文章编号: 1001- 2486(2008) 03- 0070- 06

一种改进的事件驱动系统框架*

何鸿君[1], 曹四化[1,2], 褚祖高[2], 罗 莉[1], 宁京宣[1], 董黎明[1], 李 朋[1]
(1. 国防科技大学 计算机学院, 湖南 长沙 410073; 2. 75200 部队 自动化站, 广东 惠州 516001)

摘 要: 在给出事件驱动系统安全漏洞的基础上, 指出了产生漏洞的根本原因是: 系统忽视了输入系统的事件序列之间存在的相关性; 系统无条件信任任何事件源产生的事件。针对这两个原因, 相应提出了事件序列形式安全分析模型及基于事件源的可信度评估模型, 依据这两个模型, 构建了一种改进的事件驱动系统框架。

关键词: 信息安全; 事件驱动系统; 事件序列; 事件源; 可信度
中图分类号: TP309 文献标识码: A

An Improved Event-driven System Infrastructure

HE Hong-jun[1], CAO Si-hua[1,2], CHU Zu-gao[2], LUO Li[1], NING Jing-xuan[1], DONG Li-ming[1], LI Peng[1]
(1. College of Computer, National Univ. of Defense Technology, Changsha 410073, China;
2. Automation Station, No. 75200 Army Unit, Huizhou 516001, China)

Abstract: Based on the presentation of the security vulnerability of the event-driven system, this paper points out the deep reasons for the vulnerability of system: (1) The system ignores the inherent correlation among events; (2) The system trusts events from any sources without condition. In view of these two reasons, this paper presents an analytical model for security of event sequence, and an evaluation model of trustworthiness based on event source. Furthermore, this paper constructs an improved event-driven system infrastructure based on the two models.

Key words: information security; event-driven system; event sequence; event source; trustworthiness

事件驱动系统[1]是指这样一类操作系统, 系统中的对象通过消息传递机制相互通信, 消息来自于输入事件的转换。操作系统中存在一个系统级的事件调度器(Event Dispatcher), 负责消息的调度与分配。事件既可以来源于诸如鼠标、键盘等实际设备的输入, 也可以是系统中对象相互通信的数据。

事件驱动系统在给软件使用者带来便利的同时, 也带来了诸多的安全漏洞。文献[2-7]从不同层面指出事件驱动系统的安全漏洞主要有: DOS(Deny of Service)漏洞、对运行程序的可修改漏洞、未授权的访问、执行恶意代码、事件截获与分析等。文献[3]提供的测试表明, 在随机事件序列的测试下, Windows NT 应用程序往往会难以正常工作。产生上述漏洞的原因包括[2]: 目前事件驱动系统中不存在确保事件序列正常工作的机制; 不存在区分事件来源的机制。也就是说, 系统忽视了输入系统的事件序列之间存在的相关性, 并且无条件信任任何事件源产生的事件, 这使得输入系统的事件的可信性无法保证。

为解决系统忽视事件之间相关性的问题, 本文提出了基于事件约束关系的事件序列形式安全分析模型, 该模型保证进入系统的事件序列在形式上满足相关性要求; 为解决系统无条件信任任何事件源输入系统的事件的问题, 本文提出了基于事件源的事件可信度模型, 该模型可以有效评估不同事件源发起事件的可信度, 进而依照这两个模型, 构建了改进的事件驱动系统框架, 该改进框架可以有效改善系统消息队列的安全性与消息的可信性。

* 收稿日期: 2007- 11- 06
基金项目: 国家部委基金资助项目(9140C1102060707)
作者简介: 何鸿君(1968—), 男, 副教授, 博士。

1 事件驱动系统结构

事件驱动系统[1]结构是典型的C/S结构, 由事件发生器、事件服务器及事件消费者(客户方)三个主要部分组成, 整体结构如图1所示。事件发生器负责产生输入事件(数据), 比较常见的事件发生器有鼠标、键盘等, 系统中能产生原始输入事件(数据)的进程也可以是事件发生器。服务器负责接收原始输入事件(数据), 并根据系统需求将原始输入事件解释(编码)成相应的系统消息, 尔后, 将消息提交给客户方。客户方在事件驱动系统中是指消息的接收方, 负责消息的处理。

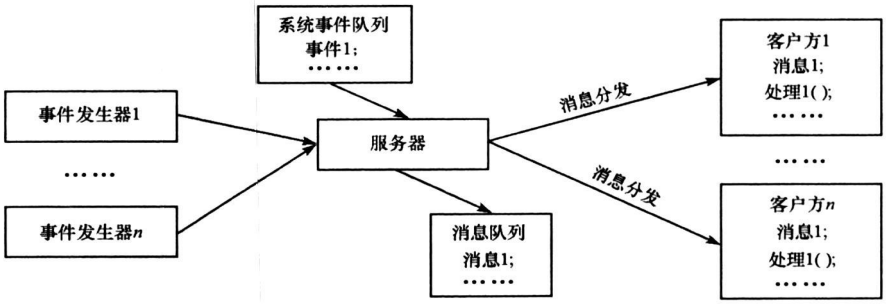

图1 事件驱动系统结构

Fig. 1 Structure of event-driven system

2 事件序列形式安全分析模型

事件驱动系统中, 某些事件之间存在固有的关联关系, 如果事件来自于正常的输入源, 则这种关联关系会自然地出现。然而, 一些恶意程序会故意向系统事件队列发送非正常关联关系的事件序列, 以达到破坏系统的目的。

2.1 事件序列形式安全分析模型

为方便起见, 用 e 表示事件发生器产生的任一输入事件, 用 E_s 表示系统中事件发生器产生的输入事件集合。

定义1 系统事件队列中按时间排序的若干个事件称为事件序列, 记为 E_L。$E_L = (e_1, e_2, \cdots, e_n)$, 其中 $e_i \in E_s$, $i = 1, 2, \cdots, n$。

定义2 令 $\oplus$ 为事件序列的"连接"运算符, Θ 为事件序列的"减法"运算符, 若 $E_L = (e_1, e_2, \cdots, e_m, e_{m+1}, \cdots, e_n)$, $E_{L1} = (e_1, e_2, \cdots, e_m)$, $E_{L2} = (e_{m+1}, \cdots, e_n)$, 则 $E_L = E_{L1} \oplus E_{L2}$, $E_{L1} = E_L \Theta E_{L2}$, $E_{L2} = E_L \Theta E_{L1}$, 称 E_{L1}、E_{L2} 为 E_L 的事件子序列。

定义3 事件 e 是约束的, 若事件 e 发生之后(前), f 必会(已)发生, 记为 $\chi(e, f) = \text{true}$($\chi(f, e) = \text{true}$)。称 f 为 e 的后(前)向匹配事件, e 为 f 的前(后)向匹配事件。否则, 事件 e 是独立的, 记独立事件为 e_d。

由定义3可以直接得到以下性质。

性质1 任意事件 e, e 要么是约束的, 要么是独立的。

定义4 E_s 的约束集 $\varphi(E_s) = \{(e_i, e_j) \mid e_i \in E_s, e_j \in E_s$ 且 $\chi(e_i, e_j) = \text{true}\}$。

定义5 设 e_i 是 E_L 的一个分量, 记 $d(e_i, E_L)$ 为事件序列 E_L 中 e_i 的约束距离。

若事件 e_i 是独立的, 则 $d(e_i, E_L) = 0$。

若 e_i 是约束的, 且存在 E_L 的分量 e_j 满足:

(1) $(e_i, e_j) \in \varphi(E_s)$, 或 $(e_j, e_i) \in \varphi(E_s)$;

(2) 事件序列 $(e_i, \cdots, e_j)$ 或 $(e_j, \cdots, e_i)$ 中, 不存在 e_m、e_n 满足 $(e_m, e_n) \in \varphi(E_s)$, 其中, $\min(i, j) \leqslant m < n \leqslant \max(i, j)$, 但不同时取"=";

则 $d(e_i, E_L) = j - i$, 称 e_j 为 E_L 中 e_i 的最近匹配事件;

否则, $d(e_i, E_L) = \infty$。

定义6 称序列 $E_L = (e_i, \cdots, e_j)$ 为约束序列, 若 $d(e_i, E_L) = j - i$ 且 $d(e_m, E_L) < j - i$, $i < m < j$。记约束序列为 $\complement(E_L)$。

性质2 若 $E_L = (e_i, \cdots, e_j)$ 为约束序列, 则 $e_{i+1}, \cdots, e_{j-1}$ 均是独立的。

证明 用反证法。

假设 $(e_{i+1}, \cdots, e_{j-1})$ 中的某分量 e_k 是约束的。

(1) 若 $d(e_k, E_L) = \infty$, 矛盾。

(2) 若 $d(e_k, E_L) \neq \infty$, 则必有 $d(e_k, E_L) \leqslant \max(j - k, k - i)$。于是:

要么存在 e_k、e_n, 满足 $(e_k, e_n) \in \varphi(E_s)$, $k < n \leqslant j$;

要么存在 e_m、e_k, 满足 $(e_m, e_k) \in \varphi(E_s)$, $i \leqslant m < k$。

由定义5, 有 $d(e_i, E_L) = \infty$。

矛盾。 □

关于事件序列的常识是: 若事件序列是安全的, 则对事件序列中任一约束事件 e 而言, 必须有相应的匹配事件出现。把这一常识表述为如下公理。

公理1 事件序列 E_L 的独立事件分量不影响 E_L 的形式安全性。设 e_i 是 E_L 的约束分量, 若 E_L 中存在 e_i 的最近匹配事件 e_j, 当 $j > i$ 时, $(e_i, \cdots, e_j)$ 是约束序列, 当 $j < i$ 时, $(e_j, \cdots, e_i)$ 是约束序列, 则 E_L 是形式安全的; 否则, E_L 是形式不安全的。

性质3 由独立事件组成的事件序列及约束序列均是形式安全的。

该性质可由公理1及定义6直接得到。

下面给出关于事件序列形式安全的相关定理。

定理1 对于事件序列 E_L, 若 $E_L = (e_{d1}) \oplus \complement(E_1) \oplus \cdots \oplus (e_{dm}) \oplus \complement(E_n)$, 则 E_L 是形式安全的; 反之亦然。

证明 (1) 先证 $\Rightarrow$

设 e 是 E_L 的任一分量。

若 e 是某一个 e_{di}, 则

根据公理1, e 不影响 E_L 的形式安全性。

若 e 是 $\complement(E_j)$ 的分量, 设 $\complement(E_j) = (e_{j1}, \cdots, e_{jk})$, 则

①若 $e = e_{j1}$, 则 e_{jk} 是 E_L 中 e 的最近匹配事件, 且 $(e_{j1}, \cdots, e_{jk})$ 是约束序列;

②若 $e = e_{jk}$, 则 e_{j1} 是 E_L 中 e 的最近匹配事件, 且 $(e_{j1}, \cdots, e_{jk})$ 是约束序列;

③若 e 为 $(e_{j2}, \cdots, e_{j(k-1)})$ 中任一元素, 则 e 为独立事件, 不影响 E_L 的形式安全性。

所以, E_L 是形式安全的。

(2) 再证 $\Leftarrow$

用反证法。

假设 E_L 不能表示成题述形式, 则

E_L 的某一分量 e 是约束的, 但 E_L 不存在 e 的最近匹配事件。

由公理1, E_L 是形式不安全的, 矛盾。 □

该定理表明, 如果一个事件序列是形式安全的, 则可以对事件序列做一个划分, 划分得到的事件子序列, 要么是独立事件组成的, 要么是约束序列。

2.2 事件序列形式安全判断算法

根据定理1, 给出事件序列形式安全判断算法。

设待判断事件序列为 E_L, 形式安全判断步骤为:

(1) 在 E_L 中寻找独立事件 e, 若找到, 令 $E_L = E_L \Theta(e)$, 转到(1); 若找不到, 记 $E_{L1} = E_L$, 转到(2)。

(2) 依次判断序列 $E_{L1} = (e_1, e_2, \cdots, e_m)$ 中元素 $e_i (i = 1, \cdots, m)$, 若找到 e_i 的最近匹配事件 e_j, 且 $(e_i, \cdots, e_j)$ 为约束序列, 令 $E_{L1} = E_{L1} \Theta(e_i, \cdots, e_j)$, 转到(2); 否则, 转到(3)。

(3) 若 $E_{L1} = \Phi$, 则 E_L 是形式安全的; 否则, E_L 是形式不安全的。

2.3 应用实例

以Windows系统的鼠标事件序列形式安全分析为例, 说明上述算法。

(1) 分析鼠标输入事件, 得: E_S = {MOVE, LEFTDOWN, LEFTUP, RIGHTDOWN, RIGHTUP, MIDDLEDOWN, MIDDLEUP, WHEEL}; $\varphi(E_S)$ = {(LEFTDOWN, LEFTUP), (RIGHTDOWN, RIGHTUP), (MIDDLEDOWN, MIDDLEUP), …}; 独立事件集为{MOVE, WHEEL}。

(2) 对事件序列 E_L 的判断过程为:

①在 E_L 中寻找鼠标独立事件MOVE 或WHEEL, 若找到, 令 $E_L = E_L \Theta(e)$, 记 $E_{L1} = E_L$, 并转到①; 若找不到, 转到②。

②依次判断序列 $E_{L1} = (e_1, e_2, \cdots, e_m)$ 中元素 e_i, 若找到 e_i 的最近匹配事件 e_j, 且 $(e_i, \cdots, e_j)$ 为约束序列, 则令 $E_{L1} = E_{L1} \Theta(e_i, \cdots, e_j)$, 转到②; 否则, 转到③。

③若 $E_{L1} = \Phi$, 则 E_L 是形式安全的; 否则, E_L 是形式不安全的。

3 基于事件源的事件可信度模型

现有事件驱动系统中, 系统无条件信任任何输入事件, 恶意程序可以很容易地向系统注入非法事件。因此, 系统应当分析事件的来源及事件的可信度。

3.1 事件可信度评估模型

定义 7 用 $T(g)$ 表示对事件发起者 g 的可相信程度。

定义 8 用 $ET(e)$ 表示事件接收者对事件安全性的相信程度。

定义 9 事件源可信度因子 $ETF(e, g)$ 表示事件 e 经事件发起者 g 发送至事件队列后, $ET(e)$ 在条件 g 下的一个修改, 反映了接收者因事件来源的可信度而增加或减少事件可信度的程度。用公式表示为: 经事件发起者发送至接收者的事件可信度的相对概率相对于事件可信度的先验概率改变的比例。即

$$ETF(e, g) = \begin{cases} \dfrac{p(e|g) - p(e)}{1 - p(e)}, & p(e|g) > p(e) \\ 0, & p(e|g) = p(e) \\ \dfrac{p(e|g) - p(e)}{p(e)}, & p(e|g) < p(e) \end{cases} \tag{1}$$

事件可信度的先验概率 $p(e)$ 表示事件接收者对事件 e 安全性固有的可信程度, 即事件接收者对其可能接收的事件的安全性的基本评估。经事件发起者发送至接收者的事件可信度的相对概率 $p(e|g)$ 表示接收者对事件发起者 g 发送的事件 e 安全性的相信程度。

(1) 当 $ETF(e, g) > 0$ 时, 由式(1)可得到:

$$ETF(e, g) = \frac{p(\neg e) - p(\neg e|g)}{p(\neg e)} \tag{2}$$

解释为接收者对事件安全性不可信度($p(\neg e)$)的先验概率相对于由事件发起者 g 发送之后不可信度的减少值。若 $ETF(e, g) = 0.3$, 表示接收者接收到由 g 发送的事件 e 不可信度减少了 30%。因而, $ETF(e, g) > 0$, 意味着接收者对由发送者 g 发送的事件 e 的可信度将得到加强。

(2) 当 $ETF(e, g) < 0$ 时, 由式(1)可得到:

$$ETF(e, g) = \frac{p(e|g) - p(e)}{p(e)} \tag{3}$$

解释为接收者对事件可信度($p(e)$)的先验概率相对于经过由事件发起者 g 发送之后可信度的减少值。若 $ETF(e,g)=-0.3$,表示接收者接收到由 g 发送的事件 e 可信度减少了30%。因而,$ETF(e,g)<0$,意味着接收者减弱由发送者 g 发送的事件 e 的可信度。

(3)当 $ETF(e,g)=0$ 时,此时,$p(e|g)=p(e)$,意味着事件由事件发起者 g 发起之后,可信程度不改变。也就是说,事件的可信度与事件发起者无关。

由式(1),容易得到如下公式:

$$p(e|g)=\begin{cases} ETF(e,g)+[1-ETF(e|g)]p(e), & ETF(e,g)>0 \\ p(e), & ETF(e,g)=0 \\ [1+ETF(e,g)]p(e), & ETF(e,g)<0 \end{cases} \tag{4}$$

通过式(4),可以求解事件的最终可信度,即 $ET(e)=p(e|g)$。称式(4)为事件可信度评估函数。$ETF(e,g)$ 是事件发起者相对于事件接收者的属性。比如:对于ATM系统,由于与其交互的输入事件来自于实际操作ATM终端的用户,而不是模拟函数,系统甚至可以认为模拟事件是恶意的。特别的,当 $ETF(e,g)=1$ 时,表示事件接收者对于事件发起者 g 发送的事件 e 是完全可信的;当 $ETF(e,g)=-1$ 时,表示事件接收者对于事件发起者 g 发送的事件 e 是完全不可信的。

3.2 事件可信性判断

定义10 用 $U(r)$ 表示事件接收者可接收事件的集合。其中,参数 r 表示特定的事件接收者。记 $U(r)=\{e_1,\cdots,e_i,\cdots,e_n\}$,$n$ 为有限数。

定义11 用 G 表示事件源的集合,$G=\{g_1,\cdots,g_k\}$。

定义12 事件源可信度因子向量 $U(e,r)$:表示特定事件接收者 r 就事件 e 而言,不同事件源的可信度因子向量,一般形式为:$U(e,r)=(ETF(e,g_1),\cdots,ETF(e,g_k))$。

定义13 事件可信度阈值向量 T:指事件接收者对接收事件可信度的要求,当事件可信度大于相应的可信度阈值时,接收者认为是正常的事件;否则,认为是恶意事件。记 $T(r)=(t_1,\cdots,t_n)$。

由此,我们给出事件可信性判断准则。

准则1 事件接收者 r 接收到事件源 g_j 发送的事件 e_i,如果 $p(e_i|g_j)\geqslant t_i$,则 r 认为事件 e_i 是可信的;否则,是不可信的。

对于要求确认完全可信与不可信的场合,有如下的强选择性与强排斥性两个性质。

性质4(强选择性) 若 $ETF(e_i|g_j)=1$,则对于事件 e_i,事件接收者必认为事件源 g_j 所发送的事件 e_i 是完全可信的。

证明 由式(4)知,

$$p(e_i|g_j)=ETF(e_i|g_j)+[1-ETF(e_i|g_j)]p(e_i)=1+(1-1)p(e_i)=1$$

于是性质4成立。 □

性质4提供了系统只接收特定事件源的方法,令该事件源 g_j 相应的 $ETF(e_i|g_j)$ 值为1即可。

性质5(强排斥性) 若 $ETF(e_i|g_j)=-1$,则对于事件 e_i,事件接收者必认为事件源 g_j 所发送的事件 e_i 是完全不可信的。

性质5提供了系统拒绝特定事件源的方法,令该事件源 g_j 相应的 $ETF(e_i|g_j)$ 值为 -1 即可。其证明类似于性质4。

3.3 应用实例

Windows系统中的事件发起者主要是输入输出设备和事件模拟函数。以ATM系统为例,如果从安全考虑出发,事件模拟函数向ATM系统发送的事件应被认为是不可信的,即认为是恶意程序的攻击行为,因而拒绝接收这种事件。下面是本节所描述模型对这类情形的处理过程。

首先,找到系统中的事件与事件源:

$$U(r=\text{ATM})=\{e_1,e_2,\cdots,e_i,\cdots,e_n\}$$

$$G=\{\text{ATM 终端输入输出交互设备, 事件模拟函数}\}$$

依系统要求, 设定可信度阈值向量与可信度因子向量:

$$T(r)=(1_1, 1_2, \cdots, 1_n)$$

$$U(e_i, r)=(1, -1)$$

按照准则 1、性质 4 与性质 5, 可知: 当系统接收到 ATM 终端交互设备发送的事件时, 认为事件是可信的; 反之, 系统接收到事件模拟函数发送的事件时, 认为事件是不可信的, 并拒绝服务这些事件。

4 改进的事件驱动系统框架

以上述两模型为依据, 构造一个改进的事件驱动系统框架, 其结构如图 2 所示。

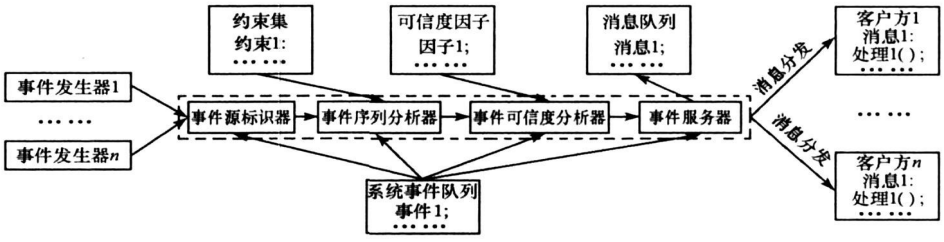

图 2 改进的事件驱动系统结构

Fig. 2 The structure of improved event-driven system

事件发生器、服务器、客户方与现有事件驱动系统中相应模块的功能一致。事件源标识器负责接收发生器的事件, 标识事件来源并发送至原始输入事件队列。事件序列分析器根据系统预先定义的事件约束集, 按照 2.2 节的算法, 分析事件序列的形式安全。事件可信度分析器根据系统预先定义的事件源可信度因子集, 按照 3.2 节准则 1, 评估事件的可信度。与原有事件驱动系统相比, 改进的事件驱动系统框架实现的功能主要有: (1) 输入系统的事件序列相关性条件得到满足, 可以阻止利用不当的事件序列破坏系统的行为; (2) 输入系统的事件可信度得到有效评估, 可以阻止恶意事件输入源对系统的攻击。

5 下一步的工作及总结

下一步的工作包括: (1) 事件序列形式安全分析模型只从形式层面讨论了事件序列的安全性, 探索事件序列本质安全的模型仍是非常有挑战性的; (2) 相较于现有事件驱动框架, 新框架增加的两层分析器的算法复杂度为 $O(n)$ (n 为事件序列中事件数), 对系统整体性能的影响需要研究。

本文的主要贡献包括: 提出了事件约束、独立、事件序列相关性等基本概念, 构建了一种事件序列形式安全分析模型; 针对事件可信度问题, 构建了基于事件源的事件可信度模型; 基于上述两个模型, 给出了一种改进的事件驱动系统框架。

参 考 文 献:

[1] Berson A. Client-server Architecture: Computer Communications[M]. New York: McGraw-Hill, 1992.

[2] Xenitellis S. Security Vulnerabilities in Event-driven Systems[C]//Proceeding of Security in the Information Society: Visions and Perspectives, 2002.

[3] Forrester J E, Miller B P. An Empirical Study of the Robustness of Window NT Applications Using Random Testing[C]//4th USENIX Windows System Symposium, 2000.

[4] Shelton C P, Koopman P, Devale K. Robustness Testing of the Microsoft Win32 API[C]//Proceeding of International Conference on Dependable Systems and Networks, 2000.

[5] Ghosh A K, Schmid V S. An Approach for Analyzing the Robustness of Windows NT Software[C]//21st National Information Systems Security Conference, 1998.

[6] Xenitellis S. A New Avenue of Attack: Event-driven System Vulnerabilities[C]//Proceeding of European Conference on Information Warfare and Security, 2002.

[7] Ghosh A K, Voas J M. Inoculating Software for Survivability[J]. Communications of the ACM, 1999, 42(7): 38-44.