# 周报：2011-03-31

07300720419 蒋菁

**本周已完成工作：**

针对SASEBO提供的测试程序SASEBO\_Checker\_AES的改进主要是以下方面：能指定明文的输入来源（键盘或文件）、能指定密钥、能将输出的密文保存在文件中。带着这些目的，我开始分析提供的源代码。

由于所有事件都由Start按键触发，所以我从Start对应的函数出发，寻找控制明文、密文、密钥的语句。Start按键对应的源代码位于Form\_Controller.cs中：

 //------------------------------------------------ Button, Start

 private void button\_start\_Click(object sender, EventArgs e)

 {

 if (button\_start.Text == "Start")

 {

 button\_start.Text = "Stop";

 int version = ctrl.module\_setup(cb\_if.SelectedIndex, cb\_lsi.SelectedIndex);

 Console.WriteLine(version.ToString("X4"));

 ctrl.tracer\_setup(cb\_lsi.SelectedIndex, tbox\_numtrace.Text, tb\_key.Text);

 ctrl.tracer\_addCompletedEventHandler(new RunWorkerCompletedEventHandler(worker\_RunWorkerCompleted));

 ctrl.tracer\_addProgressChangedEventHandler(new ProgressChangedEventHandler(worker\_ProgressChanged));

 ctrl.tracer\_run();

 }

 else

 {

 ctrl.tracer\_cancel();

 }

 }

其主要流程是：当按下Start后，按键显示变为Stop，×××，tracer\_setup()、tracer\_addCompletedEventHandler()、tracer\_addProgressChangedEventHandler()，开始运行加密进程（tracer\_run()）；当按下Stop后，进程终止（tracer\_cancel()）。

观察代码，发现参与控制的参量只有cb\_lsi.SelectedIndex、tbox\_numtrace.Text、tb\_key.Text，分别对应窗体上的LSI type、#Traces、KEY。先从容易修改的KEY入手，将

 ctrl.tracer\_setup(cb\_lsi.SelectedIndex, tbox\_numtrace.Text, tb\_key.Text);

改为

 ctrl.tracer\_setup(cb\_lsi.SelectedIndex, tbox\_numtrace.Text, File.ReadAllText("key.txt", Encoding.Default));

即指定从同目录下的key.txt读取KEY信息。

 然后添加控制向文件写入密文信息的语句，采用函数实现：

 void cipher\_out(TracerReport report)

 {

 File.AppendAllText("out.txt", format(report.text\_ans) + "\r\n", Encoding.Default);

 }

密文从text\_ans中读取（即窗口上的Answer）。其中需注意的是格式描述符，原本只写了 + "\n" 表示换行，但生成的文件内容并未产生此效果，改为 + "\r\n" 表示回车再换行，结果符合预期。再看以下代码：

 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* BackGroundWorker\_EventHandler

 //------------------------------------------------

 private void worker\_ProgressChanged(object sender, ProgressChangedEventArgs e) {

 label\_numtrace.Text = ((TracerReport)e.UserState).num\_trace.ToString();

 label\_text\_in.Text = format(((TracerReport)e.UserState).text\_in);

 label\_text\_out.Text = format(((TracerReport)e.UserState).text\_out);

 label\_text\_ans.Text = format(((TracerReport)e.UserState).text\_ans);

 }

可以看出，程序窗口上的数据是以“事件触发”方式表示的，因此在生成的out.txt中，首末两行是冗余数据，是text\_in和text\_out触发而生成的，不影响整体数据的完整性。而完成输出到文件的功能还需修改此段代码为：

 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* BackGroundWorker\_EventHandler

 //------------------------------------------------

 private void worker\_ProgressChanged(object sender, ProgressChangedEventArgs e) {

 TracerReport report = (TracerReport)e.UserState;

 label\_numtrace.Text = report.num\_trace.ToString();

 label\_text\_in.Text = format(report.text\_in);

 label\_text\_out.Text = format(report.text\_out);

 cipher\_out(report);

 label\_text\_ans.Text = format(report.text\_ans);

 }

其中优化了从类库中提取成员的代码，原本每次CYCLE总共提取四次，修改后只生成一次，并调用cipher\_out()函数。重新编译后将Trace设为10，在线运行后生成的文件和内容如下：



图1 KEY文件key.txt（预先建立）和密文文件out.txt



图2 out.txt的内容（红色圈内为冗余数据）

**遇到的问题：**

 最大的困难是无法定位生成明文的代码。一方面，与Start相关的代码中控制整个流程的参数如前文所示，只有LSI type、#Traces、KEY，而无表示明文的参量；另一方面，有理由认为明文和密钥都是通过随机函数生成，而在主函数中易找到生成密钥的代码：

 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Control

 //------------------------------------------------ Button, Start

 private void button\_key\_Click(object sender, EventArgs e)

 {

 String key\_st = "";

 for (int i=0; i<16; i++) key\_st += (rand.Next(256)).ToString("X2") + " ";

 tb\_key.Text = key\_st;

 }

而试图寻找与明文相关的代码是，Start动作代码的唯一出口是ctrl.tracer\_run()，通过Controller.cs引向类库Tracer，再经过几层引用后就指向了一个锁定的库中，由于不知道其来源和功能，因此不敢随便在内部进行改动，以防引起可能的系统错误。



图4 未知来源的引用（注意红圈）

 另一个需改进的工作是，现在外部文件的存取都是指定文件名的，如果能以弹出对话框选择文件路径和输入文件名的方式就更好了，这样的话可能需要在主函数中新建一个函数来描述这个功能。

**下周工作任务：**

 之前我已经从Agilent的网站上下载并安装了示波器的驱动程序，接下来就是通过示波器观察实验板上的电源波形和表示加密开始及结束的trigger信号，并且由此编写一个程序：当trigger信号到来时，示波器自动开始采集波形数据，直到trigger信号消失时，示波器停止采集数据。这样的话SASEBO\_Checker\_AES生成的密文就能和电源波形同步，才能正确分析两者之间的对应关系并猜测密钥。