12.4 漏洞利用工具包和勒索软件

在最后一个场景中，我们将研究 IDS 的一次告警。我们将查看从被感染的系统中采集到的实时数据包，并尝试对攻击行为进行溯源。我们将在本例中使用可能会从身边设备中发现的真实恶意软件。

Snort 的 Sguil 控制台进行了一次 IDS 告警，如图 12-31 所示。Sguil 是用于对一个或多个感应器发出的告警进行管理、查看和研究的工具。它提供一个几乎是最友好的用户界面，是在安全研究人员中很流行的工具。

在 Sguil 中有很多关于这次告警的信息。顶部的窗口 ❶ 显示了告警的概要信息。包括在告警发生时的源和目的 IP 地址、端口，协议，根据匹配的 IDS 标识产生的事件消息。在本例中，本地系统 192.168.122.145 正在与位于 184.170.149.44 的未知外部系统进行通信，外部系统使用的端口为 80，此端口通常与 HTTP 通信联系起来。这个外部系统被认为是恶意主机，告警显示，该系统与一个识别恶意通信的标识有关，并且关于此系统，我们了解的信息很少。这次通信匹配到的标识代表了由 CyptoWall 恶意软件族发起的签到流量，这意味着一个该恶意软件的变体被安装在内部系统中。

Sguil 控制台提供了匹配规则的语法 ❷ 和单个数据包中与规则相匹配的数据 ❸。我们注意到，数据信息被拆分为协议头和数据部分，与 Wireshark 中的数据包信息展现形式类似。不幸的是，Sguil 仅提供单个被匹配的数据包的信息，但是我们需要更深入的研究。下一步，在 Wireshark 中检查与这次告警相关的通信，尝试定位告警流量，并查看出现的问题。这些流量在 cryptowall4_ c2.pcapng 中。

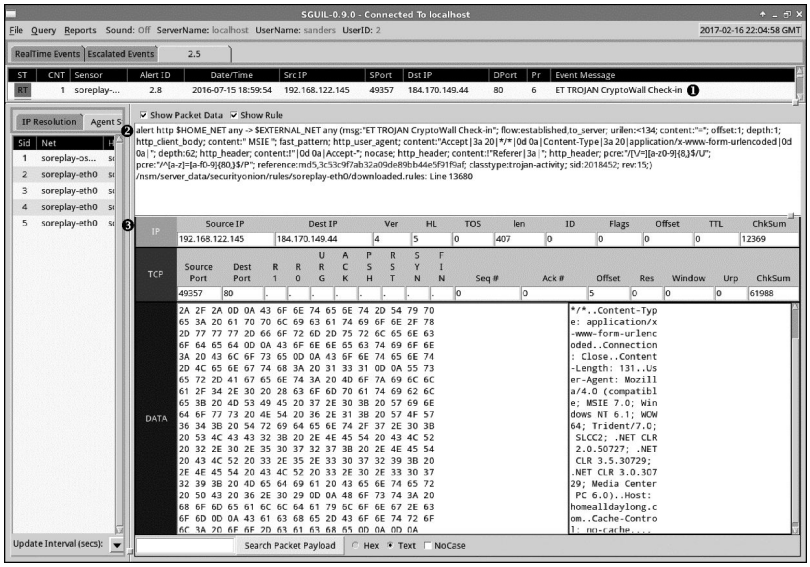

图 12-31　IDS 告警表明一次 CryptoWall 4 感染

这个抓包文件包含了告警时间前后的通信过程，数据并不是太复杂。第一次会话出现在数据包 1～16 中，我们可以跟踪会话的 TCP 流（见图 12-32）以便于查看。在抓包文件的开头，本地系统建立了到恶意主机 80 端口的 TCP 连接，之后向一个 URL ❶ 发起了一次 POST 请求，请求包含少量带有文字与数字的数据 ❷。在连接正常断开前，恶意主机回复了一个由文字和数字组成的字符串 ❹ 和 HTTP 200 OK 响应码。

如果浏览抓包文件的剩余部分，你会发现以上的序列在这些服务器之间重复出现，每次传输的数据量有差异。设置筛选条件为 `http.request.method == "POST"`，看到出现了 3 次类似 URL 结构的连接（见图 12-33）。

请求页面部分（76N1Lm.php）保持一致，但是余下的内容（被发至该页面的参数和数据）有变动。这个重复的通信序列和请求结构，与恶意软件的命令和控制（C2）行为一致，并符合此 IDS 告警标识的匹配规则。你可以在流行的 Cyrpto 研究网站 Crypto Tracker 上看到一个类似的例子，从而进一步核实。

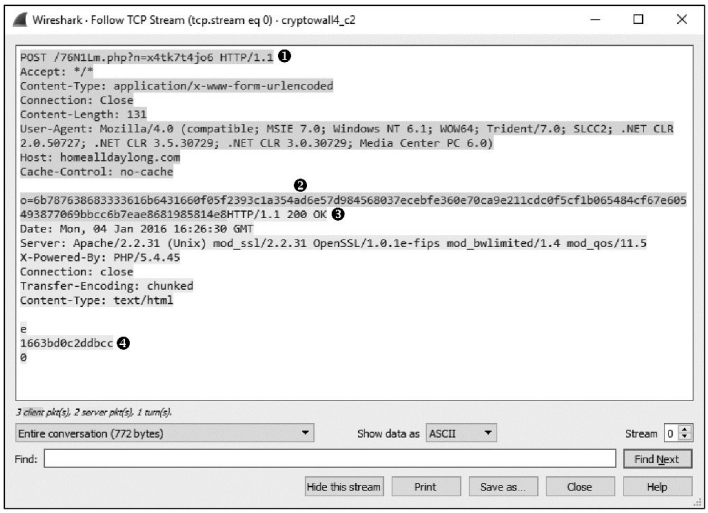

图 12-32　少量信息通过 HTTP 在服务器之间传输

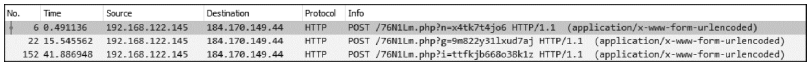

图 12-33　URL 结构显示，不同的数据被发送至同一个页面

既然已经确定发生了基于恶意软件的 C2 通信，我们就应该采取补救措施，并定位被感染的机器。当问题涉及例如 CryptoLocker 的恶意软件时，这样的措施尤为重要，因为这类恶意软件会试图加密用户数据，除非用户支付高额赎金，否则攻击者不会提供解密密钥——这类恶意软件被称为勒索软件。被勒索软件感染的补救措施不在本书的讨论范围内；在现实场景中，这是安全分析人员的后续研究内容。

通常，下一个需要考虑的问题是，内部系统最初是如何被恶意软件感染的。如果答案能够被确定，那么你可能会发现其他设备因为类似的原因被另外的恶意软件感染；或者能够开发防护工具、制定检测机制，避免可能发生的感染。

告警数据包仅显示了被感染后的 C2 序列。在采用了安全监控和持续抓包的网络中，很多感应器被设置为存储几小时或几天内的流量数据，用于取证分析。毕竟，不是每一个机构都使用了能够实时告警的设备。数据包的临时存储让我们能够查看，在前面看到的 C2 序列开始之前，被感染主机发出的数据。这些流量数据在 ek_to_cryptowall4.pcapng 中。

在这个抓包文件中有更多的数据包，新增数据包都是 HTTP 通信数据。我们已经了解了 HTTP 是如何工作的，现在让我们切入正题，设置过滤条件为 http.request，仅查看 HTTP 请求。筛选结果显示了 11 次由内部主机发起的 HTTP 请求（见图 12-34）。

图 12-34　11 个由内部主机发起的 HTTP 请求

第一个请求由内部主机 192.168.122.145 发往未知外部主机 113.20.11.49。查看数据包的 HTTP 部分（见图 12-35），我们发现用户请求了一个页面 ❶，页面入口为 Bing 搜索这个网址的结果 ❷。到目前为止，看起来一切正常。

图 12-35　一次发往未知外部主机的 HTTP 请求

接下来，内部主机在数据包 35、39、123 和 130 中发送了 4 个向另一个未知外部主机 45.32.238.202 的请求。在之前章节的例子中我们了解到，当浏览器访问的页面包含外部内容引用或第三方服务器上的广告时，从网站服务器以外的主机获取内容是很常见的。虽然请求的 URL 看起来有些杂乱，但是这个请求本身并不令人担忧。

从数据包 39 中的GET请求开始，数据开始值得注意。跟踪这次交互的 TCP 流（见图 12-36），你会发现内部主机请求了名为 bXJkeHFlYXhmaA 的

文件 ❶。这个文件名比较奇怪，并且没有文件扩展名。

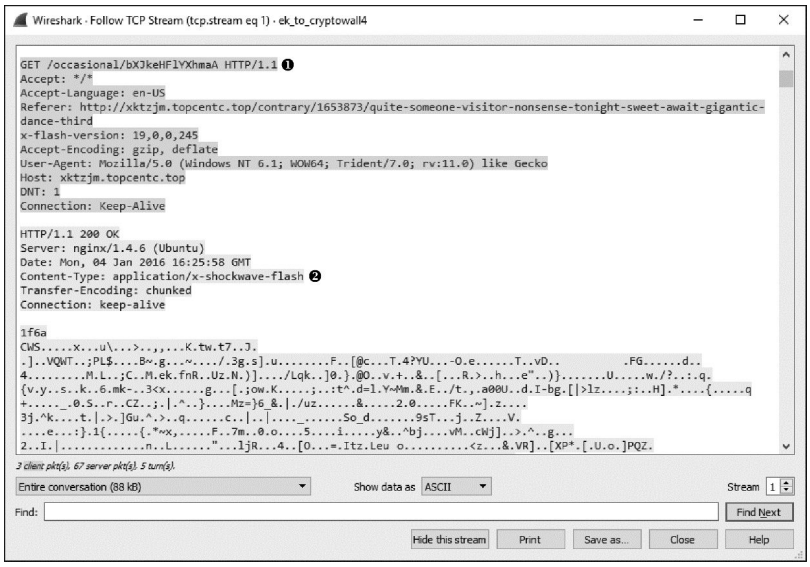

图 12-36　一个名称奇怪的 Flash 文件被下载

在更细致的检查中，我们看到 Web 服务器将文件内容识别为 x-shockwave- flash❷ 类型。Flash 是浏览器上流行的流媒体插件。设备下载 Flash 内容很正常，但是值得注意的是，Flash 由于有很多软件漏洞而臭名昭著，并且，Flash 经常不修补漏洞。随后，Flash 文件被成功下载。

Flash 文件被下载后，在数据包 130 中有一个类似的请求——请求一个名称奇怪的文件。跟踪这个 TCP 流（见图 12-37），你会看到请求的文件名为 enVjZ2dtcnpz❶。这个文件的类型没有被扩展名标明或被服务器识别。这个请求完成后，客户端下载了一个大小为 358400 字节的无法阅读的数据文件 ❷。

该文件下载完毕后不到 20s，你会看到在图 12-34 中出现过的一系列 HTTP 请求。从数据包 441 开始，内部服务器向两个不同的服务器发送相同的 C2 格式 HTTP POST 请求。我们可能发现了感染的源头。下载的两个文件是问题的原因。数据包 39 中请求的第一个文件是一个 Flash 漏洞利用程序，在数据包 130 中请求的第二个文件是恶意软件。

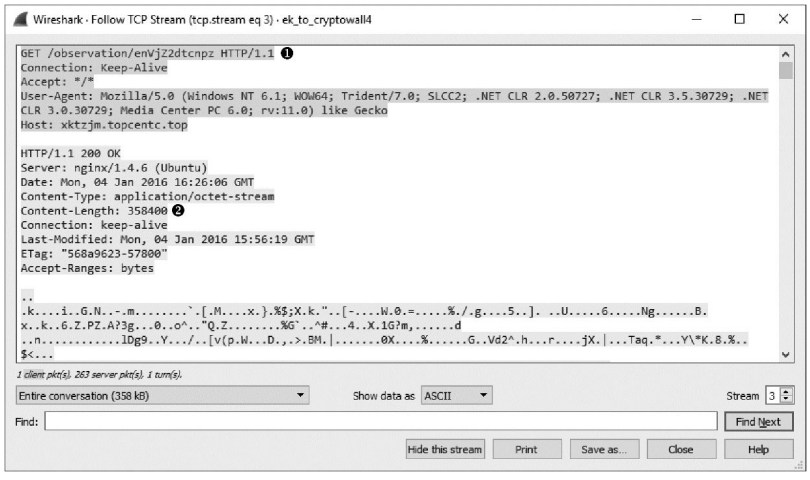

图 12-37　另一个名称奇怪的文件被下载，其文件名未被识别

> 注意
>
> 你可以使用恶意软件分析技术，对流量数据中包含的文件进行解码和分析。如果你对恶意软件逆向工程感兴趣，我推荐 MIchael Sikorski 和 Andrew Honig 的《恶意代码分析实战》（2012），另一本 No Starch Press 的书，也是我个人最喜欢的书之一。

这个场景代表了最常见的恶意软件感染技术之一。用户在浏览网页时，误入了一个被漏洞利用工具包插入了恶意重定向代码的网站。这些漏洞利用程序感染正规网站，并具备采集客户端指纹信息的功能，以确定客户端存在的漏洞。被感染的页面被称为漏洞利用工具着陆页，它的目的是，根据工具包确定的用户系统存在的漏洞，将用户客户端重定向至另一个含有对应漏洞利用程序的站点。

你刚才看到的数据包来源于 Angler 漏洞利用工具包，其可能是 2015 年和 2016 年最常见的工具包。当用户访问一个被 Angler 感染的网站时，此工具包会辨别用户系统是否存在一个特定的 Flash 漏洞。随后，用户下载一个 Flash 软件，用户系统被感染，用户下载第二个载荷——CryptoWall 恶意软件并安装。整个过程如图 12-38 所示。

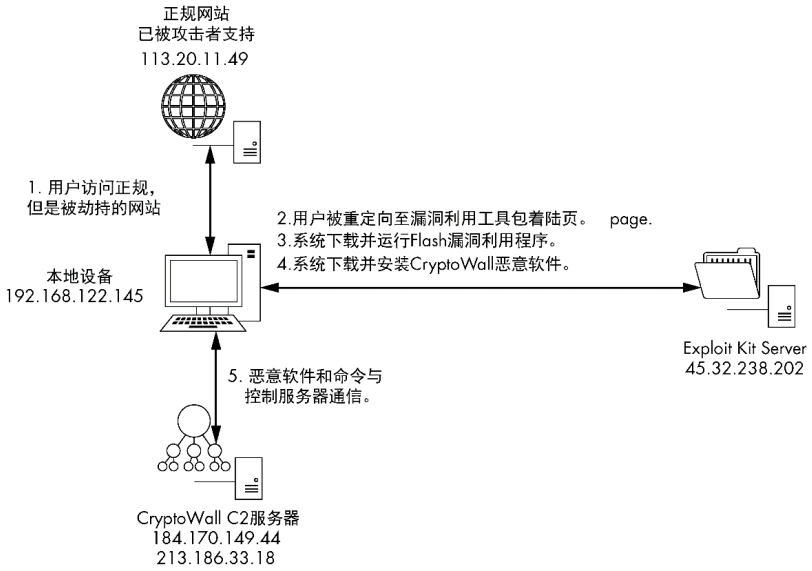

图 12-38　漏洞利用工具包感染过程