基于状态的工控协议 Fuzzing 测试技术

张亚丰 洪 征 吴礼发 周振吉 孙 贺
(解放军理工大学指挥信息系统学院 南京 210007)

摘 要 针对传统 Fuzzing 测试应用于工控系统存在测试覆盖率和有效性低、异常监测手段受限等不足,提出了一种基于状态的工控协议 Fuzzing 测试方法。该方法采用 XML 脚本对协议状态机进行描述,设计了基于协议状态机的测试序列生成算法 PSTSGM,对被测对象进行状态引导以求达到更高的命中率和覆盖率。提出了基于心跳的异常监测与定位方法 HFDLM,采用心跳探测和循环定位的方式,对被测嵌入式设备进行异常行为监测和异常用例定位。设计并实现了基于中间人代理的模糊测试原型系统 SCADA-Fuzz,对电力 SCADA 系统进行了测试。实验结果表明,利用状态引导的测试能够有效发现安全漏洞。

关键词 工控协议,模糊测试,协议状态,漏洞挖掘
中图法分类号 TP393.08 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.05.024

Protocol State Based Fuzzing Method for Industrial Control Protocols

ZHANG Ya-feng HONG Zheng WU Li-fa ZHOU Zhen-ji SUN He
(College of Command Information System, PLA University of Science and Technology, Nanjing 210007, China)

Abstract Traditional fuzzing methods for industrial control system(ICS) have shortcomings of small coverage, low effectiveness and limitation of fault monitoring in fuzzing. This paper proposed a protocol state machine based fuzzing method for ICS protocols. Firstly, it describes the protocol state machine model with XML scripts, and designs a protocol state based test sequences generating method (PSTSGM) to achieve higher coverage rate during fuzzing process. Then, it puts forward a heart-beat based detecting and locating method for faults (HDLMF). It aims to detect embedded equipment behavior faults and locate the abnormal cases via the way of heart-beat detection and loop location. On the basis of the proposed method, we designed and implemented a fuzzing tool SCADA-Fuzz, and performed tests on a power control SCADA system. Experimental results show that SCADA-Fuzz can effectively and efficiently trigger behavior faults and locate security vulnerabilities.

Keywords Industrial control protocol, Fuzzing test, Protocol state, Vulnerability mining

工业控制系统(Industrial Control System, ICS)广泛应用于电力、水处理、石油与天然气、化工、交通运输、制造业等行业[1],是国家关键基础设施的大脑和中枢。一般而言,工控系统包括数据采集与监控系统(Supervisory Control and Data Acquisition, SCADA)、分布式过程控制系统(Distributed Control Systems, DCS)、可编程逻辑控制器(Programmable Logic Controller, PLC)、远程测控单元(Remote Terminal Unit, RTU)等[1]。

随着工业信息化的快速发展,在极大提高工业生产效率的同时,也暴露了工控系统的弱点。2010年攻击伊朗布什尔核电站的"震网(Stuxnet)"病毒被认为是全球首个投入实战的网络"超级武器"[2];2011年发现的又一恶性蠕虫"Duqu"病毒[3],其目的是专门攻击工控系统和收集有关情报;2012年发现的"火焰(Flame)"病毒[4],其设计更为复杂,破坏力更强,可能已经潜伏5年之久;2014年6月,"蜻蜓组织"利用"Havex"病毒,对欧美地区的一千多家能源企业进行了攻击[5];还有最近的2015年圣诞节期间,乌克兰电网系统遭遇黑客组织利用"BlackEnergy"恶意软件的攻击,导致大规模的停电[6]。这一系列事件表明:针对工控系统的高级持续性威胁(Advanced Persistent Threats)[7]已从虚幻的好莱坞大片变成了现实,引起了各国政府对关键基础设施安全问题的高度关注。

同普通信息系统一样,攻击者之所以能够对工控系统实施攻击,一个重要的原因是工控系统中的软硬件存在可利用的安全漏洞[8]。对工控系统中的软硬件进行安全性测试,挖掘其中可能存在的安全漏洞,采取相应补救措施,对提高工控

到稿日期:2016-03-11 返修日期:2016-05-24 本文受国家自然科学基金资助项目(611032253),江苏省自然科学基金资助项目(BK2011115)资助。

张亚丰(1990—),男,硕士生,主要研究方向为信息安全,E-mail:zhangyafeng1001@126.com;洪 征(1979—),男,博士,副教授,主要研究方向为信息安全;吴礼发(1968—),男,博士,教授,主要研究方向为信息安全;周振吉(1985—),男,博士,讲师,主要研究方向为信息安全;孙 贺(1990—),男,硕士生,主要研究方向为信息安全。

系统的安全性具有重要意义。

Fuzzing 测试又称为模糊测试[9]，其使用大量半有效的数据作为目标程序的输入，通过监控程序出现的异常来发现潜在的漏洞。在工控系统中，SCADA 和 PLC 等各组件之间数据的传递均以工控网络协议为载体，工控系统组件负责解析、处理工控网络协议，其安全性直接影响着工控系统的安全。将 Fuzzing 测试技术应用于工控软硬件的脆弱性分析是当前工控安全领域的一个重要研究方向。同时研究也表明[10-11]，利用基于协议知识的 Fuzzing 测试[12]挖掘工控系统中存在的安全漏洞是一种可行且有效的方法。

目前，国内在工控协议模糊测试方向的研究还处于起步摸索阶段。匡恩网络公司发布的工控网络安全漏洞挖掘检测平台[13]具有一定的模糊测试功能；国家安全实验室于 2015 年发布了基于 Fuzzing 测试的工控通讯协议健壮性测试验证平台[14]。国外已有研究成果大都改进现有网络协议 Fuzzing 测试工具后用于对工控协议的测试。例如：美国 Tipping-Point 公司研究员 Devarajan 专门为 Sulley 开发了工控协议 ICCP，Modbus，DNP3 的模糊测试模块[15]；德国奥格斯堡应用技术大学的 Roland Koch 等人在 Scapy fuzzer 的基础上研制了 ProFuzz[16]，专门针对 Profinet 协议族进行模糊测试；Wurldtech 公司的 Byres 等人设计并实现了 BlackPeer 测试框架[17]，对两款 PLC 设备的 Modbus/TCP 协议栈进行了测试，成功发现 60 多类缺陷；Bratus 等人在通用 Fuzzing 测试器 GPF[18]的基础上实现了一个针对私有和公开的工控协议的模糊测试器 LZFuzz[19]；美国 Digitalbond 公司开发了商业测试软件 ICCPSic[20]，专门用于 ICCP 协议的测试；早前的 Mu Dynamic 公司推出了商业的 Mu 测试组件[21]，能够支持对 IEC61850，Modbus/TCP 和 DNP3 协议的模糊测试；Wurldtech公司研发的 Achilles 模糊测试仪[22]，是一款专业化的工控协议模糊测试商用工具，支持多种常见的工控协议；Codenomicon 公司推出的 Defensics 商业模糊测试工具[23]，支持 Modbus/TCP 协议；SecuriTeam 公司在商业软件 beSTORM 中增加了对 DNP3 协议的模糊测试功能[24]。

以上研究成果大多都是基于现有网络协议 Fuzzing 测试工具进行扩展的，缺乏针对协议知识的功能设计。此外，工控协议具有以下特点：1）面向会话，具备交互状态；2）报文简短，高度结构化；3）大多数缺乏认证和加密。目前，将传统的 Fuzzing技术应用于工控系统的测试主要存在以下不足：1）未考虑协议交互状态，测试的覆盖率和有效性低；2）传统的测试方式和异常监控手段受限，不支持嵌入式的工控设备。

为弥补上述不足，文中提出一种基于状态的工控协议 Fuzzing 测试方法。利用 XML 脚本对协议状态机模型进行描述，提出了基于状态机的测试用例生成算法 PSTSGM，对被测对象进行状态引导，以达到更高的有效性和覆盖率；提出基于心跳的异常监测与定位方法，用于嵌入式设备的异常监测；在 Peach[25]的基础上实现了基于中间人代理的模糊测试原型系统 SCADA-Fuzz，对电力 SCADA 系统进行了测试。实验结果表明，利用状态引导的测试提高了测试的有效性和触发漏洞的概率，并发现了更多安全漏洞；与 Peach 相比，SCADA-Fuzz 虽然耗时较大，但测试结果较好。

1　基于状态机的工控协议描述模型

目前，针对工控协议的 Fuzzing 工具大都忽略了工控协议具备交互状态这一特点，它们以单一协议报文为样本进行变异；然而，很多漏洞的触发不是由单个异常报文的输入引起的，而是由多个相互关联的测试报文以一定的次序输入，引导被测系统处于某一异常状态之下，输入畸形测试报文所导致；再者，测试中实时跟踪和记录测试目标的代码覆盖率是一项艰难的工作。因此，综合考虑被测对象的状态与模糊用例的关联，利用协议状态机来引导被测对象到达某一特定状态，然后注入模糊报文，从理论上利用对状态机的覆盖来提高覆盖率，从而挖掘出更多的潜在漏洞。

1.1　有限状态机原理

定义 1　有限状态机为一个五元组 $M=(S,I,\delta,s_0,F)$，其中，S 为非空有限状态集合；I 为输入符号集合；$\delta: S\times I\to S$ 为状态转移函数，其为一对一或一对多的映射；S_0 为初始状态；$F\subseteq S$是终止状态集合。

定义 2　给定一个有限状态机 $M=(S,I,\delta,s_0,F)$，其中 $\forall s\in S,\forall i\in I$，且 $|\delta(s,i)|\leqslant 1$，则称 M 为确定型状态机，否则称为非确定型状态机。

利用确定型状态机描述目标协议实体的交互行为时，可将协议实体的输入报文序列抽象为输入符号集 I，下一状态的迁移根据当前状态和报文输入产生。

1.2　工控协议状态机描述模型

协议状态机发生迁移之后的状态都是唯一确定的，因此，协议状态机可用确定有限自动机来进行形式化描述。以 Profibus 协议为例，说明如何利用有限状态机来描述其协议状态的迁移。图 1 是 Profibus 协议状态机。

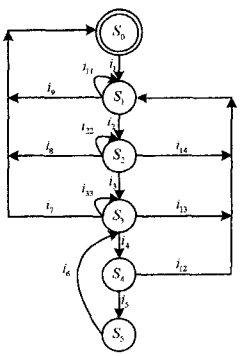

图 1　Profibus 协议状态机

其中，状态集 $S=\{S_0$：Power-ON(初态)，S_1：Wait-PRM(等待参数化完成)，S_2：Wait-CFG(等待配置完成)，S_3：Data-EXCH(数据交换)，S_4：Check-Freeze(检查冻结)，S_5：Check-Clear(检查清除)$\}$；输入符号集 $I=\{i_1$：Set_Slave_addr(设置地址)，i_2：Set_Prm(设置参数)，i_3：Chk_cfg(检查配置)，i_4：Req_FDL_Status(链路状态请求)，i_5：Freeze(冻结)，i_6：Clear(清除)，$i_7/i_8/i_9$：FATAL，$i_{12}/i_{13}/i_{14}$：ABORT，i_{11}：Slav_Diag(诊断请求)，i_{22}：Get_cfg(获取配置)，i_{33}：Data_Exchange(数据交互)$\}$；状态转移函数 $\delta=\{S_n\times i_{n+1}\to S_{n+1}\ (n=1,2,\cdots,4)$，$S_n\times i_m\to S_0\ (n=1,2,3;m=9,8,7)$，$S_5\times i_6\to S_3$，$S_n\times i_m\to S_1\ (n=2,3,4;m=14,13,12)\}$；终止状态集 $F=\{S_0\}$。

在输入符号集 I 中，每一条输入对应着与 Profibus 协议实体交互的 Profibus 协议报文。

2 基于状态机的测试序列生成

状态机作为描述完整工控协议规范的一部分，为了能够利用状态机模型对工控系统组件进行 Fuzzing 测试，还需要将协议状态机抽象模型实例化为具体的描述脚本，其转化过程如图 2 所示。以工控协议历史会话流量报文为输入，状态机推断模块根据历史会话流量推断得到工控协议状态机，然后由脚本生成模块将推断出的状态机模型转换为SCADA-Fuzz可识别的XML 描述脚本，最终测试序列生成模块结合协议状态的迁移流程以及相应的测试用例策略生成包含正常交互引导报文序列的测试序列集。

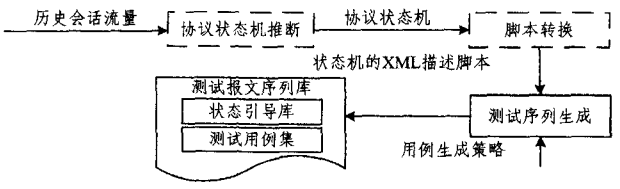

图 2 测试序列生成流程

2.1 工控协议状态机描述

本文采用协议逆向工具 Netzob[26]进行状态机推断，同时将推断出的协议状态机有向图转换为测试工具所识别的 XML 描述脚本。Netzob 可直接导出 Peach 所识别的协议描述 XML 脚本，但实践中存在以下局限性：

(1)从 Netzob 来看，导出协议描述脚本的状态模型〈StateModel〉存在局限性，只能简单用来描述报文发送和接收的交互顺序，虽然〈Action〉定义一个状态执行动作，但并没有明确指明状态的迁移目标。

(2)一方面，从 Peach 的 pit 语法来看，〈Action〉标签中可利用 when 属性定义的表达式来描述状态迁移条件(见图 3)，但 Netzob 无法自动推断生成；另一方面，描述具有复杂状态迁移的协议，一般需要多个〈StateModel〉协同，而且难以利用 when 属性来定义状态迁移条件。

```
〈StateModel name="StateModel" initial State="InitialState"〉
  〈State name="InitialState"〉
    〈Action type="input"〉
      〈DataModel ref="InputModel"/〉
    〈/Action〉
    〈Action type="changeStateM" ref="State2"
     when="int(StateModel. states['InitialState']. actions[0].
     dataModel['Type']. InternalValue)=2"/〉
  〈/State〉
  〈State name="State2"〉
    〈Action type="output"〉
      〈DataModel ref="OutputModelA"/〉
  〈/Action〉
〈/State〉
```

图 3 when 属性定义示例

(3)从测试人员的角度，图 3 所示的 when 属性定义方法略显复杂，状态迁移信息的描述方式不够直观，导致脚本的理解难度和编写复杂度较大。

(4)从测试的角度，Peach 利用 StateModel 描述的状态机在测试过程中缺乏专门的状态引导机制，测试中根据〈StateModel〉描述，期望利用模糊报文实现状态之间的迁移，而没有利用正常交互报文来进行状态引导，难以实现状态的深度测试，从而也难以发现目标处于不同协议状态时存在的安全漏洞。

针对上述问题，需要设计新描述方法表示协议状态机模型：1)将 Netzob 推断出的状态机有向图转换为具有完善状态迁移信息描述的脚本；2)简化 pit 语法对状态迁移的描述，直观表达状态迁移信息；3)避免测试人员过多纠结于复杂状态机描述的语法规范，降低脚本的编写难度；4)从测试角度，利用状态机模型实现协议实体的状态引导。考虑到协议格式脚本与协议状态机脚本的复用，协议报文格式和协议状态机分别利用不同 XML 脚本进行描述。

以 Profibus 协议为例，部分状态机描述脚本 StateMachine. xml 如图 4 所示。

```
〈?xml version="1. 0" encoding="utf-8"?〉
〈SFuzz type="StateMachine"〉
  〈Include src="Message. xml"/〉
  〈StateMachine name="ProfiStateMachine"〉
    〈!-- States--〉
    〈State name="InitialState"/〉
    〈State name="WaitPrm"/〉
    〈State name="WaitCfg"/〉
    〈State name="DataExch"/〉
    …
    〈!-- Transition--〉
    〈Trans name="i1" from="InitialState" to="WaitPrm"〉
    〈Action name="SetSlaveAddrRequest" type="output"〉
      〈Message ref="ProfiSetSlaveAddrRequest"/〉
    〈/Action〉
    〈Action name="SetSlaveAddrResponse" type="input"〉
     〈Message ref="ProfiSetSlaveAddrResponse"/〉
    〈/Action〉
    〈/Trans〉
    〈Trans name="i2" from="WaitPrm" to="WaitCfg"〉
      〈Action name="SetPrmRequest" type="output"〉
        〈Message ref="ProfiSetPrmRequest" /〉
      〈/Action〉
      〈Action name="WaitPrmResponse" type="input"〉
        〈Message ref="ProfiSetPrmResponse"/〉
      〈/Action〉
    〈/Trans〉
    〈Trans name="i3" from="WaitCfg" to="DataExch"〉
      〈Action name="ChkCfgRequest" type="output"〉
        〈Message ref="ProfiChkCfgRequest" /〉
      〈/Action〉
      〈Action name="WaitCfgResponse" type="input"〉
        〈Message ref="ProfiChkCfgResponse"/〉
      〈/Action〉
    〈/Trans〉
    …
  〈/StateMachine〉
〈/SFuzz〉
```

图 4 Profibus 协议状态机描述脚本示例

在标签〈StateMachine〉中分为描述状态节点的子标签〈State〉和描述状态迁移信息的子标签〈Trans〉。〈Trans〉中 from 表示协议实体的当前状态，to 表示在执行〈Action〉动作之后，协议实体迁移的目标状态。〈Action〉标签表示状态迁移时对应的执行动作，type="output"表示测试引擎向测试目标的输出动作，即发送特定测试报文，type="input"表示测试引擎等待接收测试目标的回复报文。〈Message〉标签表示协议报文的描述模型，ref 属性表示引用，例如图 4 中 name="i2"的状态迁移，测试引擎发送的模糊测试报文根据 ProfiSetPrmRequest 的 MABNF 范式语法模型变异生成[27]。

```
〈Message name="ProfiSetPrmRequest"〉
  〈Block name="Header"〉
    〈Number name="SD3" size="8" value="68" valueType=
     "hex" endian="network" mutable="false"/〉
    〈Number name="LE" size="8" value="04" valueType=
     "hex" endian="network" mutable="true"〉
      〈Relation type="size" of="PDUData"/〉
    〈/Number〉
    〈Number name="LER" size="8" value="00" valueType=
     "hex" endian="network" mutable="true"/〉
  〈/Block〉
  〈Number name="SDR" size="8" value="36" valueType=
   "hex" endian="network" mutable="true"/〉
  〈Block name="PDUData"〉
    〈Number name="DA" size="8" value="09" valueType=
     "hex" endian="network" mutable="true"/〉
    〈Number name="SA" size="8" value="06" valueType=
     "hex" endian="network" mutable="true"/〉
    〈Number name="FC" size="8" value="01" valueType=
     "hex" endian="network" mutable="true"/〉
    〈Number name="DSAP" size="8" value="06" valueType=
     "hex" endian="network" mutable="true"/〉
    〈Number name="SSAP" size="8" value="06" valueType=
     "hex" endian="network" mutable="true"/〉
    〈Number name="DU" size="16" value="0006" valueType
     ="hex" endian="network" mutable="true"/〉
  〈/Block〉
  〈Block name="FCSandED"〉
    〈Number name="FCS" size="8" value="0006" valueType
     ="hex" endian="network" mutable="false"/〉
    〈Number name="ED" size="8" value="16" valueType=
     "hex" endian="network" mutable="false"/〉
  〈/Block〉
〈/Message〉
〈Message name="ProfiChkCfgRequest"〉
...
〈/Message〉
...
```

图 5　Set_Prm 报文的 MABNF 描述脚本示例

由于工控协议具有报文简短、高度结构化的特点，对于协议格式的描述，本文使用 MABNF 范式语法进行描述，详见文献[27]。MABNF 范式语法在 ABNF 基础上进行了改进，引入功能函数，对协议字段在语义上实现更直观的描述，以此来增强对协议字段语义依赖关系的表达，从而利于根据功能函数描述的语义依赖关系生成有效性更高的测试用例[27]。图 5 基于 MABNF 范式语法对 profibus 协议的 Set_Prm 请求报文进行了描述。

2.2　基于状态机的测试序列生成

基于自定义的 XML 描述，协议状态机从一个抽象的数学模型转化为测试工具可识别的实例脚本。脚本解析模块对协议状态机脚本描述的状态机模型进行解析，在协议状态机基础之上，构建每个状态的正常交互报文引导序列库，确保对于一个工控协议实体组件，当其处于初始状态时，能够通过一系列的正常交互报文将工控协议实体引导至协议状态机中的任意一个工控协议状态。

为了提高测试的深度，增加覆盖率，需要对协议状态机包含的各个协议状态进行测试，从而有效发现当工控组件处于不同协议状态时存在的安全漏洞；在测试过程中，需要将工控组件通过正常的交互报文引导至需要测试的某一个工控协议状态中，在该协议状态下对工控组件实施模糊测试。

如图 6 所示，说明状态引导的测试。假设对状态 S_2 进行测试，首先向处于初态的被测对象输入状态引导报文序列 i_1，i_2，使其处于状态 S_2 下，然后利用模糊的状态迁移报文 $i_{3\text{-}Fuzzed}$ 实现对状态 S_2 的测试。

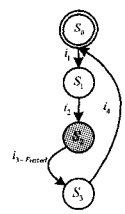

图 6　状态引导测试示意图

为了引导被测实体状态，提高测试的覆盖率，设计并实现了基于协议状态机的测试序列生成算法(Protocol State based Test Sequences Generating Method，PSTSGM)，如算法 1 所示。算法主要利用图的广度优先遍历思想对状态机有向图进行遍历，求得初始状态顶点到其他各顶点的边路径，建立从初态至任意后续状态的状态引导报文序列库。算法以协议状态机有向图为输入(第 1 行)，首先为每个状态分别建立一个测试序列 *SeqQueue* 和状态引导报文序列的优先队列 *NomMegQue*，并置为空(第 2—4 行)；然后建立一个辅助优先队列 Queue，初始状态节点 S_0 入队列(第 6 行)，当队列不为空时(第 7 行)，循环从队列中取出状态节点 s(第 8 行)，若 s 的直接后继节点 q 未被遍历(第 9—11 行)，则根据状态 s 到 q 的转移边 $Trans_{s\to q}$ 所对应的报文 MABNF 语法描述脚本的数据字段是否存在 mutable=true 的属性，若存在则利用基于 MABNF 语法变异树的测试用例生成策略(MABNF-TGM)变异相关字段，生成模糊报文(第 12—13 行)，状态 s 的测试序列即 s 的状态引导报文序列 $NomMegQue[s]$ 加上模糊的状态迁移报文 $Trans_{mutated;s\to q}$(第 16 行)，而 q 的状态引导报文序列即 s

的状态引导报文序列 $NomMegQue[s]$ 加上 s 至 q 的正常状态迁移报文 $Trans_{s\to q}$。

算法 1 基于协议状态机的测试序列生成算法(PSTSGM)

输入:状态集 States,状态迁移集 Trans,空闲初(终)态 S_0
输出:测试报文序列 SeqQueue

```
1. PSTSGM(States,Trans)
2.   for each Si in States
3.     SeqQueue[Si]←Ø;//状态 Si 的测试序列
4.     NomMegQue[Si]←Ø;//状态 Si 的正常报文序列
5.   end for
6.   Queue.put(S0);//辅助优先队列 Queue,S0 入队列
7.   while(!Queue.empty())
8.     s=Queue.get();visited[s]=true;
9.     for each state q adjacent to s
10.      if(visited[q]==false) then
11.        visited[q]=true;
12.        if(mutable(Trans_{s→q})==true)
13.          Trans_{mutated;s→q}=MABNF-TGM(Trans_{s→q});
14.        end if
15.        NomMegQue[q]=NomMegQue[s]+Trans_{s→q};
16.        SeqQueue[s]=NomMegQue[s]+Trans_{mutated;s→q};
17.        Queue.put(q);
18.      end if
19.    end for
20.  end while
21. return SeqQueue
```

PSTSGM 通过遍历协议状态机有向图,建立包含从初态至任意后续状态的状态引导报文序列的测试序列库,其主要优势在于建立了状态引导报文序列库,有助于利用状态引导报文序列对被测目标实现深度状态引导,随之利用变异的模糊报文实施测试,有助于触发目标处于不同协议状态的异常,提高测试的覆盖率,从而发现目标更多的安全漏洞。

3 异常监测

3.1 传统的异常监测方法

在传统的 Fuzzing 测试中,异常监测的方法主要是调试器跟踪和日志分析。调试器跟踪需要在被测软件所在的平台上安装本地监视器,但工控系统的运行环境封闭,像 PLC 和 RTU 等组件属于嵌入式设备,难以安装本地监视工具,所以该方法只适用于工控系统中的协议软件,无法应用于 PLC 和 RTU 等组件;日志分析主要对工控设备服务器的日志进行解析,判断工控系统软件是否产生异常,但由于 PLC 和 RTU 等组件是嵌入式设备,其计算能力、存储资源还有网络访问均受到了严格限制,难以实现异常信息的日志记录和访问。

3.2 基于心跳的异常监测与定位方法

异常监测作为整个 Fuzzing 测试过程的关键环节,其异常判别条件的选取尤为重要。由于工控设备具有高实时性和高可靠性的特征,对于点对点的请求的响应速度很快,这一特性可作为被测设备异常判别与监测的必要条件;再者,工控嵌入式组件在测试过程中,常常由于无法正确处理模糊报文从而导致 PLC 和 RTU 等组件无法响应,需要重启设备才能继续测试。鉴于此,文中提出基于心跳的异常监测与定位方法(Heartbeat based Faults Detection and Location Method,HFDLM),具有广泛的适用性。所谓心跳报文,是指以广播或单播方式向目标发送请求报文,如 PING 命令、ARP 命令或有关协议的诊断报文等,通过响应报文来判断目标是否处于正常运行状态。

由于某些工控设备是双处理器的架构,对数据交互的处理存在一定延时,若是采取传统的异常定位方法,即发现异常时就记录当前的测试用例,可能造成定位的精确度不高,甚至不准确;另外,与协议状态相关联的测试,异常并不是由单一测试用例引起的,而是先利用正常报文交互序列将目标引导至某一状态,发送模糊的报文所致,传统的异常定位方法无法准确定位引起异常的测试序列。因此,本文在基于心跳的异常监测与定位算法中引入回溯深度 L,采用逐一心跳探测与回溯定位的方式,在随后的异常重现的验证与定位中,可有效提高定位的准确度。

在算法 2 的测试用例执行阶段,若心跳探测发现拒绝服务异常,则记录当前异常测试用例及其编号 *FaultSets*(第 1 行),并将目标重启并置于初态(第 2 行)。下一步,采用逐一心跳探测与回溯定位的方式,重现和精确定位引起异常的测试用例或序列。首先设置最大回溯深度 *MaxL* 为状态集基数|*States*|或 *card*(*States*)(第 3 行)。然后逐一重现异常测试用例(第 4 行),设初始回溯深度 $L=0$(第 5 行),发送异常测试用例 N(第 6 行)。利用心跳探测,在响应延迟 *interval-time* 之内,若没有心跳响应,则说明是当前用例引起了拒绝服务异常,记录下来(第 7—9 行);若收到心跳响应(第 10—11 行),则首先重置目标于初态(第 12 行),逐步增大回溯深度 L(第 13 行),顺序发送$[N-L,N]$区间测试序列,并监测目标在发送用例 N 之后是否出现拒绝服务异常(第 14—15 行)。若出现异常,则说明拒绝服务异常由$[N-L,N]$区间测试序列联合作用所致(16—19 行);若未出现异常,则继续增加回溯深度 L,重现和定位引起拒绝服务异常的测试序列(第 20 行)。最终得到导致拒绝服务异常的测试序列集 *FaultSeqA*。

算法 2 基于心跳的异常监测和定位算法(HFDLM)

输入:测试报文序列 SeqQueue,状态集 States
输出:引起拒绝服务的报文序列 FaultSeqA

```
HFDLM(SeqQueue,States)
1.  FaultSets←ExecTests(States,SeqQueue,HeartBeat);//用例执行
2.  RESTART;//置于初态
3.  Set(MaxL);//最大回溯深度,为状态集基数|States|或 card(States)
4.  for each Msg in FaultSets do
5.    L←0;
6.    Send(param,Msg,N);//发送异常用例
7.    HeartBeat(parm,interval-time,count);//心跳探测
8.    if received no response then
9.      FaultSeqA←(Msg,N);
10.   else
11.     while(L≤MaxL) do
```

```
12.            RESTART;//置于初态
13.            L←L+1;
14.            SendSeq(parm,SeqQueue,[N−L,N]);//发送异常序列
15.            HeartBeat(parm,interval-time,count);//心跳探测
16.            if received no response then
17.              FaultSeqA←(SeqQueue,[N−L,N]);
18.              break;
19.              RESTART;//置于初态
20.            else continue;
21.            end if
22.          end while
23.        end if
24.    end for
25. return FaultSeqA
```

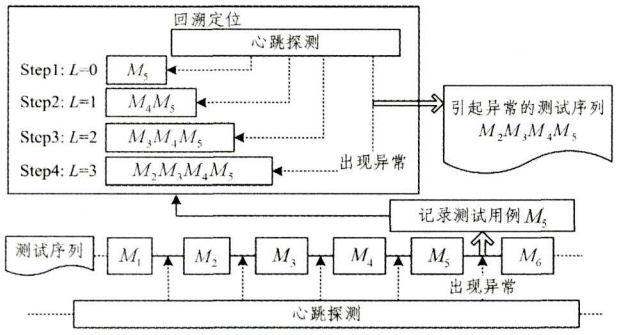

图 7　基于心跳的异常监测与定位示例

以图 7 为例，测试时，在发送用例 M_5 之后，利用心跳探测发现目标出现拒绝服务异常，则记录当前测试用例 M_5。为了精确定位引起异常的测试序列，在 M_5 的基础上进行回溯定位。首先设步长 $L=0$，向目标发送 M_5，并探测目标是否出现同样异常；然后增大步长 $L=1$，发送 M_4M_5，并探测目标是否出现同样异常；以此类推，当步长增至 $L=3$ 时，向目标发送 $M_2M_3M_4M_5$ 之后，探测到目标出现同样异常，则成功定位引起拒绝服务异常的测试序列 $M_2M_3M_4M_5$。

4　实验与结果分析

4.1　实验设计

为了验证上述方法的有效性，基于 Peach，设计并实现了原型系统 SCADA-Fuzz。

4.1.1　SCADA-Fuzz 总体架构

传统的 Fuzzing 测试工具难以直接应用于工控系统，文中设计了基于中间人代理模式的 Fuzzing 测试架构，该架构以中间人形式内联在客户端和服务端，使得测试工具能在线捕获协议报文，实施实时 Fuzzing 测试。同时，由于大多数工控协议缺乏身份认证，且报文没有加密，很容易捕获未加密的报文并对其进行变异修改，从技术上保证了基于中间人代理模式下的 Fuzzing 测试完全可行。SCADA-Fuzz 系统的总体架构如图 8 所示，包括协议采样、测试引擎、监控代理三大组件。

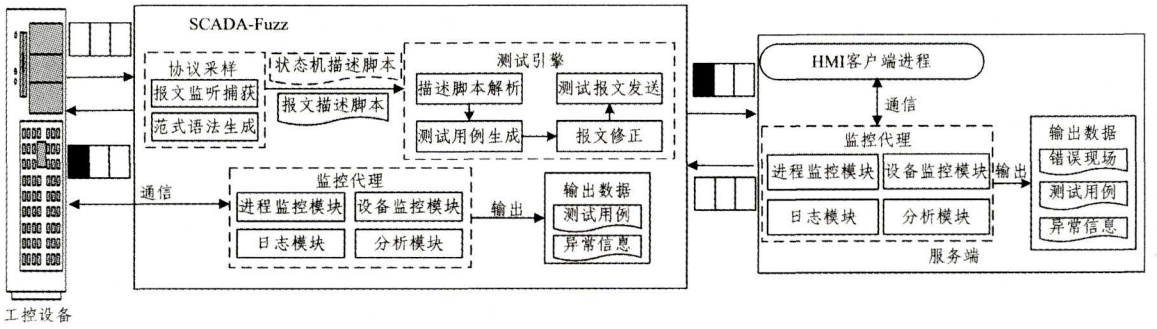

图 8　SCADA-Fuzz 架构图

SCADA-Fuzz 具备两种工作模式：在线模式和离线模式。在离线模式下，SCADA-Fuzz 首先正常交互报文引导被测对象，实施对状态的深度测试，提高测试的覆盖率，后文中称为 SCADA-Fuzz 状态引导测试；而在在线模式中，主要是为了满足实时性协议的测试，在实际中难以利用状态机对被测对象进行状态引导，因此在线模式下的测试只是在线捕获正常报文，利用 MABNF 范式语法生成测试用例发送给被测端，后文称为 SCADA-Fuzz 消息级测试。

4.1.2　测试环境

实验测试环境为某工业智能化实验室的“电力 SCADA 系统”，其系统的整体架构如图 9 所示。

该系统采用双层协议架构：基于工业以太网的 Profinet 协议层和基于现场总线的 Profibus 协议层。由于 Profinet 协议和 Profibus 协议之间不能直接相互通信，采用专用的 Profibus 网关 TH-LINK 作为代理网关接口，使两者协议得以兼容，并且对用户透明。由于实验环境中采用 TH-LINK 网关自动实现 profinet 与 profibus 之间的转换，转换过程对测试人员来说是透明的，因此文章对处于 Profinet 层的 HMI 客户端和 Profibus 层 Siemens S7-1200 之间的点对点通信，将 Profinet 与 Profibus 二者的状态机视为相同。

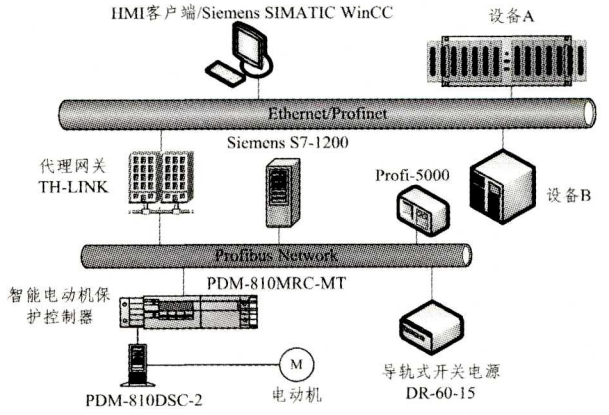

图 9　电力智能控制 SCADA 实验系统架构

HMI 客户端组态软件是 Siemens SIMATIC WinCC[28]。

运行环境:Intel(R) Core(TM) i3-3220 CPU@3.30GHz;4.00 GB RAM;Windows XP SP3。

4.2 实验结果分析

4.2.1 HMI 客户端测试

实验选择了 3 类报文类型:从站请求、从站响应和从站异常应答。对 HMI 客户端的测试实验分为 3 个部分:SCADA-Fuzz 消息级测试、SCADA-Fuzz 状态引导测试、Peach Fuzzer 测试。3 种测试均采用本地进程监控代理实时监测异常。表 1 列出了消息级测试的结果,表 2 列出了状态引导测试的结果,协议状态机如图 1 所示。

表 1 SCADA-Fuzz 消息级测试 WinCC 的结果

报文类型	用例	用时	异常数	已知异常类	未知异常类
从站请求	88916	—	907	23	12
从站响应	90874	—	912	21	11
异常应答	4952	—	194	4	3
统计	184742	142	2013	48	26

注:在线测试各类报文实时产生,无法统计同类报文的测试时间。

表 2 SCADA-Fuzz 状态引导测试 WinCC 的结果

状态	引导报文个数	用例数	用时	异常数	已知异常类	未知异常类
0	—	—	—	—	—	—
1	1	22436	10	624	22	9
2	2	41487	18	941	27	13
3	3	164356	83	2253	46	21
4	4	45715	24	201	17	6
5	5	54192	29	193	15	5
统计	—	328186	164	4302	127	54

由表 1 和表 2 可知,与消息级测试相比,SCADA-Fuzz 状态引导的测试在生成的测试用例数量上多了约 77.6%,测试用时也多了约 15.5%,这是由于需要对被测对象进行从初态开始的状态引导,其过程比较费时。但相比于消息级测试,状态引导测试触发的异常数多了 1 倍,已知异常类型数和未知异常类型数也分别多了大约 1.6 倍和 1 倍,结果明显要优于消息级测试。

表 1 和表 2 的对比结果纵向说明利用状态引导的测试虽然耗时增加,但测试结果较好;另一方面,为了将 SCADA-Fuzz 同其他模糊测试工具的效果相比,在相同的测试环境下,利用 Peach 进行了测试,测试结果如表 3 所列。

表 3 对 HMI 客户端 Peach 的测试结果

状态模型(StateModel)	测试用例数	用时/h	异常数	已知异常类	未知异常类
从站请求	92578	46	898	22	11
从站响应	94516	48	1011	19	9
从站异常应答	46714	26	323	8	6
统计	233808	120	2232	49	26

对比表 1 和表 3,SCADA-Fuzz 消息级测试的耗时较之于 Peach 多用时约 18.3%,这是由于采用在线方式需要实时对报文传输进行监听和捕捉,占用了大量的处理器资源。由于在消息级测试中,从站异常应答只能根据实时的异常主站请求报文而产生,因此在统计结果中,SCADA-Fuzz 针对从站异常应答的测试数量较少,最终得到的异常类也较少。但是针对从站请求和从站响应报文的测试,触发的异常类型数稍多于 Peach,最终消息级测试和 Peach 触发的异常类型数基本一致。

对比表 2 和表 3,虽然 Peach 对 HMI 客户端的测试从用例数量和测试时间上少于 SCADA-Fuzz 状态引导的测试,但由于 Peach 的状态模型缺乏充分的状态引导机制,同状态引导测试的效果相比,尽管 SCADA-Fuzz 耗时多,但统计到的异常数、已知异常类、未知异常类的数量要明显多于 Peach。

由于测试覆盖率难以测量,本文仅能从理论上利用状态机来提高测试的覆盖率。定义测试用例有效率 $Efec=\frac{Fn}{Tc}$,其中 Fn 表示触发的异常数,Tc 表示测试用例数量。表 4 对 3 种测试方法在测试用例有效率方面进行了对比。由表 4 可看出,SCADA-Fuzz 状态引导测试的有效率明显提高。

表 4 测试用例有效率对比

测试方法	测试用例数	触发的异常数	$Efec$/%
Peach Fuzzer	233808	2232	0.955
SCADA-Fuzz 消息级	184742	2013	1.090
SCADA-Fuzz 状态引导	328186	4302	1.311

通过对已知和未知的异常进行进一步验证分析和归类,SCADA-Fuzz 状态引导测试触发了对应的 6 个已知漏洞和 2 个未知漏洞的异常类。6 类已知异常信息验证了已公布的 WinCC 的 6 个已知漏洞,2 类未知异常信息对应着用户自定义程序的 2 个未知漏洞,如表 5 所列。

表 5 SCADA-Fuzz 状态引导测试发现的漏洞信息

漏洞编号	漏洞描述
CVE-2014-8552	WinCC 存在目录遍历漏洞,允许远程攻击者通过精心编制的数据包读取任意文件
CVE-2014-8551	WinCC 存在任意代码执行漏洞,允许远程攻击者通过精心编制的数据包执行任意代码
BUGTRAQ-58545	WinCC 存在目录遍历、缓冲区溢出安全漏洞,攻击者可利用其获取敏感信息、任意系统文件,在程序的应用上下文中执行任意代码
CVE-2011-4875	WinCC 的 2308/TCP 或 50523/TCP 端口没有正确处理数据段长度和 Unicode 字符串,可触发栈溢出,导致任意代码执行,可造成拒绝服务
CVE-2011-3321	WinCC 存在缓冲区溢出漏洞,远程攻击者可通过发送到端口 2308/TCP 的特制报文造成堆缓冲区溢出,控制受影响系统
CVE-2013-0675	WinCC 存在缓冲区溢出漏洞,可允许远程攻击者通过特制的报文造成拒绝服务和任意代码执行
未知漏洞 1	WinCC 可能存在目录遍历漏洞,允许远程攻击者通过精心编制的数据包读取任意文件
未知漏洞 2	WinCC 可能存在拒绝服务漏洞,攻击者可精心构造数据包导致程序崩溃和任意代码执行

SCADA-Fuzz 的消息级测试触发了对应的 6 个已知漏洞和 1 个未知漏洞的异常;而 Peach 的测试只触发了 6 个已知漏洞的异常。3 种测试方式触发的异常信息和对应的漏洞如表 6 所列。

由表 6 可知,Peach 的测试未发现两个未知漏洞,SCADA-Fuzz 消息级测试只发现了未知漏洞 1。经分析,由于 SCADA-Fuzz 消息级测试利用基于 MABNF 范式语法的变异策略将 Profinet 协议数据段的目录路径作为分隔符字段变异,同时对优先级字段注入"0x00FF"异常数据,二者的组合变异构造的模糊报文触发了未知漏洞 1 的非法读取内存异常,利用该漏洞可精心编制数据包,导致信息泄露,读取任意文件;而 SCADA-Fuzz 状态引导测试发现的未知漏洞 2,证明了其对目标协议状态的测试能力,由于其利用正常交互报文

引导序列 $i_1 i_2 i_3 i_4$，将被测进程引导至 Check-Freeze 状态 S_4 之后，对 Profinet 协议报头的应用标识符字段注入临界值 0xFF，同时对应用数据字段填充超长的“0xFF…”数据，二者的组合变异构造的模糊报文触发了堆溢出异常，导致程序崩溃，攻击者可利用其精心构造数据包，造成以应用进程上下文执行任意代码。

表 6　3 种测试方式的异常信息和对应漏洞对比情况

漏洞编号	异常信息	异常信息描述	Peach	SF 消息级	SF 状态
CVE-2014-8552	ReadAVonIP	非法读取结构指针	Y	Y	Y
CVE-2014-8551	WriteAV	非法写入内存	Y	Y	Y
BUGTRAQ-58545	ReadAV	非法读取内存	Y	Y	Y
CVE-2011-4875	StackCorruption	栈溢出	Y	Y	Y
CVE-2011-3321	HeapCorruption	堆溢出	Y	Y	Y
CVE-2013-0675	StackCorruption	栈溢出	Y	Y	Y
未知漏洞 1	ReadAV	非法读取内存	N	Y	Y
未知漏洞 2	HeapCorruption	堆溢出	N	N	Y

4.2.2　嵌入式设备的测试

实验以 Siemens S7-1200 为测试对象，选择了 3 类报文类型：主站请求、主站响应和主站异常应答。表 7－表 9 分别列出了对 Siemens S7-1200 利用 Peach Fuzzer 测试、SCADA-Fuzz 消息级测试和 SCADA-Fuzz 状态引导测试的结果。

表 7　对 Siemens S7-1200 设备 Peach 的测试结果

状态模型(StateModel)	测试用例数	用时/h	拒绝服务次数
主站请求 StateModel	94112	49	17
主站响应 StateModel	92108	46	15
主站异常应答 StateModel	44842	24	4
统计	231062	119	36

表 8　对 Siemens S7-1200 设备 SCADA-Fuzz 消息级的测试结果

报文类型	测试用例数	用时/h	拒绝服务次数
主站请求	91573	—	18
主站响应	89018	—	16
主站异常应答	3981	—	4
统计	224572	145	38

注：在线测试各类报文实时产生，无法统计同类报文的测试时间。

表 9　对 Siemens S7-1200 设备 SCADA-Fuzz 状态引导的测试结果

状态	引导报文个数	测试用例数	用时/h	拒绝服务次数
0	0	1232	1	2
1	1	24602	15	5
2	2	37512	22	9
3	3	124756	73	21
4	4	45325	27	6
5	5	52608	30	5
统计	—	286035	168	48

由表 7－表 9 的结果可知，Peach 测试中，Siemens S7-1200 出现 36 次拒绝服务；SCADA-Fuzz 消息级测试中，Siemens S7-1200 共出现 38 次拒绝服务；SCADA-Fuzz 状态引导测试中，Siemens S7-1200 共出现 48 次拒绝服务。当 Siemens S7-1200 设备出现拒绝服务时，无法回复心跳响应，导致整个电力 SCADA 系统无法正常工作，需要人工断电重启才能恢复正常。

实验对 SCADA-Fuzz 消息级测试的拒绝服务异常进行了多次重现验证，也正好出现 38 次拒绝服务的情况，与测试结果一致，这说明 SCADA-Fuzz 对于监测到的拒绝服务异常不存在误报。

对于 SCADA-Fuzz 状态引导的测试，实验对拒绝服务异常进行了进一步的分析。在 48 次拒绝服务的异常中，部分异常验证了 Siemens S7-1200 的 5 个拒绝服务的已知漏洞，漏洞编号及其描述如表 10 所列。

表 10　发现的漏洞编号及其描述

漏洞编号	漏洞描述
CVE-2014-2254	S7-1200 存在拒绝服务漏洞，攻击者可利用该漏洞使其进入 defect 模式，造成拒绝服务
CVE-2014-2256	S7-1200 存在拒绝服务漏洞，攻击者可通过发送特制的报文造成错误，利用该漏洞可使其进入 defect 模式，造成拒绝服务
CVE-2014-2258	S7-1200 存在拒绝服务漏洞，攻击者可通过发送特制的报文造成错误，利用该漏洞可使设备进入 defect 模式，造成拒绝服务
CVE-2013-2780	S7-1200 存在拒绝服务漏洞，攻击者可发送特制的报文造成错误，引起拒绝服务攻击
BUGTRAQ-57023	S7-1200 处理状态信息时存在错误，攻击者可发送特制的报文使设备进入 defect 模式，造成拒绝服务
未知漏洞 1	S7-1200 存在拒绝服务漏洞，攻击者可发送特制优先级标识字段和应用数据字段的报文造成错误，导致拒绝服务
未知漏洞 2	S7-1200 存在拒绝服务漏洞，攻击者可发送特制的状态信息字段和应用数据字段的报文造成错误，导致拒绝服务

表 11 对 SCADA-Fuzz 和 Peach 的漏洞发现情况进行了对比，由于 Peach 和消息级测试缺乏对 S7-1200 设备的状态的引导，导致未能发现未知漏洞 1 和未知漏洞 2；消息级测试未能发现 BUGTRAQ-57023，其是因为在线模式下，报文根据实时工作流产生，测试过程中，因为没有产生 Clear 清除报文，导致无法发现该漏洞对应的异常。

表 11　发现的漏洞情况对比

漏洞编号	危险等级	Peach	消息	状态
CVE-2014-2254	中	Y	Y	Y
CVE-2014-2256	中	Y	Y	Y
CVE-2014-2258	中	Y	Y	Y
CVE-2013-2780	高	Y	Y	Y
BUGTRAQ-57023	中	Y	N	Y
未知漏洞 1	中	N	N	Y
未知漏洞 2	中	N	N	Y

结束语　本文针对工控协议具有状态交互这一特点，提出了基于状态的工控协议 Fuzzing 测试方法，利用 XML 脚本对协议状态机模型进行描述，提出了基于状态机的测试序列生成算法 PSTSGM，对被测对象进行状态引导以求达到更高的覆盖率；提出基于心跳的异常监测与定位方法，解决了传统异常监测手段无法直接应用于工控嵌入式设备的问题；基于 Peach 实现了工控协议模糊测试原型系统 SCADA-Fuzz，对电力 SCADA 系统进行了测试。实验结果表明，利用状态引导的测试提高了测试的覆盖率和触发漏洞的概率，并发现了更多的安全漏洞；与 Peach 相比，SCADA-Fuzz 虽然耗时较大，但测试效果较好。下一步工作将考虑对相关模块进行优化，提升系统性能，缩短测试时间。

参 考 文 献

[1] 李鸿培,于旸,忽朝俭,等. 2013 工业控制系统及其安全性研究报告[EB/OL]. (2013-06-24) [2016-03-11]. http://www.nsfocus.com.cn/content/details_62_881.html.

[2] CLAYTON M. Stuxnet malware is weapon out to destroy Iran's Bushehr Nuclear Plant[R]. Christian Science Monitor, 2010.

[3] BENCSÁTH B, PÉK G, BUTTYÁN L, et al. Duqu: A Stuxnet-like malware found in the wild[R]. CrySyS Lab Technical Report, 2011.

[4] Wikipedia Flame[EB/OL]. [2016-03-11]. http://en.wikipedia.org/wiki/Flame_(malware).

[5] 李鸿培,王晓鹏,王洋. 2014 绿盟科技工业控制系统安全态势报告[EB/OL]. (2014-09-15) [2016-03-11]. http://www.nsfocus.com.cn/content/details_62_887.html.

[6] 乌克兰电网遭黑客攻击[EB/OL]. (2016-01-06)[2016-02-17]. http://www.ftchinese.com/story/001065610.

[7] TANKARD C. Advanced Persistent threats and how to monitor and deter them[J]. Network security, 2011, 2011(8): 16-19.

[8] 王清. 0day 安全:软件漏洞分析技术[M]. 北京:电子工业出版社,2008.

[9] TAKANEN A, DEMOTT J D, MILLER C. Fuzzing for software security testing and quality assurance[M]. Artech House, 2008.

[10] SHAPIRO R, BRATUS S, ROGERS E, et al. Identifying vulnerabilities in SCADA systems via fuzz-testing[M]// Critical Infrastructure Protection V. Springer Berlin Heidelberg, 2011: 57-72.

[11] DEVARAJAN G. Unraveling SCADA Protocols: Using Sulley-Fuzzer[C]// Defon Conference. 2007.

[12] HOU Y, HONG Z, PAN F, et al. Model Based Automatic Fuzzing Script Generation[J]. Computer Science, 2013, 40(3): 206-209. (in Chinese)
侯莹,洪征,潘璠,等. 基于模型的 Fuzzing 测试脚本自动化生成[J]. 计算机科学,2013,40(3):206-209.

[13] 匡恩网络. 工控网络安全漏洞挖掘检测平台[EB/OL]. [2016-03-11]. http://www.kuangn.com/product/vulnerability.

[14] LI H. From "0" to "1": China's Industrial control protocol communictaion robustness test platform based on our independent research and development[J]. Automation Exhibition, 2015(9): 70-73. (in China)
李航. 从"0"到"1":我国自主研发工控协议通讯健壮性测试平台[J]. 自动化博览,2015(9):70-73.

[15] DEVARAJAN G. Unraveling SCADA protocols: Using Sulley-fuzzer, presented at the DefCon 15 Hacking Conference[EB/OL]. [2016-03-11]. http://www.defcon.org/html/defcon-15/dc15-speakers.html.

[16] KOCH R. Profuzz. [EB/OL]. [2016-03-11]. https://github.com/HSASec/ProFuzz.

[17] BYRES E J, KUBE N, DAN H. On Shaky Ground-A Study of Security Vulnerabilities in Control Protocols[C]// 5th American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Controls, and Human Machine Interface Technology, American Nuclear Society. 2006.

[18] General Purpose Fuzzer[EB/OL]. [2016-03-11]. http://www.vdalabs.com/tools/efs_gpf.html.

[19] BRATUS S, HANSEN A, SHUBINA A. LZFuzz: a fast compression-based fuzzer for poorly documented protocols[R]. Dartmouth Computer Science Technical Report TR 2008-634, 2008.

[20] DigitalBond. ICCPSic assessment tool set released, sunrise[EB/OL]. [2016-03-11]. http://www.digitalbond.com/2007/08/28/iccpsic-assessment-toolset-released.

[21] DYNAMICS M. Mu test suite[EB/OL]. [2016-03-11]. http://http://www.mudynamics.com/products/mu-test-suite.html.

[22] Wurldtech. Achilles test platform[EB/OL]. [2016-03-11]. http://www.wurldtech.com.

[23] Codenomicon. Defensics test platform[EB/OL]. [2016-03-11]. http://www.codenomicon.com/products /protocols.shtml.

[24] BeSTORM Software Testing Framework[EB/OL]. [2016-03-11]. www.beyondsecurity.com/black-box-testing.html.

[25] Peach[EB/OL]. [2016-03-11]. http://www.peachFuzzer.com.

[26] BOSSERT G, GUIHÉRY F, HIET G. Netzob: un outil pour la rétro-conception de protocoles de communication[C]// Actes Du Symposium Sur La Sécurité Des Technologies De Linformation Et Des Communications. 2012.

[27] ZHANG Y F, HONG Z, WU L F, et al. From-syntax based fuzzing method for industrial control protocols[J]. Application Research of Computers, 2016, 33(8): 2433-2439. (in Chinese)
张亚丰,洪征,吴礼发,等. 基于范式语法的工控协议 Fuzzing 测试技术[J]. 计算机应用研究,2016,33(8):2433-2439.

[28] Siemens. SIMATIC WinCC[EB/OL]. [2016-03-11]. http://www.wincc.com.cn.

(上接第 119 页)

[9] KUANG L, ZULEMINE M. An anomaly intrusion detection method using the csi-knn algorithm[C]// Proceedings of the 2008 ACM Symposium on Applied Computing. ACM, 2008: 921-926.

[10] HU M X. Intrusion detection algorithm based on BP neural network[J]. Chinese Journal of Computers, 2012, 38(6): 148-150. (in Chinese)
胡明霞. 基于 BP 神经网络的入侵检测算法[J]. 计算机工程,2012,38(6):148-150.

[11] HUANG M M, LIN B G. Fuzzy clustering method based on genetic algorithm in intrusion detection study[J]. Journal on Communications, 2009, 30(11): 140-145. (in Chinese)
黄敏明,林柏钢. 基于遗传算法的模糊聚类入侵检测研究[J]. 通信学报,2009,30(11):140-145.

[12] ZHANG L, BAI Z Y, LUO S S, et al. Integrated intrusion detection model based on rough set and artificial immune[J]. Journal on Communications, 2013(9): 166-176. (in Chinese)
张玲,白中英,罗守山,等 基于粗糙集和人工免疫的集成入侵检测模型[J]. 通信学报,2013(9):166-176.

[13] TSAI C F, CHENG K C. Simple instance selection for bankruptcy prediction[J]. Knowledge-Based Systems, 2012, 27(3): 333-342.