Peach对Modbus功能码的模糊测试

Aug 10, 2017

♯基础知识

♯Modbus功能码

代码	中文名称	作用
01	读取线圈状态	取得一组逻辑线圈的当前状态（ON/OFF)
02	读取输入状态	取得一组开关输入的当前状态（ON/OFF)
03	读取保持寄存器	在一个或多个保持寄存器中取得当前的二进制值
04	读取输入寄存器	在一个或多个输入寄存器中取得当前的二进制值
05	强置单线圈	强置一个逻辑线圈的通断状态
06	预置单寄存器	把具体二进值装入一个保持寄存器
07	读取异常状态	取得8个内部线圈的通断状态，这8个线圈的地址由控制器决定，用户逻辑可以将这些线圈定义，以说明从机状态，短报文适宜于迅速读取状态
08	回送诊断校验	把诊断校验报文送从机，以对通信处理进行评鉴
09	编程（只用于484）	使主机模拟编程器作用，修改PC从机逻辑
10	控询（只用于484）	可使主机与一台正在执行长程序任务从机通信，探询该从机是否已完成其操作任务，仅在含有功能码9的报文发送后，本功能码才发送
11	读取事件计数	可使主机发出单询问，并随即判定操作是否成功，尤其是该命令或其他应答产生通信错误时
12	读取通信事件记录	可是主机检索每台从机的ModBus事务处理通信事件记录。如果某项事务处理完成，记录会给出有关错误
13	编程（184/384 484 584）	可使主机模拟编程器功能修改PC从机逻辑
14	探询（184/384 484 584）	可使主机与正在执行任务的从机通信，定期控询该从机是否已完成其程序操作，仅在含有功能13的报文发送后，本功能码才得发送
15	强置多线圈	强置一串连续逻辑线圈的通断
16	预置多寄存器	把具体的二进制值装入一串连续的保持寄存器
17	报告从机标识	可使主机判断编址从机的类型及该从机运行指示灯的状态
18	(884和MICRO 84)	可使主机模拟编程功能，修改PC状态逻辑
19	重置通信链路	发生非可修改错误后，是从机复位于已知状态，可重置顺序字节
20	读取通用参数（584L）	显示扩展存储器文件中的数据信息

代码	中文名称	作用
21	写入通用参数（584L）	把通用参数写入扩展存储文件，或修改之

Peach简介

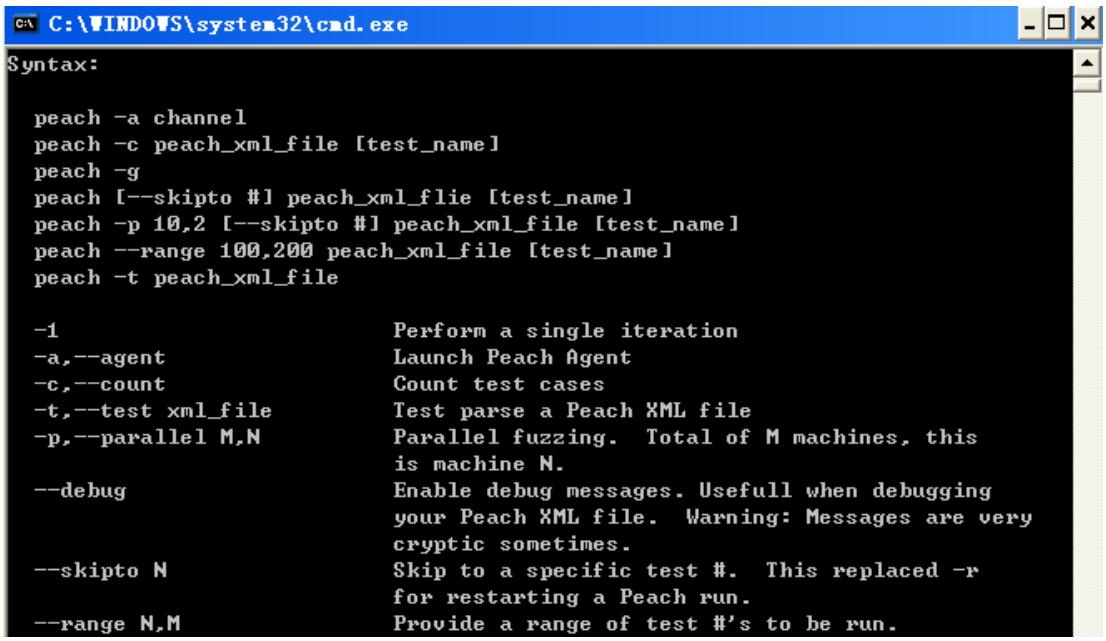
概述

Michael Eddington等人开发的Peach是一个遵守MIT开源许可证的模糊测试框架，最初采用Python语言编写，发布于2004年，第二版于2007年发布，最新的第三版使用C#重写了整个框架。

Peach支持对文件格式、ActiveX、网络协议、API等进行Fuzz测试；Peach Fuzz的关键是编写Peach Pit配置文件。

Windows下使用Peach3需要预先安装.net 4和windbg；Linux、OS X下需要安装Mono .net开发框架。

命令行参数



- -1：执行第1次测试。
- -a：启动Peach代理。不指定”channel”默认为本地代理（默认支持，无需显式启动）；“channel”可以指定为”tcp”远程代理。
- -c：统计测试用例数。
- -t：验证Peach Pit xml文件正确性。
- -p：并行Fuzz。运行Peach的机器总数为M，这是第N个。
- - debug：调试信息开关。
- - skipto：指定Fuzz跳过的测试用例数。
- - range：指定Fuzz的测试用例范围。

Peach Pit

在使用Peach进行Fuzz之前需要编写被称为”Peach Pit”的xml配置文件，其中包含着如何进行Fuzz的关键信息，如下图：

```
<?xml …版本，编码之类…>

<Peach …版本，作者介绍之类…>

<Include ...包含的外部文件… />

<DataModel >原始数据结构定义、可嵌套</DataModel>

<StateModel >测试逻辑，状态转换定义</StateModel>

<Agent >监测被测目标的反应，如crash等</Agent>

<Test >指定使用哪个StateModel 、Agent 、Publisher、

Strategy、 Logger等</Test>

</Peach>
```

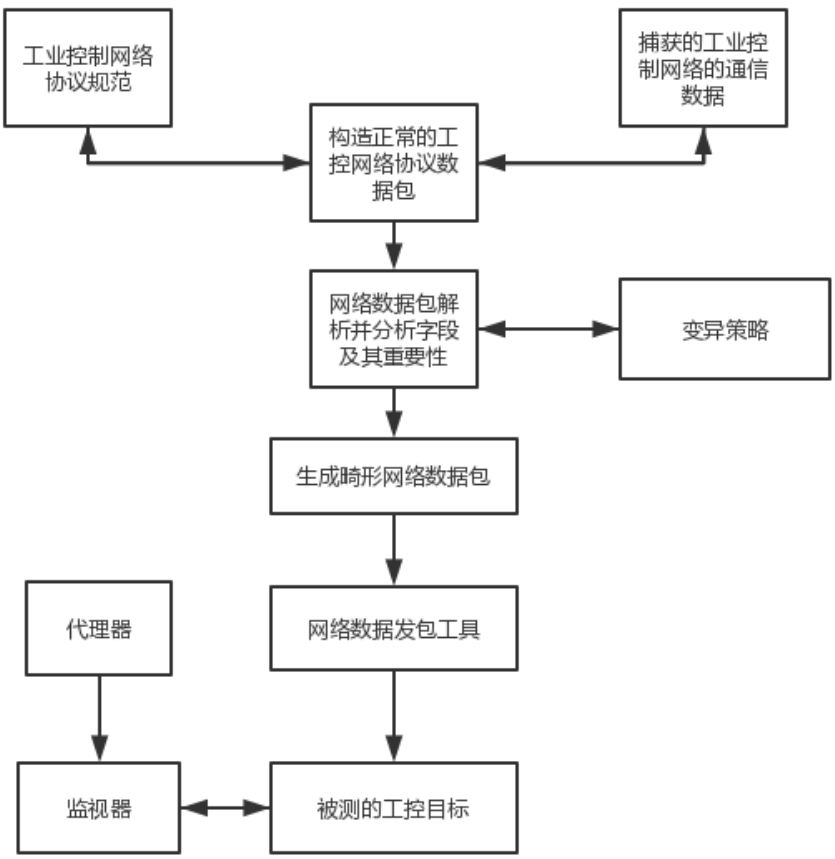
peach模糊测试

在研究网络协议模糊测试时，sulley和peach两大框架是最常见的Fuzz框架，peach相对于sulley有以下几点优势：

- 1、sulley目前已不再维护。
- 2、对sulley模糊测试编写程序，需要有一定的python语言基础。而peach是xml格式的，比较容易理解。
- 3、sulley配置环境相对繁琐，而peach配置环境相对简单（目前我手头有绿色版本，可以直接运行）。
- 4、sulley只能对网络协议进行模糊测试，而peach相对更加多样化。

Peach协议Fuzz

工作流程图



流程图具体工作原理

- 1、根据协议控制规范或者捕获工业控制网络协议数据流来构造正常的数据包；
- 2、分析正常协议的字段及其重要性；
- 3、根据分析的协议中不同的数据类型，设计有效地变异策略。
- 4、设计并实现工业控制网络协议数据包发包工具；
- 5、设计并实现代理器及监视器；
- 6、采用发包工具，将畸形数据包发送给被测工控目标；
- 7、通过监视器探测被测工控目标异常数据记录。

NetWork.xml简单分析

在Peach的目录里有个samples目录,里面有官方给出的一些Pit，以NetWork.xml文件为例做一个简单的分析，具体如下：

```
<DataModel name="HttpRequest">
    <String value="Hello World!" />
</DataModel>

<StateModel name="TheStateModel" initialState="TheState">
    <State name="TheState">
        <Action type="output">
            <DataModel ref="HttpRequest" />
        </Action>
    </State>
</StateModel>

<!-- Agents that run localy will be started automatically by Peach -->
<Agent name="RemoteAgent" location="tcp://192.168.1.190:9001">
    <Monitor name="Debugger" class="WindowsDebugger">
        <Param name="CommandLine" value="CrashableServer.exe 192.168.1.190 4242"/>
    </Monitor>

    <Monitor name="Network" class="PcapMonitor">
        <Param name="filter" value="tcp"/>
    </Monitor>
</Agent>

<Test name="Default">
    <Agent ref="RemoteAgent" />
    <StateModel ref="TheStateModel"/>

    <Publisher class="TcpClient">
        <Param name="Host" value="192.168.1.190" />
        <Param name="Port" value="4242" />
    </Publisher>

    <Logger class="Filesystem">
        <Param name="Path" value="Logs" />
    </Logger>
```

DataModel

StateModel

监听器

引入监听器

引入StateModel

测试

设置目标ip和端口

Modbus Fuzz Pit编写

由于是初次接触Peach对Modbus工控协议的模糊测试，目前也对Modbus协议不是很熟悉，所以只能在网络上找个各种资料。经过一番寻找，最终在github上找到了一份Pit，就是关于对modbus功能码的Fuzz Peach Pit。文章Reference处给出链接。简单地阅读和分析这个Pit发现，这个Pit对01 02 03 04 05 06 15 16 20 21 22 23 24 这几个功能进行Fuzz。除了22 23 24这几个功能码没有在文章开头提到，其他都在基础知识中有。关于22 23 24三个功能码具体如下

代码	作用描述
22	屏蔽写寄存器
23	读/写多个寄存器

代码	作用描述
24	读FIFO队列

另外发现，在我们找到的Pit中，缺少了Agents-Monitors部分即监听器模块部分。Agents部分可以在本地或远程运行，可以执行附加调试器的动作，看内存消耗，检测故障等。
查看官方文档可以知道Peach Fuzz根据不同的环境类型支持以下几种Monitors

Windows Monitors

- Windows Debugger Monitor
- Cleanup Registry Monitor
- Page Heap Monitor
- Popup Watcher Monitor
- Windows Service Monitor

OS X Monitors

- OS X Crash Wrangler Monitor
- OS X Crash Reporter Monitor

Linux Monitors

- Linux Crash Monitor

Cross Platform Monitors

- CanaKit Relay Monitor
- Cleanup Folder Monitor
- IpPower9258 Monitor
- Memory Monitor
- Pcap Network Monitor
- Ping Monitor
- Process Launcher Monitor
- Process Killer Monitor
- Save File Monitor
- Socket Listener Monitor
- SSH Monitor
- SSH Downloader Monitor
- Vmware Control Monitor

由于我们是对Modbus工控协议的模糊测试，这里我们可以用常见的Ping Monitor和Socket Listener Monitor。经过向一些之前有过Modbus Peach Fuzz研究的前辈的请教，得到结果是用Ping Monitor的误报率比较高，所以我们在编写是使用的是Socket Listener Monitor。

以下给出Ping Monitor和Socket Listener Monitor的编写模板

Ping Monitor

Parameters

- Host — Hostname or IP address
- Timeout — Timeout in milliseconds (optional, defaults to 1,000)
- Data — Data to send in ping packet (optional)
- FaultOnSuccess — Fault if ping is successful (optional, defaults to false)

Examples

```
<Agent name="Local">
        <Monitor class="Ping">
                <Param name="Host" value="www.google.com" />
        </Monitor>
</Agent>
```

Socket Monitor

Parameters

- Host — IP address of remote host (optional, defaults to “”)
- Interface — IP address of interface to listen on (optional, defaults to 0.0.0.0)
- Port — Port to listen on (optional, defaults to 8080)
- Protocol — Protocol type to listen for (optional, defaults to tcp)
- Timeout — Length of time to wait for incoming connection (optional, defaults to 1000 ms)
- FaultOnSuccess — Fault if no conection is recorded (optional, defaults to false)

Examples

```
<Agent name="Local">
        <Monitor class="Socket">
                <Param name="Port" value="53" />
        </Monitor>
</Agent>
```

由官方文档最终得到如下Agent

```
<Agent name="Local">
       <Monitor class="Socket">
                        <Param name="Host" value="192.168.1.100" />
                        <Param name="Port" value="502" />
       </Monitor>
</Agent>
```

在TEST模块里引用Agent模块

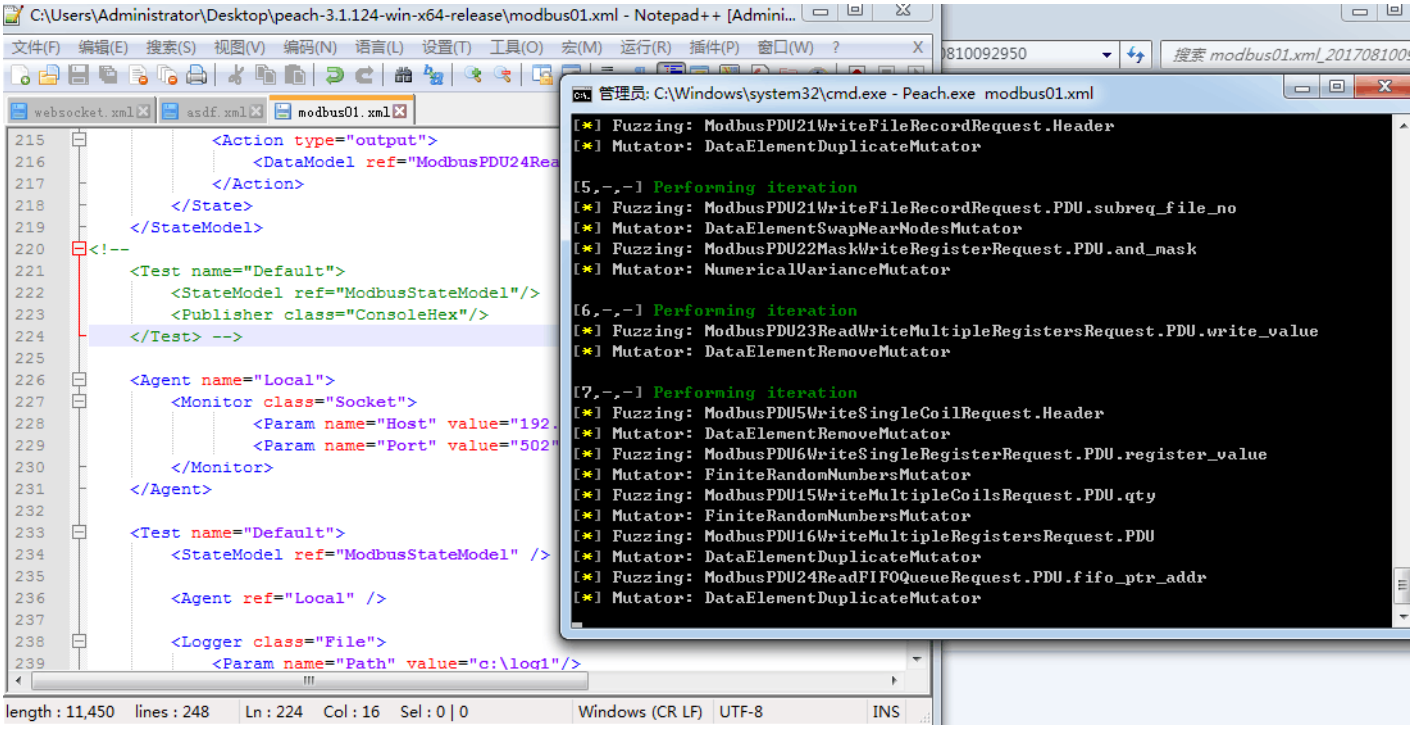
```
<Test name="Default">
        <StateModel ref="ModbusStateModel" />

        <Agent ref="Local" />

        <Logger class="File">
                <Param name="Path" value="c:\log1"/>
        </Logger>
        <Publisher class="tcp.Tcp">
                <Param name="Host" value="192.168.1.100"/>
                <Param name="Port" value="502"/>
        </Publisher>
</Test>
```

#运行测试

先简单的说明下这个Pit的基本原理就是给工控设备发送合法的功能码，然后对数据部分进行变异。在这个过程用监听器监听，如果出现崩溃就会出现崩溃日志。这个过程是很漫长的过程，经过我跟Modbus Peach Fuzz的前辈进行的交流，得知他们的模糊测试点都是经过逆向找到的，所以Pit是不能发给我的。而我们这边目前只能对功能码进行模糊测试。据我了解对Modbus协议的模糊测试，需要先提取固件，然后对固件逆向分析，找到模糊测试的点，国内做的很少。



♯Github Link

Modbus Peach Pit

♯Reference

Github Peach Fuzzer PIT Files
PeachPit官方文档
浅析Peach Fuzz
工控网络协议模糊测试：用peach对modbus协议进行模糊测试

上一篇 下一篇