一种结合包检测与流检测的 SECS2 流量识别方法

唐 璇 严 明* 万仕贤
(复旦大学计算机科学技术学院 上海 200000)

摘 要 要对网络数据包所采用的应用层协议进行识别,保证半导体生产环境的安全,使用传统的基于服务端口和特征字的识别方式都具有一定的局限性,无法达到所需的准确度。针对这种情况,提出一种基于 HSMS(High Speed Message Services)头部信息和 SECS2 数据本身固定模式的识别模型,结合深度包检测、深度流检测、机器学习等技术对 SECS2 流量进行识别。实验结果表明,该模型能有效地识别 SECS2 数据包,误判率仅为 0.598 8%,相比传统识别方式,误判率降低了 29.469 6%。

关键词 SECS2 HSMS 流量识别 深度包检测 深度流检测

中图分类号 TP3 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2024.09.019

A SECS2 TRAFFIC IDENTIFICATION METHOD COMBINING PACKET INSPECTION AND FLOW INSPECTION

Tang Xuan Yan Ming* Wan Shixian
(*School of Computer Science, Fudan University, Shanghai 200000, China*)

Abstract In order to identify the application layer protocol used in the network packet, to ensure the security of the semiconductor production environment, the traditional recognition methods based on server port and feature words have certain limitations and cannot achieve the required accuracy. In view of this situation, a recognition model based on HSMS header information and the fixed pattern of SECS2 data is proposed, which combined deep packet inspection, deep flow inspection, and machine learning to identify the SECS2 traffic. Experimental results show that this model can effectively identify SECS2 packets, and the misjudgment rate is only 0.598 8%, which is 29.469 6% lower than the traditional identification method.

Keywords SECS2 HSMS Traffic identification Deep packet inspection Deep-flow inspection

0 引 言

随着计算机与工业技术的融合发展,传统的工业制造领域逐渐从人工机台操作转向自动化机台操作,一系列符合工业控制规范的通信协议也应运而生。为了减少工控系统[1]接入互联网的难度,减少工控网络和通信网络产生数据通信冲突的可能性,越来越多的工业控制系统软件选择采用统一的通信协议和标准,这也就意味着工控网络被恶意攻击的风险加剧,需要更加严格精密的工控网络流量分析技术实现对生产网络的监控,以针对性地保护工控业务的安全。

在高度自动化的半导体制造厂中,通常使用计算机集成制造系统 CIMS(Computer Integrated Manufacturing System)[2]来控制并监控半导体产品生产的各个阶段,实时了解机台的工作状况并调整,以求最低的过程失误。而在半导体制造过程的各个时期中,由于设备具有个体性差异,CIMS 的自动化管理难度也随之增大。另外,不同厂商生产的设备之间并没有共同的通信协议,如果使用不同厂商制造的设备,半导体公司需要自行建立软件连接,导致了巨大的经费消耗。基于此,SEMI(Semiconductor Equipment and Materials Institute)

收稿日期:2021-04-14。2019 年工业互联网创新发展工程项目。**唐璇**,硕士生,主研领域:网络空间安全。**严明**,工程师。**万仕贤**,硕士生。

制定了半导体设备通讯标准接口 SECS(Semiconductor Equipment Communication Standard),以此指定了 CIMS 与设备之间的标准化接口,设备制造商只需要提供符合 SECS 标准的设备,就能使其快速地整合进 CIMS 当中,缩短了设备配置时间及成本,SECS 协议也在半导体制造业中被广泛使用。

随着计算机系统在半导体制造厂中的应用,半导体生产网络环境的安全问题不容忽视,但现在普遍缺乏针对半导体生产网络的流量检测机制,若面临安全威胁,则会造成巨大损失。要实现对特定网络流量的安全检测,需要对应用层协议[3]内容进行解析,并对解析出的信息进行分析,当发现不合法数据时发出警告,进而实现对生产过程的监控,故识别出网络数据包所采用的应用层协议是实现协议解析的基础。当前对网络通信协议的识别主要使用两种方式:基于服务端口的识别方式和基于包头特征或关键字的方式。

基于服务端口的识别方式[4],主要针对协议所使用的特征端口号,对协议进行识别,例如,在 HTTP 规范中,HTTP 协议通常使用 80 端口作为通信端口,若使用该识别方法对 HTTP 协议进行识别,仅需检查数据包的通信端口字段是否为 80 端口,即可完成对 HTTP 协议的识别。但基于端口的识别方式原理较为简单,有一定的局限性,有些 HTTP 服务程序未必使用 80 端口,有些无关软件也可能恶意占用 80 端口。此外网络上的有些应用软件采用复杂的通信机制,可能会使用动态随机化的端口,或应用镜像端口以保护应用进程的安全,这使得基于服务端口的识别精度大打折扣,已经无法满足准确识别的需求。基于包头特征或关键字的识别方式,主要针对应用层协议携带的具有鲜明特征的头部,对协议进行识别。例如 SMTP 协议,在其头部具有 EHELO、MAIL FROM、RCPT TO 等指令,可利用这些特征对其进行识别,这种方式有效地解决了基于端口识别方式带来的局限性。但在实际的网络数据包传输中,含有应用层头部的数据包占比较少,当进入大段数据负载传输时,明显的协议特征已不复存在,这使得基于头部特征的识别方式在单包识别模式下的效率及准确率也并不理想。

在半导体制造环境中,机台众多,虽然习惯上将 5000 端口作为 HSMS 协议上 SECS 通信的服务端口,但实际上协议服务端口可任意指定。此外,有些其他软件也采用 5000 端口作为服务端口,例如很多木马软件就采用 5000 端口。以上两种方式的局限性针对 SECS 协议依然存在,若使用传统方式,可能无法捕获自定义端口的 SECS 通信数据,甚至误把其他数据当成 SECS 数据进行解析,对整个生产环境造成安全隐患,因此对于 SECS 协议的自动协议识别是亟待解决的问题。

我们将 SECS 通信端口设为 5000 端口,并对数据包进行抓取,在不使用端口作为条件进行识别情况下,使用 Wireshark 对已知的 SECS 通信数据进行识别,Wireshark 将 SECS 协议误判为 RSL 协议,出现的误判情况如图 1 所示。

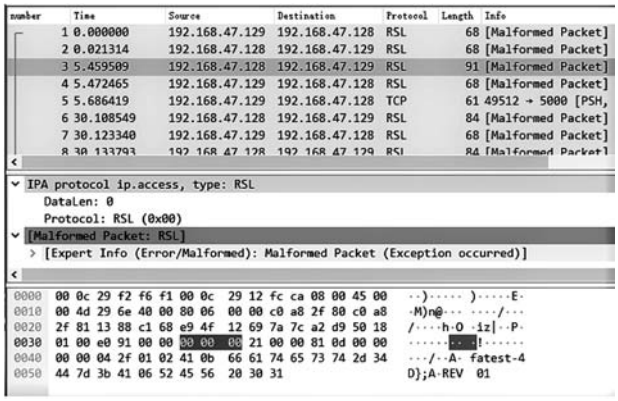

图 1 Wireshark 误判情况

除了端口和包头,SECS 协议自身具有一定的格式,考虑到这种格式可以作为协议识别的依据,为识别提供较高的可能性,本文充分利用 SECS 协议自身所具备的特性,提出对于 SECS2 数据包的流量识别模型,在不依据端口号的情况下,对于 SECS2 数据包的 HSMS 头部和数据片段进行识别,并对模型进行实验测试。

1 SECS2 数据包

1.1 HSMS 报文

SECS2 作为应用层协议,其类型数据通常被封装在 TCP 协议的 Message 当中进行传输,而在 TCP 协议封装的 Message 中,HSMS 报文作为头部出现在 SECS2 数据内容的开始阶段,当 Message 很长时,则需要进行分包传输。分包之后,只有靠前的一个或几个数据包中含有 HSMS 头部信息,紧随其后的其他数据包中,均为 SECS2 数据内容,即若能够检测出 HSMS 头部的存在,即可基本判定该数据包及其后的数据包均为 SECS2 数据。

HSMS 报文格式如图 2 所示。报文的消息长度占 4 个字节,信息头部占 10 个字节,共为 14 字节。其中,信息头部包含 6 部分信息,分别为 Session ID、Header Byte 2、Header Byte 3、PType、SType、System Byte。以下分别对这 6 部分信息进行说明。

图 2　HSMS 报文格式

Session ID:占 2 字节,为 16 位无符号整数,用来关联控制信息[5]以及其后的数据信息。当 Session ID 的值为 0 时,表示此条数据为发送数据信息或建立通信的第一个 Request;当 Session ID 的值为 1 ~ 65 535 时表示通信已建立,此 Session ID 由返回的 Response 产生,后续的控制信息都会使用这一非零的 Session ID,以标识同一条通信所产生的数据信息。

Header Byte 2:占 1 字节,用来表示 HSMS 协议不同阶段的消息特征。当 Header Byte 2 的值为 0 时,表示此条数据为控制信息,不包含数据负载;而对于数据信息,Header Byte 2 中使用 W-bit 来表示是否需要回复,剩下的 bit 用来体现 SECS Stream 信息。

Header Byte 3:占 1 字节,同样用来表示 HSMS 协议不同阶段的消息特征。当 Header Byte 3 的值为 0 时,表示此条数据为控制信息,不包含数据负载;而当其不为 0 时,表示此条数据为数据信息,用来体现 SECS Function 信息。

PType:占 1 字节,为 8 位无符号整数。用来表示消息头部和消息文本的编码类型,通常设计中其取值为 0,表示编码类型为 SECS-II。

SType:占 1 字节,为 8 位无符号整型数。用于表示消息类型为控制信息或数据信息。当取值为 0 时表示数据信息,否则表示控制信息。会话类型不同取值表示的消息类型如表 1 所示。

表 1　SType 取值与消息类型关系表

取值	表示消息类型
0	Data message
1	Select request
2	Select response
3	Deselect request
4	Deselect response
5	Link test request
6	Link test response
7	Reject request
9	Separate request

System Byte:占 4 个字节,在整个通信过程中用来标识特定的传输阶段,每条数据信息和控制信息的 System Byte 必须互不相同。但对于一条回复信息必须和其请求信息有相同的 System Byte。

1.2　SECS2 报文

在半导体制造环境中,机台众多,协议端口可任意指定,而对于无固定 TCP 端口的应用层协议,应用层的头部一般位于连接或交互会话的开始阶段,而协议最鲜明的特征在应用层协议的头部,例如 HTTP 协议可根据 GET、POST 操作指令为特征识别,SMTP 协议可根据 EHELO、MAIL FROM、RCPT TO 等指令为特征识别。而当传输应用层协议的数据负载时,明显的协议特征已经不存在,即若对于随机截获的一段数据包,很有可能并不具有明显的协议特征。

对于在现实的网络流量中随机截取 SECS2 数据包,很大概率上截取到数据负载,而无法通过识别 HSMS 头部进行 SECS2 数据的识别,对于截取到的这种数据包,我们通过对 SECS2 数据字段格式的识别,来判断该数据包是否可能为 SECS2 数据包。

SECS2 协议数据字段格式定义如图 3 所示。数据的第一个字节(8 bits)表示控制信息,控制信息后为数据负载。其中,控制信息主要标识两个信息:后续数据负载的数据类型(3 ~ 8 bits);后续数据负载长度信息所占的字节长度或数据元素数量(1 ~ 2 bits)。值得注意的是,数据之间可能存在嵌套关系。

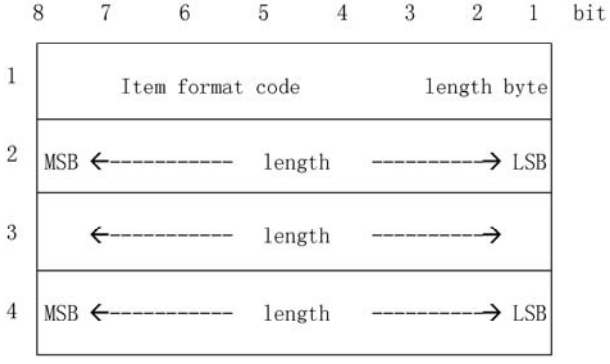

图 3　SECS2 协议数据字段格式定义

以下通过一个例子说明 SECS2 数据字段格式。

Eg. bit　87654321

00000001　List

00000011　Have 3 elements

00100001　First element: Binary Item

00000001　1 byte long

00000100　Alarm set, category 4

01100101　Second element: 1-byte integer Item

00000001	1 byte long
00010001	Alarm 17
01000001	Third element: ASCII Item
00000111	7 characters
01010100	ASCII T
00110001	ASCII 1
00100000	ASCII space
01001000	ASCII H
01001001	ASCII I
01000111	ASCII G
01001000	ASCII H

第一个字节为控制信息,3 ~8 bits 为$(000000)_2 = (00)_8$,数据类型枚举值及其对应关系如表 2 所示,依据表 2 所知,表示数据类型为 List,1 ~2 bits 为$(01)_2 = (1)_{10}$,表示后面的一个字节长度为数据负载的长度信息或个数信息。即第二个字节$(00000011)_2 = (3)_{10}$表示该 List 含有 3 个元素。

表 2 数据类型枚举值及其对应关系

Format Code(Binary/Octal) Bit 876 543	Meaning
000 000/00	List
001 000/10	Binary
001 001/11	Boolean
010 000/20	ASCII
010 001/21	JIS-8
010 010/22	2-Byte character
011 000/30	8-Byte integer(signed)
011 001/31	1-Byte integer(signed)
011 010/32	2-Byte integer(signed)
011 100/34	4-Byte integer(signed)
100 000/40	8-Byte floating point
100 100/44	4-Byte floating point
101 000/50	8-Byte integer(unsigned)
101 001/51	1-Byte integer(unsigned)
101 010/52	2-Byte integer(unsigned)
101 100/54	4-Byte integer(unsigned)

第一个元素首字节的 3 ~8 bit 为$(001000)_2 = (10)_8$,数据类型为 Binary,第二个字节为$(00000001)_2 = (1)_{10}$表示该数据长度为 1 Byte,即为第三个字节的值 00000100。

第二个元素首字节的 3 ~8 bit 为$(011001)_2 = (31)_8$,数据类型为 1-Byte integer,第二个字节为$(00000001)_2 = (1)_{10}$表示该数据长度为 1 Byte,即为第三个字节的值$(00010001)_2 = (17)_{10}$。

第三个元素首字节的 3 ~8 bit 为$(010000)_2 = (20)_8$,数据类型为 ASCII,第二个字节为$(00000111)_2 = (7)_{10}$表示包含 7 个 ASCII 数据,即为随后 7 个字节的值,分别为"T""1"" ""H""I""G""H"。

综上,以上表示一个含有 3 个元素的 List,分别是一个 binary 数据、一个 integer 数据、一串 ASCII 数据"T1 HIGH"。

当前 SECS 数据包普遍采用 GEM/SECS2/HSMS/TCP/IP/Ethernet 的方式进行封装,其通信传输有以下特性:(1) 通信双方端口可自定义;(2) 协议本身的原语没有明显的客户端、服务端的区分,即机台也可以主动向 EAP 主机发送信息;(3) HSMS 头部信息可能会被分割在两个 IP 包中,导致数据传输中,每一个数据包都不含有完整包头;(4) HSMS 包头具备的特征不明显,若采用基于包头关键字的识别方式,存在误判的可能性;(5) 当进行 HSMS 通信会话时,HSMS 包头仅存在于第一或第二个数据包当中,后续传输 SECS2 数据的数据包均没有 HSMS 头部。

由于以上特性,若采用传统识别方式,如基于服务端口的识别方式和基于包头特征、关键字的方式,存在漏判、误判等可能性,可能会导致识别效率及准确率都十分低下。但 SECS2 数据本身具有其固定模式,此模式为 SECS2 数据识别提供了可能性,而目前的协议解析类工具并未很好利用其数据模式,本文将充分利用 SECS2 数据自身的模式特性,实现对 SECS2 数据的高效精准识别。

2 识别模型

在本识别模型中,将服务端口排除在识别的因素之外,主要基于 SECS2 数据的 HSMS 头部信息和 SECS2 数据本身的固定模式,对其进行识别,识别过程基于深度包检测(Deep Packet Inspection,DPI)和深度流检测(Deep Flow Inspection,DFI)相结合的方式。

深度包检测技术[4]是一种对应用层数据进行流量识别和特征检测技术,在检测过程中,可对应用层数据进行深度解码,并对数据包载荷部分的特征字段进行检测。该技术包含基于特征字、应用层网关、行为模式

的识别技术,分别根据不同协议的特殊指纹信息(特定字符串、特定的 Bit 序列等[6])、业务流、实施的行为动作等,对协议进行识别。在本识别模型中,我们考虑将深度包检测技术运用于对 HSMS 头部特征字段以及 SECS 数据包载荷的识别中,利用 HSMS 头部和 SECS 数据本身具有的特殊模式和规律,对数据包进行识别判定。

深度流检测技术[7-8]是一种对网络数据流宏观行为特征进行分析识别的技术,其主要对大量样本进行标记,并利用机器学习技术进行分类模型[9]的训练,最终达到利用网络流量的平均数据包长度、数据包到达时间间隔等宏观特征对网络流量进行分类,深度流检测技术[8]无须对数据包进行解包和载荷信息提取,识别速度较快。本识别模型中,主要利用深度流检测技术,对数据包进行宏观辅助识别[10],以提高识别结果的精确度和可信度。

本模型主要分为两个子模型,分别为 HSMS 头部信息识别模型、SECS2 数据负载识别模型,接下来对模型进行整体介绍,并分别介绍两个子模型。

2.1　总体设计

模型使用 HashMap 存储会话连接的信息和状态,设计 HashMap 节点的数据结构,以数据包的五元组(源 IP 地址,源端口号,目的 IP 地址,目的端口,会话 ID)进行两字节异或的方式作为 HashKey,在 HashMap 中存储会话连接信息和状态。当未知数据包进入模型,先对其进行预处理,在 HashMap 中查找是否有这条连接的信息,如果没有,则新建一个节点,对这条连接进行存储。如果有,查看其状态,当出现下列情况,可以判定后续数据包为 SECS2 数据包:(1) 前面的数据包中,已经拿到了 HSMS 头部,且连接还未关闭;(2) 前面的数据包已经被判定为 SECS2 数据段,且连接还未关闭。如若不属于以上两种情况,则无法根据之前的识别进行该未知数据包的判别,进入到正式的识别模型中。

首先进入 HSMS 头部信息识别子模型,对数据包进行 HSMS 头部的检测,如果检测到 HSMS 头部,则可以判定该数据包为 SECS2 数据包,否则进入 SECS2 数据负载识别子模型,对数据包进行进一步的识别,检测数据包的格式是否符合 SECS2 标准。

识别过程结束以后,对该数据包进行综合评估,最后输出识别结果,并对该条会话连接状态进行更新。总体的设计模型结构如图 4 所示。

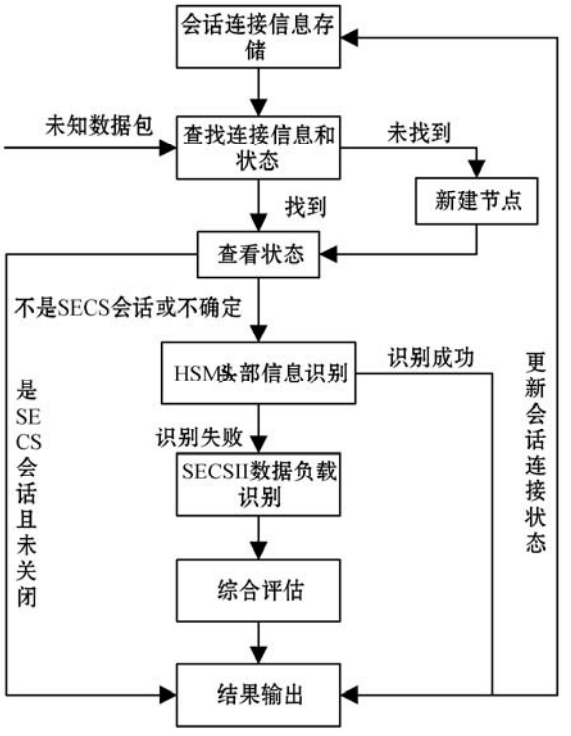

图 4　模型结构总体设计

2.2　HSMS 头部识别模型

在传统的识别方法中,通常对端口号进行识别,若指定端口号,对该服务端口数据,通常采用关键字识别的方法,对数据包的 HSMS 头部对应位置数据进行识别,并判定其是否为 HSMS 头部数据。以 Wireshark 的识别逻辑为例,当我们指定某个服务端口为 HSMS 协议通信的情况下,Wireshark 将会调用 HSMS 解析插件对该端口数据进行解析,其解析核心代码如下:

```
if (tvb_reported_length(tvb) < HSMS_MIN_LENGTH) return 0;
if ((tvb_get_ntohl(tvb,0) + 4) != tvb_reported_length(tvb)) return 0;
sessionId = tvb_get_ntohs(tvb,4);
Byte2 = tvb_get_guint8(tvb,6);
Byte3 = tvb_get_guint8(tvb,7);
pType = tvb_get_guint8(tvb,8);
sType = tvb_get_guint8(tvb,9);
if ((sType ==8) || (sType ==10) || (sType >127)) return 0;
switch (sType)
{
case STYPE_SECS_DATA:
if (Byte2 ==0) return 0;
  break;
case STYPE_SELECT_REQ:
case STYPE_DESELECT_REQ:
case STYPE_SEPARATE_REQ:
if ((Byte2 !=0) || (Byte3 !=0)) return 0;
if (tvb_reported_length(tvb) >HSMS_MIN_LENGTH) return 0;
```

```
    break;
  case STYPE_SELECT_RSP:
  case STYPE_DESELECT_RSP:
  if (Byte2! =0) return 0;
  if (tvb_reported_length(tvb) >HSMS_MIN_LENGTH) return 0;
    break;
  case STYPE_LINKTEST_REQ:
  case STYPE_LINKTEST_RSP:
  if (sessionId! =0xFFFF) return 0;
  if (Byte2! =0) return 0;
  if (Byte3! =0) return 0;
  if (tvb_reported_length(tvb) >HSMS_MIN_LENGTH) return 0;
    break;
  case STYPE_REJECT_REQ:
  if (tvb_reported_length(tvb) >HSMS_MIN_LENGTH) return 0;
  break;
  }
```

以上代码主要为 Wireshark 针对指定端口数据包的开始 14 字节进行 HSMS 头部判断。首先进行数据包长度的判断,当数据包长度小于 HSMS 最小长度时,排除该数据包具有 HSMS 头部的可能性,之后进行消息长度和数据包长度的比较,当两者不相等时,排除该数据包具有 HSMS 头部的可能性,当两者相等时,将对应位置上的值分别取出作为 Session ID、Header Byte2、Header Byte3、PType、SType 的值。对取出的 SType 的值进行判断,SType 的值通常取 0 ~ 7、9,不使用 8、10 和大于 127 的值,11 ~ 127 可能会在辅助标准中用到。故当 SType 的值取到 8、10 或大于 127 时,证明判断有误,排除该数据包具有 HSMS 头部的可能性。当 SType 的值在合理范围内时,结合 SType 不同的值对应的消息类型,对数据包进行识别,SType 取值与消息类型的对应关系见表 1。

当 SType =0,表示该条信息为数据信息,此时若 header Byte2 =0,表示此条信息为控制信息,则识别错误,排除该数据包具有 HSMS 头部的可能性。SType = 1,3,9 时,表示该条为控制信息,此时 header Byte 2&3 应当都等于 0,如果不是则判定为非 HSMS 数据;此时若数据长度大于 HSMS 最小长度,则说明该数据包不是控制信息,含有数据负载,也判定为非 HSMS 数据。当 SType = 2,4 时,表示该条信息为回复信息,header Byte 2 必须为 0,若不是则判定为非 HSMS 数据;此时若数据长度大于 HSMS 最小长度,则说明该数据包不是控制信息,含有数据负载,也判定为非 HSMS 数据。当 SType = 5,6 时,若出现 Session ID 不为最大值 0xFFFF、header Byte 2 不为 0、header Byte 3 不为 0,或者数据长度大于 HSMS 最小长度的情况,均判定该数据包不含 HSMS 头部。当 SType = 7 时,若数据长度大于 HSMS 最小长度,则说明该数据包不是控制信息,含有数据负载,也判定为非 HSMS 数据。

在 Wireshark 的识别逻辑中,主要是根据 HSMS 协议在应用中的规律,对 HSMS 包头的关联取值进行判断,试图避免将其他数据包误判为含有 HSMS 头部的情况。这种判断方式不但需要在识别之前进行端口号的指定,且识别逻辑中依然含有漏洞,存在误判的可能性。例如,当截获的消息长度未达到 HSMS 头部长度字段时,即数据包截获不完整时,将产生漏判;当数据包前 4 字节所代表的整数值恰好等于当前截获的消息长度,第 7 字节不为 0,第 10 字节为 0 时,将会被误判为含有 HSMS 头部的数据包,其类型为 Data Message,这种可能性虽然非常小,但依然存在。

在本模型中,当对未知数据包进行了初步判定,认为该数据连接历史信息不足以对此数据包做出判定时,进入 HSMS 头部信息识别子模型。HSMS 识别模型中,利用机器学习的方式,对已经标注的具有 HSMS 头部的 SECS2 协议数据包集进行学习。提取出各会话阶段的 HSMS 特征字符,训练样本集,提取各特征字的可能取值范围、HSMS 头部可能所在位置及其长度信息,生成可持久化存储特征库。另外,提取出平均数据包长度、数据包到达时间间隔等宏观特征,作为深度流检测的判定参数,在流层面对数据进行辅助识别,将提取的参数分别记作 $P_1, P_2, \cdots, P_n$,利用机器学习结果分别计算出以上参数的特征阈值,分别记作 $\beta_1, \beta_2, \cdots, \beta_n$。

当读入未知包,对其进行预处理,根据数据链路层中的数据链路类型,判断数据链路层协议,之后向上提取网络层数据,若判定为 IP 协议报文则继续向上提取,判定其是否为 TCP 协议报文,若确为 TCP 协议报文,则将其载荷提取出来,按照 HSMS 头部所在相对位置进行 HSMS 头部的识别;否则认为该包不可能为 SECS2 数据包,直接输出结果。

当在未知包中提取出了 TCP 协议载荷,根据 HSMS 头部格式对其对位进行判定,并对判定结果进行累计,另外,对各辅助参数进行计算,并与这些参数的特征阈值 $\beta_1, \beta_2, \cdots, \beta_n$ 进行比较,对比较结果进行累计。若对位数据均符合格式要求,在合理范围内,且辅助识别结果均达到阈值,认为该数据包具有 HSMS 头部,必为 SECS2 数据包,输出结果;若有数据不符合格式,或辅助参数没有达到阈值,认为该数据包不能认定为 HSMS 头部或为具有 HSMS 头部的数据包,进入数据载荷识别,对其进一步进行判定。

HSMS 头部信息识别模型如图 5 所示。

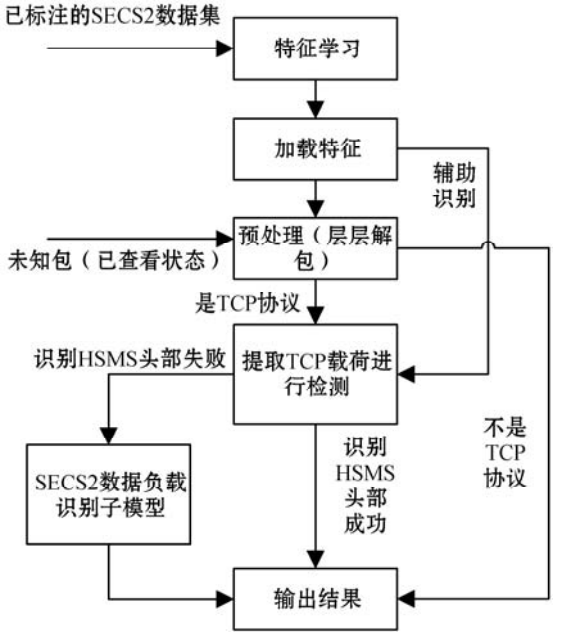

图 5　HSMS 头部识别

值得注意的是,在本轮识别中,对 HSMS 头部的识别应当严格,当某数据包对应位置、流层面参数等均满足具有 HSMS 头部的条件时,即能够确信数据包具有 HSMS 头部时,才将其直接判定为 SECS2 数据包,否则均将其输入到下一步识别模型中,进行进一步判定。而对于进入到下一轮识别的数据包,将其本轮识别综合判定结果权值一同输入下一轮子识别模型,作为一个辅助识别条件,影响最终的综合判定结果,尽可能提高模型整体的识别准确率。

对比传统的识别方式,本文方法为了进一步降低误判率,不仅考虑以上 HSMS 头部的基本合规准则,还将结合其在实际运行过程中的各字段的常规取值来综合判定数据包是否存在 HSMS 头部,并结合机器学习的方式,对数据包的宏观特征进行辅助识别,提高识别的准确性和可信度。

2.3　SECS2 识别模型

当数据包不能认定为 HSMS 头部或为具有 HSMS 头部的数据包时,HSMS 头部信息识别失败,进入 SECS2 数据负载识别模型。截取未知数据包的数据载荷(PAYLOAD),对单字节逐个扫描,寻找第一个满足下列条件的 Byte,即其值属于 SECS2 类型字段枚举值,将此 Byte 作为判定切入点,将切入点字节作为类型字节 Type_Byte1,提取类型值 Type 以及长度字节数 $L_{\text{en\_Bytes1}}$。根据长度字节数 $L_{\text{en\_Bytes1}}$ 提取后续的 $L_{\text{en\_Bytes1}}$ 个字节作为长度值 l_{ength}。根据类型值 type,判断是否属于 ASCII 类型,如果是 ASCII 类型,则判断后续跟随的 l_{ength} 个字节是否为一个 ASCII 字符串;否则根据长度值 l_{ength} 跳过相应字节,提取下一个类型字节 Type_Byte2,继续判断该字节是否属于协议限定的枚举值范围。如果 Type_Byte2 属于枚举值范围,则将该字节作为下一个切入点继续分析后续数据,如此跳跃检测直至边界或跳出边界;若某次跳跃后到达某字节不是应当出现的枚举值,说明之前的切入点 Byte 有误,跳回切入点 Byte 的后一位 Byte 并继续寻找。若跳跃到边界或跳出边界依旧未发现一段数据符合设定的 SECS2 规律,则认为该数据段基本不可能属于 SECS2 数据。SECS2 数据负载识别模型结构如图 6 所示。

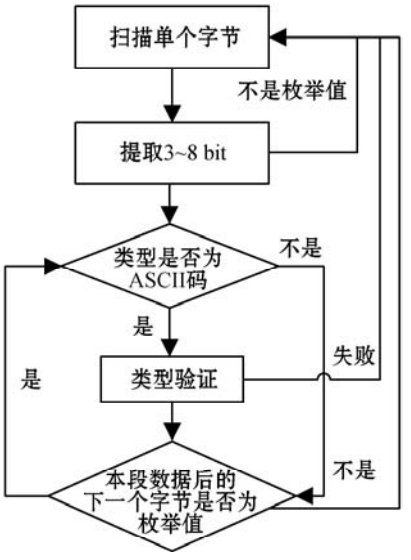

图 6　SECS2 数据负载识别模型

对符合 SECS2 规律的数据进行占比统计,并进行权值计算。调研设计二维的概率函数,便于综合进行二维结果的评定——A 为能够判定为 SECS2 数据的部分占 PAYLOAD 部分的比例;B 为识别过程中取得的权值。由于这两个维度的特征,对概率函数有下列要求:

当 A 较大 B 较小时,概率设置不应过高,以排除是包的长度过短,导致并没有太多数据满足 SECS2 数据格式,却显示了较多的占比的情况。

当 A 较小 B 较大时,概率设置不应过高,以排除是包的长度过长,导致将很多无关数据误判为 SECS2 数据,显示了较高的权值的情况。

以上两种情况,当 A、B 的值差距越大时,概率应当越低,差距越小,在一定范围内,概率越高。

当 A 较小 B 较小时,概率设置不应高,显然两个维度都未合格。

当 A 较大 B 较大时,应该将其判定为高概率 SECS2 数据包。

3　实验及分析

3.1　实验方案

为了让实验结果有较高的可分析性,在实验前,我们需要构建实验需要的数据包集合。

首先不进行 SECS 通信,对背景数据包进行抓取,并对其进行过滤,去掉背景数据包中端口号为 5000 的数据包,剩下的数据包应为非 5000 端口的非 SECS2 数据包。然后打开模拟软件进行 SECS 通信,将通信端口号设置为 5000,进行 SECS2 数据包的抓取。最后将两次抓取的数据包混合在一起,作为实验识别使用的数据包集合。在此集合中,当端口号为 5000 时,必为 SECS2 数据包,否则必不是 SECS2 数据包,当将端口号为 5000 的数据包判别为非 SECS2 数据包时,和将端口号为非 5000 的数据包判别为 SECS2 数据包时,将此次判别认定为误判。因为本实验中流量识别代码并未将 5000 端口作为识别依据,因此实验结果可以反映不依赖服务端口的识别准确率。

构建完数据包集合后,使用模型对该集合中的数据包进行一一识别,并对识别结果和端口号进行输出,作为第一组的数据结果。另外设置第二组为对照组,使用传统的 Wireshark 对此数据包集合进行识别,对结果进行记录,作为第二组的数据结果。

对两组数据结果进行分析,比较两组识别方式的误判率;对模型的误判率做出推算,并比较推算与实际的误判率。基于三者误判率的结果,对识别模型和实验做出总结。

3.2 误判率分析

结合模型的识别判定过程,我们对模型的误判情况进行分析,并对误判率做出粗略的推算。我们认为误判主要分为两大类情况:将 SECS2 数据包误判为非 SECS2 数据包;将非 SECS2 数据包误判为 SECS2 数据包。分别称它们为情况 A 和 B,接下来分别对 A、B 情况进行分析。

当情况 A 发生,有如下几种可能场景:

(1) 数据包长度短且具有 HSMS 头部。由于定位失误,导致进行错位判别,HSMS 头部识别出现误判,数据包进入数据负载识别,而由于其长度太短,综合评定时,数据占比较高,而权值过低,有可能会在此环节再次出现误判。

(2) 数据包长度短且不具有 HSMS 头部。数据包进入数据负载识别可能会出现误判,理由同(1)。

当情况 B 发生,有如下几种可能场景:

(3) HSMS 头部误判。数据包对应位置碰巧符合 HSMS 头部格式,从而通过识别误判为 SECS2 数据包,由于数据包头部位置数据一般有其含义,包含数据信息,排除人为构造,自然情况下非 SECS2 数据包的数据符合了 HSMS 头部格式的可能性非常小,可以忽略不计。

(4) 数据部分误判。当数据包进入数据负载识别模块,对单字节进行读取识别,若数据包长度较长,一般不会产生误判,而当数据长度较短时,是可能出现误判的,理由同(1)。

在以上分析中,我们忽略了数据分片、漏包等问题的存在,而现实情况下这些都是可能出现的,有如下几种可能场景:

(5) 若出现数据分片,相邻的两个包在特定情况下,前面的包有头部无数据负载或有少量数据负载,后面的包无头部有数据负载。对前面的包:若头部出现误判,无数据负载,则数据包误判;若头部出现误判,有少量数据负载,则同情况(1)出现误判;若头部出现误判,有足够长的数据负载,则进入数据判别阶段,不易出现误判。对后面的包:当其数据负载过短,可能会出现误判,而当其负载足够长,则不易出现误判,理由同上。

(6) 漏包情况。我们对单个包进行检测,一般情况下,漏包并不影响判定结果。

综上六种可能情况所述,当出现误判情况,主要是由于数据包过小导致的,其他情况下,误判概率较小,均可以忽略不计。我们使用两台机器模拟生产环境中的机台通信,并对数据包进行抓取,对抓取到的数据包进行长度统计,发现长度及其占比大致情况如表 3 和图 7 所示。

表 3 数据包长度及占比

数据长度范围	计数	占比/%
0 ~ 19	0	0
20 ~ 39	0	0
40 ~ 49	32	2.021 5
50 ~ 59	239	15.097 9
60 ~ 69	512	32.343 7
70 ~ 79	126	7.959 6
80 ~ 159	427	26.974 1
160 ~ 319	224	14.150 3
320 ~ 639	3	0.189 5
640 ~ 1279	16	1.010 7
1 280 ~ 5 119	2	0.126 3
5 120 and greater	2	0.126 3
总计	1 583	100

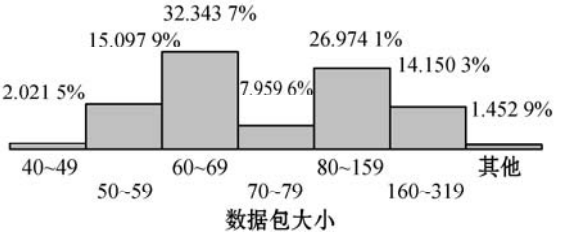

图 7 数据包长度及占比情况

如果认为长度低于 50 Byte 的数据包为短包,则在生产环境下的短包大约占比为 2.021 5%,假设通过模型的短包全部误判,则推算产生的模型的误判率应为 2.021 5%,现实情况下,误判率应当低于推算误判率,即低于 2.021 5%。

3.3　实验结果

对实验数据进行统计,结果如表 4 所示。

表 4　实验结果

识别方式	模型识别	Wireshark 识别	推算分析
Port = = 5000&true	698	0	—
Port ! = 5000&false	1 626	1 635	—
Port = = 5000&false	5	703	—
Port ! = 5000&true	9	0	—
误判率/%	0.598 8	30.068 4	2.021 5

实验中我们发现,当在 Wireshark 中不使用 5000 端口作为 SECS2 数据的判断依据时,Wireshark 无法识别 SECS2 数据,无论其是否具有 HSMS 头部。传统的识别方式误判率高达 30.068 4%,而本文模型识别误判率仅为 0.598 8%。

通过表 4 的数据,可以看出误判率满足以下关系:模型真实误判率 < 推算误判率 < Wireshark 误判率。推理正确。

4　结　语

本文提出对于 SECS2 数据包的流量识别模型,在不依据服务端口号的情况下,对于 SECS2 数据包的 HSMS 头部和数据片段进行识别,并对模型进行实验测试。模型识别结果与普通的 Wireshark 识别有较明显的改善效果,提升识别准确度,将误判率控制在一定范围内。

根据 Wireshark 的识别逻辑,当含有 HSMS 头部的数据包丢失,后续的数据包中,若不能检测到完整的 HSMS 头部,都将无法正常判定出 SECS 数据。我们对实验数据包进行实验,当去掉前面具有 HSMS 头部的数据包,单独抽取出不含 HSMS 头部的 SECS 数据包进行 Wireshark 识别,即使指出其端口号为 5000,该数据包依然没有被正确识别为 SECS 数据包,数据包原识别结果和误判结果对比如图 8 所示。在这种情况下,本文模型表现出了更高的识别精度,对于单个数据包,就算无法识别出 HSMS 头部信息,也可以尝试寻找到 SECS 数据负载部分,若有需要,可以在综合评估部分的数据占比统计时,提取出被认为是 SECS 数据的碎片数据段,也可进一步验证识别的准确性。

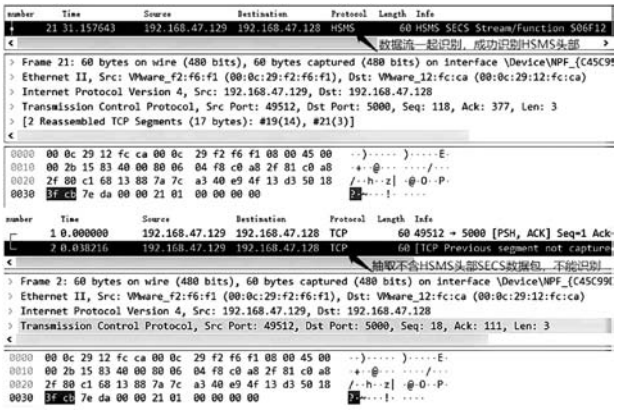

图 8　Wireshark 识别不含 HSMS 头部单包误判结果

为进一步提高识别准确率,使用深度流检测等技术时,对数据流的整体信息进行把控,设置更多辅助参数,如业务流持续时间、平均流速率等;还可以对更庞大的 SECS 数据进行训练,使得模型的识别准确率更高,最终是否能够将误判率降低到趋近于 0,需要在以后的工作中对模型进行改良,并验证模型的准确性和可用性。

参　考　文　献

[1] 董健,韩鹏军. 面向工控的协议解析方法应用研究[J]. 信息技术与网络安全,2021,40(2):1 - 6.

[2] 马继山. 基于非 SECS/GEM 的半导体生产远程监控系统[D]. 南京:东南大学,2018.

[3] 陈亮,龚俭,徐选. 应用层协议识别算法综述[J]. 计算机科学,2007,34(7):73 - 75.

[4] 王旭东,余翔湛,张宏莉. 面向未知协议的流量识别技术研究[J]. 信息网络安全,2019(10):74 - 83.

[5] 蔡乐,石荣,许都. 基于关联规则挖掘的未知协议特征提取方法[J]. 电子信息对抗技术,2016,31(6):18 - 23,57.

[6] 宋疆,张春瑞,张楠,等. 基于数据报指纹关系的未知协议识别与发现[J]. 计算机应用研究,2012,29(12):4604 - 4606,4614.

[7] 林平,余循宜,刘芳,等. 基于流统计特性的网络流量分类算法[J]. 北京邮电大学学报,2008,31(2):15 - 19.

[8] 雷东,王韬,赵建鹏,等. 面向比特流的未知协议识别与分析技术综述[J]. 计算机应用研究,2016,33(11):3206 - 3210,3250.

[9] Lu Q, Lin J, Su Z, et al. System design of network data classification based on deep packet inspection[J]. Journal of Physics: Conference Series,2021,1738(1):012118.

[10] Feng T, Man D, Fu H, et al. Unknown protocol identification based on improved K-Means ++ algorithm[J]. Journal of Physics: Conference Series,2020,1646(1):012023.