面向模糊测试的工业控制协议逆向方法研究

刘俐媛，霍朝宾，贺敏超
（华北计算机系统工程研究所，北京　100083）

摘　要：模糊测试是传统的漏洞挖掘方法之一，在工业控制领域，针对一些不公开协议规范的私有协议，由于无法获取协议格式及语义信息，模糊测试在对其脆弱性分析上存在用例冗余度高、测试效率低等问题。通过将协议逆向与模糊测试相结合，提出了应用动态污点分析的协议逆向方法，从函数和指令级分析工业控制协议报文在程序运行中的轨迹信息，依据轨迹日志构造协议语法树，生成语义信息，指导测试用例生成，并使用 Peach 模糊测试框架进行测试，证明了该方法能有效减少测试冗余，提高测试效率。

关键词：工业控制；协议逆向；模糊测试；动态污点分析
中图分类号：TP309　　**文献标识码**：A　　**DOI**：10. 19358/j. issn. 2097 - 1788. 2023. 03. 002
引用格式：刘俐媛，霍朝宾，贺敏超. 面向模糊测试的工业控制协议逆向方法研究[J]. 网络安全与数据治理，2023，42(3)：8 - 12.

Research on reverse method of industrial control protocol for fuzz testing

Liu Liyuan，Huo Chaobin，He Minchao
(National Computer System Engineering Research Institute of China，Beijing 100083，China)

Abstract：Fuzz testing is one of the traditional vulnerability mining methods. In the field of industrial control，for some private protocols that do not disclose protocol specifications，fuzz testing has problems of high redundancy of use cases and low testing efficiency in vulnerability analysis due to the inability to obtain protocol format and semantic information. By combining protocol inversion with fuzz testing，this paper proposes a protocol inversion method based on dynamic stain analysis. It analyzes the track information of industrial control protocol message in program operation from function and instruction level，constructs protocol syntax tree based on track log，generates semantic information，and guides test case generation. The Peach fuzz test framework is used to test，which proves that this method can effectively reduce test redundancy and improve test efficiency.

Key words：industrial control；protocol reverse；fuzz testing；dynamic taint analysis

0　引言

在工业互联网背景下，工业控制系统在更加开放和智能的同时也面临着更大的安全挑战。其中工业控制协议作为工业控制系统通信的语言，在工业控制系统的安全研究上是至关重要的一部分。工业控制系统本身与互联网不同，系统本身存在很多未公开的私有协议，而这些协议没有经过公开的测试，协议本身可能会存在某些漏洞或缺陷，通过协议格式分析进行防御的手段无法对未知协议奏效。所以针对协议规范未知的自定义协议，通过协议逆向方法进行分析变得非常重要，为后续模糊测试等漏洞挖掘方法提供基础技术支撑。

协议逆向是对网络中存在的未公开的协议通过对其在网络中的输入输出、系统行为和程序执行流程进行监控和分析，提取协议规范描述信息的过程[1]。在此基础上，模糊测试依据逆向分析出的协议格式及状态构建测试用例，能够分析发现系统中的漏洞并及时修补，对于应对工业控制系统在实际运行中遇到的风险有重大意义。本文通过协议逆向的方法提取出工业控制协议的格式信息和语义信息，指导模糊测试生成测试用例进行漏洞挖掘。

协议逆向主要分为两个研究方向：基于报文的协议逆向和基于指令执行轨迹的协议逆向[2]。两种方法的分析对象有所不同，第一种以网络流量作为研究对象，通过网络抓包工具获取大量报文作为预处理的样本数据，研究各字段值的变化和特征，从而提取协议格式信息；第二种则是需要分析协议程序的报文解析过程，通过程

序处理域边界的方式获得协议字段信息。

由于大部分工业控制协议属于二进制数据流传输信息，不像文本数据具有显著协议特征，因此基于报文的方法很难提取出较为准确的二进制协议格式。而基于指令执行轨迹的方法则是通过程序内部的数据调用过程进行分析，得到的格式信息更为准确，此方法主要利用动态污点分析技术[3]实现报文解析。对于模糊测试技术的发展，从 Sulley[4]、Peach[5]等模糊测试工具的研发使用到对模糊测试用例生成方法[6]的改进研究，一直在致力于测试效率的提高，目前已经取得了一定成就。

针对以上研究，本文将协议逆向和模糊测试相结合，使用基于动态污点分析的方法，分析程序中的指令执行轨迹，得到协议格式并指导测试用例生成，通过 Peach 模糊测试框架进行模糊测试并对比，证明该方法能有效提高模糊测试的漏洞挖掘效率。

1　面向模糊测试的工业控制协议逆向方法

模糊测试在当前的工业控制环境下更多被用于对已知协议进行安全漏洞检测。而协议逆向则针对未知或私有协议，根据逆向结果构造测试用例来分析协议中的安全隐患。将两者结合，能提高对工业控制私有协议的漏洞挖掘效率，同时协议逆向结果也可以应用于工业防火墙中对工业控制协议的识别与检测以及指令级控制等。

本文主要分为两个模块：协议逆向模块和模糊测试模块。协议逆向模块使用动态污点分析方法，通过对工业控制协议的执行程序进行插桩，并记录污点数据在程序中的执行轨迹，分析日志得到协议格式；模糊测试模块则是将格式信息作为构建 pit 文件的数据模型生成测试用例。总体框架如图 1 所示。

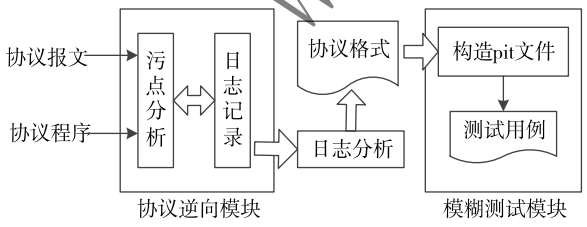

图 1　总体框架图

1.1　动态污点分析

本文利用 Intel 公司的 Pin[7]动态污点插桩工具，对报文在协议可执行程序中的执行轨迹进行污点标记，从函数和指令两个层面记录污点数据的传播过程，分析各字节数据在程序函数和系统寄存器或内存中的传播和使用方式，得出协议格式和语义信息。

通过对工业控制协议特点的分析，协议消息由字段块序列组成，其中字段块是具有某种意义的最小连续字节数据序列。例如 Modbus TCP 协议的报文格式如图 2 所示。

事务处理标识符	协议标识符	长度	单元标识符	功能码	数据
2字节	2字节	2字节	1字节	1字节	*N*字节

图 2　Modbus TCP 报文格式

在协议程序接收到报文后，调用插桩接口对网络函数插桩，将函数中的协议数据所在的内存区域进行标记，以便后续对污点的追踪。

1.2　污点传播

针对协议的污点传播需要考虑程序处理过程中哪些指令会影响污点传播的准确性，同时提高程序的运行效率[8]。由于本实验在 Linux x86 环境下运行，且协议在程序读取阶段不涉及浮点数运算，对整数的运算更多集中在 x86 指令集中，因此本文只考虑 x86 整数指令。

x86 指令集包含许多复杂指令，但如果分析全部指令，工程量过大且有些指令对于污点传播没有直接影响。例如，控制类指令是对 CPU 功能进行的控制，对于分析协议格式没有太大帮助，所以主要从运算类、传送类、转移类指令考虑污点的传播规则。传播规则如下：

（1）源操作数→目的操作数：

if（istainted（left_ op））then taint（right_ op）

指令例如：mov，moves，movesz 等。

（2）源和目的操作数→目的操作数：

if（istainted（left_ op）｜ istainted（right_ op））then taint（right _ op）

指令例如：add，sub（算数运算），and，or，xor（逻辑运算），shr，shl（位运算）等。

（3）单操作数：

pushebp：taint（mem）← taint（reg）

push qwordptr［ebp - 0x4］：taint（mem）← taint（mem）

not：removetaint（reg）

（4）结果清零：

当源操作数和目的操作数是同一寄存器，如指令 xor edi，edi，sub esi，esi 时，结果为零，需要清除污点信息；指令 and eax，0x0 结果也为零时同样需要清除污点。

而当存在一段连续的污点内存时，系统中的内存拷贝函数可以对多字节实现一次性拷贝，从函数级进行污点传播更为高效。例如，在监控到 memcpy 函数对内存进

行拷贝时，memcpy_ point 函数判断是否有污点数据，有的话则对拷贝的地址标记污点。

1.3 污点过程记录

主要记录的污点过程包括函数调用信息和处理污点数据的相关指令，并以数组方式进行保存。通过在函数插桩时获得返回地址，记录函数执行前和执行后的进入退出信息，获得完整的调用链，最终形成日志文件，以便后续分析语法及语义信息。

1.4 字段边界分析

日志记录了污点传播的整个函数和指令的调用过程，通过函数之间的调用关系和执行顺序，构建语法层次树，表示协议的各个字段和函数之间的层次关系，具体算法如下：

算法 1：语法层次树生成

```
输入：日志数组 logs
输出：语法树
1 root = FunctionNode ();
2 node = root;
3data_ node = DataNode ();
4func_ node = root;
5 for each log ∈ logs do
6    tag = log. tag;
7    if tag = = FUNCTION then
8        iflog. state = = access then
9            add node← data_ node;
10data_ node = DataNode ();
11func_ node = FunctionNode (log. name);
12           addnode. children ← func_ node;
13           node = func_ node;
14       else iflog. state = = exit then
15           add node← data_ node;
16func_ node = func_ node. parent;
17           node = func_ node;
18   else if tag = = INSTRUCTION then
19       adddata_ node ← log. offsets;
20 return root;
```

其中，以函数节点作为根节点，将含有数据的报文节点 data_ node 作为子节点插入，遍历日志记录，当某条记录项类型属于函数时：（1）如果函数状态是进入，当存在被污染的报文字节时，将该部分的 data_ node 插入当前节点的子节点中，并在当前节点新建子节点作为当前节点。（2）如果函数是退出状态时，将节点返回至上一个父节点。如果某条日志属于指令时，将该指令处理的报文字节插入 data_ node。最终形成一个完整的语法层次树。

1.5 语义分析

大部分工业控制协议的使用场景和主要功能都有相似之处，通过对常见协议进行分析可以将协议分为固定字段、地址字段、长度字段、序号字段、功能标识字段、数据字段、校验字段等。例如，协议标识符、发送或接收地址、序号字段等可以通过大量报文比对识别，而对于长度字段、功能标识字段、数据字段、校验字段则需要通过其在协议中的执行过程来识别。

在动态污点分析的基础上，语义分析可以借助于污点上下文，通过分析跳转和比较指令，并结合不同字段在汇编指令层的执行逻辑，将污点中符合该字段执行逻辑的报文字节识别并标注语义。

功能标识字段一般是决定报文类型或报文存储区域的固定值，在汇编指令中有多个 cmp 指令与数值的比较，可以判断为功能标识字段；如果只与一个值比较则可以判定为固定字段。

长度字段一般与报文的长度有关，在接收函数中，专门有一个参数用来存储长度信息，可以对该参数进行识别。

在数据字段中一般包含寄存器起始地址字段、数量字段等，数量字段能够传递后续数据块的数量信息，一般以循环形式进行条件判断，通过识别循环可以判断该字段。循环检测[9]算法通过对程序运行中 cmp 指令所在的循环位置进行检测，具体算法如下：

算法 2：循环检测算法

```
输入：日志中基本块数组 bbls，函数数组 funcs
输出：所有循环中的基本块位置记录
1 loops = dict ();
2trace, result = list ();
3 for eachbbl ∈ bbls , each func ∈ funcs do
4while trace not empty and trace [ - 1]. rsp <
func. rsp do
5    remove loops [traces [ - 1]];
6    remove trace [ - 1];
7if func, id ∉ loops then
8    loops [func. id] ← list (bbl);
9    add trace ← func;
10   if Match (bbl, loops) then
11    add result ← bbl;
12add loops [func. id] ← bbl;
13 return result;
```

通过观察协议源码，发现针对数量字段的判断使用的是 for 循环语句。基本执行流程如图 3 所示，A、B、C、D 符合汇编程序中基本块的执行顺序，当循环变量为 2 时，基本块 B 和 C 会重复执行 2 次，执行顺序为 B、C、B、C、B 。Match（）函数通过判断当前基本块地址与循环中存入的基本块 B 或 C 的地址是否相同输出循环体位置信息。

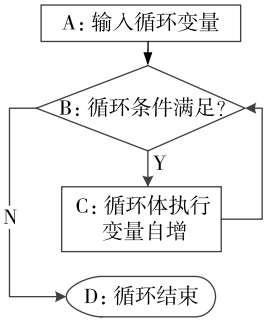

图 3　for 循环流程图

2　实验结果与分析

2.1　实验设计

本文实验设计主要分为两部分，第一部分是对协议报文进行逆向分析，实验选取 Modbus TCP 协议中的 libmodbus 和 freemodbus 两个协议实现，使用动态污点分析方法获得工业控制协议的格式信息。第二部分是直接使用 Peach 框架对 Modbus TCP 进行模糊测试，大多数测试用例会被拒绝；而使用本文方法，将逆向结果根据不同字段进行变异并构造 pit 文件进行模糊测试，构造的测试用例更高效。

实验从报文样本中选取几条报文，将选取的报文发送到 Modbus 的协议程序中，启动 pin 插桩程序，跟踪程序处理协议流量的轨迹，生成日志记录。后续对日志记录进行分析判别协议的字段边界和语义信息。本文方法的实现过程如图 4 所示。

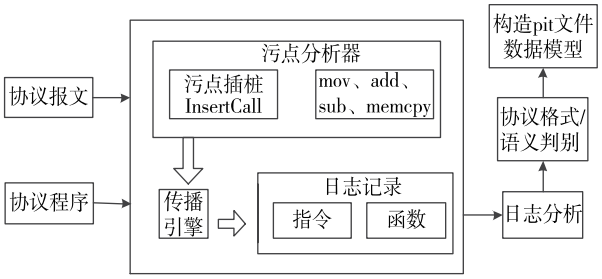

图 4　本文方法过程图

通过对日志进行分析得到协议语法层次树，图 5 为 libmodbus 生成的语法层次树，从图中可以看到［0，1］、［8，9］、［10，11］是长度为 2 字节的字段，［6］、［7］字节长度为 1，其中数字表示所接收消息的位置偏移量。结合语义识别，表 1 中对 libmodbus 和 freemodbus 协议的逆向分析结果与 Wireshark 的结果进行了比较，“√”属于正确识别的字段。从表 1 中可以看出，libmodbus 中对功能码字段和数量字段进行了正确识别，但未识别出长度字段；而 freemodbus 识别效果较好。

在模糊测试部分，首先直接使用 Peach 模糊测试框架对 Modbus TCP 进行模糊测试，由于本文主要研究对工业控制私有协议的模糊测试方法，因此将报文作为未知协议进行测试，不对数据流中具体字段做变异策略处理，直接对报文设定随机变异生成测试用例。

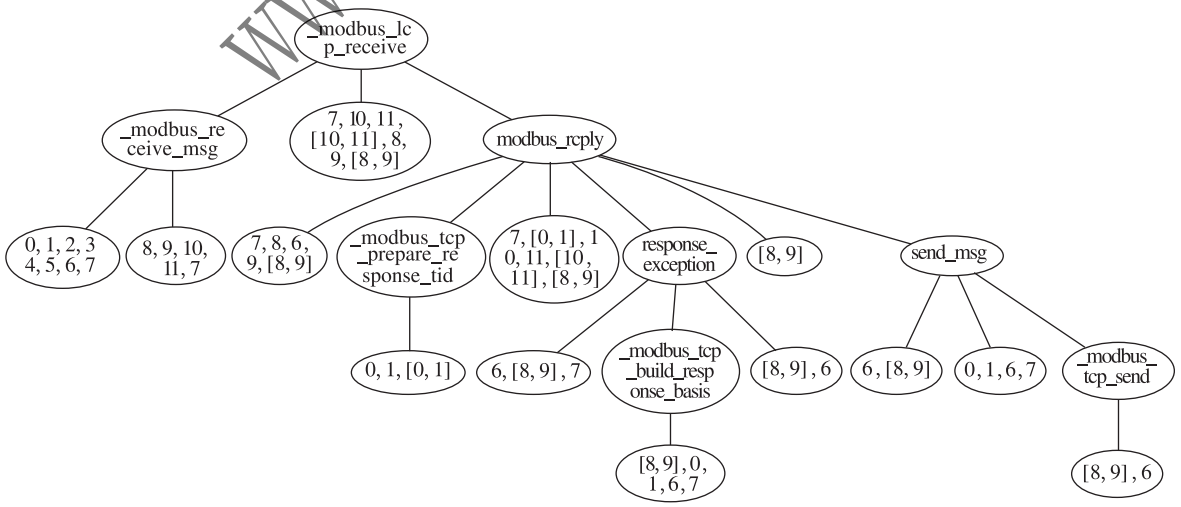

图 5　libmodbus 语法层次树

表 1　字段标识比对表

字段名称	Wireshark		libmodbus	freemodbus	
	字段大小	字段大小	语义匹配	字段大小	语义匹配
事务处理标识符	2	√	—	√	—
协议标识符	2			√	√
长度	2			√	√
单元标识符	1	√	√	√	√
功能码	1	√	√	√	√
起始地址	2	√		√	
数量字段	2	√	√	√	√

使用本文方法，在得到协议语法格式和语义信息后，根据不同字段设定变异策略，通过 Publisher 模块将测试报文发送给被测目标，监控目标程序运行情况。

2.2　结果分析

本文对测试目标协议设定测试执行时间为 2 h，通过 Wireshark 进行旁路监听，捕获测试期间报文并记录导致异常的用例和异常响应信息，分析测试用例的有效性。其中测试效率表达式为：

$$\text{测试效率} = \frac{\text{响应数量}}{\text{请求数量}} \tag{1}$$

测试结果对照见表 2 所示。

表 2　测试结果对照表

测试方法	请求数量	正常响应数量	异常响应数量	执行时间	测试效率
本文方法	5 652	2 592	2 760	2 h	94.7%
Peach	9 204	2 736	4 468	2 h	78.3%

实验结果显示直接使用 Peach 模糊测试框架生成的测试用例数较多，且大部分测试用例是无效的，被测目标并没有对其产生响应；而结合协议逆向并依据其协议语法有针对性的对不同字段进行变异，能够生成更有效的测试用例，避免了大多数用例直接被系统拒绝而造成测试效率低下。通过对比两种方法的异常响应数量及测试效率，本文方法提高了测试用例质量，降低了执行测试的开销，提升了测试效率。

3　结论

本文针对目前工业控制私有协议模糊测试过程中存在测试用例冗余度高，测试效率低等问题，提出了在对工业控制协议进行协议逆向的基础上生成测试用例的方案，通过动态污点分析方法对协议执行程序进行污点标记和跟踪，分析协议执行过程，利用语法树对语法格式进行划分得到字段边界信息，同时分析程序执行中特定的指令生成语义信息，最后利用得到的协议格式和语义设定针对各字段的变异策略并生成测试用例。实验结果表明，与直接进行模糊测试相比，该方法能有效减少冗余的测试用例，提高测试效率。

参考文献

[1] 潘璠，吴礼发，杜有翔，等. 协议逆向工程研究进展 [J]. 计算机应用研究，2011，28(8)：2801 - 2806.

[2] CABALLERO J，SONG D. Automatic protocol reverse - engineering：message format extraction and field semantics inference [J]. Computer Networks，2013，57(2)：451 - 474.

[3] CABALLERO J，YIN H，LIANG Z，et al. Polyglot：automatic extraction of protocol message format using dynamic binary analysis [C] //Proceedings of the 14th ACM Conference on Computer and Communications Security，2007：317 - 329.

[4] Sulley [EB/OL]. (2018 - 03). http：//www. fuzzing. org/wp - content/SulleyEpyDoc/.

[5] Peach [EB/OL]. (2015 - 07 - 21). http：//www. peachFuzzer. com.

[6] 尚文利，李文轩，陈春雨，等. 基于特征矩阵的工业控制协议模糊测试方法 [J]. 信息与控制，2020，49(4)：436 - 443，454.

[7] LUK C K，COHN R S，MUTH R，et al. Pin：building customized program analysis tools with dynamic instrumentation [C] //Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'05)：190 - 200.

[8] 代伟，刘智，刘益和. 基于二进制代码的动态污点分析 [J]. 计算机应用研究，2014，31 (8)：2497 - 2501，2505.

[9] MOSELEY T，GRUNWALD D，CONNORS D A，et al. Loopprof：dynamic techniques for loop detection and profiling [C] // Proceedings of the 2006 Workshop on Binary Instrumentation and Applications (WBIA)，2006.

（收稿日期：2023 - 02 - 17）

作者简介：

刘俐媛（1997 - ），女，硕士研究生，主要研究方向：工业控制系统信息安全技术。

霍朝宾（1983 - ），男，硕士，高级工程师，主要研究方向：工业控制系统信息安全技术。

贺敏超（1989 - ），男，硕士，工程师，主要研究方向：工业控制系统信息安全技术。

版权声明

凡《网络安全与数据治理》录用的文章，如作者没有关于汇编权、翻译权、印刷权及电子版的复制权、信息网络传播权与发行权等版权的特殊声明，即视作该文章署名作者同意将该文章的汇编权、翻译权、印刷权及电子版的复制权、信息网络传播权与发行权授予本刊，本刊有权授权本刊合作数据库、合作媒体等合作伙伴使用。同时，本刊支付的稿酬已包含上述使用的费用，特此声明。

《网络安全与数据治理》编辑部