其中的一些是高度可配置的，并且提供了许多自定义攻击。通过下拉菜单为当前有效负荷选中源。每个有效负荷源在下面都一一介绍。

有效负荷源(Payload Sources)

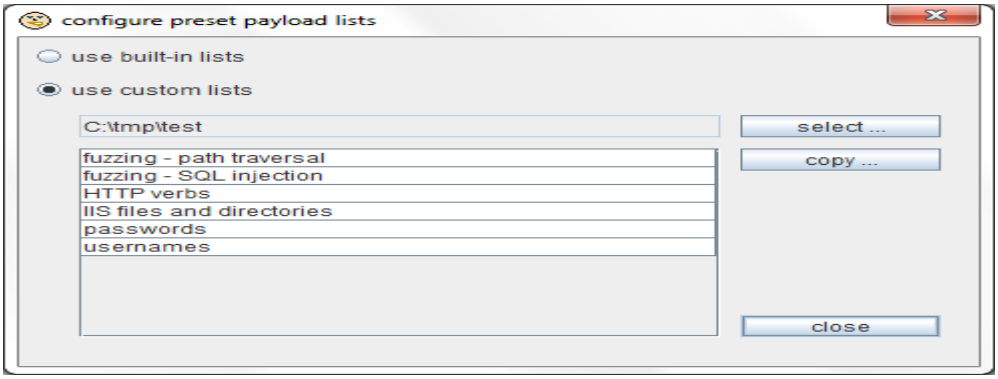
Preset list

这是一个简单的有效负荷源，配置有效负荷的预设列表：



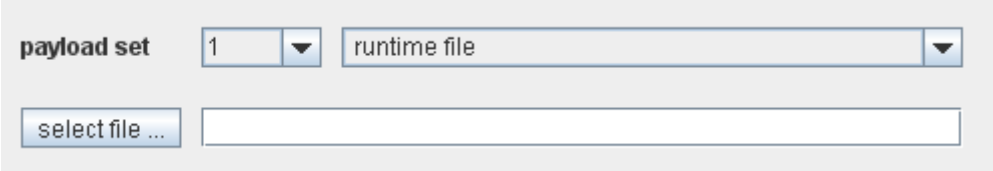
配置列表的主控制台在面板的右下方。可以通过文本框和”add”按钮手动地添加项。可以通过下拉菜单” add from list”来添加有用的有效负荷预设列表，包括常用的用户和密码，以及用来探测像 SQL 注入的这样常规漏洞的字符串。”load”按钮是用来导入文件的。”paste”按钮式用来添加粘贴板上的列表项。”delete”按钮删除选中的项，”clear”按钮删除列表里的所有项。

你可以通过” add from list”菜单自定义有效负荷的预设列表。首先，选择 Intruder 菜单里的” configure preset payload lists”，然后选择你自己的包含有效负荷文件的目录。你可以使用”copy”按钮把 Burp 的内置有效负荷列表复制到你自定义的目录，和你的有效负荷列表一起使用：



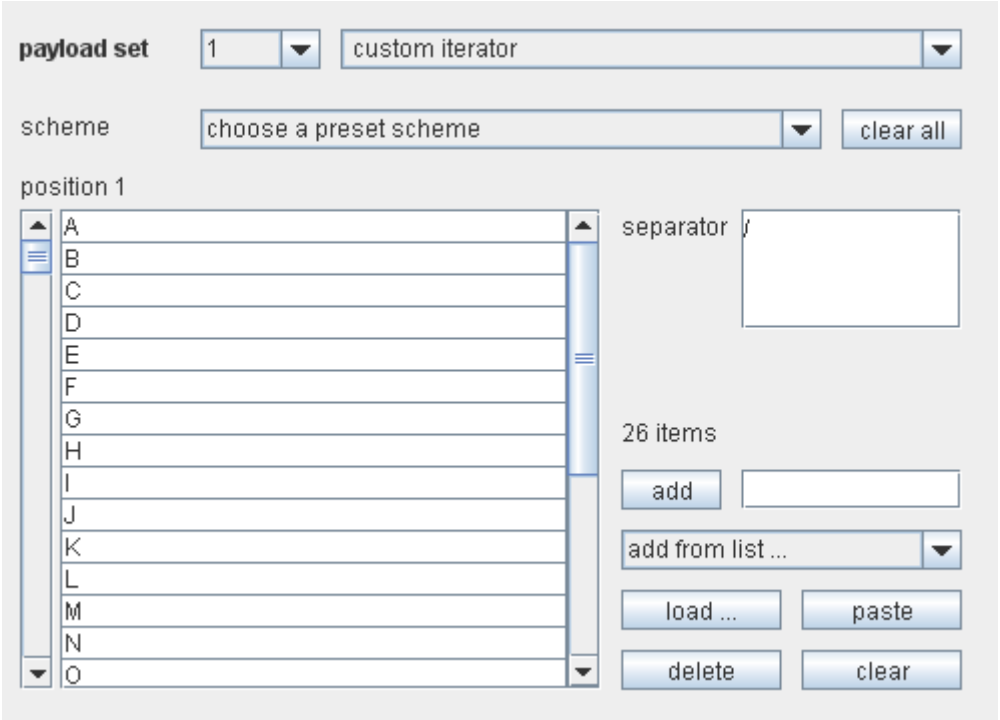
Runtime file

这个有效负荷源配置了一个外面的文本文件，在运行时可以通过这个文件读取里面的有效负荷。当需要一个非常大的预设有效负荷列表时，这很有用了，避免了把整个列表都放到内存里。从文件里的一行读取一个有效负荷，因此有效负荷不会包含换行符。



Custom iterator

这个有效负荷源提供了一种强大的方式，通过给出的模板，来产生自定义的字符和其他项的排列。例如，使用表单里个人编号 AB/12，工资应用程序来识别单个人员；你需要通过迭代所有的人员编号来获取所有个人细节。



自定义的迭代器定义的用来产生排列的位置不能超过 8 个。每个位置使用一个列表来配置，以及一个可选的分隔符，这个分隔符是被插入到那个位置和下一个之间。在上面的例子中，位置 1 和 2 是用 A-Z 项来配置，位置 3 和 4 用 0-9 项来配置，并且位置 2 会被设置分隔符/。当执行攻击时，自定义的迭代器会迭代每个位置里的项，以覆盖到所有的排列可能。因此，在这个例子中，有效负荷的总数量等于 26*26*10*10.

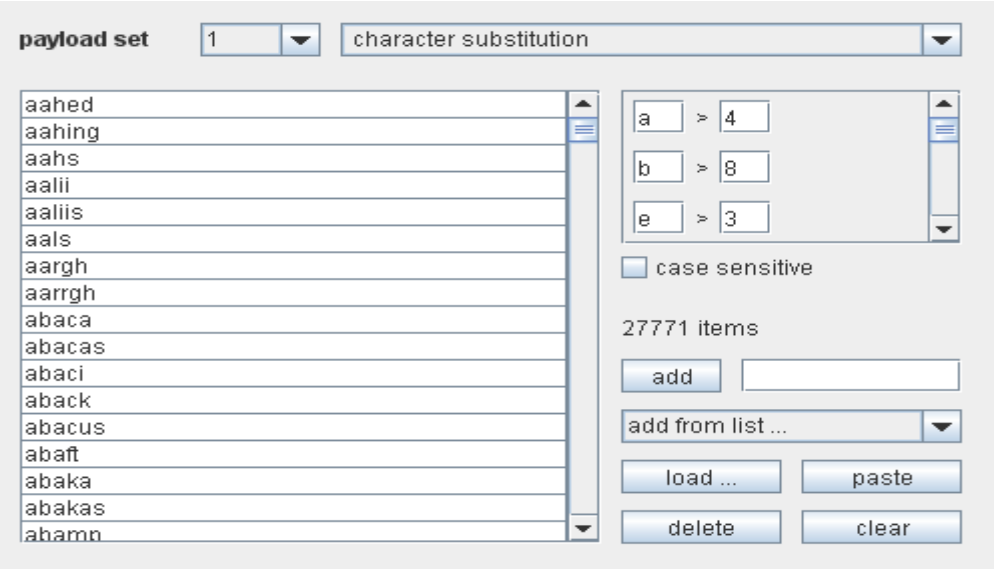
“scheme”下拉菜单用来为自定义迭代器选择一个预设的配置。这些可以被许多标准的攻击或者自定义攻击的修改使用。可用的 scheme 有”目录/文件扩展名”，这些可以被用来枚举 web 内容，”密码+数字”可以为密码猜测攻击提供一个扩展的字符列表。

在右下方得控制按钮是用来配置每个位置上的项。他们的功能和在 preset list source 中一样。”clear”按钮是删除自定义迭代器上所有位置里配置。

Character substitution

这个有效负荷源需要一个有效负荷项的预设列表，在自定义规则下，通过用不同字符把替换项里的单个字符，然后从这些项里产生许多有效负荷。这个有效载荷源在密码猜测攻击

中很有用。例如，在字典单词上产生许多变化：



在右下方的控制项是用来配置预设项的列表。它们的功能和上面的一样。

右面的复选框列表是用来配置提交规则。当攻击执行时，字符提交源会按顺序地处理每个预设项。对于每个项，它都会产生一些有效负荷，来包含通过提交规则提交的字符的所有排列。例如，对于上面截图里的第一个项，就会产生下面的有效负荷：

aahed

4ahed

a4hed

44hed

aah3d

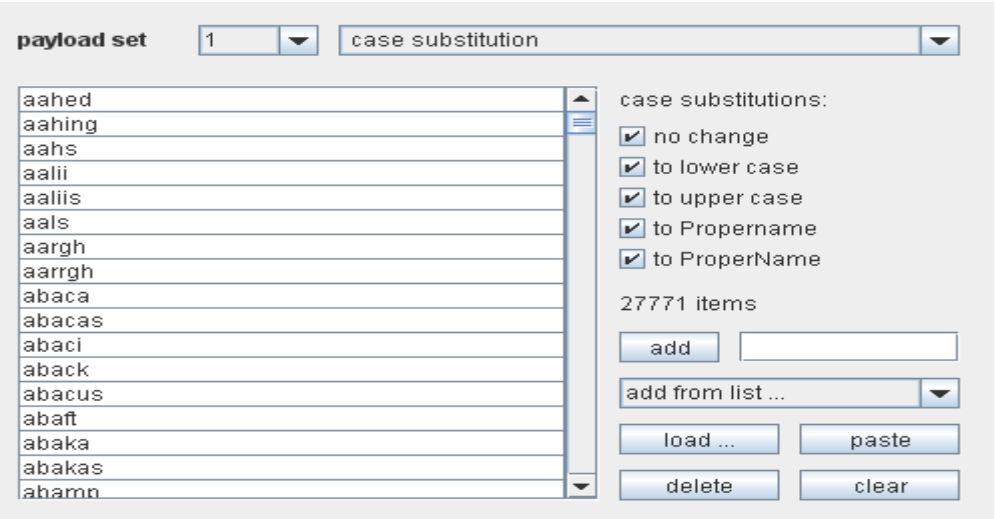
4ah3d

a4h3d

44h3d

Case substitution

这个有效负荷源需要一个有效负荷项的预设列表，调整每一项里的字符，从这些项里产生出一个或多个有效负荷。这个有效负荷源在猜测密码攻击时，会很有用。例如，在字典单词上产生许多变化：



右下方得控制项是用来配置预设项的列表。它的功能和上面的一样。

右边的复选框是用来配置提交规则。可用的规则执行下面的功能：

no change — 这个项是让不要修改有效负荷集合。

to lower case — 所有的字母被转换成小写的，并且结果添加到有效负荷集合。

to upper case — 所有的字母被转换成大写的，并且结果添加到有效负荷集合。

to Propername — 项里的第一个字母转换成大写，后面的字母转换成小写，结果添加到有效负荷集合。

to ProperName — 项里的第一个字母转换成大写，后面的字母不变，结果添加到有效负荷集合。

当攻击执行时，提交源会按顺序执行每一个预设项。对于每个项，它都会使用选中的提交规则产生一个有效负荷。如果这个规则产生了一个全新的有效负荷，它就会被加到有效负荷集合里(如，重复的有效负荷会被丢弃)。例如，对于上面截图里的第一个项，就会产生下面的有效负荷：

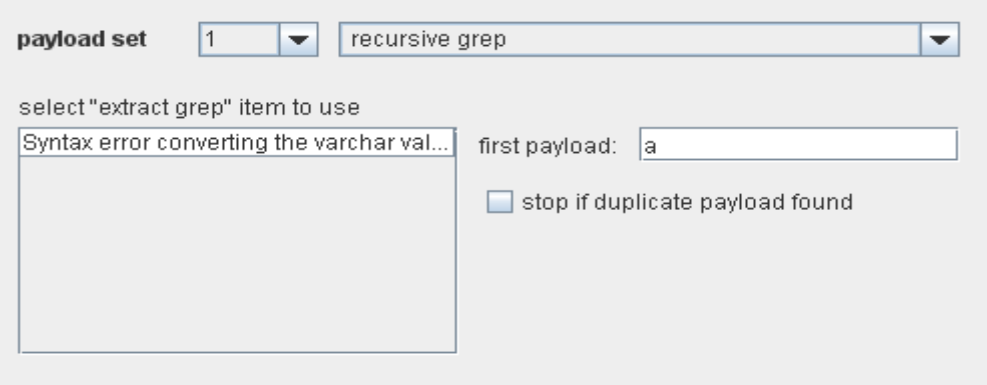
aahed

AAHED

Aahed

Recursive grep

这个有效负荷源带有”extract grep”功能(下面会介绍)。它允许通过对早期请求的响应来递归地产生有效负荷。”extract grep”功能会通过匹配一个正则表达式来捕获一个服务器响应的一部分。和"recursive grep"一起，从先前的服务器响应捕获的文本将用作后续请求的有效负荷。



这可以用在许多的枚举任务里。例如，可以用来通过 SQL 注入来枚举数据库的内容，递归查询格式为:

union select name from sysobjects where name>'a'

服务器的错误信息会泄露一个数据库对象的名字：

Syntax error converting the varchar value 'accounts' to a column of data type int.

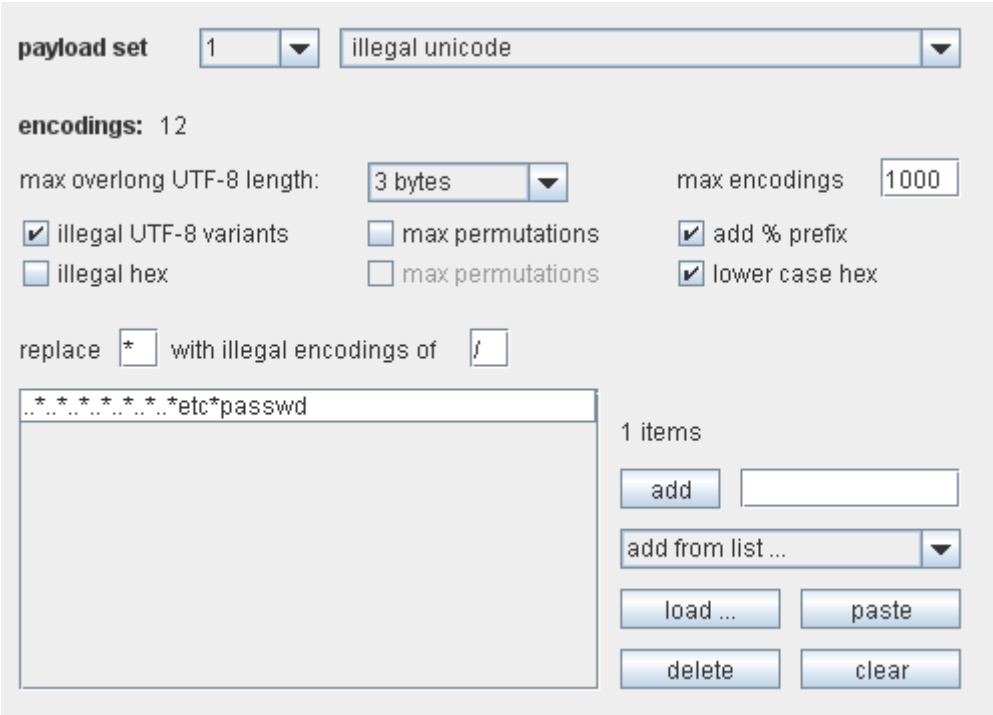
使用’ accounts’，再查询确认下一个对象。使用 recursive grep 有效负荷可以简单地自动执行这个任务，进而快速地枚举出数据库里的所有对象。

有效负荷里使用的第一个请求必须是手动指定的。当重复连续地发现 recursive grep 项时，就可以停止了，因为这就说明了枚举已经完成。注意由于有效载荷的本质属性，使用它的攻击就不能进行多线程请求。

Illegal Unicode

这个有效负荷源需要一个有效负荷项的预设列表，在每一项里用指定字符的非法 Unicode 编码替换字符本身，从这些项里产生出一个或者多个有效负荷。在尝试回避基于模

式匹配的输入验证时，这个有效负荷会有用的，例如，在防御目录遍历攻击时../和\..序列的期望编码的匹配。



右下方的控制项是用来配置预设项的列表。它的功能和上面的一样。

顶端的 2 个文本框是来配置在每个预设项里提交的字符(*那里)，以及为非法编作基础的字符(/那里)。可以通过字母的 ASCII 编码或者 2 位十六进制编码来指定这个字母 — 这对于指定那些无法打印的字符是很有用的，如空字符。

中间的控制项是用来产生非法编码的类型。在下面解释：

maximum overlong UTF-8 length Unicode 编码方案允许最多使用 6 字节表示一个字符。使用一种类型就可以正确地表示出(0x00-0x7F) Basic ASCII 字符。然而，使用多字节的 Unicode 方案也能表示出它们(如，”overlong”编码)。下拉菜单用来指定是否使用超长编码，以及应该设定的最大使用型号。

illegal UTF-8 variants 如果选择的最大超长 UTF-8 长度为 2 字节以上，这个选项是可用的。当使用多字节编码一个字符时，第一个字节后面的字节应该用 10XXXXXX 这样的二进制格式，来指出后续的字节。然而，第一个字节里最有意义的位会指出后面还有多少后续字节。因此，Unicode 编码例程会安全地忽略掉后续字节的前 2 位。这就意味着每个后续字节可能有 3 个非法变种，格式为 00XXXXXX，01XXXXXX 和 11XXXXXX。如果选中这个选项，则非法 Unicode 有效负荷源会为每个后续字节生成 3 个附加编码。

max permutations 如果选择的最大超长 UTF-8 长度为 3 字节以上，这个选项可用，并且选中” illegal UTF-8 variants”。如果” max permutations”没被选中，则在生产非法变种时，非法 Unicode 有效负荷源会按顺序处理每个后续字节。为每个后续字节产生 3 个非法变种，并且其他的后续字节不会改变。如果” max permutations”被选中了，然而，非法 Unicode 有效负荷源会为后续字节生成所有的非法变种排序 — 如，多个后续字节会同时被修改。在目标系统上回避高级模式匹配控制时，这个功能就会很有用。

illegal hex 这个选择基本上一直可用。当使用超长编码和后续字节的非法变种(如果选中)生成非法编码项列表时，通过修改由此产生的十六进制编码可能会迷惑到某种模式匹配控制。十六进制编码使用字符 A—F 代表十进制 10—15 的值。然而有些十六进制编码会把 G

解释为 16，H 为 17，等等。因此 0x1G 会被解释为 32。另外，如果非法的十六进制字符使用在一个 2 位数的十六进制编码的第一个位置，则由此产生的编码就会溢出单个字节的大小，并且有些十六进制编码只使用了结果数字的后 8 个有效位，因此 0x1G 会被解码为 257，而那时会被解释为 1。每个合法的 2 位数的十六进制编码有 4—6 种相关的非法十六进制表示，如果使用的是上面的编码，则这些表示会被解释为同一种十六进制编码。如果"illegal hex"被选中，则非法 Unicode 有效负荷源会在非法编码项列表里，生成每个字节的所有可能的非法十六进制编码。

max permutations 如果选中的最大超长 UTF-8 长度为 2 字节以上并且" illegal hex"也被选中，则这个选项可用。如果" max permutations"没被选中，在生成非法十六进制编码时，非法 Unicode 有效负荷源会按顺序处理每个字节。对于每个字节，会生成 4—6 个非法十六进制编码，其他的字节不变。如果" max permutations"被选中，然而，非法 Unicode 有效负荷源会为所有的字节，生成非法十六进制的所有排序 — 如，多个字节会被同时修改。在目标系统上回避高级模式匹配控制时，这个功能会非常有用。

add % prefix 如果选中这个选项，在产生的有效负荷里的每个 2 位数十六进制编码前面，都会插入一个%符号。

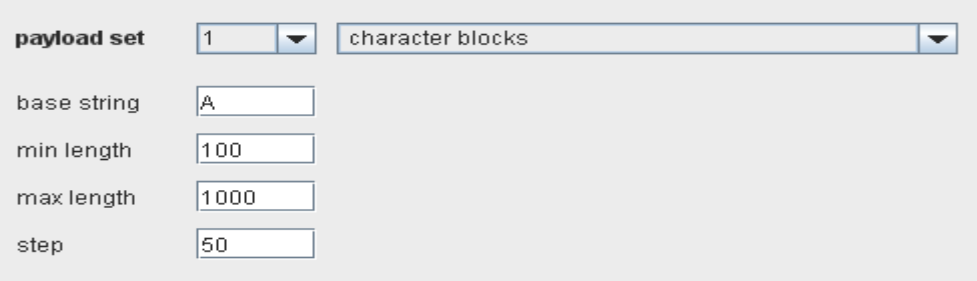
lower case hex 这个选项决定了是否在十六进制编码里使用大小写字母。

max encodings 这个选项为会产生的非法编码数量放置了一个上界。如果大量使用超长编码或者选中了最大排序，这个选项会很有用，因为那会生成大量的非法编码。

当攻击执行时，这个有效负荷源会迭代所有预设项列表，在非法编码集合里，每个预设项替换每个项里的指定字符的所有实例。

Character blocks

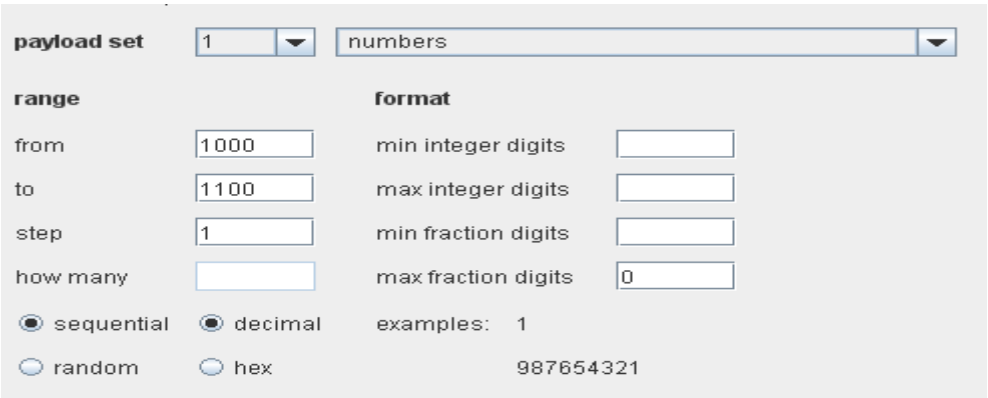
使用一个给出的输入字符串，这个有效负荷源产生指定大小的字符块。在对本机(非托管)上运行的软件进行探测缓冲区溢出和其他边界条件漏洞时，这个选项很有用。



'string'字段指定了输入字符串，从这里产生字符块。'min'和'max'字段指定了产生字符块的最小和最大长度。'step'字段指定了每个字符块的长度增量。

Numbers

这个有效负荷产生的数字，是顺序的或者随机的，使用一个指定的格式：

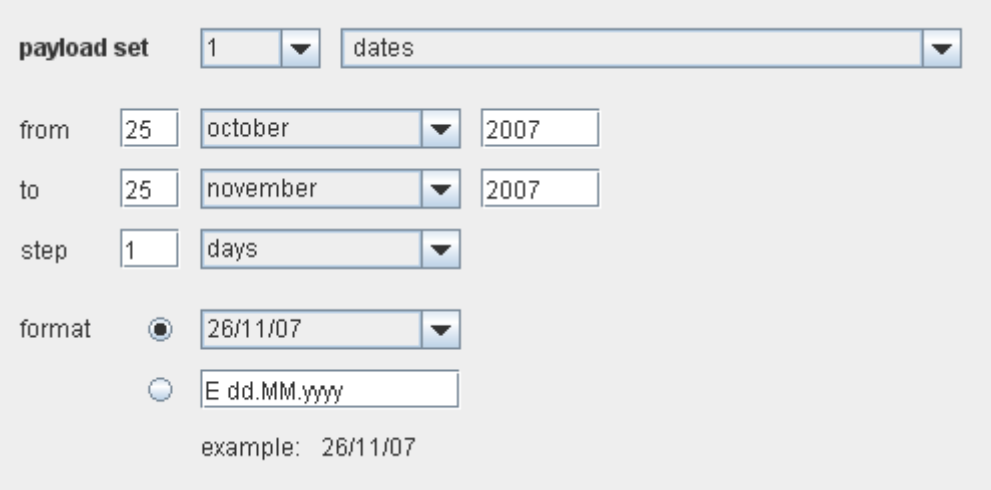


‘from’和‘to’字段指定了产生的最小和最大的数。如果选中‘sequential’，数字就以‘from’字段里的值作为起点，以‘step’字段里的值做为增量。如果选中‘random’，‘howmany’字段指定生成数字的数量。数字是以十进制或者十六进制的格式生成。如果选中十六进制，则‘form’，‘to’以及‘step’字段必须是十六进制整数；否则是十进制的整数或分数。

在右手边的控制项指定了要用到的数字格式。

Dates

这个有效负荷在一个指定的范围内，在一个指定的间隔内，以一个指定的格式，产生日期。这个选项在数据挖掘(拖出在不同天里的订单条目)或者暴力攻击(猜测一个由生日组成的用户认证)中有用。

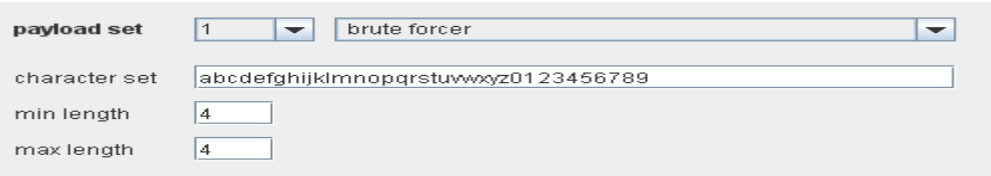


产生的日期是从‘from’控制项里指定的日期开始，使用‘step’控制项里指定的间隔来增加，直到或者包含了‘to’控制项里的指定日期。在‘format’下拉菜单里可以选择一些预设的日期格式，或者自定义一个可以在文本字段里输入的日期格式。下面的例子说明了可以用来指定自定义的日期格式的编码：

E Sat
EEEE Saturday
d 7
dd 07
M 6
MM 06
MMM Jun
MMMM June
yy 03
yyyy 2003
/ . : etc / . :

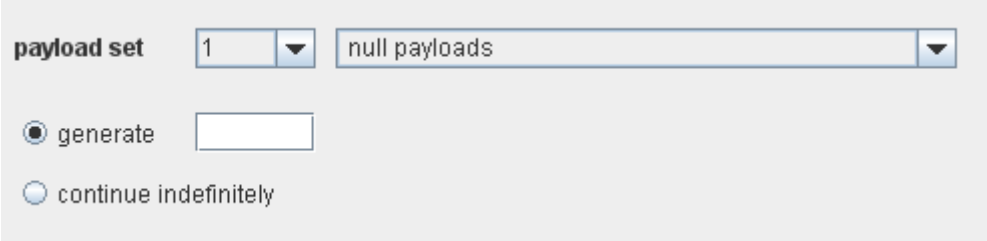
Brute forcer

这个有效载荷源产生一个指定长度的有效载荷集合，这里面包含了指定字符集的所有排序可能。



Null payloads

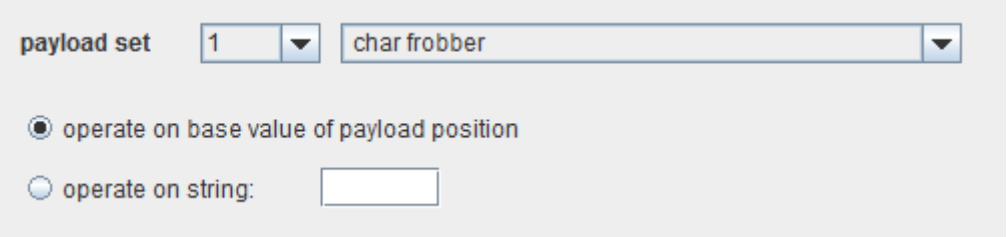
这个有效载荷源能产生‘null’有效载荷 — 如，0 长度的字符串。它可以产生一个指定数量的空有效载荷，也可一直继续下去。



当一次攻击需要重复同一请求，而不是做修改的基础模板，这个有效载荷就有用了。要完成这个工作，把一对位置标记一起放在请求模板的一个位置。这个可以在许多攻击中使用，例如测序分析中获取 cookie，应用层的拒绝服务攻击，这里通过重复发送请求以在服务器上执行一个超载任务，保持一个用在其他地方进行断断续续测试的会话令牌。

Char frobber

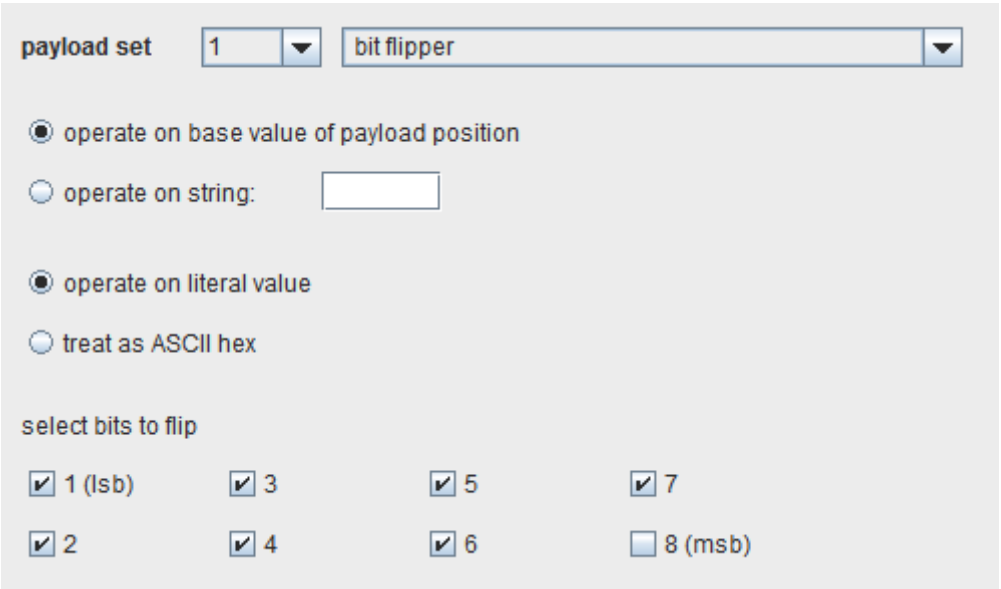
这个有效载荷是在每个有效载荷位置现有基值或者知道的字符串上进行操作的，它通过一次一个字符，在基类字符串上循环，把那个字符 ASCII 码加 1。



当测试参数值或部分值哪个更影响应用程序的响应，这个有效载荷源就有用了。特别是在测试复杂的会话令牌的哪一部分实际上是用来跟踪会话状态的。如果你修改会话令牌里的单个字符的值后，在你的会话里，请求仍然能被处理，则就有可能是这个字符不是用来跟踪你的会话的。

Bit flipper

这个有效载荷是在每个有效载荷位置现有基值或者知道的字符串上进行操作的，它通过一次一个字符，在基类字符串上循环，翻转顺序上的每(指定的)一位。



你可以配置位翻转要么为了操作字面基础值，要么把基础值当成 ASCII 十六进制字符串。例如，如果基础值是‘ab’，则通过字面字符串操作和翻转所有的位，产生下面的有效载荷：

`b
cb
eb
ib
qb
Ab
!b
ább
ac
a`
af
aj
ar
aB
a"
a â

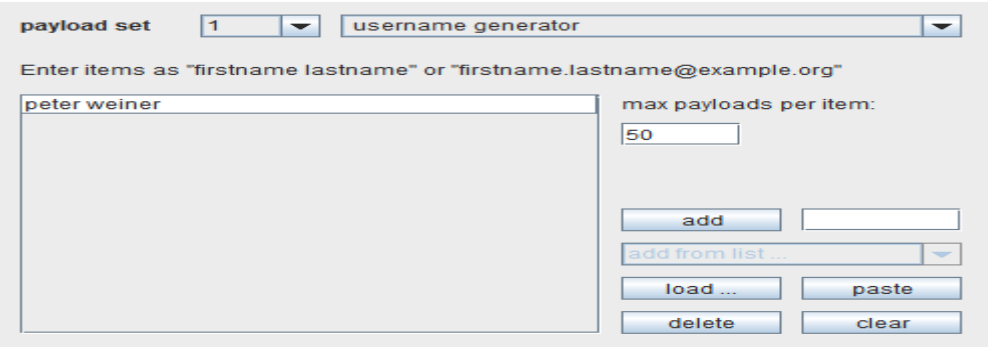
如果是‘ab’当做一个 ASCII 十六进制字符串，然后翻转所有位，就得到下面有效载荷：

aa
a9
af
a3
bb
8b
eb
2b

这个有效载荷源非常有用在对 char frobber 相似的情况下，但你需要有细微控制权。例如，如果一个会话令牌或者其他参数值包含一个有意义的数据，但这个数据使用了一种 CBC 模式密码加密了，就由可能通过修改前面密码块里的位来系统地改变部分加密数据。在这种情况下，你可以在有效载荷源上使用位翻转来确定修改加密值里的单个位的影响，弄清楚应用程序是否有漏洞。

Username generator

这个有效载荷源需要人名作为输入，使用许多常规方案来产生用户名。



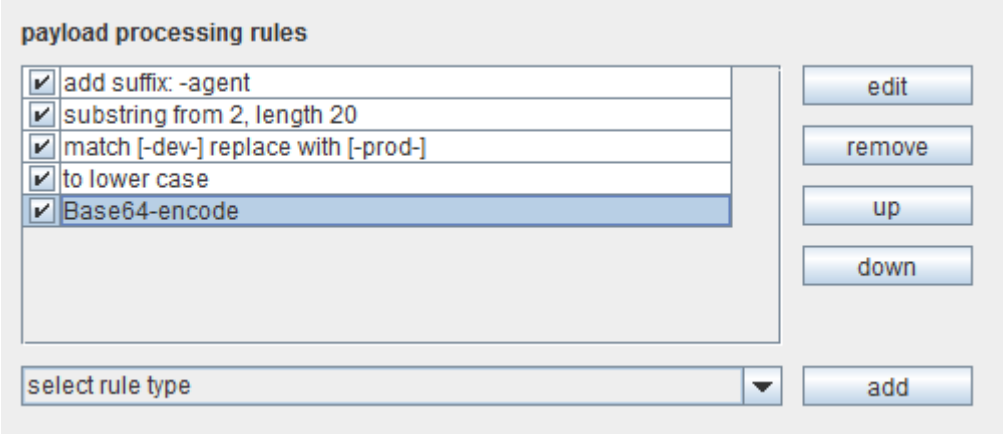
例如，提供‘peter weiner’这个名字，就会产生 115 个可能的用户名，如下：

peterweiner
peter.weiner
weinerpeter
weiner.peter
peter
weiner
peterw
peter.w
wpeter
w.peter
pweiner
p.weiner
weinerp
weiner.p
等等。。

如果你的目标是一个特殊人物的用户，这个有效载荷源就有用了，并且你不知道应用程序里使用的用户名或电子邮件地址方案。

有效载荷的处理(Payload processing)

对于每个有效载荷集合，除了使用有效载荷的‘源’，在每个有效载荷上可能执行定义的许多附加处理过程。选中有效载荷源执行所有操作后，这个处理过程就开始了：

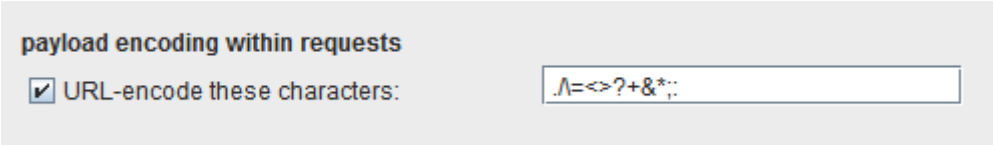


执行队列中的定义规则，并且通过切换开关来帮助排除配置中的问题。可用下面类型的规则:

1.添加前缀
2.添加后缀
3.匹配/替换
4.子字符串(从一个指定的偏移到一个指定的长度)
5.转换子字符串(从有效载荷的结尾索引的子字符串)
6.修改状况(一些为 case substitution 的有效载荷源选项)
7.编码(URL,HTML,Base64,ASCII 十六进制以及多平台的组合字符串)
8.解码(URL,HTML,Base64,ASCII 十六进制以及多平台的组合字符串)
9.哈希表
10.添加原来的有效负荷(如果你需要在同一个有效载荷里包含原来的和哈希格式，这会

有用)。

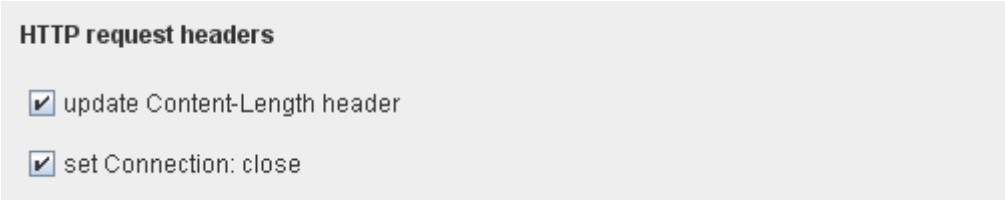
最后，你可以配置最终的有效载荷里的字符，把它进行 URL 编码，这样在 HTTP 请求里更安全地传输。



建议使用最终的 URL 编码配置，而不是一个有效载荷过度规则。因为 payload grep 选项是用来在使用最终的 URL 编码之前为显示的有效载荷检查响应。

选项卡(Options tab)

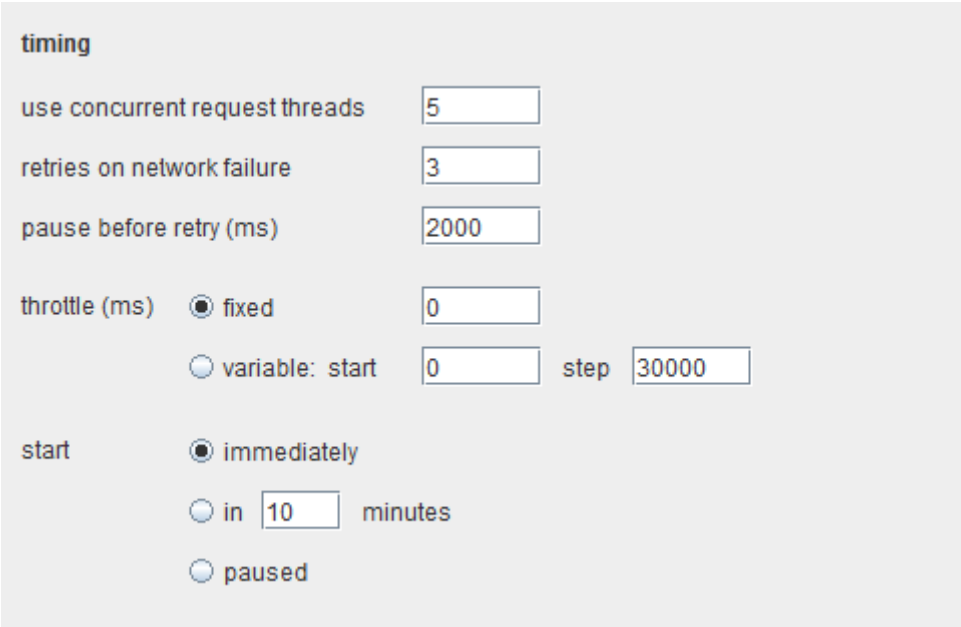
这个选项里包含了许多配置控制着单个攻击行为的选项。



如果选中‘update Content-Length header’框，Burp Intruder 会使用每个请求的 HTTP 主体长度的正确值，添加或更新这个请求里 HTTP 消息头的内容长度。这个功能对一些需要把可变长度的有效载荷插入到 HTTP 请求模板主体的攻击是很有必要的。这个 HTTP 规范和大多数 web 服务器一样，需要使用消息头内容长度来指定 HTTP 主体长度的正确值。如果没有指定正确值，目标服务器会返回一个错误，也可能返回一个未完成的请求，也可能无限期地等待接收请求里的进一步数据。

如果选中‘set Connection: close’框，则 Burp Intruder 会添加或更新 HTTP 消息头的连接来请求在每个请求后已关闭的连接。在多数情况下，这个选项会让攻击执行得更快。

注意：早期的 Burp Intruder 版本在这里包含往请求里添加 cookie 头的选项，这是依据不同请求的响应。现在这些配置被删除了，你可以使用 suite-wide session handling support 来代替。



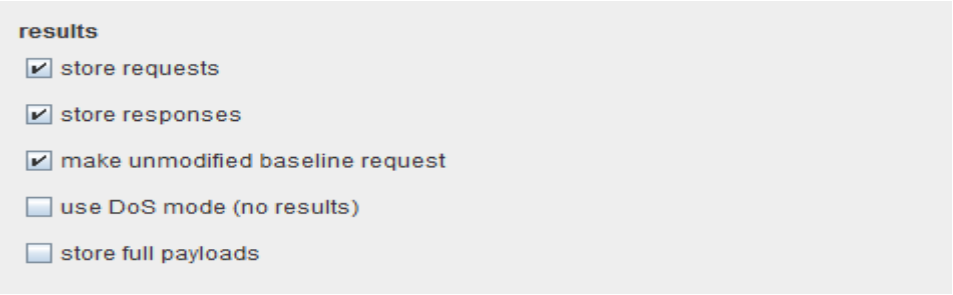
concurrent threads 设置决定了攻击是否使用单线程或多线程来同步地加载请求。使用多

线程能迅速地加快一次大型攻击，影响时间的主要因素就是处理请求和接收响应之间的延时。这可以用来测试应用程序漏洞的并发处理。这也可用来增加应用层拒绝服务的效果。

retry 设置决定了如果产生网络错误(如，连接被拒绝或超时)，Burp 会重发一个请求的次数，以及等待的时间间隔。

throttle 设置用来配置请求之间需要的延时。可能会需要一个固定的延时作为隐形的防护措施，来保留带宽和处理能力，以避免影响其他活动，这样就可以定期执行请求操作，如保持一个断断续续地用在其他地方测试的会话令牌存活。一个可变的延时会对自动探测会话超时值很有用。

start 设置决定了攻击在加载时是否立即执行，或者在一个指定的延时后开始，或者一直等到选中‘恢复’命令。如果配置的一个攻击需要在一些未知点上执行，或者为以后的使用保存，那么这个功能就有用了。

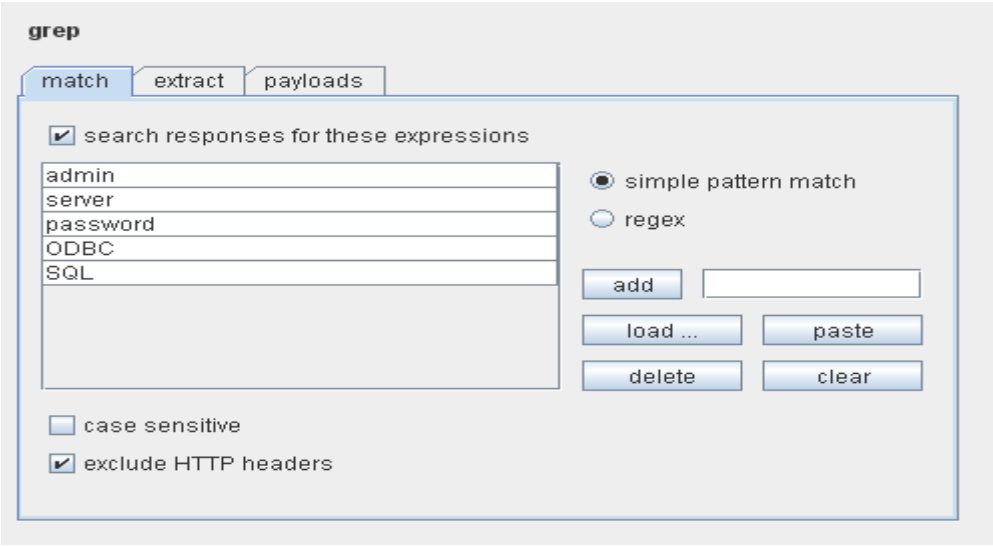


storage 设置决定了攻击是否会保存单个请求和响应的内容。保存请求和响应需要在消耗临时目录里的磁盘空间，但能让你在攻击时完整地查看它们，如果需要可以重复发请求，也可以把它们发送到其他 Burp 工具上。

如果选中‘make unmodified baseline request’，这时除了配置攻击请求，Burp 还会除了模板请求，使用所有有效载荷位置来设置它们的基础值。这样请求在结果表格里会以#0 项显示出来。

如果选中‘DoS mode’，则攻击会和平常一样地处理请求，但不会等待处理服务器返回的响应。当每个请求都处理完后，关闭 TCP 连接。这个功能可以通过重复地发送请求，使服务器执行超负荷任务，来对有漏洞的应用程序执行应用层的拒绝服务攻击。

如果选中‘store full payloads’，Burp 会完整地保存每一个结果的有效载荷值。这个选项会消耗一些内存，如果你想在运行时执行某种操作，这个就可能是需要的了，如修改有效载荷的 grep 设置，重新处理一个使用修改请求模板的请求。



‘grep’设置是用在运行时，配置在服务器响应里执行的模式匹配测试。这有 3 种测试类型：

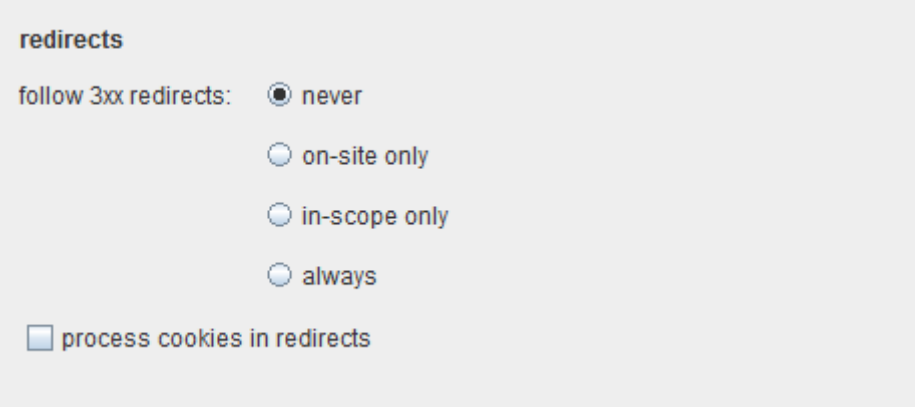
1. match grep — 这个是用来检查每个服务器响应里指定表达式，要么简单的模式匹配，要么 Perl—like 正则表达式。对于每个指定的表达式，攻击会在结果表里包含一个列来指明是否找到一个匹配。基本功能有广泛的应用，例如：密码猜测攻击，扫描‘密码错误’或者‘登陆成功’的短语，SQL 注入漏洞测试，扫描包含‘ODBC’‘error’的消息，等等。

如果使用正则表达式来匹配表达式，可能会包含换行符。

2. extract grep — 这个是用来检查每个服务器响应里指定表达式，是否存在紧随匹配表达式的要提取的文本(知道指定的符号或者最大长度)。对于每个指定的表达式，攻击都在结果列表里包含一个从服务器响应里提取的文本的列。这个功能可以用来进行数据挖掘，通过数据挖掘能获得 web 页面里的有用信息，并且需要一个提取这些信息的自动化方法。例如，如果你获得了一个通向管理员用户的页面，通过它能够修改由 URL 查询字符串指定的 ID 的用户的账户信息，这时重复地通过用户 ID 来提取每一位用户的用户名和密码。

3. payload grep — 这个是用来检查每个服务器响应中的用在相关请求中的有效载荷字符串。这个功能在探测跨站点脚本和其他响应注入漏洞中会有用。这漏洞产生在用户把输入动态插入到应用程序的响应里。

如果选中‘match against pre-encoded payloads’，则会搜索响应里的每一个应用编码之前的有效载荷字符串的原来格式。设置的这些常常都是很有必要的，例如，如果你使用 XSS 测试有效载荷里的印刷字符，这通常需要在有效载荷处理选项里的 URL 编码，但如果应用程序有漏洞，就会在响应里显示出每个编码格式。



这重定向设置控制了 Burp Intruder 在执行攻击时，是否跟踪 HTTP 重定向(如，有 3xx 状态码和包含一个新 URL 的 Location header)。如果配置跟踪重定向，则 Intruder 在接收到一个重定向时，会请求这个重定向 URL(如果需要，最多跟踪 10 个重定向)，并在结果里记录下后续响应的细节。在结果表格的一个列里会显示是否为每个结果跟踪重定向。你可以配置是否只跟踪站点(如，相同协议，主机和端口)重定向，或者只跟踪范围内(在目标范围定义的)重定向，或者跟踪所有的。

当一个应用程序对许多类型的输入都返回一个 3XX 的响应，这个跟踪重定向的选项就会有用，在请求重定向目标时，会返回应用程序处理请求的感兴趣的特征。例如，当使用模糊技术测试常规漏洞时，应用程序会频繁地返回一个到错误页面的重定向，这个页面可能包含了关于错误本质的有用信息，通过这个错误能诊断出像 SQL 注入这样的问题。

注意在跟踪重定向时，大多数情况下，要使用单线程进行攻击，例如，如果应用程序存储了下一个指向重定向目标的请求返回的会话信息。同样要注意自动地跟踪重定向有时会给你的攻击带来一些麻烦 — 例如，一些应用程序会对一些恶意请求响应到一个注销页面的重定向，这时跟踪重定向会导致你的会话终止，然而它不来不应该这样做。

如果选中‘process cookies in redirects’，当跟踪重定向上时，任意设置为 3XX 响应 cookie 都会被提交。例如，当你尝试暴力进行登录挑战时，通过重定向页面来指示登录结果，并且每次的登录尝试都会有创建一个新的会话，这个就需要这个选项了。

启动一次攻击(Launching an attack)

为了创建一次新的攻击，使用控制面板选项来设置需要的配置信息，然后选择 Intruder 菜单上的‘start attack’。要想加载一个保存的攻击，选择 Intruder 菜单上的‘open saved attack’，然后选择所需的文件。

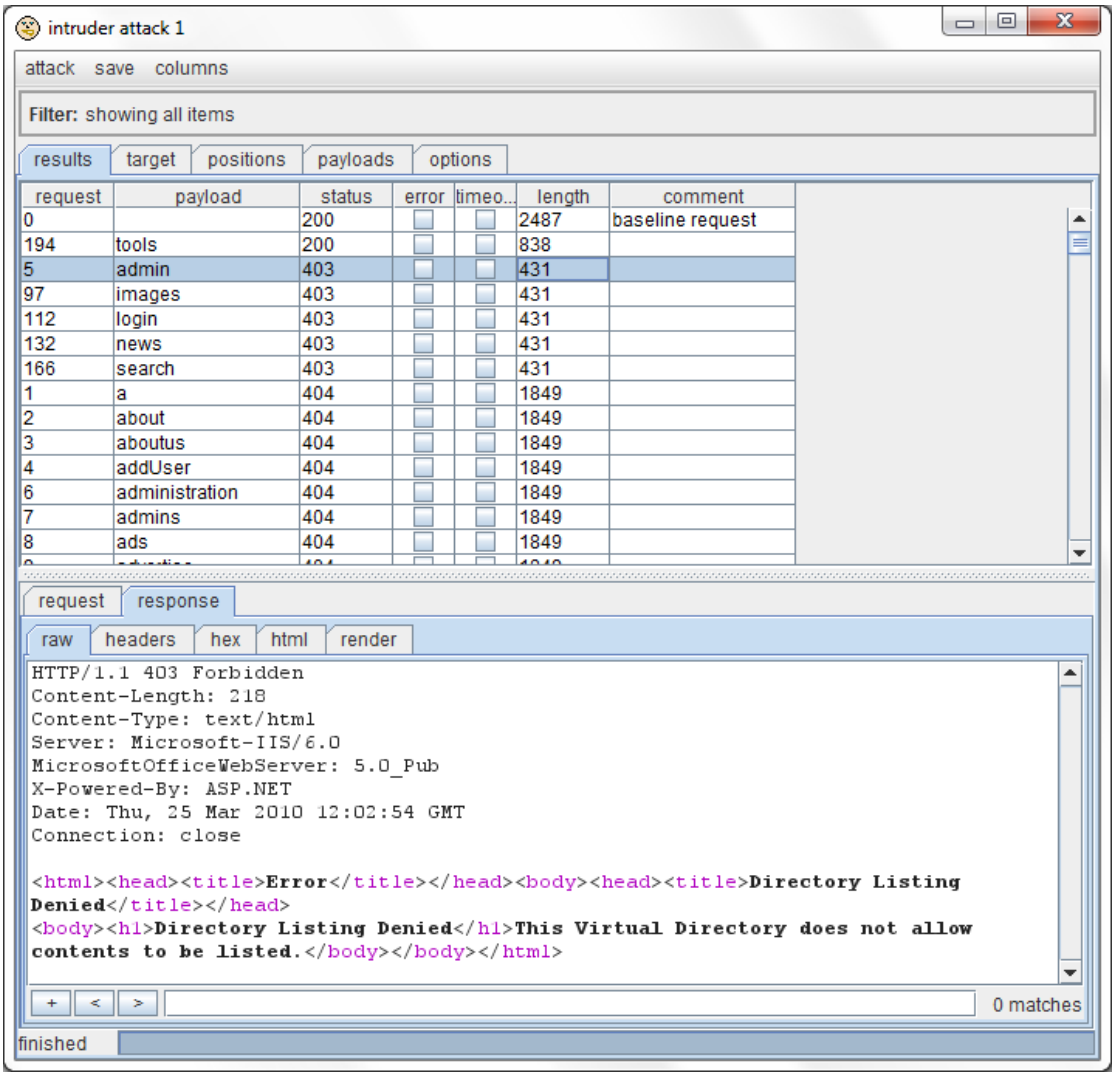
当执行一次新的攻击时，会在指定的配置信息上执行许多验证审查。这包括验证有效载荷位置和有效载荷集定义是否正确，时序和 grep 设置是否可行，等等。有些故障会产生一些阻碍执行攻击的错误；产生的其他警告则会被忽略。

每次攻击都打开了一个分离的窗口。这个窗口显示出攻击产生的结果，使你能够实时修改攻击配置，同样也包含了的一系列控制选项，有保存结果，服务器响应和攻击自身。

注意：当修改一个正在运行的攻击配置时，你应该小心地处理，在修改之前最好暂停。

结果选项(Results tab)

下面显示的就是一次执行的攻击的结果视图例子，在目标 web 站点上枚举的基本内容：



这次攻击使用了 sniper 攻击类型，对一系列常用的 web 目录进行请求。对于这次攻击，结果视图显示了默认的每次请求次数，使用的有效载荷位置，插入的有效载荷，从服务器上接收的 HTTP 状态码，是否产生了超时或错误，以及服务器响应的长度。附加的结果列显示

了，每个请求的"received response"和"finished response"计时器，以及接收到的 cookie。许多配置选项，如 grep 功能，这会让附加列在结果视图里显示出来。使用‘view’菜单来隐藏和显示这些列。通过单击结果列的相关标题头，来对整个结果集进行排序(按住 shift 单击可以转换排序方式)。按住 Ctrl+单击标题可以复制该列的内容。[专业版]

一个对攻击结果进行有效解释的关键部分，定位在感兴趣或成功的服务器响应，以及产生这些响应的请求。通常情况下感兴趣的响应分为下面几种：

1.一个不同的 HTTP 状态码。

2.一个不同长度的响应。

3.某个表达式存在或不存在。

4.一个错误或超时的产生。

5.接收或完成响应的时间。

例如，在一个探索内容的测试中，请求一个现存的资源，会返回一个长短不一的‘200 OK’的响应，然而请求一个不存在的资源时，会返回一个‘404 not found’的响应，或者返回一个包含固定长度的自定义出错页面。在一次密码猜测攻击中，登陆失败会产生一个包含关键字‘login failed’的‘200 OK’的响应，然而成功的登陆会产生一个‘302 Object moved’的响应，或者一个包含关键字‘welcome’的不同长度的’200 OK’的响应。

Burp Intruder 可以提供一些帮助来确认上面的那些区别。grep 功能可以用来标记那些已知关键字的响应，或者从页面的关键部分提取感兴趣的信息。在结果视图里，可以通过单击列标题来对结果排序，或者按住 shift+点击标题来倒置排序。在上面的例子里，HTTP 状态码是感兴趣结果的主要区别，并且可以通过排序查明这些。

你可以通过添加批注和加亮来注释一个或多个项：

request	payload	status	error	timeo...	length	comment
0		200			2487	baseline request
194	tools	200			838	
5	admin	403			431	unprotected admin interface
97	images	403			431	nothing of interest
112	login	403			431	public login
132	news	403			431	old content - buggy
166	search	403			431	
1	a	404			1849	

你可以使用最左边度的列表里的下拉菜单来加亮单个项：

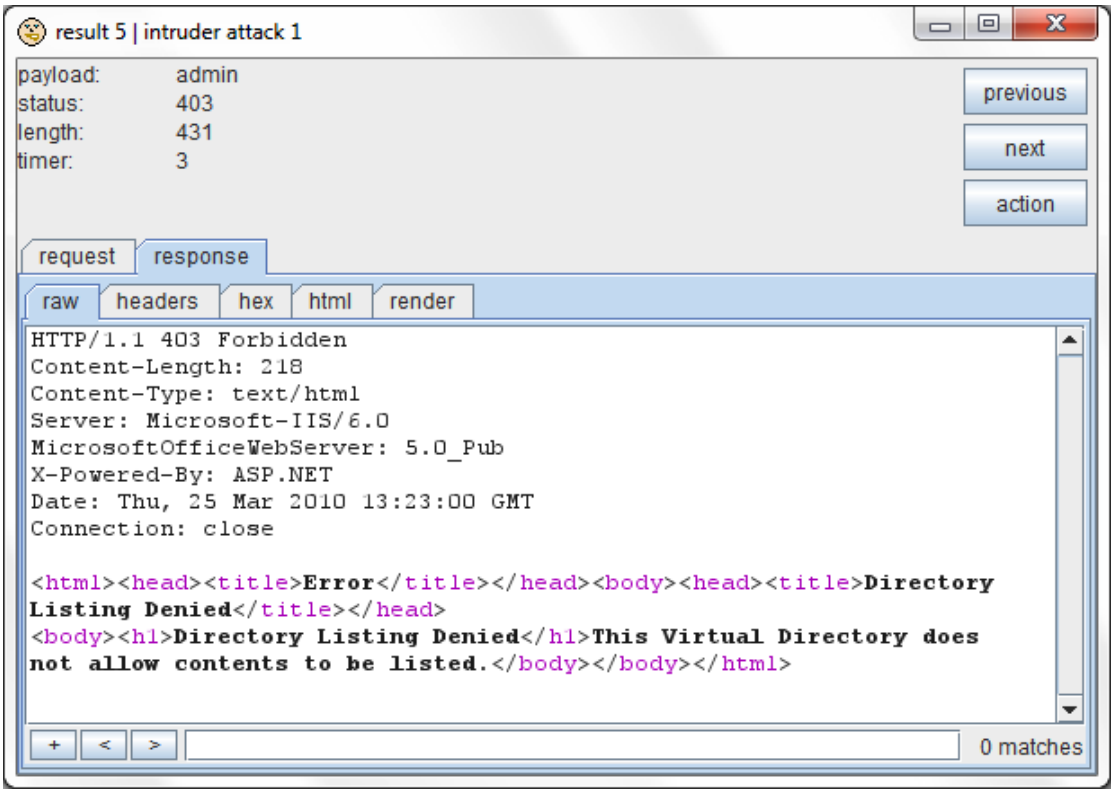
request	payload	status	error	timeo...	length	comment
0		200			2487	baseline request
194	tools	200			838	
5	admin	403			431	unprotected admin interface
5	images	403			431	nothing of interest
5	login	403			431	public login
5	news	403			431	old content - buggy
5	search	403			431	
5	a	404			1849	
5	about	404			1849	
5	aboutus	404			1849	
5	addUser	404			1849	
5	administration	404			1849	
5	admins	404			1849	
8	ads	404			1849	

你可以通过双击并编辑单元格就地对单个项进行注释：

request	payload	status	error	timeo...	length	comment
0		200			2487	baseline request
194	tools	200			838	
5	admin	403			431	unprotected admin interface
97	images	403			431	nothing of interest
112	login	403			431	public login
132	news	403			431	old content - buggy
166	search	403			431	
1	a	404			1849	

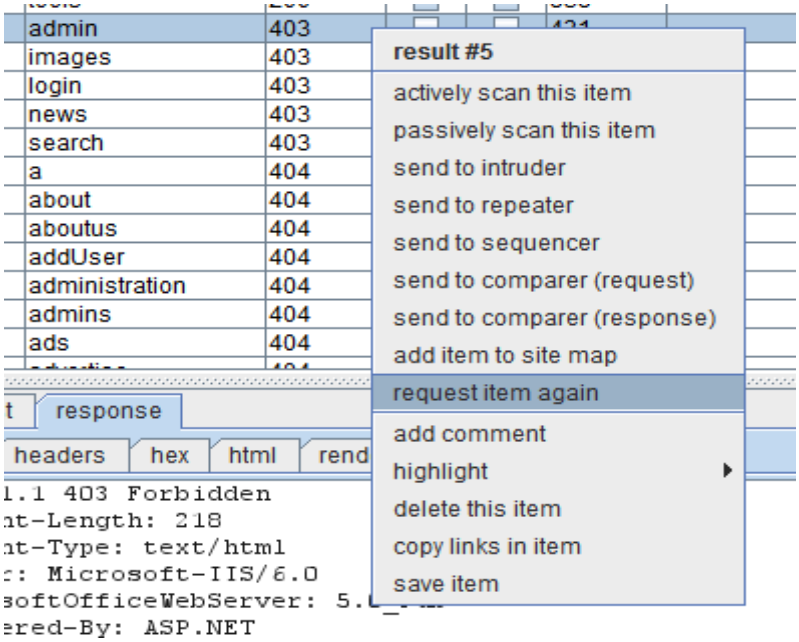
当你已经注释好感兴趣的请求时，以后你就可以使用列排序和显示过滤器来快速地找到这些项。

如果攻击被配置为保存请求或/和响应，你就可以通过预览表格来查看这些，或者通过双击来显示这些请求和响应的细节。这些显示提供了细节分析和呈现出每个 HTTP 消息。"previous"和"next"按钮可以用来循环结果集。如果结果视图里的表已被排序，则结果会以当前序列显示在那个视图里。



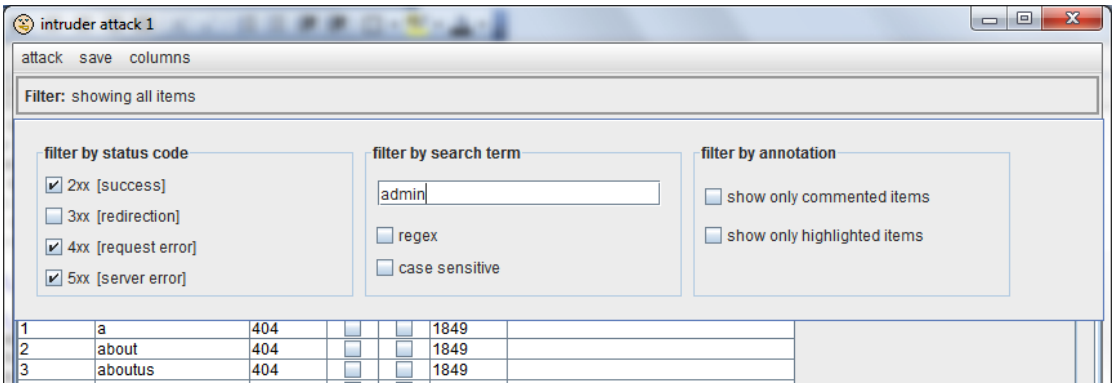
如果攻击是被配置来跟踪重定向，在初始清求和最后响应的旁边，所有的中间响应和请求都会被显示出来。

如果你使用’action’按钮来发送请求或响应到其他 Burp Suite 工具里，如 Repeater。你可以在结果表里的项上右击来显示一个有许多选项的上下文菜单：



你可以发送选中的项到其他工具，添加多个项到 Suite 站点地图上，使用批注和加亮来注释这些项，或者把项标记为重新请求。如果网路错误或其他问题影响了一些结果，这个选项就会有用了。如果你在攻击时修改了一些基础请求模板或者其他的项，如果可以，要被重发的请求会使用当前的配置进行重组。于是，例如，如果你的应用程序会话在攻击中被部分地终止了，你可以修改基础会话模板使用一个新的会话令牌，并重新处理任何失败的请求，使用你的新会话来执行。

在结果表的顶端是一个过滤栏，你可以使用它来隐藏一些结果，有 HTTP 状态码，搜索项，以及用户使用的注释：



和过滤器一样，通过选中结果表里的一个或多个项，然后选择上下文菜单里的'delete'，你可以永久地删除结果里的项。

结果菜单(Results menus)

结果视图里包含了许多有控制攻击命令的菜单，和保存结果，服务器响应和攻击自身。它们将在下面介绍。

攻击菜单(Attack menu)

这里包含了暂停，继续，重复攻击的指令。

保存菜单(Save menu)

attack — 这个是用来保存当前攻击的一个副本以及结果。从 Burp Intruder 控制面板里，加载保存文件作进一步使用。

results table — 这个是用来把结果列表保存为一个文本文件。可以选择单个的行或列，以及整个表来保存。可以配置区域分隔符。这个功能是用于把结果导出到一个电子表格上作进一步分析，或者保存单个列(如使用提取 grep 功能)到一个输入文件里，用于作进一步攻击或者其他工具上。

server response — 这个是用来保存接收到的服务器对请求作出的完整响应。这些可以到自己的文件(按顺序编号)里，也可以串联到序列里的单个文件。

attck configuration — 这是用来保存当前执行攻击(不是结果)的配置，使你能够加载那个配置到 Intruder 主控制面板来配置出一个相同或相似的攻击。

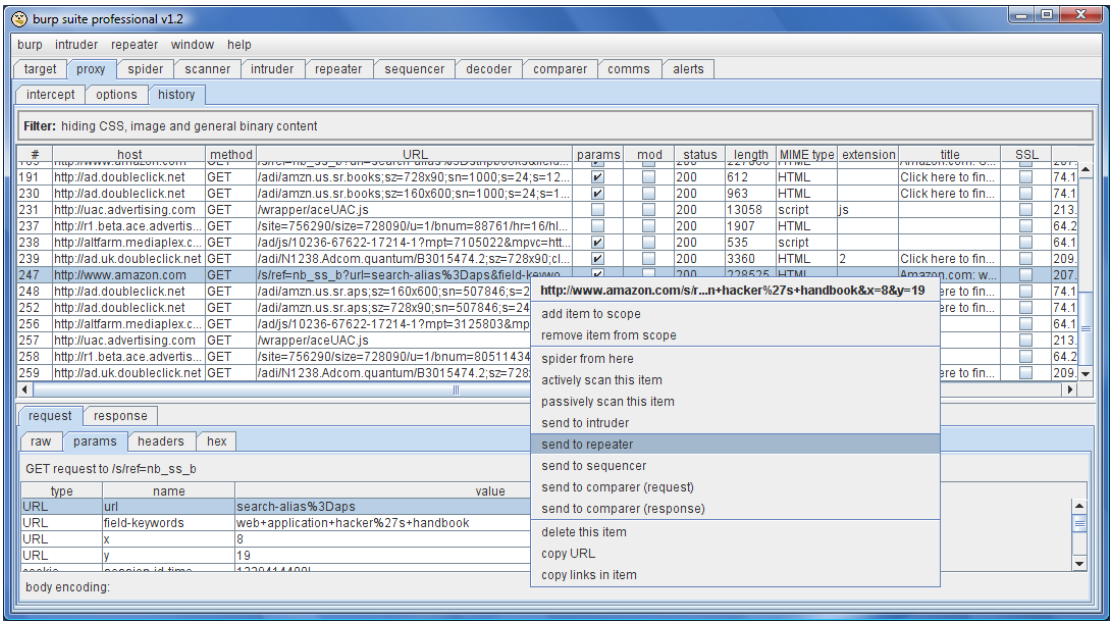
视图菜单(View menu)

这包含了查看或隐藏结果表里的每个可用的数据列(这些列的可用性取决于当前攻击的配置)。

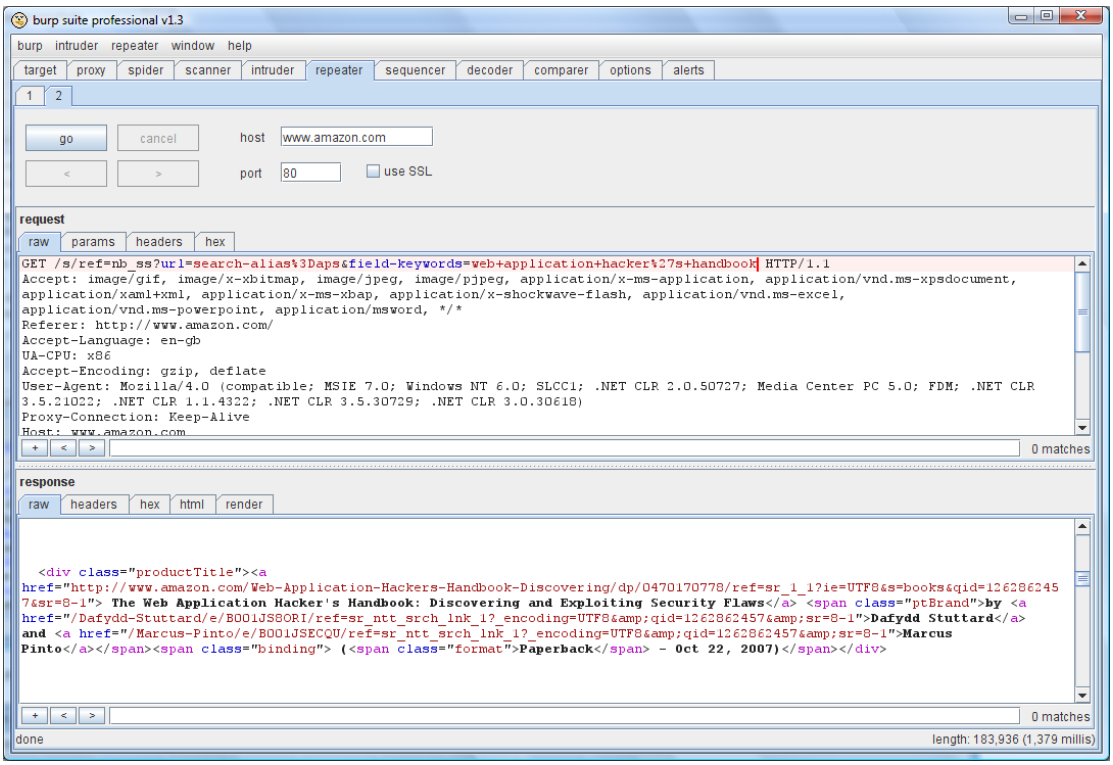
Burp Repeater Help

使用 Burp Repeater(Using Burp Repeater)

Burp Repeater 是一个手动修改并补发个别 HTTP 请求，并分析他们的响应的工具。它最大的用途就是和其他 Burp Suite 工具结合起来。你可以从目标站点地图，从 Burp Proxy 浏览记录，或者从 Burp Intruder 攻击结果上的请求，发送到 Repeater 上，并手动调整这个请求来微调对漏洞的探测或攻击。



当你从其他工具上发送请求到 Repeater 上时，这些请求会得到它自己的选项。每个选项有自己的请求和响应窗口，以及自己的记录。面板的上半部允许你配置目标的主机和端口，以及请求的细节。你可以手动地完成这些信息，然而当你从其他 Burp Suite 工具上发送一个请求时，相关的细节都会为你完成了：



当你配置好一个请求，单击'go'按钮发送它到服务器。响应会在显示窗口的下半部显示出来。对请求和响应二者，许多消息视图是可用的：

raw — 这显示纯文本格式的消息。在文本面板的底部有一个搜索和加亮的功能，可以用来快速地定位出消息里的感兴趣的字符串，如出错消息。搜索栏左边的弹出项让你能控制状况的灵敏度，以及是否使用简单文本或者十六进制搜索。

params — 对于包含参数(URL 查询字符串，cookie 头，或者消息体)的请求，这个选项把这些参数分析为名字/值的格式，这就可以简单地随他们进行查看和修改了。

headers — 这里是以名字/值的格式来显示 HTTP 的消息头，并且也以原始格式显示了消息体。

hex — 这里允许你直接编辑由原始二进制数据组成的消息。如果在文本编辑器修改，某种类型的传输(如，MIME 编码的浏览器请求)包含了可能损坏的二进制内容。为了修改这类消息，应该使用十六进制编辑器。

HTML/XML — 对于包含了这些格式内容的响应，这里提供了一个消息体的颜色语法格式视图。

render — 对于包含 HTML 或者图像内容的响应，这里会以可见的形式显示出这些内容，就像显示在你浏览器那样。

AMF — 对于 Action Message Format 格式的请求和响应，显示了一个编码消息的视图树。如果可编辑，你可以双击视图树上的单个节点来修改它们的值。

viewstate — 对于包含 ASP.NET ViewState 参数的请求，这会把 ViewState 中的内容进行反序列化，使你能够查看任何敏感项包含的数据。也指示了 ViewState MAC 项是否可用(也就是 ViewState MAC 是否可修改)。

在任何请求和响应上，右击它们就会产生一个上下文菜单，来进行下面的操作：

send to — 你可以发送任何消息，或者选中的部分消息到其他 Burp Suite 工具上，来执行进一步操作或分析。

find references — [仅限专业版]你可以使用这个功能在所有的 Burp 工具上来搜索 连接到当前项的 HTTP 响应。

discover content — [仅限专业版]你可以使用这个功能来发现那些不是连接到由浏览或网络爬虫发现的内容的内容和功能。

schedule task — [仅限专业版]你可以使用这个功能来创建一些在定义的时间和间隔内自动运行的任务。

change request mothod — 针对请求，你可以在 POST 和 GET 之间自动地切换请求方法，并使用相关的请求参数稳定地加载这些请求。这个选项可以用来以潜在的恶意请求来快速地测试到应用程序的极限参数的位置。

change body encoding — 针对请求，你可以在应用程序/X—www—form URL 编码和 multipart/form-data 之间进行切换任意消息体的编码方式。

copy URL — 这个功能是把当前的完整 URL 复制到粘贴板上。

copy to file — 这个功能允许你选择一个文件，然后把消息的内容复制到这个文件里。这对二进制内容很方便，然而通过粘贴板会产生一些问题。在选中的文本上进行复制操作，如果什么也没选中，则复制整个消息。

paste from file —这个功能允许你选择一个文件，然后把文件里的内容粘贴到消息里。这对二进制内容很方便，然而通过粘贴板会产生一些问题。粘贴会替换选中的文本，如果什么也没选中，则在光标的位置插入。

save item — 这个功能让你指定一个文件用来保存 XML 格式的选中的请求和响应，这里包含了所有相关的元数据，如响应长度，HTTP 状态码和 MIME 类型。

convert selection — 这些功能让你能够以各种方案快速地对选中的文本进行编码解码。

URL-encode as you type — 如果这个选项被打开，则像&和=这样的字符会被你输入的相等的 URL 编码替换。

你可以使用<和>按钮来后退和前进浏览当前选项的请求记录，如果需要，可以修改任何请求。

选项(Options)

‘repeater’菜单控制 Burp Repeater 的各方面的行为。

你可以创建一个新的空白项，删除一个现有项，或者恢复一个选项的标题来帮助你继续你的工作。

如果’ update Content-Length header’框被选中，Burp Repeater 使用一个特殊请求的 HTTP 主体长度的正确值，来更新每个请求的消息头(如果需要可以添加消息头)的内容长度。这个功能在手动修改 HTTP 主体时是很有用的，这可能会改变它的长度。HTTP 规范和大多数的 web 服务器都需要使用消息头的内容长度指定 HTTP 主体长度的正确值。如果没指定正确值，则目标服务器就会返回一个错误，也可能返回一个未完成的请求，或者无限期地等待接收请求的下一数据。

如果’ unpack gzip / deflate’框被选中，Burp Repeater 在显示之前会把 gzip / deflate 格式压缩的内容解压。

重定向设置控制着 Burp Repeater 是否会跟踪 HTTP 重定向(那些有 3XX 状态码的和包含新 URL 的本机消息头)。下面的选项是可用的：

1.Nerver — Repeater 不会跟踪任何重定向。

2.On-site-only — Repeater 只会跟踪同一 web 站点的重定向。如，在本地请求中使用的有相同的主机，端口，协议 URL。

3.In-scope only — Repeater 会只跟踪那些在 Suite-wide 目标范围(定义在目标选项卡)内的 URL。

4.Always — Repeater 会跟踪任意 URL。你应该小心地使用这个选项 — web 应用程序偶尔会以重定向的方式转发你的请求参数到第三方 web 站点，于是通过跟踪这个重定向你不经意间的就攻击了一个不打算攻击的应用程序。

当 Repeater 接收到一个配置来跟踪的重定向时，它会请求这个重定向(如果需要，最多跟踪 10 个重定向，不再多了，这样为了避免无限地循环)。在一个重定向面板上显示了从重定向 URL 得到的响应。消息的状态指示出是否跟踪了一个重定向，以及有多少重定向。

当应用程序对多种输入都返回一个 3XX 响应时，这个跟踪重定向的选项就有用了，因为当请求重定向目标时，会返回应用程序处理你请求的许多感兴趣的特征。例如，当探测常规漏洞时，应用程序会频繁地返回指向错误页面的重定向 — 这个页面会包含错误本质的有用信息，这可被用来诊断像 SQL 注入的问题。

如果选中’ process cookies in redirects’选项，如果跟踪了一个到相同域名的重定向，则要重新提交任何设置有 3XX 响应的 cookie。

注意当 Burp Repeater 接收到一个不是配置为自动跟踪的重定向响应，会在 Repeater 接口的顶部显示一个’ follow redirect’按钮。这允许你查看后，手动跟踪这个重定向。这个功能是用来穿过重定向序列里的每个请求和响应。如果这些选项被设置在上面的’ process cookies’配置里，则用这些手动的重定向来处理这些新 cookie。

‘action’子菜单包含了和右击请求或响应面板的可用的一样的项。

Session Handler Help

会话处理的挑战(Session handling challenges)

当对应用程序进行模糊测试或扫描时，会常常遇到一些问题：

1.应用程序会因为防守或其它原因终止了进行测试的会话，并且其余的测试连续也是无效的了。

2.有些应用程序改变每个请求必须提供的节点(例如，来阻止请求欺骗攻击)。

3.有些功能在测试这个请求之前，需要一系列的请求，才能让应用程序进入一个接收测试请求的合适状态。

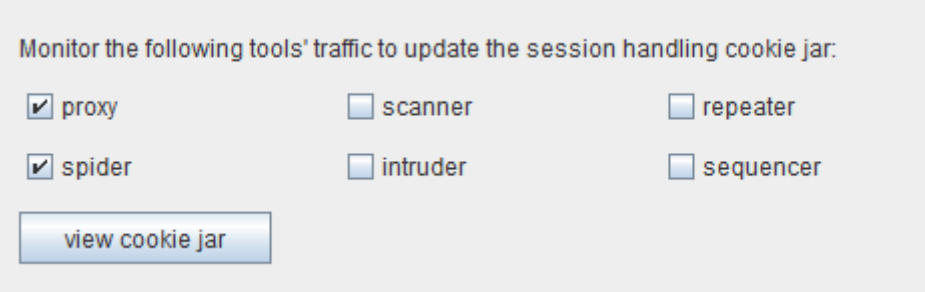
当你进行手动测试时，所有的这些问题都能产生，并且手动解决这些问题常常显得无聊，这会降低你对下面测试的欲望。

Burp 包含了一系列的功能来帮你解决这些情况，让你继续你的手动的和自动的测试，Burp 会在后台为照看这些问题。所有的会话相关的配置都可以在主选项卡里的会话选项卡里找到。

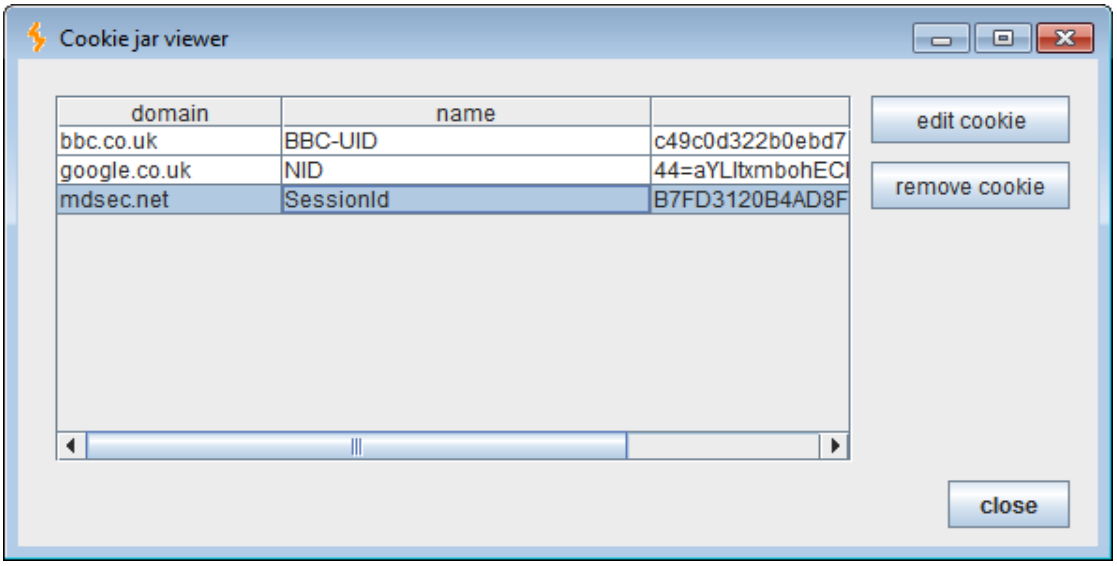
Burp 的 cookie 容器(Burp's cookie jar)

Burp 保存了一个追踪 cookie 的 cookie 容器，这个容器可以用在许多应用程序会话中。这个 cookie 容器被所有的 Burp 工具共享。响应中设置的 cookie 会存储在 cookie 容器中，这些 cookie 会被自动地添加到即将发送的请求里。

所有的这些都是可配置的，例如，你可以为 Proxy 和 Spider 接收到的 cookie 来更新 cookie 容器，以及 Burp 会自动地把 cookie 添加到 Scanner 和 Repeater 发送的请求里。在主选项卡里的会话选项卡显示了 cookie 容器的配置：



如上面显示的，默认的 cookie 容器是依靠 Proxy 和 Spider 的传输进行更新的。你可以查看 cookie 容器的内容并按照自己的意愿来手动编辑 cookie：

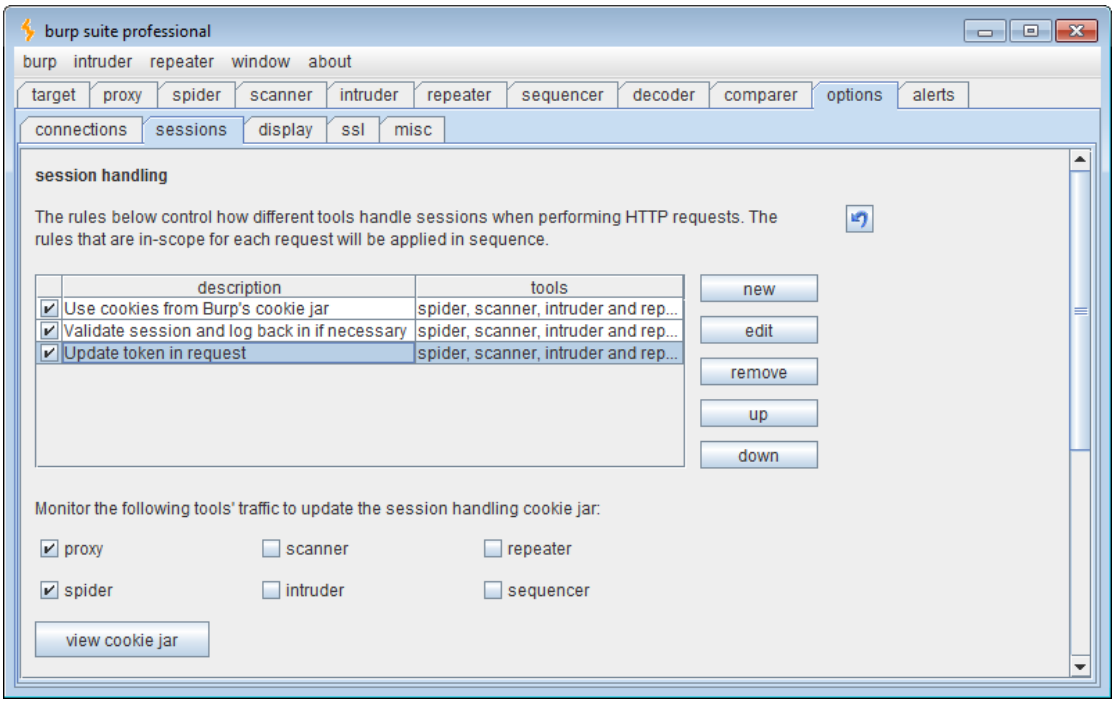


除了 Proxy 其他的所有工具，都要检查 HTTP 响应以确认新 cookie。除了 Proxy，从浏览器进入的请求也被检查。当有个应用程序设置了一个永久 cookie，这个 cookie 出现你的浏览器里，并且还需要这个 cookie 来适当处理你的会话时，这个会有用了。Burp 依据通过代理的请求来更新它的 cookie 容器，这就意味着，即使在你访问时应用程序不更新 cookie 的值，所有需要的 cookie 都会被添加到 cookie 容器里。

Burp 的 cookie 容器遵循 cookie 的域范围，用一种方式来模仿 Internet Explorer 的解释 cookie 的处理规范。不需遵循路径范围。

会话处理规则(Session handling rules)

Burp 让你定义了一个会话处理规则列表，这让能完全控制着 Burp 是怎样来处理应用程序的会话处理机制和相关的功能。在主选项卡里的会话选项卡来配置这些规则：



每个规则包含了一个范围(规则应用的对象)和操作(规则做什么)。对于 Burp 要发送的每一个请求，它决定要使用哪一个规则来处理请求，并且按顺序执行每个规则的动作(除非条件检查操作确定该请求不需要进一步的处理)。

每个规则定义的范围，是根据处理请求的以下特征而设定的：

1.处理请求的 Burp 工具。

2.请求的 URL。

3.请求里的参数名。

每个规则执行的一个或多个动作。实施以下动作：

1.添加会话处理 cookie 容器中的 cookie。

2.设置一个特定的 cookie 或参数值。

3.检查当前会话是否可用，并有条件地在结果上执行一些子操作。

4.为恢复浏览器的会话提示用户。

5.运行一个宏。

6.运行一个 POST 请求宏(这来处理当前请求，并执行下一个宏)。

所有的这些操作都是高度可配置的，可以以任意的方式结合起来处理任何会话处理机制。能运行任意宏(定义在请求队列的)，并能更新结果上的指定 cookie 和参数值，允许你通过部分自动扫描或 Intruder 攻击来自动重新登录到一个应用程序。恢复浏览器会话提示让你

和登录机制一起工作，这些机制有输入物理令牌的一个数字或者解决一个 CAPTCHA 类型的难题。

通过创建多个不同范围和操作的规则，你可以为 Burp 应用到不同应用程序和功能的行为定义一个层次结构。例如，在一个特殊的测试上你可以定义一下规则：

1.对所有的请求，添加 Burp 的 cookie 容器里的 cookie。

2.对一个特定域的请求，验证那个应用程序的当前会话是否存活，如果不是，运行一个宏重新登录到应用程序，并且使用结果里的会话令牌更新 cookie 容器。

3.对特定的包含__CSRF 令牌参数 URL 的请求，首先运行一个包含__CSRF 令牌值的宏，然后在提出请求时使用。

怎样配置 Burp 来完成这些的细节内容将在接下来的部分讲到。

宏(Macros)

Burp 的会话处理功能的关键部分就是运行宏的能力，和定义在会话处理的规则一样。宏是一个单个或多个请求的预定义序列。典型的使用宏的情况有：

1.获取一个检查当前会话是否存活的一个应用程序页面(如用户的主页面)。

2.执行一个包含新的存活会话的登录。

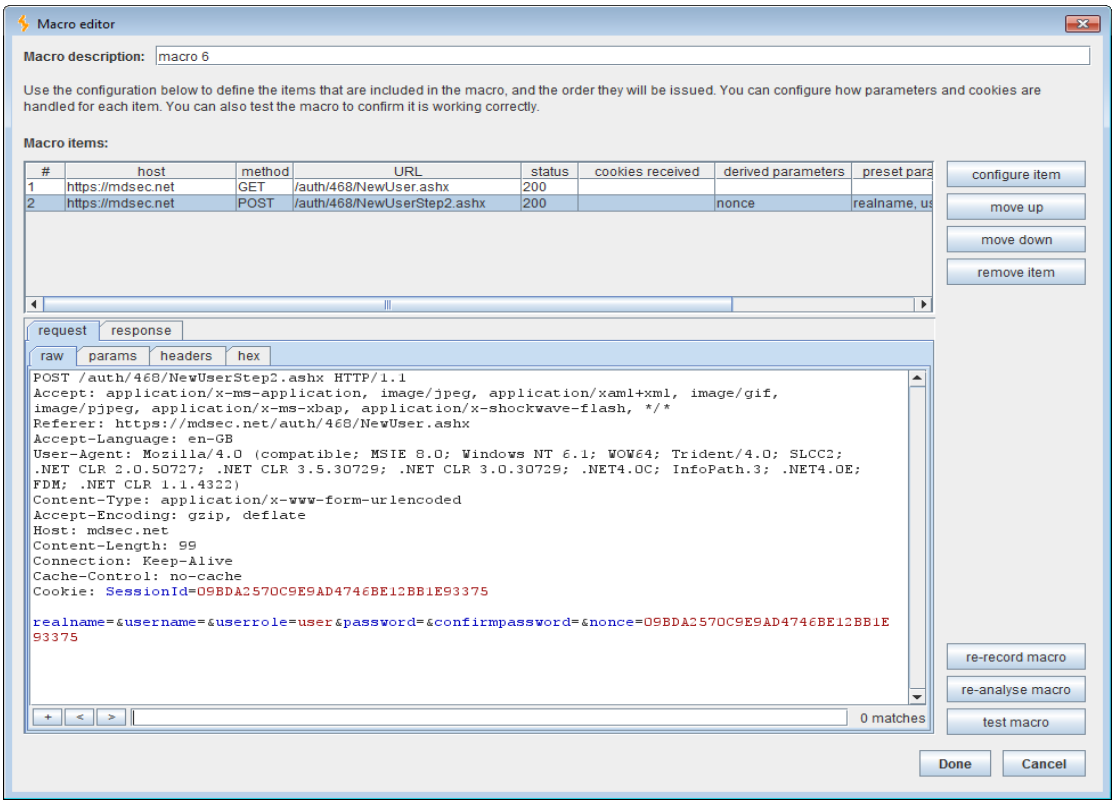
3.包含一个用在其他请求里的令牌或随机数。

4.当在多步处理过程扫描或模糊测试一个请求时，执行必须的先前请求，来让应用程序进入一个接收目标请求的状态。

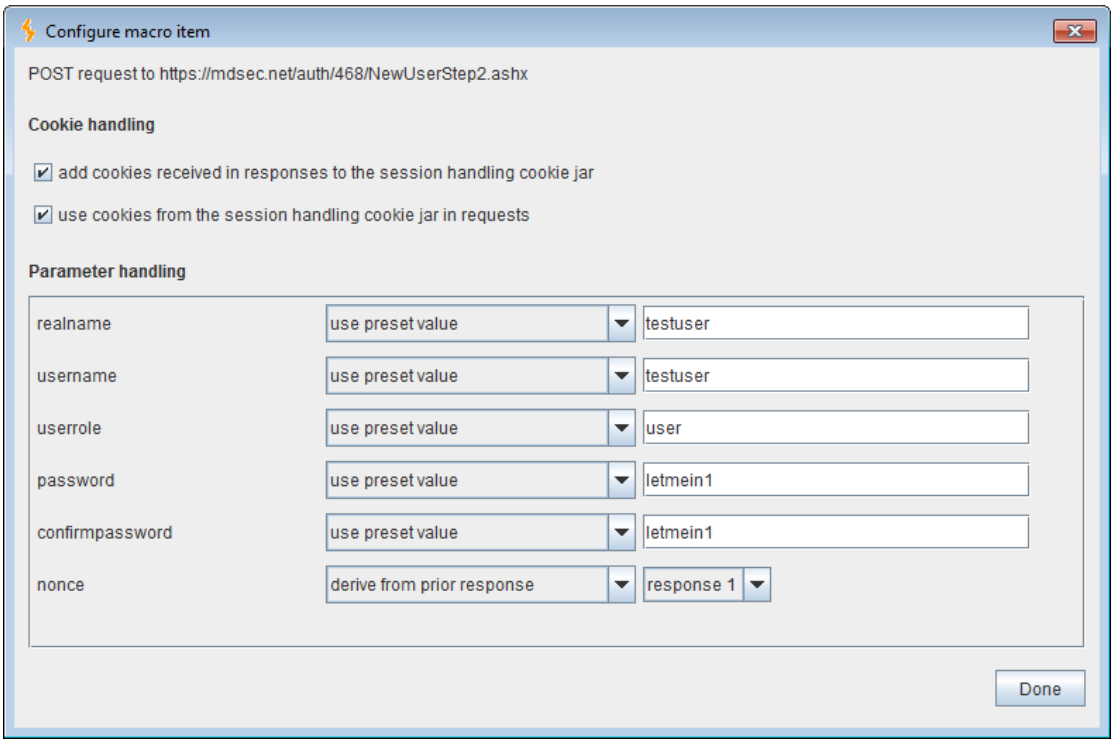
5.在一个多步处理过程中，攻击请求发出后，完成剩下的处理步骤，以确认要执行的操作，或者从处理结果获取结果或出错信息。

使用浏览器记录下宏。当定义一个宏，Burp 会显示一个 Proxy 理论记录的视图，从这里你可以为宏来选择要使用的请求。你可以从先前产生的请求里选择，或者重新记录这个宏并且从历史记录选择这个项。

当你记录下这个宏，宏编辑器在宏里显示出这个项的细节，你可以预览，以及按照需要来配置：



和基础的请求序列一样，每个宏都包含了一些重要配置信息，怎样处理序列里的项，以及项之间的相互依存关系：



对于宏里的每个项，可进行以下配置：

1.是否应该把会话处理 cookie 容器里的 cookie 添加到请求里。

2.从响应里接收的 cookie 是否应该被添加到会话处理的 cookie 容器里。

3.对于请求里的每个参数，是否应该使用一个预设值，或者使用一个宏以前的响应里的派生值。

4.在更新的参数值里，关键字符是否进行 URL 编码。

在一些多阶段处理过程中，从一个先前的响应派生一个参数值的能力通常是很有用的，并且在这种情况下，应用程序能更好地利用逆 CSRF 令牌。当定义了一个新的宏，Burp 会通过确认由先前响应决定值的参数(构造字段值，重定向目标，查询连接里的字符串，等等)，来自动尝试查找和这一类的关系。你可以在使用宏之前，简单地预览并编辑 Burp 使用的宏配置。下面，可以隔离测试配置的宏，以及预览完整的请求/响应序列，来检查是不是你需要的功能。

使用示例(Worked example)

让我们观察一个只能通过认证会话使用的应用程序功能，以及使用后面的令牌来抵御 CSRF 攻击。你想测试那些基于输入像 XXS 和 SQL 注入的漏洞的功能。执行自动测试这个功能，会面临两个挑战：(a)确保使用的会话存活(b)获取每个请求里使用的可行令牌。Burp 的会话处理功能能处理这两个挑战。

要做到这样，我要定义一些会话处理规则。这些规则将应用到我们要测试的功能发出的请求上，使用功能的工具是 Intruder,Scanner 和 Repeater：

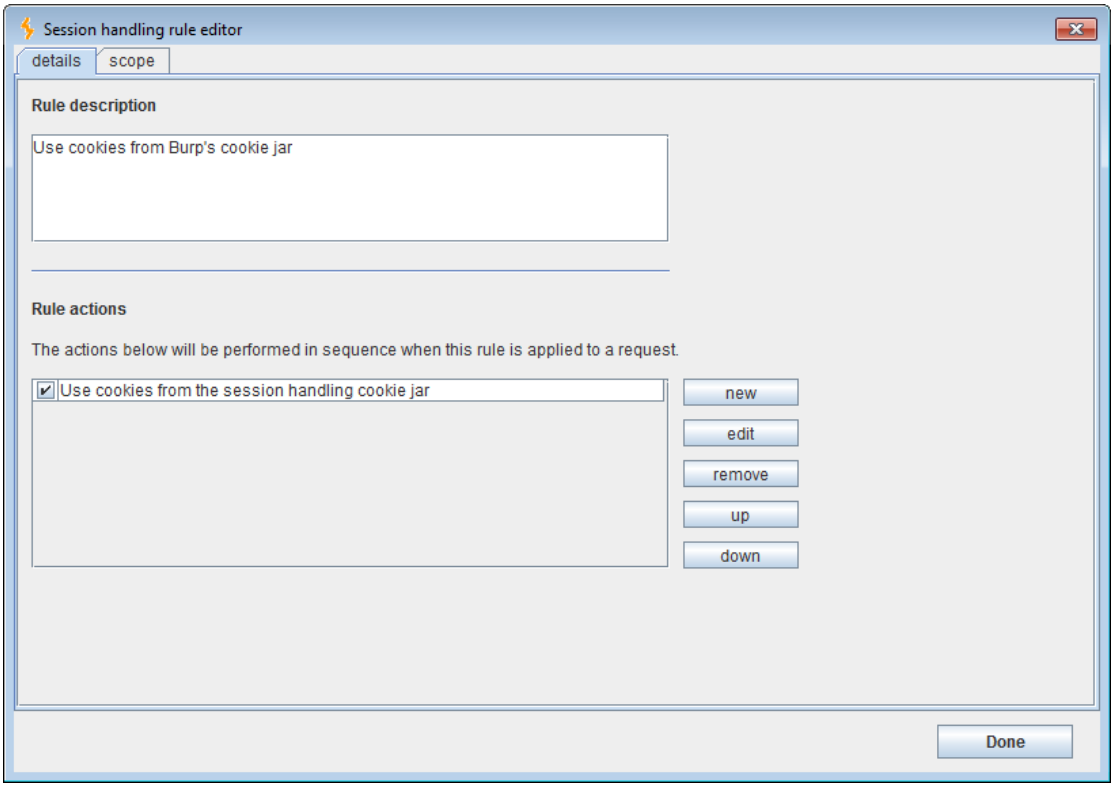
1.通过请求应用程序里的用户登录界面来检查当前会话是否有效，然后检查响应以确认用户是否登录进来。

2.如果没登录进来，重新登录以获得一个有效会话。

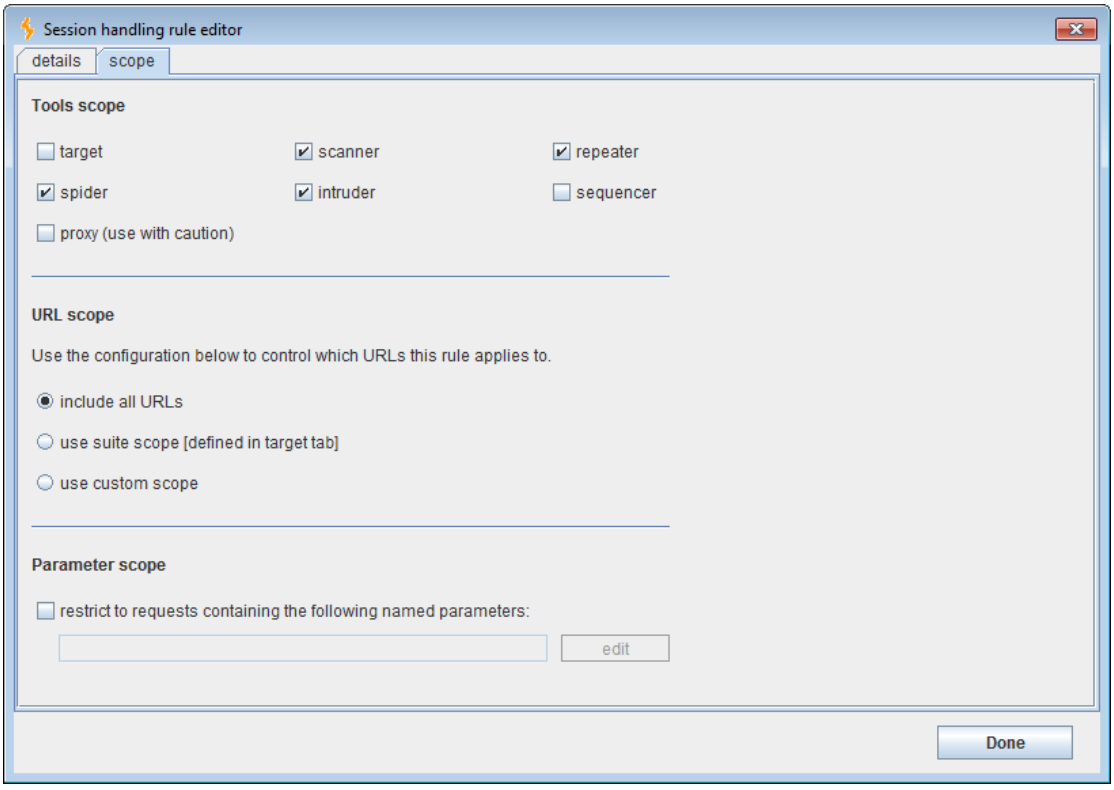
3.请求我们将要测试的包含提交的页面。这个格式在一个隐藏的模块里包含了我们需要的逆 CSRF 令牌。

4.对我们使用逆 CSRF 令牌的值来测试的功能，更新请求。

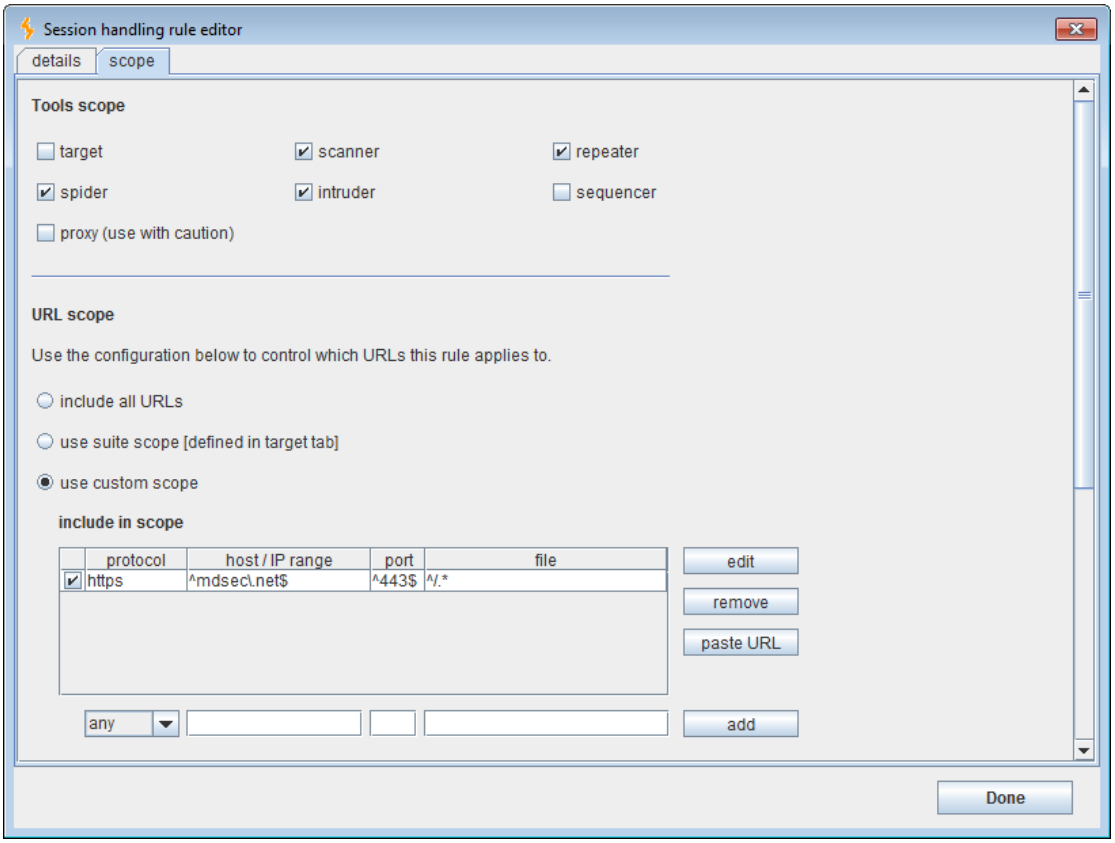
在大多数情况下，我们需要使用 Burp 自己的会话处理的 cookie 容器，所以在第一条规则里，我们就告诉 Burp 把 cookie 容器里的 cookie 添加到每个请求里。这样，实际上，也就是 Scanner 和 Spider 工具的默认规则，于是我们就只修改 Intruder 和 Repeater 工具的默认规则即可。这规则执行一个操作，在下面显示：



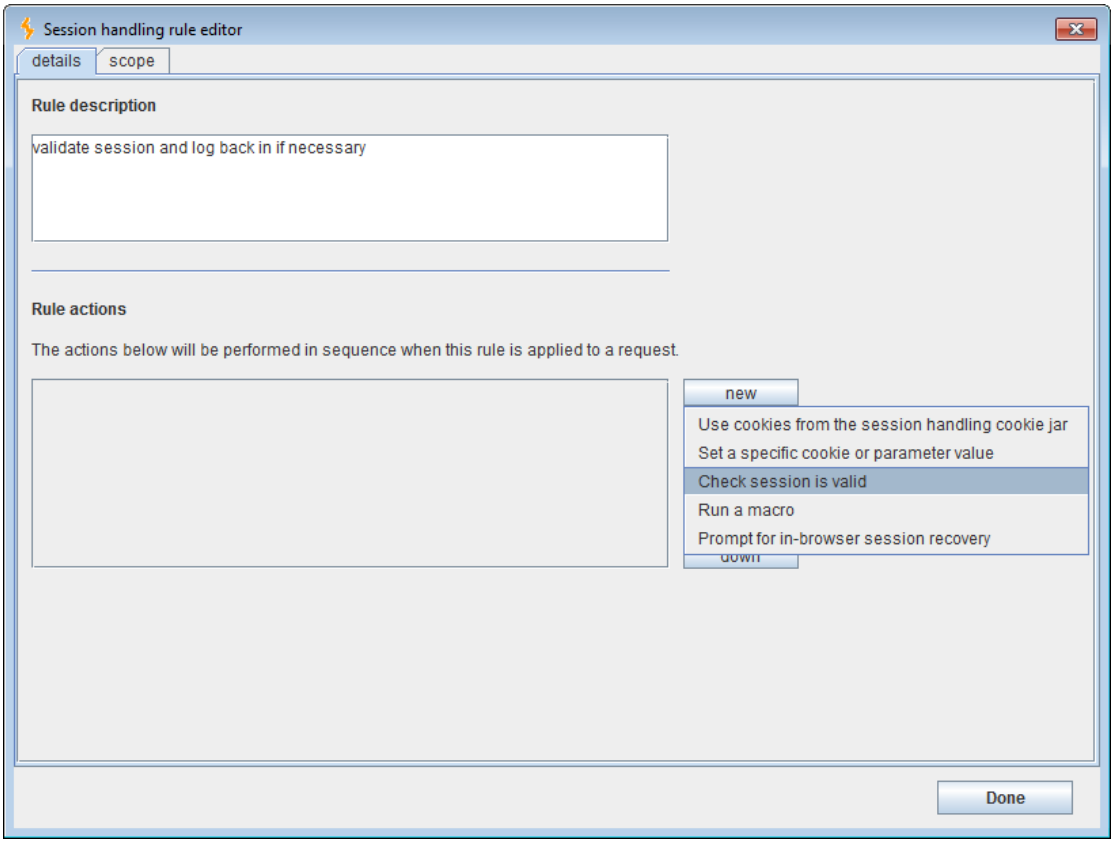
规则的定义的范围来包含相关的工具，并对所有 URL 适用：



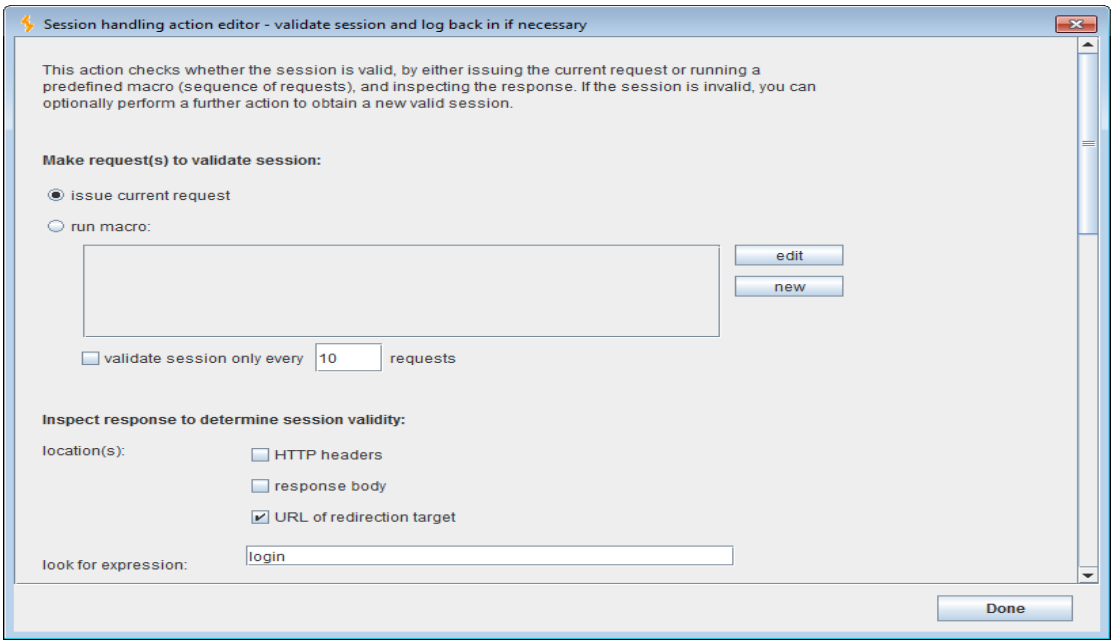
下一步，我需要检查目标应用程序上的当前用户会话是否可用。假设我想要把这些规则应用到目标应用程序的所有请求上，我可以把它定义在整个应用程序域的范围内：



然后，我们添加一个合适的描述以及一个‘check session is valid’类型的操作：



这里打开了这类型操作的编辑器，它里面包含了许多配置项：

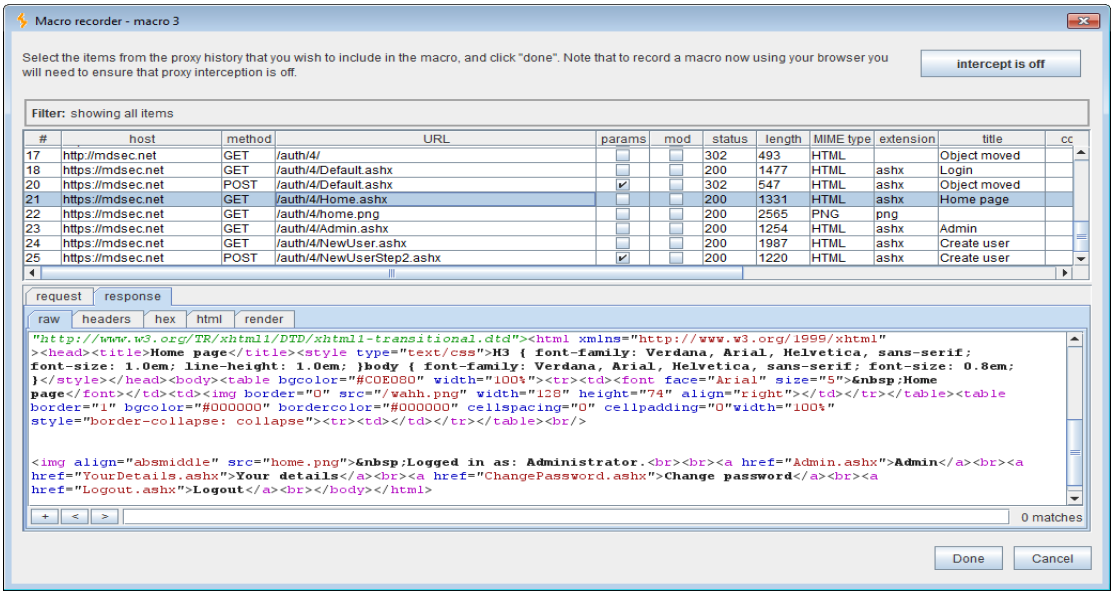


选项的第一步设置是确定了 Burp 使用哪一个请求来验证当前会话。这些选项是：

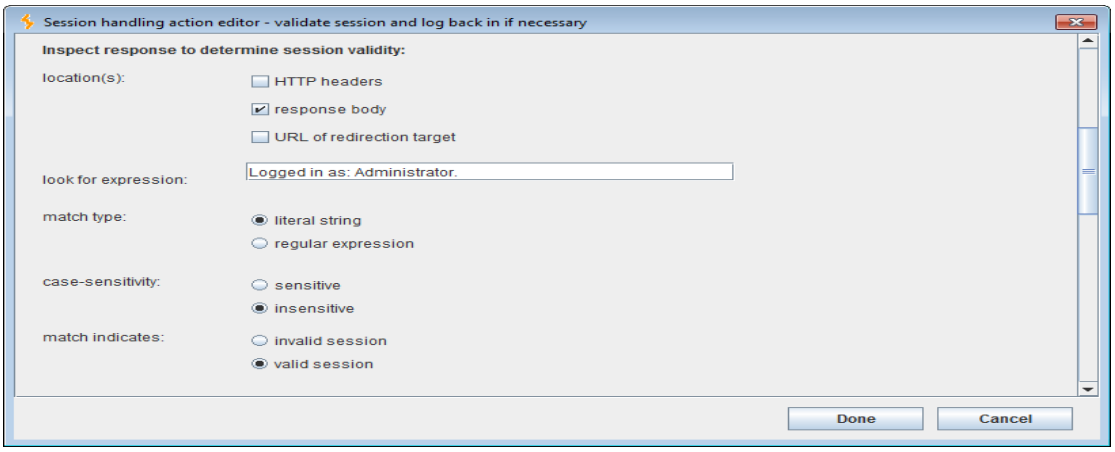
1.发送当前正在处理的实际请求。如果应用程序会对有常规响应签名的会话外请求作出回应，如一个登录重定向时，选上这个选项是明智的。

2.运行一个宏，来发送一个或多个请求。如果是为了确认会话是否有效，这就是一个理想的选项，你需要请求一些标准的项，如用户的主页。这个选项也会是最好，在需要使用进一步规则来修改当前处理的请求 — 例如(像在当前情况下)，更新请求里的逆 CSRF 令牌。如果选项是为了运行一个选中的宏，你需要进一步选择是否要对每一个请求或者其中的 N 个请求。如果应用程序对未预料的输入会积极地终止响应里的会话，建议你验证每次会话；否则你可以只定期验证会话来加快速度。

在当前的实例中，我要运行一个宏来获取应用程序里的用户登录界面，以检测他们的会话是否有效。要这样做，我们需要单击上一截图里的‘new’按钮，来定义自己的宏。这里打开了宏记录器，让我们来选择希望包含在宏里的请求。当前状况下，我们只需要选择用户登录页面里的 GET 请求：



选项的第二部设置是‘check session is valid’操作控制 Burp 怎样检查宏里的响应来确定会话是否有效。许多选项是可用的，下面列出了当前情况下我们需要的配置：



在这里，我们使用 cookie 容器里的 cookie 配置了 Burp 进行更新请求，并且在会话有效时，把用户重新登录到应用程序。为完成需要的配置，我们需要定义一个延伸规则来处理在我们要测试功能里逆 CSRF 令牌。我们要测试请求像这样：

POST /auth/4/NewUserStep2.ashx HTTP/1.1

Content-Type: application/x-www-form-urlencoded

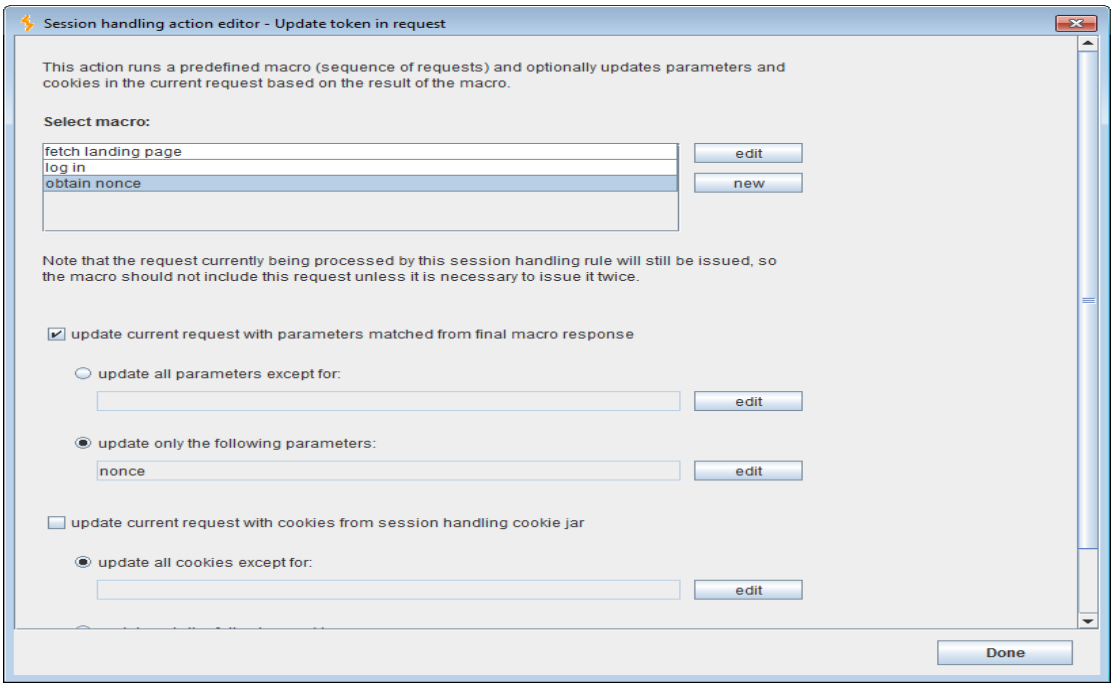
Host: mdsec.net

Content-Length: 137

Cookie: SessionId=39DD9F0CB979BFB431005524A4010244

realname=testuser&username=testuser&userrole=user&password=letmein1&confirmpassword=letmein1&nonce=938549246127349541173

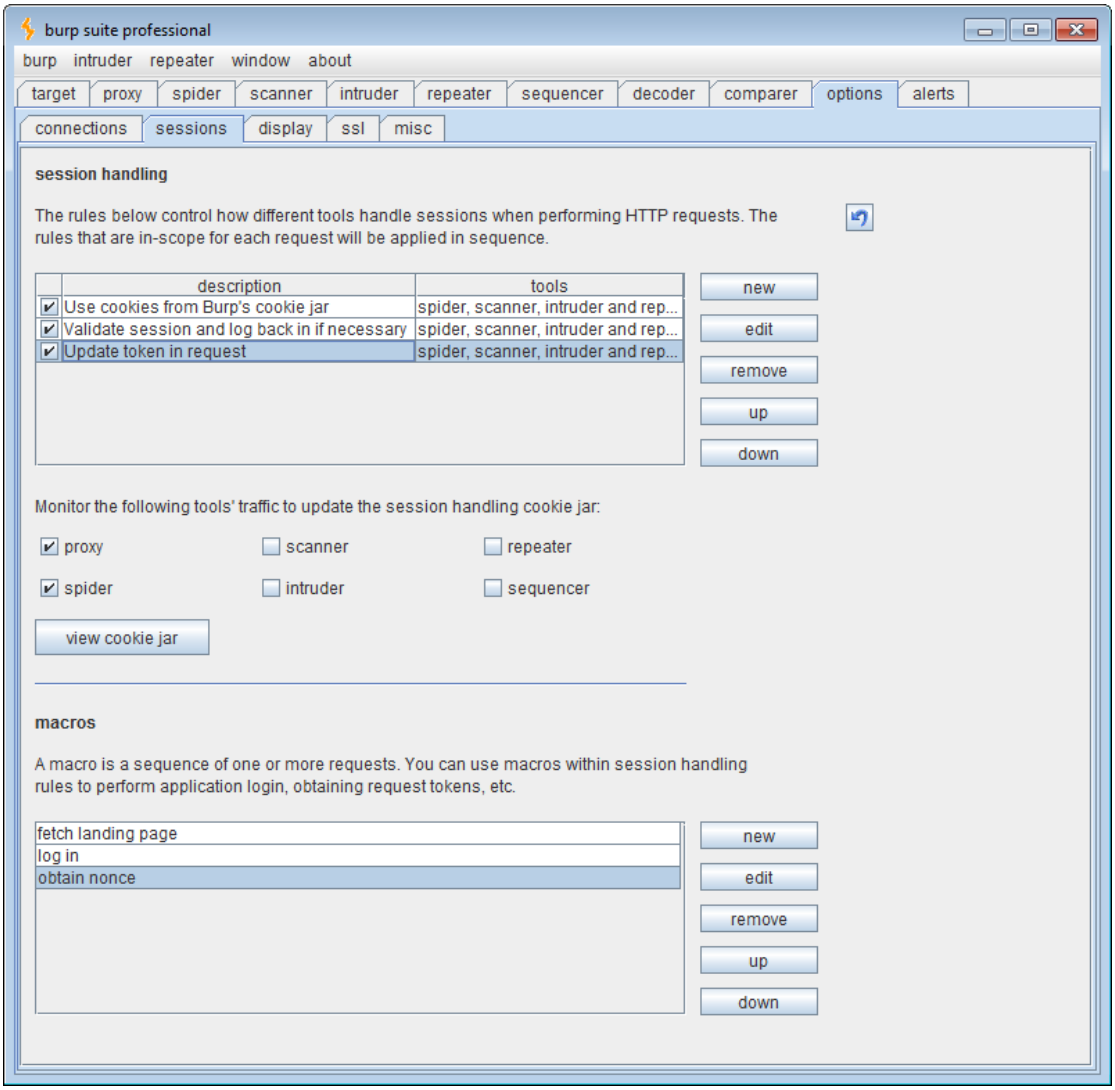
为确保这个功能正确地处理了我们的请求，需要每个请求提供的随机数有效。在应用程序里的产生上面请求的格式的隐藏区域提供了这个随机数值。因此，我们需要运行一个宏来获取包含这个格式的页面，并使用随机参数值来更新当前请求。我们使用一个有‘run macro’类型的操作来添加一个延伸规则，并像下面这样来配置它：



在上面的配置里，我们指定了 Burp 应该运行一个新宏，来获取包含逆 CSRF 令牌的格式，和从宏响应中获取随机参数，以及在请求里更新。另外，我们可以选择‘update all parameters’选项，Burp 会自动地尝试请求里的参数和在宏响应指定的值进行匹配。

在规则范围内，明显地需要定义在比整个应用程序域更窄的范围。例如，我们可以把这个规则只定义在上面请求里 URL 上。如果应用程序只能使用一些位置上的逆 CSRF 令牌。然而，在一些应用程序上，许多功能都使用令牌，在这种情况下，令牌会获取到不止一种功能。我们可以定义一个使用所有域的规则，但仅限于包含一个指定参数名字的请求。在这种方法下，无论何时向应用程序提交的请求中都会包含一个逆 CSRF 令牌，这个规则执行后，Burp 会获取一个新的有效令牌来用在请求上。

这个配置，有它的 3 个会话处理规则和 3 个宏，在 Burp 主界面里就像这样：



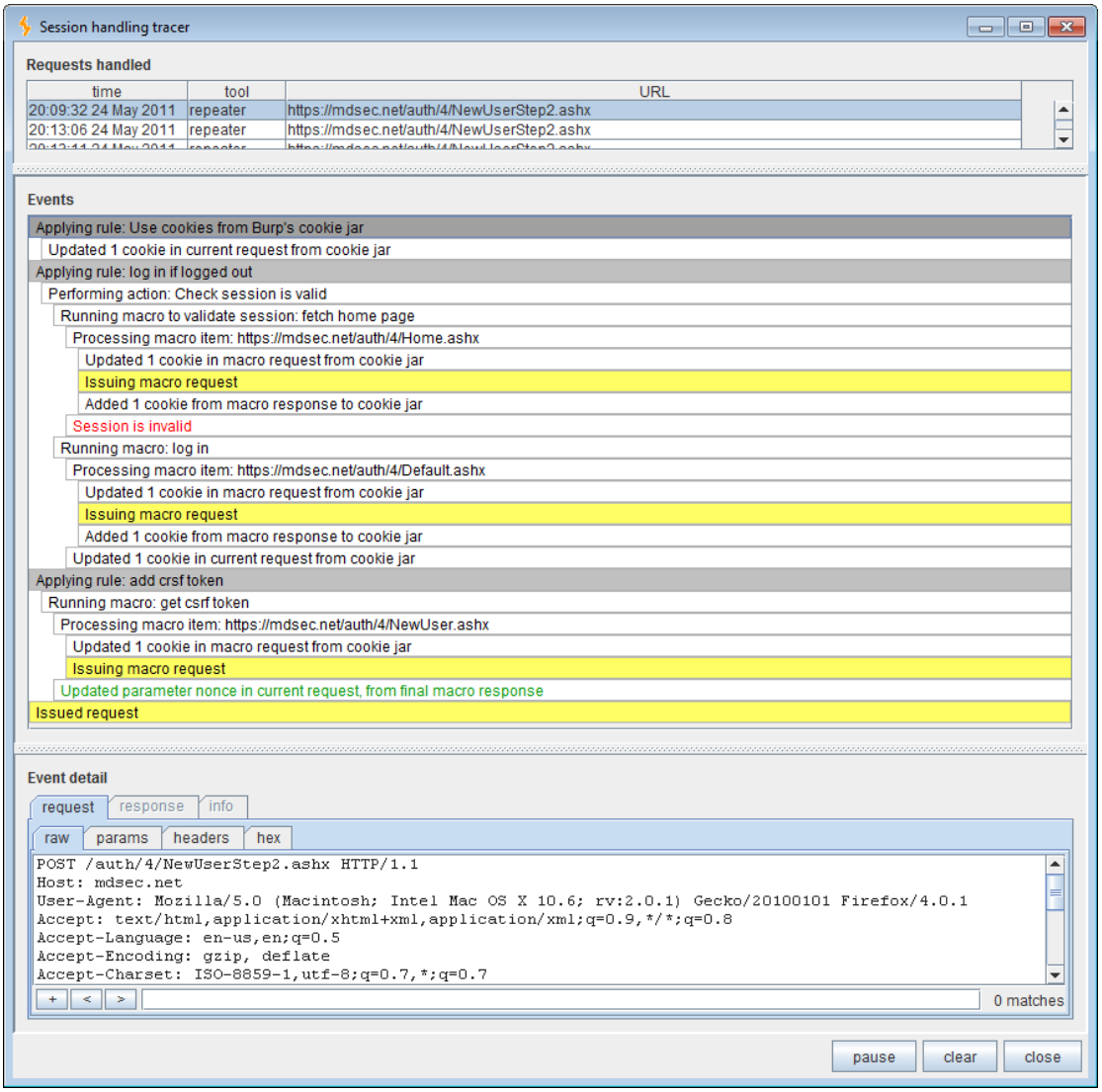
你可以通过在应用程序上注销，向 Burp Repeater 发送已认证的有令牌保护的请求，来测试配置的运行状况以及确认它是否执行了需要的操作。这些请求可能会花费比平常长的时间才能返回，因为 Burp 在幕后提交了许多其他请求，以验证你的会话，如果需要可以再次登录，并获取请求使用的令牌。

如果你发现你的规则不按你打算的方式工作，你可以使用 session handling tracer 来解决这个问题。

一旦你为会话处理规则正确地运行而感到高兴时，你就可以把这些请求发送到 Burp Intruder 或者 Scanner，来以正常地方式执行你的自动化测试。

会话处理跟踪器(session handling tracer)

配置需要把 Burp 的会话处理规则应用到现实世界里的应用程序功能上，这往往是很复杂的，并且很容易产生错误。Burp 提供了一个跟踪功能，来排除会话处理配置的故障。当 Burp 把会话处理规则应用到一个请求上时，这里显示了执行的所有步骤，让你清楚地看到怎样来处理和更新请求的。你可以通过 options / sessions / view sessions tracer 来打开会话处理跟踪器：



Burp 工具的集成(Integration with Burp tools)

关于会话处理功能对一些其他 Burp 的功能影响，需要注意以下几点：

1.这里有一个默认的会话处理规则来更新那些由 Scanner 和 Spider 用 Burp 的 cookie 容器里 cookie 产生的所有请求。这确保了所有 spidering 和扫描的请求都是在会话里产生的，并且还使用你的浏览器来维持了一个有效的会话。这也意味着，在主动扫描队列里的项，是从一个稳定文件里加载过来的，并会在你当前的会话里扫描，而不是保存状态文件的那个激活的会话。如果这不是你想要的行为，在执行扫描之前，你应该使这个默认会话处理规则不可用。

2.如遇到会话处理规则在请求发送之前，就把它修改了(例如，更新一个 cookie 或其它参数)，一些 Burp 工具，为了清晰，会显示出最终更新的请求。这使用在 Intruder,Repeater 和 Spider 工具。在 Scanner 报告里显示发送的请求的同时，还继续显示原来的请求，在需要

的地方，就能方便清楚地和基础请求进行对比。为了观察一次扫描发出的最后一个请求，和会话处理器一样，你可以先把请求发送到 Burp Repeater，然后在发送它。

3.当 Scanner 或者 Intruder 提交一个请求，这请求操纵了一个由会话处理操作影响 cookie 或参数时，这个操作不会应用到那个请求上，这是为了避免干扰正在进行的测试。例如，如果你正在使用 Intruder 对请求里的所有参数进行模糊测试，并且你已经配置了一个会话处理规则来更新请求里的‘sessid’cookie，当 Intruder 对其他参数进行模糊测试时，这个‘sessid’cookie 就会被更新。当 Intruder 对‘sessid’cookie 自身进行模糊测试时，Burp 会把‘sessid’的值作为 Intruder 的有效载荷，并不会更新它，因为它是正常的。