基于混合马尔科夫树模型的 ICS 异常检测算法

张仁斌 [1,2] 吴 佩 [1] 陆 阳 [1] 郭忠义 [1]

摘 要 针对工业控制系统中现有异常检测算法在语义攻击检测方面存在的不足, 提出一种基于混合马尔科夫树模型的异常检测算法, 充分利用工业控制系统的阶段性和周期性特征, 构建系统正常运行时的行为模型 — 混合马尔科夫树. 该模型包含合法的状态事件、合法的状态转移、正常的概率分布以及正常的转移时间间隔等 4 种信息, 基于动态自适应的方法增强状态事件的关联度并引入时间间隔信息以实现对复杂语义攻击的检测, 语义建模时设计一种剪枝策略以去除模型中的低频事件、低转移事件以及冗余节点, 当被检测行为使得模型的以上 4 种信息产生的偏差超过阈值时, 判定该行为异常. 最后, 基于 OMNeT++ 网络仿真环境构建一个简化的污水处理系统对本文算法进行功能性验证, 并利用真实物理测试床的数据集对算法的检测准确度进行性能验证. 验证结果表明, 本文算法能有效消除人机交互和常规诊断等操作带来的噪声影响, 对复杂语义攻击具有较高的检出率, 且能识别传统的非语义攻击.

关键词 工业控制系统, 语义攻击, 异常检测, 混合马尔科夫树模型

引用格式 张仁斌, 吴佩, 陆阳, 郭忠义. 基于混合马尔科夫树模型的 ICS 异常检测算法. 自动化学报, 2020, **46**(1): 127−141

DOI 10.16383/j.aas.2018.c170493

Anomaly Detection Algorithm in ICS Based on Mixed-Order Markov Tree Model

ZHANG Ren-Bin[1, 2] WU Pei[1] LU Yang[1] GUO Zhong-Yi[1]

Abstract Aiming at the problem that the existing anomaly detection algorithms in industrial control systems are insufficient in semantic attacks, an anomaly detection algorithm based on hybrid Markov tree model is proposed. The periodic characteristics of industrial control system is utilized to construct the system normal behavioral model at runtime — mixed-order Markov tree. The model includes four kinds of information: legal state events, legal state transition, normal probability distribution and normal transition time interval. Based on the dynamic adaptive method, the relation between state events are enhanced and the time interval information is added to realize the detection for more complex semantic attacks. When modeling, a pruning strategy is designed to remove low frequency events, low transition events and redundant nodes in the model. When the detected behavior makes the deviation of the above four kinds of information of the model exceed the threshold, the system detects the anomaly. Finally, a simplified sewage treatment system based on OMNeT++ network simulation environment is constructed to verify the function of the proposed algorithm, and a data set of a real physical testbed is used to verify the detection accuracy of the algorithm. Experimental results show that the proposed algorithm can effectively eliminate the noise of human interaction and normal diagnosis, and has high detection rate for complex semantic attacks and can also identify traditional non-semantic attacks.

Key words Industrial control system (ICS), semantic attacks, anomaly detection, mix-order Markov tree model

Citation Zhang Ren-Bin, Wu Pei, Lu Yang, Guo Zhong-Yi. Anomaly detection algorithm in ICS based on mixed-order Markov tree model. *Acta Automatica Sinica*, 2020, **46**(1): 127−141

工业控制系统 (Industrial control system, ICS) 主要用于工业生产过程中的各种监控被广泛应用于能源、电力、化工、污水处理、石油天然气等大型国家基础设施行业. 近年来, 由于远程管理控制的需求不断扩大, 之前相对封闭独立的控制系统已经变得更加开放化和互联化. 这一方面提升了 ICS 远程管理的能力, 给操控者带来了极大的便利, 另一方面却也将其自身暴露在各种网络攻击的威胁之下.

ICS 面临的安全威胁中, 除了传统的攻击, 如 DoS (Denial of service) 攻击、缓冲区溢出攻击、信息收集等攻击之外[1], 一种依赖于对控制过程以及物理设备了解的详细程度而采取的攻击也在尝试最大化地破坏工业基础设施, 称为 "语义攻击"[2]. 语义攻击建立在对协议、软件、硬件以及生产控制过程深入理解的基础上, 攻击者通过精心构造一组看

收稿日期 2017-09-01 录用日期 2018-01-29
Manuscript received September 1, 2017; accepted January 29, 2018
国家重点研发计划专项 (2016YFC0801804, 2016YFC0801405), 中央高校基本科研业务费专项资金资助 (PA2019GDPK0074) 资助
Supported by National Key Research and Development Projects (2016YFC0801804, 2016YFC0801405), the Fundamental Research Funds for the Central Universities of China (PA2019GDPK0074)
本文责任编委 袁勇
Recommended by Associate Editor YUAN Yong
1. 合肥工业大学计算机与信息学院 合肥 230601 2. 工业安全与应急技术安徽省重点实验室 合肥 230601
1. College of Computer and Information, Hefei University of Technology, Hefei 230601 2. Anhui Province Key Laboratory of Industry Safety and Emergency Technology, Hefei 230601

似 “合法” 的消息序列来破坏正常的工业生产. 这些消息序列单纯从协议层的角度来看是合法的, 然而从系统当时所处的上下文语境与状态来看, 却违反了工业生产过程中的一些语义限制. 2010 年发生的 “震网” 蠕虫[3] 就是一个典型的语义攻击, 它改写并隐藏了可编程逻辑控制器 (Programmable logic controller, PLC) 的代码, 通过构造恶意但技术上合法的指令来改变离心机的速度, 对伊朗的纳坦兹核设施控制系统造成了极大的破坏.

现有语义攻击可分为两种类型[4]: Order-based 语义攻击和 Time-based 语义攻击. Order-based 语义攻击指攻击者以不正常的顺序发送消息指令[5]; Time-based 语义攻击指攻击者以不正常的频率发送消息指令[6]. 传统的网络入侵检测方法针对这种基于消息序列的语义攻击显得无能为力. 因此需要结合 ICS 的一些语义特征, 设计一种能够识别语义攻击的异常检测系统.

针对语义攻击问题, 有学者提出以数据为中心的异常检测研究方法, 利用发生攻击之后网络数据以及其他过程变量的变化特征来进行异常检测. Hadziosmanovc 等[2] 直接对 ICS 过程变量进行分析, 从设备的网络通信中提取变量值, 使用自回归模型的方法对数据进行监督和限制, 当数据值使模型发生异常偏离或者超过了变量的取值范围时, 入侵检测系统发出警告. 该方法能在一定程度上解决语义攻击的检测问题, 但也面临着一系列问题, 如建模时对浮点类型数据的分类错误导致系统出现较高的漏报, 建模时需要无污染的数据或需对数据进行预处理, 否则, 模型参数受少数异常点影响较大[7]. 文献 [8−10] 中利用数据挖掘中聚类或者分类的相关算法进行异常数据的识别. 叶石罡[8] 在 AVC 系统中采用 K-Means 聚类算法提取电压的特征曲线, 该方法不仅受限于 K-Means 算法自身的缺陷并且未能很好地考虑 AVC 系统中其他参数变量的特征规律. Jiang 等[9] 采用支持向量机算法对数据采集与监控 (Supervisory control and data acquisition, SCADA) 系统中的异常进行检测, 但在训练支持向量机模型时, 未考虑到观测变量中相关数据间的关联性. Zhao 等[10] 采用基于 K-Means 聚类的自适应模糊 C 均值法检测能源系统异常数据, 虽然该算法考虑了相关数据间的关联性, 但是却没有考虑数据的时序特征以及训练样本中出现噪声数据的可能性. 以数据为研究目标的异常检测方法对系统的先验性知识要求低, 可以通过自学习的方法从大量数据中寻找正常行为的规律. 然而, 这些基于数据建模的异常检测方法还存在以下问题: 1) 不能很好地处理混合属性数据的相似度计算问题; 2) 异常检测结果易受参数 (如聚类半径) 影响, 而当前对于参数的选择仍然没有简单有效的方法[11]; 3) 仅考虑到数据空间上的相关性, 而未考虑到数据在时间序列上的关联性[12].

ICS 系统通常包括 PLC、现场设备 (如传感器、执行器), 以及上层管理接口 (Human machine interface, HMI). HMI 定期对 PLC 进行数据采集, 以监控底层的生产过程, 当出现异常时可发送控制指令对生产过程进行干预. PLC 通过顺序执行梯形逻辑指令控制传感器的数据采集和执行器的执行, 通过重复一系列周期性操作完成每轮工业生产过程. 无论是 HMI-PLC 层还是 PLC-现场设备层, 主从设备的交互通常都具有明显的阶段性特征[13], 这种高度的阶段性和周期性特征, 使我们可以提取系统正常运行时的行为特征, 对其分析建模, 从而利用正常行为模型构建基于异常的入侵检测系统.

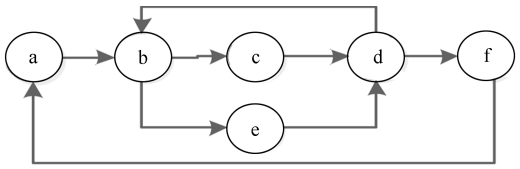
许多国内外研究学者利用 ICS 的阶段性和周期性特征, 提出了一系列针对语义攻击的检测方法. Goldenberg 等[14−15] 基于工业控制协议 Modbus[16], 采用确定有限自动机 (Deterministic finite automata, DFA) 的方法对 Modbus/TCP 网络数据包进行建模, 基于 Modbus 交互数据包中功能码具有高度阶段性特征的假设, 为每个 HMI-PLC 交互通道构建 DFA 模型. 但是在实际控制过程中, Modbus 通信流只有在过滤掉一些人为操作和随机产生的数据包丢失、重传、延迟等噪音之后, 系统才会呈现出阶段性特征, 因此当该方法应用于实际的 Modbus 通信样本时, 会产生大量的 DFA 状态节点, 使得 DFA 模型的规模相当庞杂, 此外, 该方法受限于 DFA 算法本身的局限性, 只考虑到状态转移的合法性, 而未考虑到转移概率在进行异常检测时的重要作用, 从而无法检测出 Time-based 语义攻击所带来的概率分布异常. Adepu 等[17] 基于混合自动机实现状态约束, 考虑执行器和传感器的关联关系, 即执行器的状态转移受传感器约束, 传感器的状态转移受执行器约束, 该方法能较好地解决语义攻击的检测问题, 但是该方法需要结合具体的 ICS 过程文档, 且需要将约束条件硬编码在 PLC 控制逻辑中, 在 PLC 运行过程中不断检查约束条件是否成立, 因此该方法具有很强的局限性, 缺乏灵活性. 李晓航等[18] 针对状态转移概率不确定的延迟马尔科夫跳变系统, 设计了自适应观测器来同时估计执行器和传感器故障, 实现对其异常状态和故障的检测, 该方法相对于基于精确状态转移概矩阵的故障估计方法更具有通用性, 并可以同时估计状态、执行器和传感器故障, 但是该方法主要适用于故障检测领域, 当应用于异常检测领域时, 由于未考虑到 ICS 各组件之间的交互性特征, 因此并不适用于对 ICS 整个系统进行建模以实现入侵检测. Caselli 等[4, 19−20]

利用离散时间马尔科夫链 (Discrete time Markov chains, DTMC) 模型对 SCADA 消息序列进行建模. 该模型由一阶马尔科夫链构成, 模型中的每个节点代表了合法的状态事件, 节点之间的转移关系代表了合法的状态转移, 每个状态节点维护了一张转移概率表, 表明了转移的分布情况. 此模型通过检测未知状态异常、未知转移异常以及概率分布异常, 能够检测出简单的语义攻击. 然而由于该模型基于一阶马尔科夫链, 假设当前状态只与前一个历史状态存在关联性, 即事件相关度仅为 1, 这种低事件相关度模型在面对复杂语义攻击时存在明显的不足. 针对一阶马尔科夫模型的低相关性问题, 有学者提出变阶马尔科夫模型. Begleiter 等[21] 分析并比较了多种变阶马尔科夫模型在分子生物学、文本以及音频领域中的应用情况, 但是这些模型均为概率预测模型, 即通过条件概率来判断当前序列的合法性, 且文中未将这些方法应用在 ICS 领域来验证这些方法在异常检测时的有效性. 有学者通过一种变阶马尔科夫模型 — 概率后缀树模型 (Probabilistic suffix trees, PST) 来实现 ICS 领域的异常检测[22−23]. 该模型扩大了状态事件的关联性, 提升事件的相关度至树的最大深度 3. 但是不同于 DTMC 方法, 该模型中树节点的转移概率表不是通过比较概率分布变化来发现异常, 而是如文献 [21] 中多种模型一样仅仅作为一种概率预测模型来判断当前消息的合法性, 当这种可能性低于一定阈值时, 系统检测出异常. 该模型能够检测出未知状态以及未知转移异常, 但是却无法检测出 Time-based 语义攻击所带来的概率分布异常.

本文针对以上研究方法的不足, 提出一种基于混合马尔科夫树模型 (Mix-order Markov tree model, MMTM) 的异常检测算法, 充分利用 ICS 的阶段性和周期性特征, 构建系统正常运行时的行为模型 — 混合马尔科夫树. 该模型包含合法的状态事件、合法的状态转移、正常的概率分布以及正常的转移时间间隔等 4 种信息, 当被检测行为使得模型的以上 4 种信息发生较大偏差时, 系统检测出异常. 相比 DFA 模型, 本文算法利用了转移概率以及转移时间间隔信息, 从而能够检测出复杂的语义攻击; 相比 DTMC 模型, 本文算法增强了状态事件的关联度, 在必要时将决定当前状态的历史状态从一维追溯到多维, 从而将一阶马尔科夫模型扩展为变阶马尔科夫模型; 与 PST 模型相比较, 本文算法没有事先设定事件最大相关度, 即树的最大深度, 而是通过一种自适应的方法动态扩展分支状态节点的历史相关状态, 并且不再将树节点的转移概率表仅仅作为一种概率预测表, 而是通过比较建模与检测时的概率分布变化来发现概率分布异常. 此外, 本文算法在建模阶段针对人机交互、PLC 常规诊断等操作所引入的噪音设计了一种剪枝策略, 去除了模型中的低频事件和低转移事件, 维持了系统的阶段性特征, 解决了现有异常检测算法因为对噪音敏感而造成的漏报率高的问题; 并且通过去除冗余节点, 简化了系统模型, 从而提高了系统的检测效率.

1　复杂语义攻击

DTMC 模型[4] 基于一阶马尔科夫模型, 假设当前状态只与前一个历史状态存在关联性, 即事件相关度为 1. 该模型能够检测简单的 Time-based 以及 Order-based 语义攻击. 然而仅仅从前一个历史状态到当前状态的转移关系和概率分布情况来判断消息序列的合法性在面对复杂语义攻击时存在明显的不足. 以某一状态序列 “abcdbedfabcdbedfabcdbedf···” 为例, 构建的 DTMC 模型如图 1 所示.



b 节点转移分布表

目标	频率	概率
c	3	0.5
e	3	0.5

图 1　序列 “abcdbedfabcdbedf···” 对应的 DTMC 模型

Fig. 1　DTMC model for the sequence“abcdbedfabcdbedf···”

针对该模型, 可能存在以下两种复杂语义攻击:

1) 分支节点攻击

对于分支节点 b, b→c 以及 b→e 的转移均是合法的, 攻击者通过颠倒 “bcd” 和 “bed” 子序列的顺序, 构建消息序列 “abedbcdfabedbcdfabedbcdf···”. 该消息序列经 DTMC 模型的检测, 由于未出现未知状态, 所有状态转移均合法, 且转移分布表统计得到的概率分布情况与正常消息序列建模时相同, 因此 DTMC 模型不会报出任何异常. 然而该消息序列由于不符合正常的工业控制过程, 可能会对系统造成意想不到的破坏;

2) 周期性攻击

攻击者以极短的时间间隔连续发送大量的正常消息序列 “abcdbedfabcdbedfabcdbedf···”, 该序列符合正常生产过程中的周期性特征, 因此该序列经 DTMC 模型的检测, 不会报出任何的异常. 但是由于时间间隔明显缩短, 设备以一种异常的频率在工作, 这种频率可能会给设备带来极大的损坏.

本文主要针对以上两种复杂语义攻击提出MMTM 异常检测方法.

2 模型设计

如图 2 所示, 本文在 ICS 的 HMI-PLC 层以及PLC-现场设备层之间的交换机上接入监听器, 以记录每个交互层之间的通信数据, 利用该通信数据构建系统正常运行时的行为模型. ICS 的阶段性特征体现在 HMI-PLC、PLC-现场设备的交互中, 如果对所有交互层数据包统一建模, 即将图 2 中多个监听器记录的数据联合建模, 系统的阶段性以及周期性特征会减弱甚至丧失, 最终会构建一个相当大规模的 MMTM, 此模型过于复杂且检测效率和检测准确率均会下降, 因此本文采用各交互层分离建模的方法, 对每个 HMI-PLC 以及 PLC-现场设备交互层行为单独进行分析.

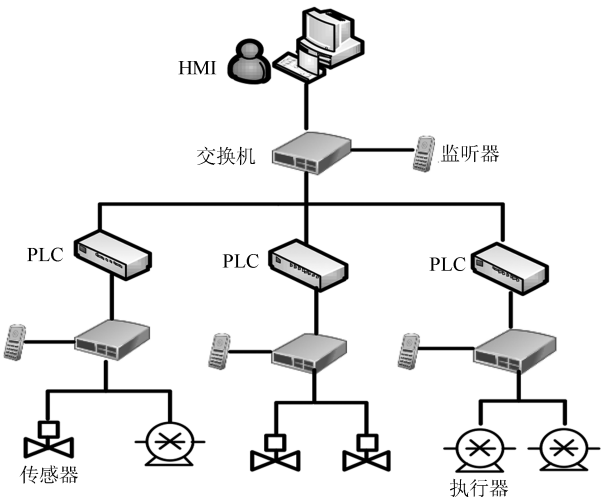


图 2 ICS 监听器布局示意图

Fig. 2 ICS listener layout

在构建混合马尔科夫树之前, 需要根据原始消息序列构建状态转移图, 该转移图包含一系列合法的状态节点以及状态转移关系. 状态节点的定义根据不同的应用层协议而有所不同. 以 Modbus 协议为例, 状态可表示为对某一现场设备的读或者写操作, 如状态 a 表示读温度传感器操作 (对应 Modbus 的读保持寄存器), 状态 b 表示打开阀门操作 (对应 Modbus 的写线圈). 消息序列中的每条消息都可以映射为某一特定的状态. 状态 a 到状态 b 的转移关系表明, 在消息序列中, 至少有一条属于状态 b 的消息是紧跟在一条属于状态 a 的消息之后的. 此外, 根据状态转移关系, 我们可以获取每个节点的出入度信息, 该信息可用于在后续建模阶段判断状态节点是否为分支节点和单一入度节点.

构建完状态转移图, 并将原始消息序列转化为状态序列和与之对应的时间间隔序列之后, 就可以构建混合马尔科夫树. 依次根据状态序列中的值对马尔科夫树进行更新, 动态自适应地将分支树节点扩展为多阶马尔科夫以增强状态事件的关联度. 当待扩展的树节点为单一入度节点时, 说明该状态的前一历史状态是确定的, 从而无需将更多的历史状态信息添加到马尔科夫树中, 至此该树节点扩展结束. 在上述混合马尔科夫树的构建过程中, 每个树节点更新并维护一张转移分布表, 该分布表表明了合法的状态转移关系, 记录了正常情况下的转移概率信息和转移时间间隔信息.

2.1 模型构建流程

基于前文模型描述, 本文混合马尔科夫树模型的主要构建流程如图 3 所示.

图 3 说明如下:

1) 原始消息

监听点捕获到的原始数据包信息;

2) 渠道划分

根据原始数据包的 IP 地址划分成多个渠道, 为每个渠道单独建模;

3) 过滤器

对原始消息序列进行过滤, 过滤掉单一的请求或者响应包, 只保留能匹配成功的请求响应包, 通过此操作能有效减少建模过程中由于随机产生的数据包丢失和重传而引入的噪音;

4) 状态转移图

对过滤之后的消息序列构建状态转移图, 获得每个状态节点的出度/入度信息, 并将消息序列转化为状态序列和与之对应的时间间隔序列;

5) 马尔科夫树

利用上述步骤中获取的状态节点出入度信息, 状态序列以及时间间隔序列构建混合马尔科夫树, 每个树节点维护一张转移分布表, 该分布表包含转移目标、转移概率以及转移时间间隔等信息, 例如, 图 3 中渠道 n 所构建的马尔科夫树中节点 b 的转移分布表如表 1 所示.

表 1 节点 b 转移分布表

Table 1 The transfer table of node "b"

转移目标	转移频率	转移概率	时间间隔总和	时间间隔均值
c	3	0.5	0.3 ms	0.1 ms
e	3	0.5	0.3 ms	0.1 ms

2.2 模型检测机制

利用建模过程中得到的状态转移图以及混合马尔科夫树, MMTM 可检测如下 4 种异常:

1) 未知状态异常

被检测消息不属于状态转移图中的任何一个状态节点. 此异常可用于检测非语义攻击, 如 Modbus 功能码异常攻击、缓冲区溢出攻击等, 因为发生以上攻击时, 通常会出现新的状态事件;

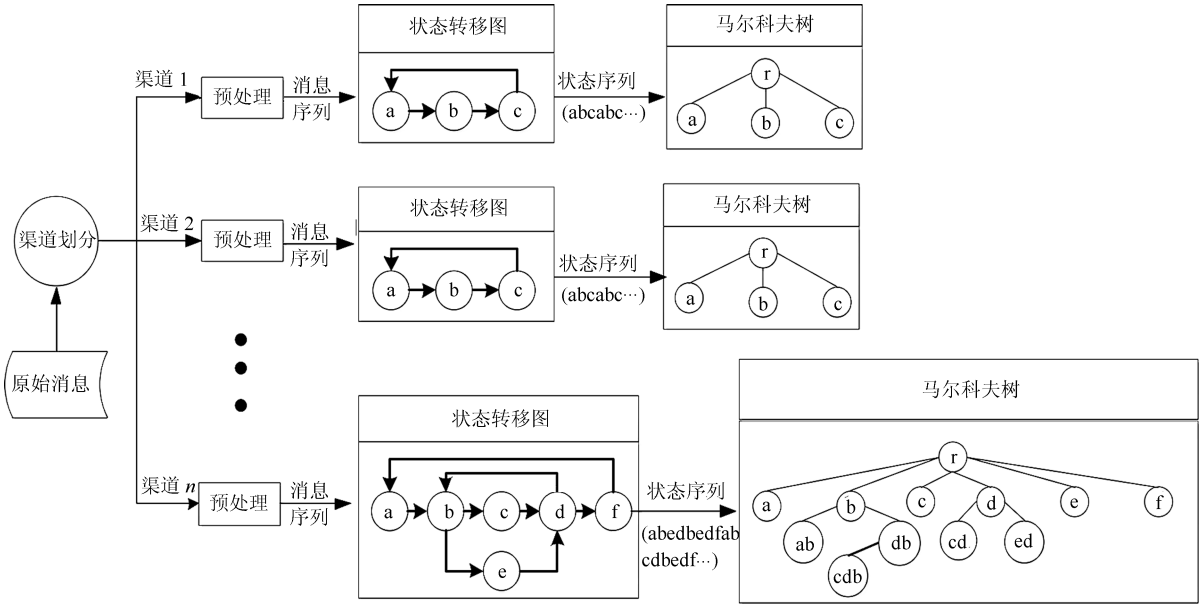


图 3　模型构建流程

Fig. 3　Model building process

2) 未知转移异常

混合马尔科夫树中所有与当前待检测状态相关的历史状态树节点的转移分布表中均不包含转移目标为待检测状态的表项. 此异常可用于检测 Order-based 简单语义攻击以及前文描述的分支节点复杂语义攻击. 由图 3 中的混合马尔科夫树可知, 针对分支节点 b, MMTM 不是仅仅从历史状态 b 来判断状态 c、e 的合法性, 而是追溯到状态 a, 从联合状态 ab 来判断 c、e 的合法性. 通过训练正常消息序列, 可以得知 ab→c 的转移是合法的, 而 ab→e 的转移是非法的, 由此可知, MMTM 通过发现 “未知转移异常” 来检测分支节点语义攻击;

3) 概率分布异常

对某一树节点的转移分布表而言, 检测过程中统计得到的转移概率值与训练过程中统计得到的转移概率值偏差超过阈值. 此异常可用于检测信息收集攻击, Time-based 语义攻击以及 DoS 攻击 (大量数据包丢失时, 概率分布将发生变化);

4) 时间间隔异常

对多个树节点的转移分布表而言, 检测过程中统计得到的时间间隔均值信息与训练过程中统计得到时间间隔均值信息偏差超过阈值. 此异常可用于检测前文描述的周期性复杂语义攻击.

3　建模过程

3.1　构建状态转移图

在构建混合马尔科夫树之前, 需要根据消息序列构建状态转移图, 以获得每个状态节点的出入度信息, 在构建马尔科夫树时状态节点的出入度信息可用于判断其是否为分支节点和单一入度节点. 另外, 此过程还需将消息序列转化为相应的状态序列以及对应时间间隔序列, 以便于在后续阶段直接对简化的状态序列和时间间隔序列进行操作, 而无需重新扫描原始消息序列, 从而减少了处理大量原始消息序列的时间开销.

3.1.1　基于 Modbus 的状态事件定义

状态转移图与 DTMC 模型类似, 包含一系列状态节点以及各状态节点之间的转移关系, 每个状态节点代表一种具体的操作, 但每个状态节点无需像 DTMC 一样维护一张转移分布表, 以维护转移信息, 概率信息或者时间信息. 不同的应用层协议对状态事件的定义是不同的, 但是这对 MMTM 的构建来说是相对独立的. 本文以 Modbus 协议为例, 以一个五元组来定义状态事件 ⟨EventId, UnitId, FuncCode, pduData, LastTime⟩, 其中:

1) EventId: 状态事件的唯一标识符;

2) UnitId: Modbus 协议中的从设备标识符;

3) FuncCode: Modbus 协议中的功能码;

4) pduData: Modbus 协议中的数据单元部分, 包括线圈/寄存器地址、线圈/寄存器数量、写内容等信息;

5) LastTime: 最后一条属于该状态的消息的时间戳.

通过以上状态事件的定义, 消息序列中任何一条 Modbus 消息均可映射成唯一的状态事件. 此外, 每个状态事件还包含以下两个统计属性:

1) ID: 状态节点的入度值

2) OD: 状态节点的出度值

3.1.2 转移图构建算法

构建状态转移图时, 对原始消息序列中的每条消息依次执行如下操作:

1) 根据状态事件的五元组定义完成消息到状态的映射, 若映射成功, 则直接执行 3), 否则执行 2);

2) 根据当前消息创建新的状态事件, 执行 3);

3) 更新当前状态与前一历史状态的转移关系, 将邻接矩阵对应位置的值置 1; 将当前状态的 LastTime 值设为当前消息的时间戳, 执行 4);

4) 将当前状态的标识符 EventId 添加到状态序列中; 将当前消息时间戳与前一历史状态 LastTime 的差值添加到时间间隔序列中, 重复执行 1).

最后, 根据邻接矩阵的值, 计算每个状态节点的入度以及出度值.

以上处理过程可用如下算法 1 进行描述.

算法 1. 状态转移图构建算法

输入: Message Sequence

输出: Sates, adjArray[ ][ ], stateSequence, durationSequence

```
for all msg ∈ Message Sequence do
{
    currentState = mapState(msg);
    if(currentSate == NULL)
    {
      currentState = createState(msg);
      States.add(currentState);
    }
    adjArray[preState.EventId][currentState.
    EventId] = 1;
    currentState.LastTime = msg.time;
    stateSequence.add(currentState.EventId);
    durationSequence.add(msg.time-
    preState.LastTime);
preState = currentState;
}
for all state ∈ States
{
    Caculate state.ID;
    Caculate state.OD;
}
```

为了进一步论述上述过程, 以如下 4 条消息对象组成的消息序列为例, 详细说明状态转移图的构建过程:

1) TransId = “1”, UnitId = “1”, FuncCode = “1” (读线圈), pduData = “Address = 0; Quantity = 1”, Timestamp = “2016.06.01, 12:15:20.032”;

2) TransId = “2”, UnitId = “1”, FuncCode = “3” (读保持寄存器), pduData = “Address = 1; Quantity = 2”, Timestamp = “2016.06.01, 12:15:20.048”;

3) TransId = “3”, UnitId = “1”, FuncCode = “1” (读线圈), pduData = “Address = 0; Quantity = 1”, Timestamp = “2016.06.01, 12:15:21.168”;

4) TransId = “4”, UnitId = “1”, FuncCode = “3” (读保持寄存器), pduData = “Address = 1; Quantity = 2”, Timestamp = “2016.06.01, 12:15:22.192”.

上述消息序列构建状态转移图的第一阶段如图 4 所示. 对于消息序列中的消息 1) , 消息到状态的映射失败, 因为当前未创建任何状态事件, 因此创建新的状态 a, 该状态表示从起始地址 0 读取一个线圈的值; 消息序列中的消息 2) , 表示从起始地址 1 读取两个保持寄存器的值, 无法映射到状态 a, 因此创建新的状态 b, 将该消息时间戳 “2016.06.01, 12:15:20.048” 与状态 a 时间戳 “2016.06.01, 12:15:20.032” 的差值添加到时间间隔序列中, 并且设置邻接矩阵 adjArray[1][2] = 1.

上述消息序列构建状态转移图的第二阶段如图 5 所示. 消息序列中的消息 3) 根据状态事件的定义可映射到状态 a, 因此将状态 a 的 LastTime 值设为当前消息的时间戳 “2016.06.01, 12:15:21.168”, 将该消息时间戳与状态 b 时间戳 “2016.06.01, 12:15:20.048” 的差值添加到时间间隔序列中, 并且设置邻接矩阵 adjArray[2][1] = 1; 消息序列中的消息 4) 可映射为状态 b, 因此将状态 b 的 LastTime 值设为当前消息的时间戳 “2016.06.01, 12:15:22.192”, 并且将该消息时间戳与状态 a 时间戳 “2016.06.01,

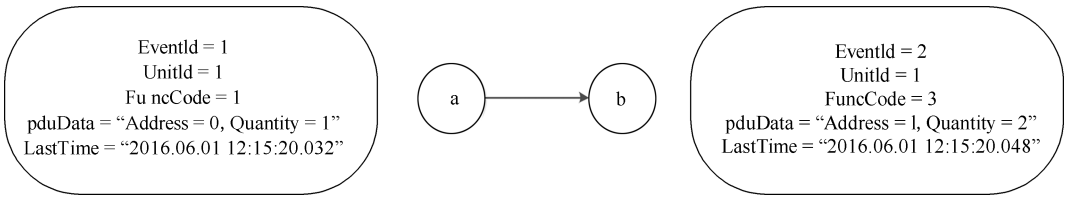


图 4 状态转移图构建阶段一

Fig. 4 Construct state transition graph–phase I

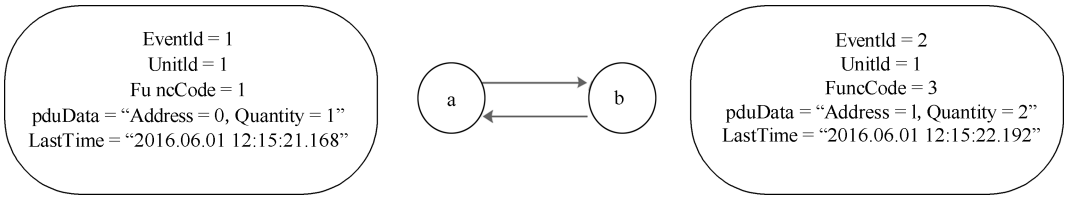


图 5　状态转移图构建阶段二

Fig. 5　Construct state transition graph–phase II

12:15:21.168” 的差值添加到时间间隔序列中. 最终转化得到的状态序列为 “1, 2, 1, 2”, 对应的时间间隔序列为 “0.016, 1.12, 1.024”.

3.2　构建混合马尔科夫树

在构建状态转移图之后, 利用状态节点的出入度信息, 状态序列以及时间间隔序列构建混合马尔科夫树, 动态自适应地将分支节点扩展为多阶马尔科夫, 增强状态事件的关联度, 结合更多的历史状态信息判断当前状态转移的合法性.

3.2.1　树骨架构建算法

构建马尔科夫树, 首先需要构建树的骨架, 并同时建立每个树节点的转移分布表. 其算法描述如下算法 2 所示.

算法 2. 马尔科夫树构建算法

输入: State Sequence

输出: root

create root node;

for all $\text{state}_i \in$ State Sequence do

{

　if($\text{state}_i! \in$ root.childNode)

　addChildNode(root, state_i);

　add/updateTrans(state_i, state_{i+1});

　if (state_i.OD! = 1)

　{　int selfLoop = 1;

　for(int $k = i-1; k \geq 0; k--$)

　{

　if (state_k == state_{k+1}) selfLoop++;

　else selfLoop = 1;

　if ($\text{state}_k! \in \text{state}_{k+1}$.childNode)

　addChildNode (state_{k+1}, state_k);

　add/updateTrans (state_k, state_{i+1});

　if(state_k.ID == 1 ‖ selfLoop == 3)

　break;

　}

　}

}

构建时, 需要对状态序列中的值依次执行以下操作:

1) 创建根节点 root;

2) 判断当前状态节点 $state_i$ 是否已经包含在 root 的孩子节点中, 若是则直接执行 4), 否则执行 3);

3) 将当前状态添加到 root 的孩子节点中, 执行 4);

4) 添加或者更新当前状态节点 $state_i$ 关于 $state_{i+1}$ 的转移分布表项, 修改相应的转移频率 $weight$ 以及累计时间间隔 $durationTotal$, 执行 5);

5) 通过判断当前状态节点的出度是否为 1 以判断其是否为分支节点, 若是, 则执行 6), 否则继续执行 2);

6) 依次将历史状态 $state_{i-1}$, $state_{i-2}$, $\cdots$, $state_k$ 扩展为后一状态 $state_{k+1}$ 的孩子节点, 并且更新相应节点关于 $state_{i+1}$ 的转移分布表项, 直到 $state_k$ 为入度为 1 的节点或者同时扩展相同的节点超过三次, 继续执行 2).

以状态序列 “abcdbedfabcdbedfabcdbedf” 为例, 详细说明上述构建过程. 状态序列对应的状态转移图如图 6 所示, 为方便说明, 假设时间间隔序列为 “0.1, 0.1, 0.1, 0.1$\cdots$”. 首先, 创建马尔科夫树的根节点 root; 然后, 依次根据状态序列中的值对马尔科夫树进行更新. 序列中第一个状态值 “a”, 未包含在 root 的孩子节点中, 因此将 “a” 添加到 root 的孩子节点中, 并将序列中后续状态值 “b” 添加到树节点 “a” 的转移分布表中: 转移目标为 “b”, 转移频率为 1, 累计时间间隔为 0.1. 由图 6 可知状态节点 “a” 的出度为 1, 为非分支节点, 无需对非分支节点进行变阶扩展, 继续对序列中第二个状态值 “b” 进行处理. 将 “b” 添加到 root 的孩子节点中, 并将序列中后续状态值 “c” 添加到树节点 “b” 的转移分布表中: 转移目标为 “c”, 转移频率为 1, 累计时间间隔为 0.1. 由图 6 可知状态节点 “b” 为分支节点, 因此需对该树节点进行变阶扩展, 将序列中前一历史状态 “a” 扩展为树节点 “b” 的孩子节点 “ab”, 并将序列中后续状态值 “c” 添加到树节点 “ab” 的转移分布表中: 转移目标为 “c”, 转移频率为 1, 累计时间间隔为 0.1. 由图 6 可知状态节点 “a” 入度为 1, 为单一入度节点, 此时变阶扩展结束, 继续对序列中第三个状态值 “c” 进行处理. 类似处理其他状态值.

图 7 说明了上述构建过程. 最终构建的混合马尔科夫树模型如图 8 所示.

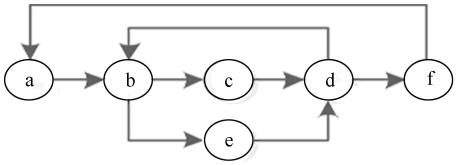


图 6 序列 “abcdbefabcdbefabcdbedf” 对应状态转移图

Fig. 6 State transition graph of sequence “abcdbedfabcdbedfabcdbedf”

3.2.2 转移概率/时间间隔均值统计

在构建树的基本框架之后, 需要对每个节点的转移分布表进行统计, 计算求得转移概率以及时间间隔均值. 记 “cdb” 状态节点的出现频率为 *node.weight*, 其转移分布表中 e 表项的转移频率为 *trans.weight*, 时间间隔累计和为 *trans.durationTotal*, 则该表项中:

1) 转移概率 = *trans.weight*/*node.Weight*

2) 时间间隔均值 = *trans.durationTotal*/*trans.weight*

3.2.3 去噪剪枝策略

本文假设建模时所采用的训练数据集是系统正常运行时的数据, 其在理想状态下不应该包含任何攻击数据包, 然而系统正常运行时可能由于人为操作或者 PLC 的常规诊断操作引入一些噪音, 这些噪音在系统建模时会被当作合法事件. 若不去除这些噪音, 攻击者可能会绕开异常检测系统的检测, 利用这些噪音执行攻击. 因此, 本文设计了一种剪枝策略, 去除马尔科夫树中低频状树节点, 状态转移表中的低转移事件, 并且剪去包含重复转移信息的冗余节点. 记马尔科夫树中根节点的频率为 *root.weight* (即状态序列的大小), 某一状态节点的出现频率为 *node.weight*, 该节点转移分布表中某转移项的出现概率为 *trans.probability*.

1) 低频事件

指 $node.weight/root.weight < minEvtProb$, $minEvtProb$ 为最小状态事件概率阈值. 当出现低频事件时, 将相应状态节点连同其孩子节点一起剪去.

2) 低转移事件

指 $trans.probability < minTransProb$, $minTransProb$ 为最小转移概率阈值. 当出现低转移事件时, 将该表项从对应的转移分布表中删除.

3) 冗余节点

指孩子节点与父节点的转移分布表相同, 孩子节点所包含的转移信息是重复并且多余的, 出现这种情况时, 通常表现为父节点只包含单一孩子节点, 此时应当剪去父节点中所包含的单一孩子节点.

图 9 举例说明了某马尔科夫树模型的去噪以及剪枝过程.

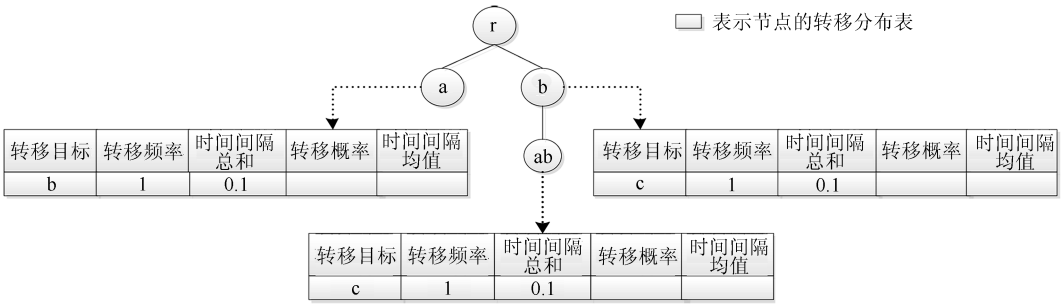


图 7 子序列 “abc” 对应部分马尔科夫树骨架

Fig. 7 Part of the Markov tree of sub-sequence “abc”

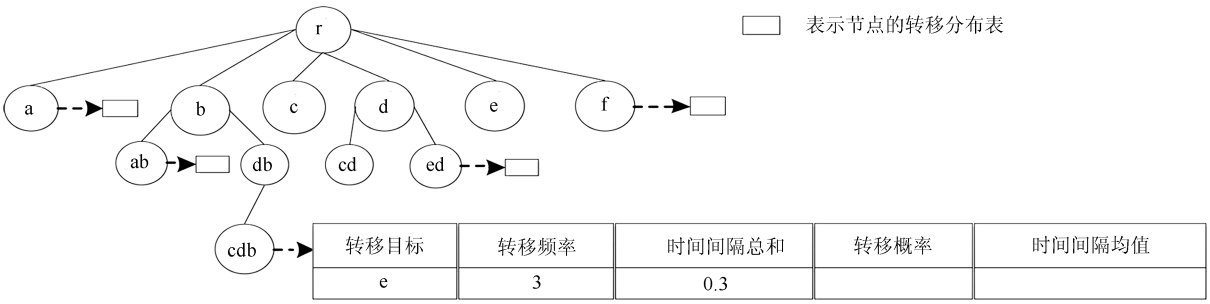


图 8 序列 “abcdbefabcdbefabcdbef” 对应完整马尔科夫树骨架

Fig. 8 Complete Markov tree of sequence “abcdbedfabcdbedfabcdbedf ”

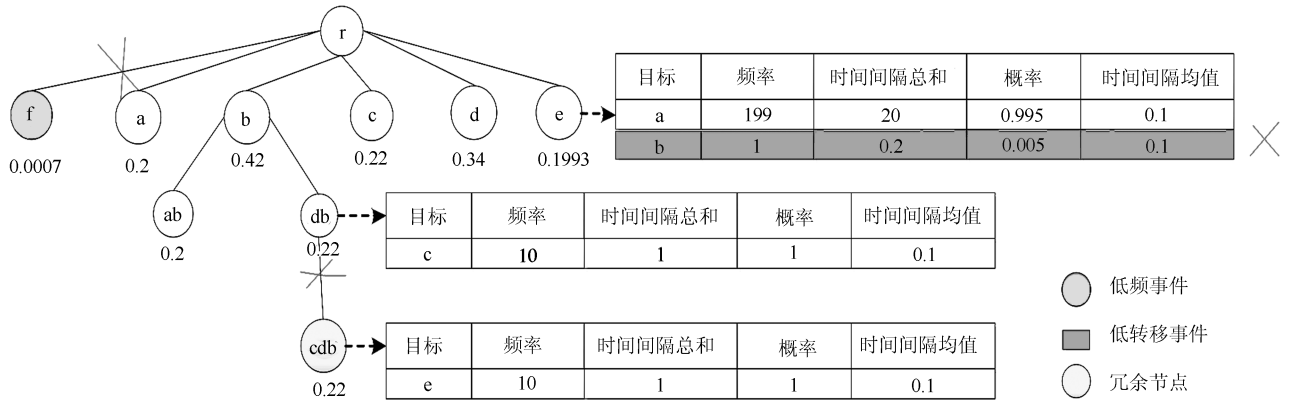


图 9　去噪剪枝过程示例

Fig. 9　The process of denoising and pruning

4　检测过程

在上述训练建模阶段, 获得了系统正常运行时的行为模型 — 混合马尔科夫树, 下文将利用该模型对异常行为进行检测.

4.1　消息映射机制

首先, 将待检测消息序列转化为状态序列以及相应的时间间隔序列, 其处理过程与建模时第 3.1.2 节所描述的过程类似, 只是此处不再根据消息内容创建新的状态事件, 也无需更新邻接矩阵以获取各状态节点的出入度, 而是直接将消息映射为建模中已创建的状态事件, 若映射失败, 则检测出 “未知状态异常”.

4.2　混合马尔科夫树最大匹配追踪算法

在获取了状态序列以及时间间隔序列之后, 对状态序列执行以下操作:

1) 按顺序依次获取状态序列中的值 $state_i$;

2) 从根节点出发, 沿父亲–孩子节点寻找 $state_{i-1}$,$state_{i-2}$, $\cdots$, $state_k$ 的匹配节点, 直至匹配失败, 获取最大匹配节点 $state_{k+1}$, 执行 3);

3) 判断 $state_i$ 是否在 $state_{k+1}$ 的转移分布表中, 若是, 则执行 4), 否则检测出 “未知转移异常”, 检测结束;

4) 更新相关表项的转移频率以及时间间隔总和信息, 执行 1).

4.3　转移概率/时间间隔均值统计

在处理完状态序列中的所有值且系统未报出 “未知转移异常” 之后, 需要统计各个树节点转移分布表中的转移概率和时间间隔均值, 其统计方法与第 3.2.2 节所描述方法相同. 记某一树节点转移分布表中, k 表项的建模时转移概率为 $trans_k.trainProbability$, 时间间隔均值为 $trans_k.trainDurationAvg$, 检测时转移概率为 $trans_k.detectProbability$, 时间间隔均值为 $trans_k.detectDurationAvg$.

1) “概率分布异常” 检测

若式 (1) 成立, 则检测出 “概率分布异常”, 其中 $probabilityThreshold$ 为概率偏差阈值, 本文中设置为 0.1.

2) “时间间隔异常” 检测

若式 (2) 成立, 则检测出 “时间间隔异常”, 其中, $trans_k$ 表示任一转移分布表中表项, $func(trans_k)$ 定义如式 (3) 所示, N 为总的转移分布表项数目; $abnormalThreshold$ 为异常度阈值, 本文中设置为 0.4. 式 (3) 中 $durationThreshold$ 为时间间隔偏差阈值, 本文中设置为 0.4.

$$|trans_k.detectProbability - trans_k.trainProbability| > probabilityThreshold \tag{1}$$

$$\frac{\sum_{k=1}^{N} func(trans_k)}{N} > abnormalThreshold \tag{2}$$

$$func(trans_k) = \begin{cases} 1, & 若 \dfrac{|trans_k.detectDurationAvg - trans_k.trainDurationAvg|}{trans_k.trainDurationAvg} > durationThreshold \\ 0, & 否则 \end{cases} \tag{3}$$

5 模型验证与结果分析

为了验证本文所提出的 MMTM 的有效性, 本文设计了两种验证方法, 并与现有的异常检测算法在检测能力、误报率、漏报率等方面进行对比. 首先搭建一个仿真的工控系统, 采集该系统正常运行时的网络通信数据包用以建模, 并在该系统中实现模拟攻击, 采集系统遭受攻击之后的通信数据包用以异常检测, 从而对模型进行功能性验证. 其次, 为了对 MMTM 的误报率和漏报率进行有效的评估, 需要大量且接近真实工控系统的网络通信数据, 因此本文引入真实物理测试床的网络通信数据集对模型进行性能验证, 有力地说明了本文方法检测的准确性, 且便于同其他检测方法在误报率和漏报率等方面进行对比.

5.1 仿真环境搭建

本文基于 OMNeT++[24] 网络仿真环境构建一个简化的污水处理系统, 并在该仿真系统中模拟实现 Modbus 通信协议, 基于该协议实现整个 ICS 的控制过程. 该 ICS 系统由一个 HMI、一个 PLC 以及多个现场设备组成. 现场设备主要包括: 水槽、注水阀门、排水阀门、加热器、水位传感器以及温度传感器. PLC 的控制过程主要包括三个循环的步骤: 水槽注水, 水加热, 水槽排水. HMI 负责向 PLC 端定期采集并监控底层数据的变化情况, 监控的数据包括: 阀门的开关状态, 加热器的开关状态, 当前水位, 当前温度, 最高限制水位以及最高限制温度. 此外, 在 HMI-PLC 以及 PLC-现场设备交互层的中间交换机 Switch 上接入监听器, 用以记录网络通信数据. 该仿真系统架构如图 10 所示.

5.2 数据提取

在污水处理仿真系统中, 通过监听器记录系统中的通信数据包, 该数据包包括应用层 Modbus 协议数据, TCP 层协议数据以及 IP 层协议数据. 对原始数据包进行解析, 提取与本系统建模检测相关的特征向量, 去除无用信息, 最终得到的数据格式如下:

Time: 13.13
srcAddress/port: 192.168.1.4/502
destAddress/port: 192.168.1.3/502
transId: 9
slaveId: 5
funcCode:0x03
Data: 2; 0; 20

其中, Time 代表数据包发送的时间; srcAddress/port、destAddress/port 源目的 IP 地址作为划分多个渠道的主要依据; transId 用于匹配 Modbus 协议的相应请求包, slaveId 代表从设备对象, 如阀门, 传感器等; funcCode 表示具体的 Modbus 操作, 如读写寄存器等; Data 代表 Modbus 协议的数据单元部分, 包含寄存器地址以及读写数值等信息.

5.3 模型训练

由于本仿真系统中 HMI-PLC 交互层只涉及两个简单的周期性 Modbus 请求响应包 — 读多个线圈值和读多个寄存器, 而 PLC-现场设备层涉及更复杂的交互控制过程, 其通信数据包更具有多样性, 因此本文只对 PLC-现场设备层进行分析建模. 在仿真系统中采集系统正常运行一天的数据集 Dataset, 通过第 3.1.2 节中的分析, 获得周期性状态序列为 “abbbbbcdeeeeefefeegbbbbbhabbbbbcdeeeeefefeegbbbbbh···”, 每个状态代表一种具体的操作, 如 “a” 代表打开注水阀门, “b” 代表读水位寄存器等. 最终构建的混合马尔科夫树模型如图 11 所示.

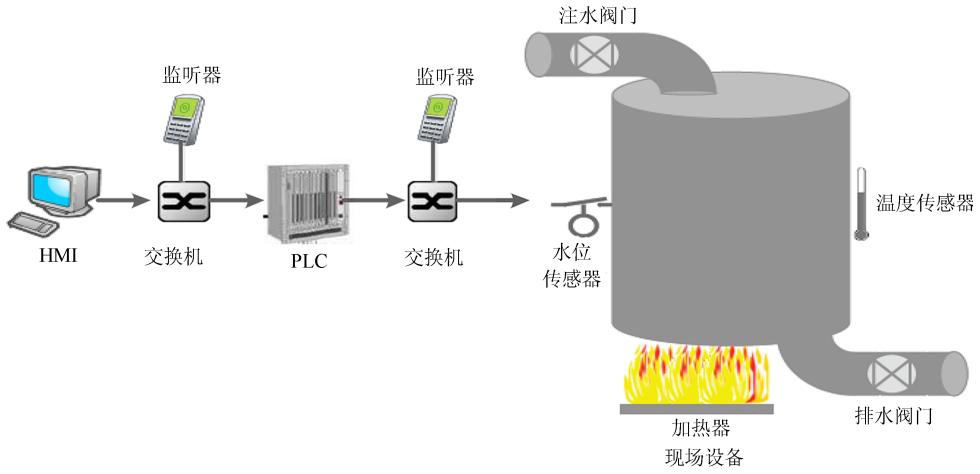
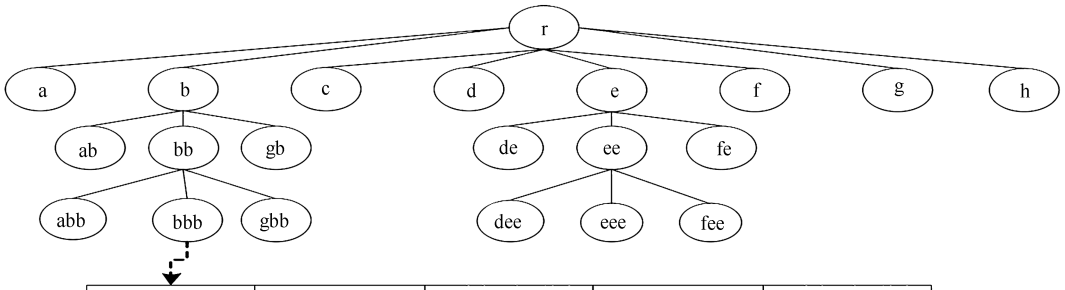


图 10 污水处理仿真系统

Fig. 10 Simulated sewage treatment system



转移目标	转移频率	时间间隔总和	转移概率	时间间隔均值
b	4 172	62 580	0.66	15
c	1 043	1 679	0.17	1.61
h	1 043	1 408	0.17	1.35

图 11　仿真系统建模结果

Fig. 11　The model of the simulation system

5.4　模拟攻击与异常检测

为了验证 MMTM 对各种类型攻击检测的有效性, 本文在 OMNeT++ 仿真环境的交换机处利用劫持篡改方式, 修改 PLC 和现场设备的 Modbus 通信数据包, 从而模拟实现了多种攻击, 并对攻击之后的系统进行数据采集, 获得了多种攻击数据集, 用图 11 中的训练模型分别对其进行检测.

1) 非语义攻击

在交换机处随机劫持一部分数据包进行篡改, 如将 Modbus 功能码变为 “0x08”(诊断操作), 从而构成功能码异常攻击; 或者将 Modbus 功能码变为 “0x06”(写单一寄存器), 同时将数据单元部分变为 “0;0;1;0;255;1;0;1;1;0”(10 字节), 因为写单一寄存器操作时, 数据单元部分按照规定应为 2 字节的寄存器地址加 2 字节的写入值, 由此可知该篡改操作构成了典型的缓冲区溢出攻击. 经 MMTM 的检测, 由于建模阶段不包含此种类型的状态事件, 所以检测时报出 “未知状态异常”.

2) Order-based 语义攻击

通过交换机劫持篡改方式, 修改 PLC 和阀门 1、加热器间的交互过程, 将正常的关闭阀门 1 之后再打开加热器过程变为先打开加热器再关闭阀门 1, 即将正常状态序列 “abbbbbcdeeeeefefeegbbbbbh···” 中的 “cd” 子序列颠倒次序, 构成异常状态序列 “abbbbbdceeeeefefeegbbbbbh···”. 该状态序列经 MMTM 的检测, 报出 “bbb→d 未知转移异常”.

3) Time-based 语义攻击

通过交换机劫持方式, 多次重放某数据包序列, 将正常状态序列 “abbbbbcdeeeeefefeegbbbbbh···” 中的 “ef” 子序列重复多次, 构成异常状态序列 “abbbbbdceeeeefefefefefefefeegbbbbbh···”. 该状态序列经 MMTM 的检测, 报出 “fe 节点概率分布异常”.

4) 分支节点语义攻击

通过交换机劫持篡改方式, 调整分支状态节点 “e” 的分支顺序, 构造异常状态序列 “abbbbbcdefeeeeefeegbbbbbh···”, 该状态序列经 MMTM 的检测, 报出 “de→f 未知转移异常”.

5) 周期性语义攻击

直接在 PLC 控制逻辑中, 随机选取一部分消息 (多于 40 %), 利用 OMNeT++ 延迟函数 scheduleAt 将这部分消息的发送时间缩短至原来的一半, 获得消息时间间隔明显缩短的攻击数据集, 该数据集经 MMTM 的检测, 报出 “时间间隔异常”.

5.5　模型对比分析

将正常数据集 Dataset 分别用 DFA 模型、DTMC 模型以及 PST 模型进行训练, 并用训练得到的模型依次检测以上 5 种攻击数据集, 将检测结果与本文提出的 MMTM 进行对比, 其结果如表 2 所示.

表 2　各模型攻击检测结果对比

Table 2　Comparison of detection results of each method

检测模型	功能码异常攻击	缓冲区溢出攻击	Order-based 语义攻击	Time-based 语义攻击	分支节点语义攻击	周期性语义攻击
DFA	检测成功	检测成功	检测成功	检测失败	检测失败	检测失败
DTMC	检测成功	检测成功	检测成功	检测成功	检测失败	检测失败
PST	检测成功	检测成功	检测成功	检测失败	检测成功	检测失败
MMTM	检测成功	检测成功	检测成功	检测成功	检测成功	检测成功

由表 2 可知, MMTM 相比 DTMC、PST 以及 DFA 模型表现出更完整的检测能力, 除了能够检测传统的非语义攻击、Time-based、Order-based 的简单语义攻击之外, 还能够检测分支节点攻击和周期性攻击等更复杂的语义攻击.

5.6 误报率与漏报率评估

5.6.1 数据集来源

为了对 MMTM 的检测准确率进行评估, 并将该方法同其他异常检测方法在误报率、漏报率等方面进行对比, 本文使用安全水处理模型 (Secure water treatment, SWaT) 数据集[25] 作为评估各方法检测能力的数据来源. SWaT 是新加坡科技设计大学搭建的一个小规模、完全可操作的实物仿真平台, 它模拟了一个大型的现代水处理工厂的处理过程, 每分钟能生产 5 加仑的过滤水, SWaT 包含 6 道工艺过程, 如超滤、脱氯、反渗透等, 每个处理过程由一个 PLC 进行单独控制[26]. 该组织收集了系统连续操作运行 11 天的网络交互数据包, 执行器以及传感器的值变化情况, 其中有 7 天是模型正常运行时的数据记录, 另外 4 天是系统遭受 36 种攻击时的数据记录. 这 36 种攻击包含传统的非语义攻击, 也包含通过数据包劫持篡改方式所实现的语义攻击, 如修改传感器的测量值, 致使 PLC 发出错误的执行器指令; 或者直接修改执行器指令, 阻止阀门的正常关闭等. 由于 SWaT 中包含 7 个交互控制层: HMI-PLCs 交互层, 以及 6 个 PLC-现场设备交互层. 本文对 7 个交互控制层分别建模并进行检测评估.

5.6.2 误报率评估

为了测试 MMTM 对正常数据的误报情况, 并与其他算法进行误报率对比, 本文将 SWaT 数据集的正常数据集分为两部分: 将其中前 3 天的数据集作为训练数据, 将后 4 天的数据集作为待检测数据, 对 7 个交互控制层分别利用不同算法进行建模并测试的结果如表 3 所示.

表 3 误报率评估

Table 3 False positive rate evaluation

交互控制层	MMTM	DFA	DTMC	PST
HMI-PLCs	1.21 %	1.28 %	1.197 %	1.208 %
PLC1-现场设备	0.85 %	1.151 %	0.83 %	0.828 %
PLC2-现场设备	0.93 %	1.226 %	0.921 %	0.93 %
PLC3-现场设备	1.124 %	1.205 %	1.12 %	1.108 %
PLC4-现场设备	1.09 %	1.199 %	0.97 %	1.05 %
PLC5-现场设备	1.075 %	1.27 %	1.066 %	1.073 %
PLC6-现场设备	0.974 %	1.13 %	0.97 %	0.97 %

通过分析可知, 各交互控制层模型中产生的大部分误报是网络通信过程中的数据包延迟而造成的“未知转移异常”. 此外, DFA 算法相比于其他算法误报率最高, 主要是由于其将请求和响应数据包都作为状态节点和输入符号参与到自动机的构建和检测中, 而 MMTM、DTMC 以及 PST 算法均事先将请求包和响应包进行匹配, 只有匹配成功的数据包对才会参与到模型的构建和检测中, 因此有效避免了因数据包丢失和重传操作而造成的 “未知转移异常” 所带来的误报. 最后, 还有一小部分误报是由于人机交互操作和 PLC 自诊断操作所引起的. 由于本文算法在建模时采用了去噪剪枝策略, 将一些人机交互操作和 PLC 自诊断操作数据包作为噪音去除, 当待检测正常数据集中恰好出现以上类型数据包时, 系统识别为异常, 发出警报给操作人员, 操作人员根据异常详细信息最终确认是否发生真实攻击. 因此, 本文算法相比于 DFA、DTMC 以及 PST 算法, 误报率略微高出一点, 但是以误报率为代价, 换取低漏报率在对可靠性以及安全性要求较高的 ICS 中是十分值得的.

5.6.3 漏报率评估

针对现有语义攻击检测算法因为对噪声敏感而造成的漏报率高的问题, 本文提出去除训练模型中低频事件和低转移事件的剪枝策略. 为了测试 MMTM 对异常数据检测的漏报情况, 且验证本文的去噪剪枝策略对漏报率的改进效果, 并与其他算法进行对比, 本文将 SWaT 数据集中系统连续 4 天遭受多种语义以及非语义攻击时的交互数据包作为待检测数据, 利用剪枝前 MMTM、剪枝后 MMTM、DFA、DTMC 以及 PST 模型分别对以上测试数据集进行异常检测, 其漏报率统计结果如表 4 所示.

表 4 漏报率评估

Table 4 False negative rate evaluation

交互控制层	剪枝前 MMTM	剪枝后 MMTM	DFA	DTMC	PST
HMI-PLCs	0.694 %	0.58 %	1.328 %	1.10 %	1.093 %
PLC1-现场设备	0.498 %	0.432 %	1.124 %	0.995 %	1.01 %
PLC2-现场设备	0.607 %	0.55 %	1.301 %	1.027 %	1.02 %
PLC3-现场设备	0.778 %	0.501 %	1.48 %	1.139 %	1.115 %
PLC4-现场设备	0.41 %	0.329 %	1.119 %	0.971 %	1.13 %
PLC5-现场设备	0.492 %	0.47 %	1.138 %	0.89 %	1.011 %
PLC6-现场设备	0.197 %	0.203 %	0.897 %	0.704 %	0.703 %

通过分析可知, 在各交互控制层中, 本文 MMTM 漏报率均低于 DFA、DTMC 和 PST 模型, 其中 DFA 模型的漏报率最高, DTMC、PST 模型次之. 主要是因为相较于另外三个算法, 本文算法在未知转移以及概率分布异常时检出率较高. DFA

和 DTMC 模型进行未知转移检测时, 仅关联一个历史状态, 因此面对分支节点语义攻击时检出率较低; PST 模型和 DFA 模型由于未对建模和检测时的概率分布情况进行对比, 因此面对 Time-based 语义攻击时检出率较低; 此外, 可以看到本文的剪枝后 MMTM 相较于剪枝前 MMTM 以及其他模型漏报率有显著提升, 主要是因为训练数据集中包含一部分人机交互操作和 PLC 常规自诊断操作, 当未有效去除这些操作所引入的噪音, 且测试数据集中恰好包含与此噪音相同的数据包时, 其他模型不识别此异常, 而剪枝后的 MMTM 将其识别为异常, 由此降低了系统的漏报率, 有效防止了攻击者利用诊断数据包进行信息收集攻击等. 最后, 剪枝后的 MMTM 仍然存在一小部分漏报, 主要是由于剪枝过程中相关阈值设置的偏低, 导致一些噪音仍未被有效剔除.

在第 3.2.3 节设计去噪剪枝策略时, 提到了两种阈值: 最小状态事件概率阈值 *minEvtProb* 和最小转移概率阈值 *minTransProb*, 这两个阈值分别和低频事件以及低转移事件的定义有关. 当阈值设置的过低时, 一些发生较为频繁的人机交互和 PLC 自诊断操作未被判定为噪音从而无法被有效的去除, 导致模型漏报率提高; 而当阈值设置的过高时, 一些小概率正常事件会被误判为是噪音从而被剔除, 导致模型的误报率提高. 本文在上述实验过程中将阈值 *minEvtProb* 设置为 0.001, *minTransProb* 设置为 0.01, 阈值设置的偏低, 导致模型仍然存在一小部分漏报. 为了进一步降低模型的漏报率, 并保持误报率在可接受的范围内, 下文将对阈值的选取进行优化调整.

在选定阈值 *minTransProb* 为 0.01 的情况下, MMTM 在阈值 *minEvtProb* 取不同值时的误报率和漏报率情况如图 12 所示.

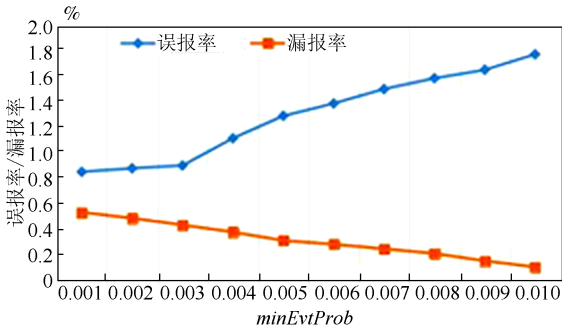


图 12　不同 *minEvtProb* 阈值下误报率和漏报率

Fig. 12　The FAR and MAR for different values of *minEvtProb*

由图 12 可知, 漏报率随着 *minEvtProb* 的增大而缓慢减小, 其变化过程较为平缓; 当 $0.001 \leq minEvtProb \leq 0.003$ 时, 误报率随着 *minEvtProb* 的增大而缓慢变大, 而当 $minEvtProb \geq 0.003$ 时, 误报率急剧上升, 说明此时由于 *minEvtProb* 设置的过大, 导致建模过程中一部分正常操作被误当作“低频事件”而去除, 从而导致检测过程中部分正常操作被错误的识别为“未知状态异常”. 因此, 为了在不引起误报率显著提升的前提下, 适当降低漏报率, 应当将 *minEvtProb* 设置为 0.003.

在选定阈值 *minEvtProb* 为 0.003 的情况下, MMTM 在阈值 *minTransProb* 取不同值时的误报率和漏报率情况如图 13 所示.

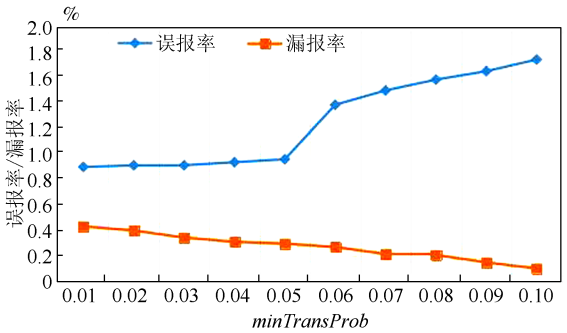


图 13　不同 *minTransProb* 阈值下误报率和漏报率

Fig. 13　The FAR and MAR for different values of *minTransProb*

由图 13 可知, 当 *minTransProb* 由 0.01 增大到 0.1 时, 漏报率呈现缓慢下降趋势; 当 $0.01 \leq minTransProb \leq 0.05$ 时, 误报率随着 *minTransProb* 的增大缓慢上升, 而当 $minTransProb \geq 0.05$ 时, 误报率开始急剧上升, 说明此时由于 *minTransProb* 设置的过大, 导致建模过程中一部分正常状态转移被误当作“低转移事件”而去除, 从而导致检测过程中部分正常转移被错误的识别为“未知转移异常”. 因此, 为了在不引起误报率显著提升的前提下, 适当降低漏报率, 应当将 *minTransProb* 设置为 0.05.

6　结束语

本文主要针对 ICS 面临的语义攻击问题, 充分利用系统的阶段性和周期性特征, 提出基于混合马尔科夫树模型的异常检测算法. 该算法通过动态自适应的方式调整消息序列的关联度, 在必要时增强事件的关联性, 并且引入时间间隔信息, 从而能够检测出更加复杂的语义攻击. 另外通过去除建模过程中低频事件、低转移事件以及冗余节点的剪枝策略, 弥补了现有异常检测算法因为对噪音敏感而造成的漏报率高的问题. 模型验证测试结果表明, 该算法相比于现有的异常检测算法, 不仅能够检测出传统的非语义攻击、Time-based 以及 Order-based 简单语义攻击, 对各种复杂语义攻击表现出更完整的检测

能力, 并且通过剪枝策略降低了系统的漏报率, 简化了系统模型, 提高了系统的检测效率. 在今后的研究工作中, 将利用各种不同类型的、真实的工业控制系统通信数据, 完善本文提出的算法, 并进行更进一步的测试, 并与现有的其他攻击检测算法在误报率、漏报率以及时间效率等方面进行对比; 另外, 还将深入关注并探索各种其他类型的攻击方式, 以不断验证和完善算法对不同攻击方式的检测能力.

References

1 Barbosa R R R, Sadre R, Pras A. Towards periodicity based anomaly detection in SCADA networks. In: Proceedings of IEEE 17th Conference on Emerging Technologies & Factory Automation. Krakow, Poland: IEEE, 2012. 1−4

2 Hadžiosmanović D, Sommer R, Zambon E, Hartel P H. Through the eye of the PLC: semantic security monitoring for industrial processes. In: Proceedings of the 30th Annual Computer Security Applications Conference. New Orleans, Louisiana, USA: ACM, 2014. 126−135

3 Falliere N, Murchu L O, Chien E. W32. Stuxnet Dossier. White Paper, 2011

4 Caselli M, Zambon E, Kargl F. Sequence-aware intrusion detection in industrial control systems. In: Proceedings of the 1st ACM Workshop on Cyber-Physical System Security. Singapore, Republic of Singapore: ACM, 2015. 13−24

5 Fovino I N, Carcano A, De Lacheze Murel T, Trombetta A, Masera M. Modbus/DNP3 state-based intrusion detection system. In: Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications. Perth, WA: IEEE, 2010. 729−736

6 Ellis J, Fisher D, Longstaff T A, Pesante L, Pethia R D. Report to the President′s Commission on Critical Infrastructure Protection, Special Report CMU/SEI-97-SR-003, Carnegie Mellon University, Pittsburgh, PA, 1997.

7 Almaö G. Comparison of robust regression methods in linear regression. *International Journal of Contemporary Mathematical Sciences*, 2011, **6**(9): 409−421

8 Ye Shi-Gang. Application of Data Mining Techniques in the AVC System Analysis [Master thesis], South China University of Technology, China, 2014
(叶石罡. 数据挖掘技术在 AVC 系统分析中的应用研究 [硕士学位论文], 华南理工大学, 中国, 2014)

9 Jiang J M, Yasakethu L. Anomaly detection via one class SVM for protection of SCADA systems. In: Proceedings of the 2013 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery. Beijing, China: IEEE, 2013. 82−88

10 Zhao J, Liu K, Wang W, Liu Y. Adaptive fuzzy clustering based anomaly data detection in energy system of steel industry. *Information Sciences*, 2014, **259**: 335−345

11 Huang Yong. Research of Anomaly detection system for Industrial Control System [Master thesis], Chongqing University of Technology, China, 2014
(黄勇. 面向 ICS 的异常检测系统研究 [硕士学位论文], 重庆理工大学, 中国, 2014)

12 Liu Qiang, Qin S Joe. Perspectives on big data modeling of process industries. *Acta Automatica Sinica*, 2016, **42**(2): 161−171
(刘强, 秦泗钊. 过程工业大数据建模研究展望. 自动化学报, 2016, **42**(2): 161−171)

13 Nivethan J, Papa M. A SCADA intrusion detection framework that incorporates process semantics. In: Proceedings of the 11th Annual Cyber and Information Security Research Conference. Oak Ridge, TN, USA: ACM, 2016. Article No. 6

14 Goldenberg N, Wool A. Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems. *International Journal of Critical Infrastructure Protection*, 2013, **6**(2): 63−75

15 Kleinmann A, Wool A. A statechart-based anomaly detection model for multi-threaded SCADA systems. In: Proceedings of International Conference on Critical Information Infrastructures Security. Berlin, Germany: Springer International Publishing, 2015. 132−144

16 Pan Hong-Yue. Design and implementation of communication based on MODBUS protocol. *Measurement Technique*, 2002, (4): 37−38
(潘洪跃. 基于 MODBUS 协议通信的设计与实现. 计量技术, 2002, (4): 37−38)

17 Adepu S, Mathur A. From design to invariants: detecting attacks on cyber physical systems. In: Proceedings of the 2017 IEEE International Conference on Software Quality, Reliability and Security Companion. Prague, Czech Republic: IEEE, 2017. 533−540

18 Li Xiao-Hang, Zhu Fang-Lai. Simultaneous estimation of actuator and sensor faults for uncertain time-delayed Markovian jump systems. *Acta Automatica Sinica*, 2017, **43**(1): 72−82
(李晓航, 朱芳来. 延迟不确定马尔科夫跳变系统的执行器和传感器故障同时估计方法. 自动化学报, 2017, **43**(1): 72−82)

19 Yang Xin-Xu, Wang Chang-Shan, Wang Dong-Qi, Zheng Li-Na. An intrusion detection system based on hidden Markov model. *Computer Engineering and Applications*, 2005, **41**(12): 149−151
(杨新旭, 王长山, 王东琦, 郑丽娜. 基于隐马尔可夫模型的入侵检测系统. 计算机工程与应用, 2005, **41**(12): 149−151)

20 Kleinmann A, Wool A. Automatic construction of statechart-based anomaly detection models for multi-threaded industrial control systems. *ACM Transactions on Intelligent Systems and Technology*, 2017, **8**(4): Article No. 55

21 Begleiter R, El-Yaniv R, Yona G. On prediction using variable order Markov models. *Journal of Artificial Intelligence Research*, 2004, **22**(1): 385−421

22 Zheng Qi, Jiang Sheng-Yi, Tang Yong. Applying probabilistic suffix tree to intrusion detection. *Computer Engineering and Applications*, 2010, **46**(23): 79−81
(郑琪, 蒋盛益, 汤庸. 概率后缀树在入侵检测中的应用研究. 计算机工程与应用, 2010, **46**(23): 79−81)

23 Yoon M K, Ciocarlie G F. Communication pattern monitoring: improving the utility of anomaly detection for industrial control systems. In: The Workshop on Security of Emerging Networking Technologies. San Diego, CA, USA, 2014.

24 Varga A. The OMNET++ discrete event simulation system. 2001.

25 Goh J, Adepu S, Junejo K N, Mathur A. A dataset to support research in the design of secure water treatment systems. In: Proceedings of the 11th International Conference on Critical Information Infrastructures Security. Paris, France, 2016.

26 Secure Water Treatment [Online], available: http://itrust.sutd.edu.sg/research/testbeds/secure-water-treatment-swat/, September 2017.

张仁斌 合肥工业大学副教授. 2013 年于合肥工业大学获得博士学位. 主要研究方向为工业互联网安全, 物联网安全.
E-mail: hfutzrb@163.com
(**ZHANG Ren-Bin** Associate professor at Hefei University of Technology. He received his Ph. D. degree from Hefei University of Technology in 2013. His main research interest covers industrial internet security and IoT security.)

吴 佩 合肥工业大学硕士研究生. 主要研究方向为工业互联网安全. 本文通信作者. E-mail: dorademon@163.com
(**WU Pei** Master student at Hefei University of Technology. Her main research interest is industrial internet security. Corresponding author of this paper.)

陆 阳 合肥工业大学教授. 2002 年于合肥工业大学获得博士学位. 主要研究方向为物联网工程, 分布式控制系统.
E-mail: luyang.hf@126.com
(**LU Yang** Professor at Hefei University of Technology. He received his Ph. D. degree from Hefei University of Technology in 2002. His research interest covers IoT engineering and distributed control system.)

郭忠义 合肥工业大学教授. 2008 年于哈尔滨工业大学获得博士学位. 主要研究方向为偏振智能信息处理, 先进光通信技术, 复杂电磁环境.
E-mail: guozhongyi@hfut.edu.cn
(**GUO Zhong-Yi** Professor at Hefei University of Technology. He received his Ph. D. degree from Harbin Institute of Technology in 2008. His research interest covers the polarization based intelligent information processing, advanced optical communication, and complex electromagnetic environment.)