**Mirai源码中的SYN弱口令扫描**

05 Nov 2016 TAGS : [ [技术相关](http://rootkiter.com/goto_tags.html#技术相关) [Mirai](http://rootkiter.com/goto_tags.html" \l "Mirai) [Iot安全](http://rootkiter.com/goto_tags.html" \l "Iot安全) [僵尸网络](http://rootkiter.com/goto_tags.html#僵尸网络) [SYN端口扫描](http://rootkiter.com/goto_tags.html#SYN端口扫描) [恶意样本](http://rootkiter.com/goto_tags.html#恶意样本) ]

**SYN口令扫描原理**

最近在细读Mirai的源码，在它之中用到了若干有趣的技术，先把SYN扫描弱口令的代码过一下。其实这个技术本并不新鲜，早在有些扫描器中已经支持，但从代码层面分析的文档并不多见，所以在这里写一段，给日后留个念想。

细细想来，我也从业许久，但就从没见过如此恶心的代码，作者竟然能把整个syn口令扫描的核心代码堆在了一个函数中，活生生把一个功能函数扩充到了583行（line.57 – line. 640）。代码缩进的最深处，可以达到9层（4\*9=36个空格）之深，逐行品读时常常读一屏忘一屏。可以说这段代码磨练了我的意志，锻炼了我的耐心，是出家修行必备之良品。。。

吐槽完闭，该上正文了。

本文档用到的源码链接如下： <https://github.com/jgamblin/Mirai-Source-Code/blob/master/mirai/bot/scanner.c>

**广撒网多捞鱼**

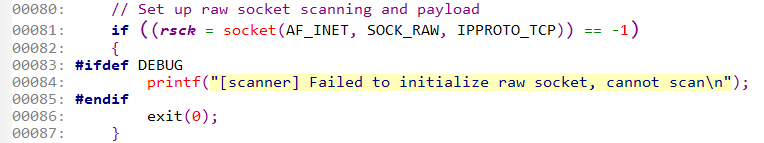
SYN弱口令扫描，之所以能有极高的扫描速度，与其口令枚举速度无关，而是同其寻找爆破目标的速度有关。这依赖于其能够同时对多个IP进行端口探测（而且单线程即可）。

想要在所有联网的主机中找到开放指定端口的主机，常规的思路，就是遍历IP，挨个调用connect函数去尝试连接，成功的便是开启了相应端口，反之则是没有开启端口。但connect函数调用后要经历TCP协议的三次握手，才会得到结果，在大批量扫描中，这是相当难以接受的。

所以几乎在所有常见的“抓鸡扫描器”中都以SYN扫描的形式进行端口状态判断。

**套接字初始化**

想要进行网络操作，socket套接字是无法避免的，SYN扫描的套接字初始化，和常见的TCP套接字有些区别，详情可参看源码第81行，如下图示：



该套接字初始化代码长得有些独特，因为网上最常见的代码一般都长成下面的样子：

sockfd = socket(PF\_INET, SOCK\_STREAM, 0);

当然也有长这个样子的：

sockfd = socket(PF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

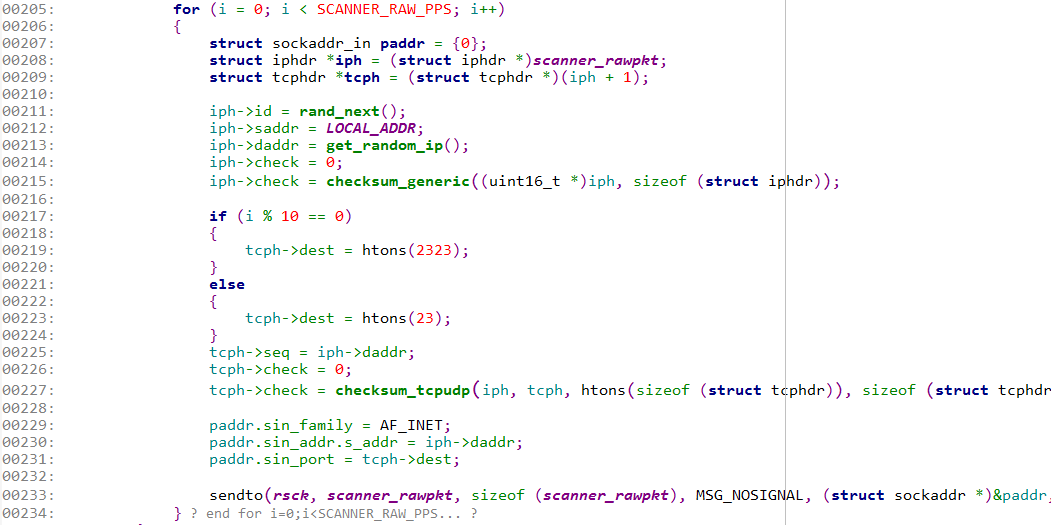
中间的参数SOCK\_STREAM表明该套接字将以流的形式进行读写操作，但SOCK\_RAW这个参数却表明该套接字是一个原始套接字，没有经过任何的封装，该种形式的套接字，除了能收发传统的TCP/UDP数据包外，还可以接受ICMP、IGMP等种类的数据包。这是SOCK\_STREAM套接字无法做到的。当然，这也需要一定的权限才能做到，现代操作系统下，要求用户必须具有管理员权限才能使用该类功能。

从这样一条简单的初始化代码就不难发现，该套接字的一生注定不凡。

**群发SYN**

TCP三次握手中，第一个操作就是发送SYN信号，它代表发起者的连接意向，当服务端收到该信号后，如果连接条件允许就会回复ACK信号，并正式进入TCP会话协商阶段。这是TCP的基本原理，搞不懂的小伙伴出门左转去问Google。

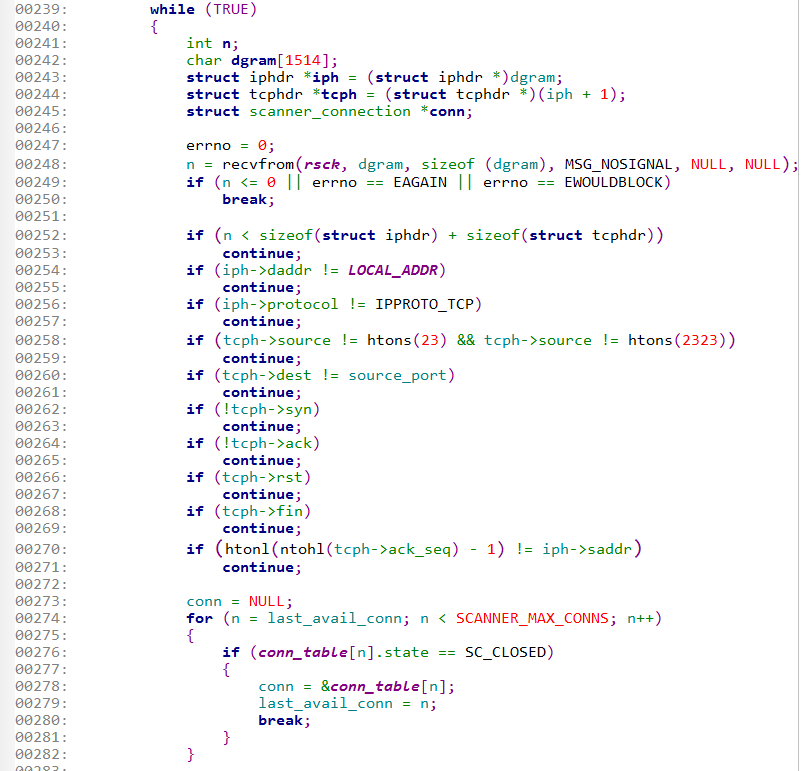
此时SOCK\_RAW的套接字就可以发挥作用了，从这个创建好的套接字可以用最快的速度向所有目标IP发送数据包。而数据包的构造要完全参照TCP的SYN包格式去构造。详情可参见源码207到234行。如下图所示：



205行是个for循环，遍历所有的目标IP。循环体中210至232行为数据包的构造代码，比较值得关注的是215行和227行，这两行代码是校验和计算函数，数据包的接收端会通过这个结果判断数据包是否在传输过程中发生错误，如果这个值算错的话，该数据包会被目标无情抛弃，因为数学不好的人没朋友。233行，就是发送函数了，发送的目标由paddr字段指定。

**地址筛选**

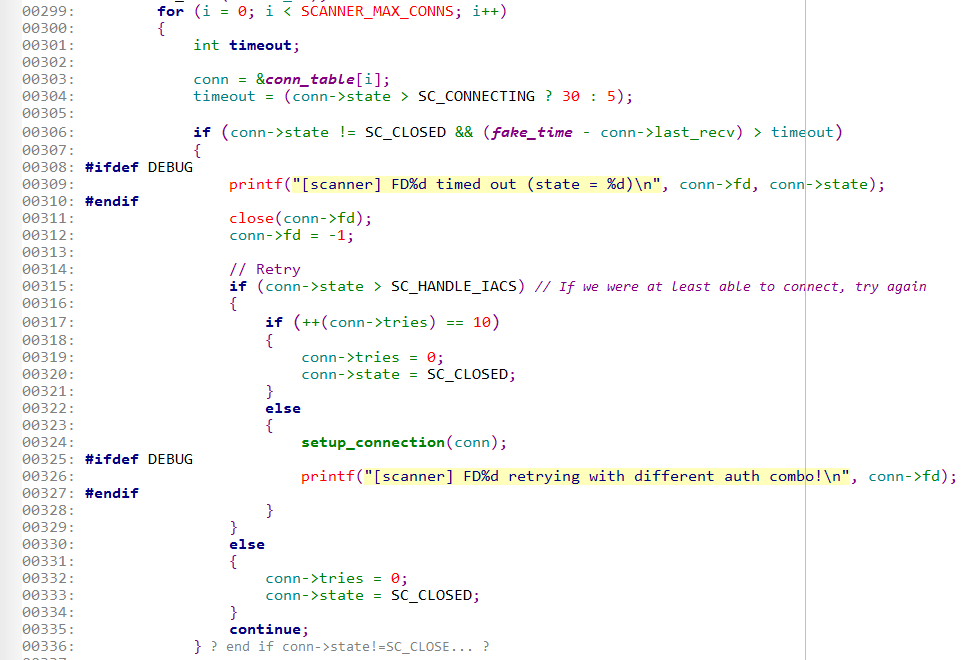
ACK请求发送后，无论对方指定端口是否开放，都会拿到一条回复数据包，通过对该数据包分析就能得知那些IP的端口开放了。代码从239行到282行，如下图所示：



248行依次收取每个主机的回包，249行到271行则负责对收到的数据包进行过滤，将一切不符合端口开放要求的IP进行跳过，能够执行到273行代码就表明，目标的端口是开放的，有必要记录下来等待后续的口令破解。

**依次建立TCP会话**

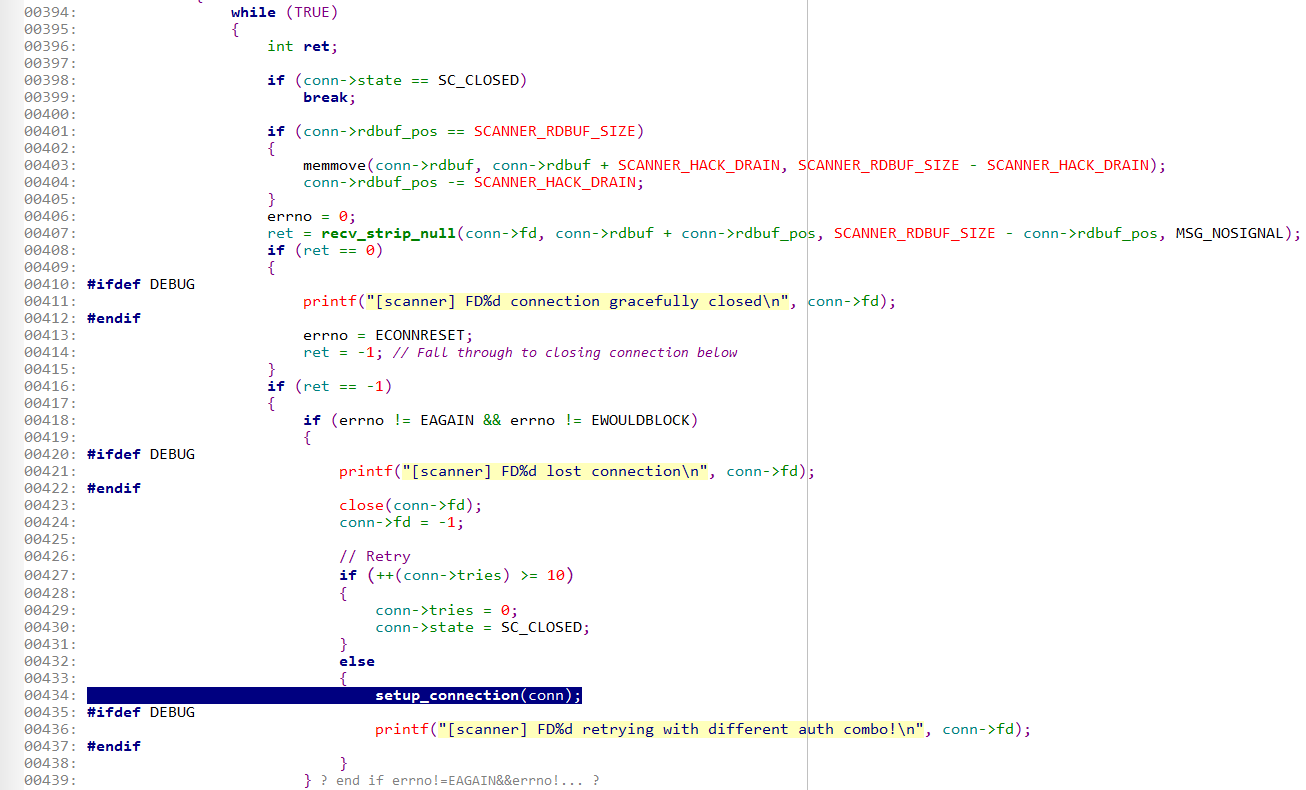
将所有开放目标端口的IP地址集中起来后，依次建立TCP会话，为后续破解做准备，此处代码无甚亮点，忽略即可，代码行号299至336行。



**口令枚举**

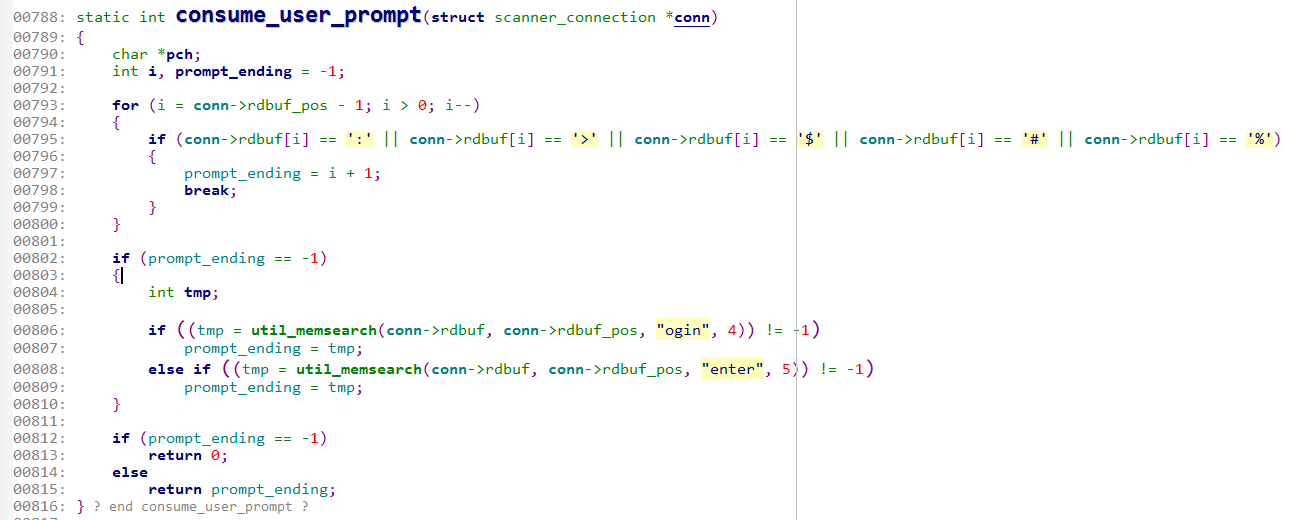
从394开始的while循环，是口令枚举的全部过程，该循环体占地极大，一直持续到整个函数结尾，Yeah你没算错，这个循环体一共有240多行，没法完整截图了，一段一段慢慢看吧。

396到441行，是从服务端收取数据的代码，如果第一次连接收取到的就是telnet的banner信息，如果非第一次连接收到的就是交互数据。当数据收取失败时，意味着socket中断，还需要重新连接（详见434行）



在这个大循环的内部，竟然还存在另一个while循环，使用这种编码方式的同学，都被编程老师骂死了，但种种迹象表明，作者目前还尚在人世，由此推测，Mirai的开发者一定是学校除名，无师自通，自学成才。

该循环起始于445行，这个while循环一直延续到634行，该while循环用于详细处理爆破过程的交互，它会从服务端发过来的数据分析中，推测当前爆破状态：是需要提供账号，还是要提供密码，再或者已经找到了账号密码，要检查下目标环境是否可用，是否能具有种植bot的必要。这里只选择一个和账号名有关的函数（consume\_user\_prompt，在788行）来分析，其他函数类似。



793行到801行，是对输入提示符的测试，常见的输入提示符基本也就这几种，枚举一轮就可以了。802行到810行，是对当前输入要求的检查，“ogin”是“Login”或“login”单词的尾字符，而“enter”则是“回车”的英文单词。检查无误后，基本可以断定当前状态是等待本地输入账户名的状态了。通过这样几行代码来判定账号名输入状态，还是挺好玩的。

**结束**

Mirai的源代码从编码质量上看，基本处在没老师教的状态，但其中用到的一些基础技术，却很值得细细玩味。比如，和跨平台有关的编译技巧，以及跨平台开发技巧，本文提到的SYN扫描技巧，还有未提到的“看门狗喂狗代码”等。这些东西从理论上了解是一种滋味，但被代码实现后，又发酵出另一番滋味。