Modeling and Analyzing Logic Vulnerabilities of E-commerce Systems at the Design Phase

WangYang Yu, Lu Liu, XiaoMing Wang, Ovidiu Bagdasar, and John Panneerselvam

***Abstract*—E-commerce systems have become tremendously popular and important for modern business processes in the world of digital economy. E-commerce business processes rely on the distributed and concurrent interaction process among web applications of participants, such as clients, merchants, third-party payment platforms (TPPs) and bank systems. Such complex business interactions bridge the gap of trustiness among participants and introduce new security challenges in the form of logical vulnerabilities, which are prevalent in the business process at the application level. The most pressing challenge is to guarantee security throughout the checkout process at the conceptual design phase such that the logic errors can be detected before the actual implementation. Maintenance and repair of implemented e-commerce systems can be extremely costly. To this end, this paper proposes a novel modeling and analyzing methodology for multi-participants and multi-sessions e-commerce interaction processes based on Colored Petri nets (CPN). Firstly, we define a novel model that can efficiently depict the key properties of e-commerce business interaction processes. Secondly, several modeling principles are formulated based on the design specification of e-commerce systems. Finally, the concept of *Transaction-Logical Consistency* is defined to analyze and verify the logical vulnerabilities of e-commerce systems. Through a discussed case study, we demonstrate the feasibility and applicability of the proposed methodology and its efficiency in detecting problems those can potentially lead to logical vulnerabilities.**

***Index Terms*—E-commerce systems, Petri nets, Business interaction, Logical vulnerability.**

I. Introduction

E-COMMERCE with multi-participants has rapidly evolved worldwide, and becomes increasingly popular in the global economy. The prevalence of online network and smart mobile devices contribute significantly to the rise of e-commerce [1], [2]. E-commerce systems characterize extreme design complexities, such that the dynamism of business interactions among heterogeneous participants imposes an increased level of practical challenges in implementing a perfectly secure checkout process. Many web stores today often utilize web Application Programming Interfaces (APIs) of TPPs such as Google Checkout, AliPay and PayPal for enabling their cashier services. As a result, today's e-commerce businesses have become increasingly hybrid, with their program logic being distributed across multi-participants, including the servers and their clients, along with various third party API service providers [2], [3].

Despite the third party service providers bridging the gap of trustiness between merchants and users, their involvement complicates the logic flow in the checkout process. Even a minor logic vulnerability can lead to serious impacts in mission critical financial applications, thus logic vulnerabilities pose serious threats to the security of e-commerce applications [1], [4], [5]. In such a hybrid system, coordinating the involved participants in a secure fashion is highly challenging, logic flaws are pervasive in modern e-commerce systems [3], [6], [7], [8], [9], [10]. Application and business logic refers to application-specific functionality and behaviors in concurrent business interactions. A logic vulnerability typically exists when a user abuses legitimate application-specific functionality against developers' intentions [11]. Developers are usually unaware of the need to implement sufficient server-side authorization checks to the received request parameters. As a result, vulnerable servers are exposed to malicious users who can potentially alternate the control and data flows through concurrent interactions. The success of such attacks mainly relies on logical errors or lack of server-side validations [12]. Integrity and logic validation mechanism errors are usually the keys for a successful attack. Logic vulnerabilities of e-commerce systems allow malicious users to purchase products using fabricated payments [1].

Such issue cannot be solved by solely relying on existing techniques like protocol verifications, because different integrators usually follow different ways of incorporating and using API services. Furthermore, misunderstandings among the involved participants often arises logic vulnerabilities in the business processes [3], [6]. Logic vulnerabilities are prevalent in real-world services that adopt mainstream protocols. Suppressing the need for integrating security protocols in the online payment systems allow malicious users to successfully get through the purchase process without properly paying for the services [7], [13]. A number of methodologies to test the classes of vulnerabilities of the web applications have been proposed in the past [13], [14], [15]. Davide Balzarotti *et*

This paper is supported in part by the Open Research Fund of Anhui Province Engineering Laboratory for Big Data Analysis and Early Warning Technology of Coal Mine Safety, China under Grants CSBD2022-ZD05, by the Fundamental Research Funds for the Central Universities under Grants GK202205039, and by the Natural Science Foundation of Shaanxi Province under Grants 2021JM-205. *Corresponding Authors (Lu Liu and XiaoMing Wang).*

W. Yu and X. Wang are with the Key Laboratory of Modern Teaching Technology, Ministry of Education, Xi'an 710062, China, with the School of Computer Science, Shaanxi Normal University, Xi'an 710119, China, and also with the Key Laboratory of Intelligent Computing and Service Technology for Folk Song, Ministry of Culture and Tourism, Xi'an 710062, China (E-mail: ywy191@snnu.edu.cn and wangxm@snnu.edu.cn).

L. Liu and J. Panneerselvam are with the School of Informatics, University of Leicester, Leicester LE1 7RH, UK (E-mail: l.liu@leicester.ac.uk and j.panneerselvam@leicester.ac.uk).

O. Bagdasar is with the School of Electronics, Computing and Mathematics, University of Derby, Derby DE22 1GB, UK (E-mail: o.bagdasar@derby.ac.uk).

Manuscript received ***, 2023; revised ***, 2023.

al. developed a vulnerability analysis approach that characterizes both the extended state and the intended workflow of a web application [16]. A static detection method of logic vulnerabilities in e-commerce web applications was proposed in [1]. Eric Chen *et al.* introduced an approach of building provably secure multiparty online services, which is called the Certified Symbolic Transaction (CST) [9]. A system that offers security protection to vulnerable web application integrations was presented in [3]. In the work of [17], a scanning method was proposed to resolve the issue of parameter tampering vulnerabilities.

However, existing approaches focus mostly on the testing or detection of implementation systems at the code-level. In fact, design-level vulnerabilities are indeed a major source of security issues in web applications. For example, in Microsoft's security push, about 50% of the security issues has been identified due to design-level flaws [18]. System development engineering is a comprehensive discipline that involves many activities such as requirements engineering, design and specification, implementation, testing, and deployment. The activity of constructing a model is typically done in the design phases of system development. This may in turn significantly shorten the implementation and testing phases and further decrease the number of flaws in the final system. Different from non-formal models in software engineering, this work focuses on the formal modeling of online shopping systems. In most cases, formal models are more accurate and complete than traditional design documents. This makes that the exploration and construction of the formal model can generate a more solid foundation for the implementation phase [19].

Formal methods are mathematical techniques for specifying and verifying correctness and trustworthiness of software systems. The U.S. Department of Defense Trusted Computer System Evaluation require that the highest level of security classification (the A-class) use formal specification and verification techniques [20]. Petri net-based approaches have been presented to model and verify the correctness and soundness of workflow and concurrent systems [21], [22], [23], [24], [25]. Significant progress has been done on cooperative systems and composition of web services based on Petri nets [26], [27], [28]. However, most of the existing works focus on the correctness and soundness of workflow, cooperative systems, and composition of web services, but fail to concern about the financial security issues. Logic vulnerabilities can be linked directly to the financial losses of legitimate users, and the impact is often severe.

E-commerce business interaction is a typical distributed and concurrent system in the open Internet, the recent developments of which has opened a range of security challenges. A major reason is that the typical online distributed system handles process concurrency in a number of fashions and ways. It is very easy for a designer to neglect significant security interactions and validations during this process concurrency, which might lead to logical vulnerabilities in the system design. To cope with this issue, it is crucial to provide a methodology which enables the detection of logical vulnerabilities of system designs, prior to the actual implementation and deployment. Logic errors in the implementation system might cause irreparable damage, compensation for such defects and modifying the program can be extremely costly.

Furthermore, performing an efficient systematic security analysis of the design specification is still an open issue. Logic vulnerabilities still lack a formal definition, but, in general, they are often the consequence of an insufficient validation of the business process of web applications [13]. Thus, modeling and verifying the e-commerce business interaction processes at the design phase is an important requirement in the current digital economy. With this in mind, this work focuses on the business interaction process at the design phase, and proposes a novel formal methodology for modeling and analyzing logical vulnerabilities of multi-participants and multi-sessions e-commerce systems. Important contributions of this paper are as follows.

1) Firstly, this paper proposes a novel formal model, called *Interactive Business Process Fusion (IBPF) net*, to depict distributed, multi-participants and multi-sessions e-commerce business interaction processes based on CPN. The business flow, logical structures, data definitions, interaction behaviors and key properties can be depicted easily using *IBPF*.

2) Secondly, this paper introduces a novel modeling scheme for *IBPF* according to the design specification, including control structures, data structures and modeling procedure. With this scheme, the business interaction process can be established based on *IBPF*.

3) Thirdly, this paper presents the analyzing principles for logical vulnerabilities, with formal definitions. *Transaction-logical consistency* is defined as the basic property of e-commerce business interaction processes. Then, a state analyzing method is proposed for discovering logical vulnerabilities.

The remainder of this paper is organized as follows. Section 2 discusses a case study. Section 3 presents the modeling scheme. Section 4 proposes the analyzing methods and Section 5 concludes this paper.

II. Case Study

E-commerce systems are facing a wide range of logic vulnerability cases in the recent past, such as the ones in the integration of Interspire and PayPal Express [6], [7], [9], Mongolia [29], osCommerce 2.3.1 with PayPal Payments Standard [13]. After reviewing various logic vulnerability cases, this paper presents an e-commerce business interaction process derived from real scenarios. For illustrating our method clearly, it is abridged, and we only concern about the key functions. The key features of the business process are shown in Fig. 1.

Merchant systems can incorporate various payment methods of TPP, and Fig. 1 shows an integration case. The Dotted rectangles represent the pseudo code of $ValidatePara$, $FinishOrder$, and $UpdateStatus$. During the checkout process, the merchant makes two calls to the TPP. The first one notifies the TPP with an upcoming payment (Step 2) with authentication data (identity). Then, the TPP sends a message attached with a payment token for identifying the payment transaction, which the merchant passes to the shopper with transaction gross (Step 4). The shopper then presents token and gross to the TPP. The TPP sets and confirms certain information about the payment (Step 5). After that, TPP redirects

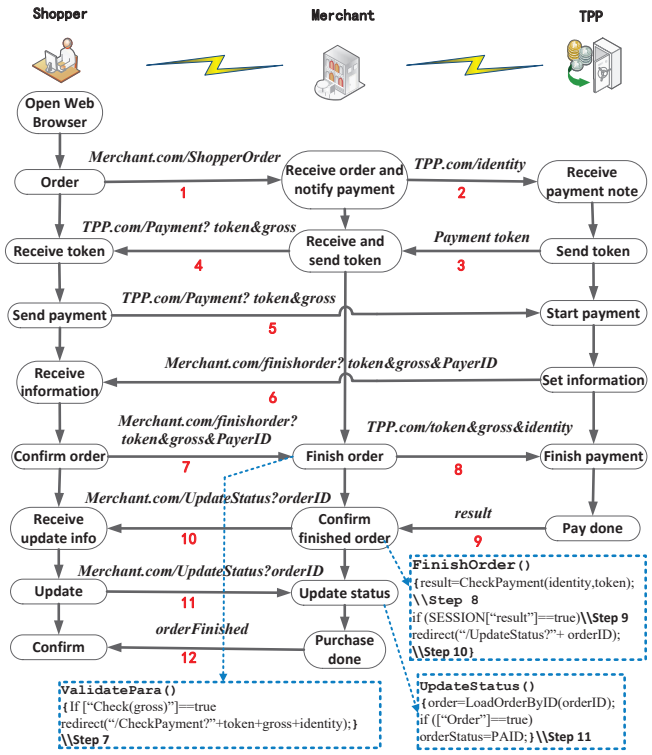


Fig. 1. A case of an e-commerce business process.

the shopper's browser to merchant API $FinishOrder$ with payerID, gross and token as arguments (Step 6). The merchant directly contacts the TPP to check and complete the payment. The function $ValidatePara$ is used to validate the integrity of the transaction gross, and the payment is completed by Steps 8 and 9, where the TPP is contacted to complete the fund transfer. If the identity and other payment information is valid, TPP records the payment and returns $result = true$. This result is saved in the session variable SESSION["result"]. By this time, the payment is completed, and the merchant updates the status of the order. Since the browser needs to be in synchronization with merchant state, the merchant cannot directly call the merchant-side API, thus needs to redirect the shopper's browser to call merchant API $UpdateStatus$, by passing orderID as an argument, which updates the status of the order (Step 11). Then, the merchant retrieves the order using orderID, to set the order status as "PAID" if the session variable is true (SESSION["result"]=true).

For security, sometimes, digital sign and encryption are used in data transmission. However, for enhancing efficiency and usability of the distributed business interaction process, some messages are not signed in the checkout process, which cannot be regarded as a security weakness, as the merchant directly verifies the data integrity with the TPP. This interactive mechanism guarantees the security in many real cases [1], [2], [6], [7], [13]. In the above discussed scenario, the two calls sent to TPP ensure the data consistency among the merchant and TPP. Supplemented by the key validating function APIs, such as $ValidatePara$, $FinishOrder$, and $UpdateStatus$, this complex interactive mechanism of the business process assures security, as illustrated in Fig.1.

However, the aforementioned business interaction process is still characterize logic vulnerabilities. As long as a valid orderID assumes a session in the success state, $updateStatus$ marks the corresponding order as PAID, no matter whether the payment is successful or not. When the shopper obtains a valid orderID for an unpaid expensive order, the orderID can be replaced in Step 11, so that he/she can use the current session state in order to change the order status as PAID. This enables the shopper to get through the checkout process by paying relatively low price for an expensive item [7], [13]. Except this logical vulnerability, the case may has other ones. For a design specification, although it appears to be secure, it is impossible to judge whether it is secure or not, and it is impossible to know in advance where and what the vulnerabilities are. This paper uses this case to illustrate our proposed methodology.

III. Modeling Scheme

In this section, we propose the modeling rules and formal definitions, including data definitions for real e-commence systems, naming scheme, and the structures used in constructing e-commerce business interaction processes.

A. Data Definitions and Naming Rules

CPN is a graphical language used to construct the models of concurrent systems and analyzing their properties. CPN ML embeds the Standard ML language and extends it with constructs for defining color sets and declaring variables [19]. The color sets can be defined as the data types corresponding to the real e-commerce systems, and the variables or constants in the modeling process. According to the specification of the case discussed in Section 2, based on the CPN ML language, we define the data elements used for modeling as follows.

colset User = with a1 | a2;
var u, u1: User;
colset Merch = with b1 | b2;
var m, m1: Merch;
colset Tpp = with c1 | c2;
var t, t1:Tpp;
*colset UserPara= product User*BOOL;*
*colset UserMerPara= product User*Merch*BOOL;*
*colset TPPara= product Tpp*BOOL;*
*colset MerPara=product Merch*BOOL;*
var orderIDBOOL, grossBOOL, orderFinishedBOOL, ShopperOrderBOOL, tokenBOOL, identityBOOL, payerIDBOOL, resultBOOL: BOOL;
colset Parameter = union orderID: UserMerPara + gross: UserMerPara + orderFinished: MerPara + ShopperOrder: UserPara + token: TPPara + payerID: TPPara+result: TPPara + identity: MerPara;

The colsets *User*, *Merch* and *Tpp* are used to represent the participants, such as the user (shopper), merchant and TPP. *UserMerPara*, *UserPara*, *MerPara* and *TPPara* are the types that depict the trading parameters used in the trading process which are passed and exchanged among different participants. Such trading parameters are constructed using *union* and *product* types of CPN. *TP*, *User* and *Mer* are the prefixes of the data produced by the TPP, user (shopper) and merchant respectively. *UserMer* represents the prefix of the data type closely associated with the user (shopper) and merchant.

The e-commerce systems with logic flaws are vulnerable to Web Parameter Tampering, depending on the operation of

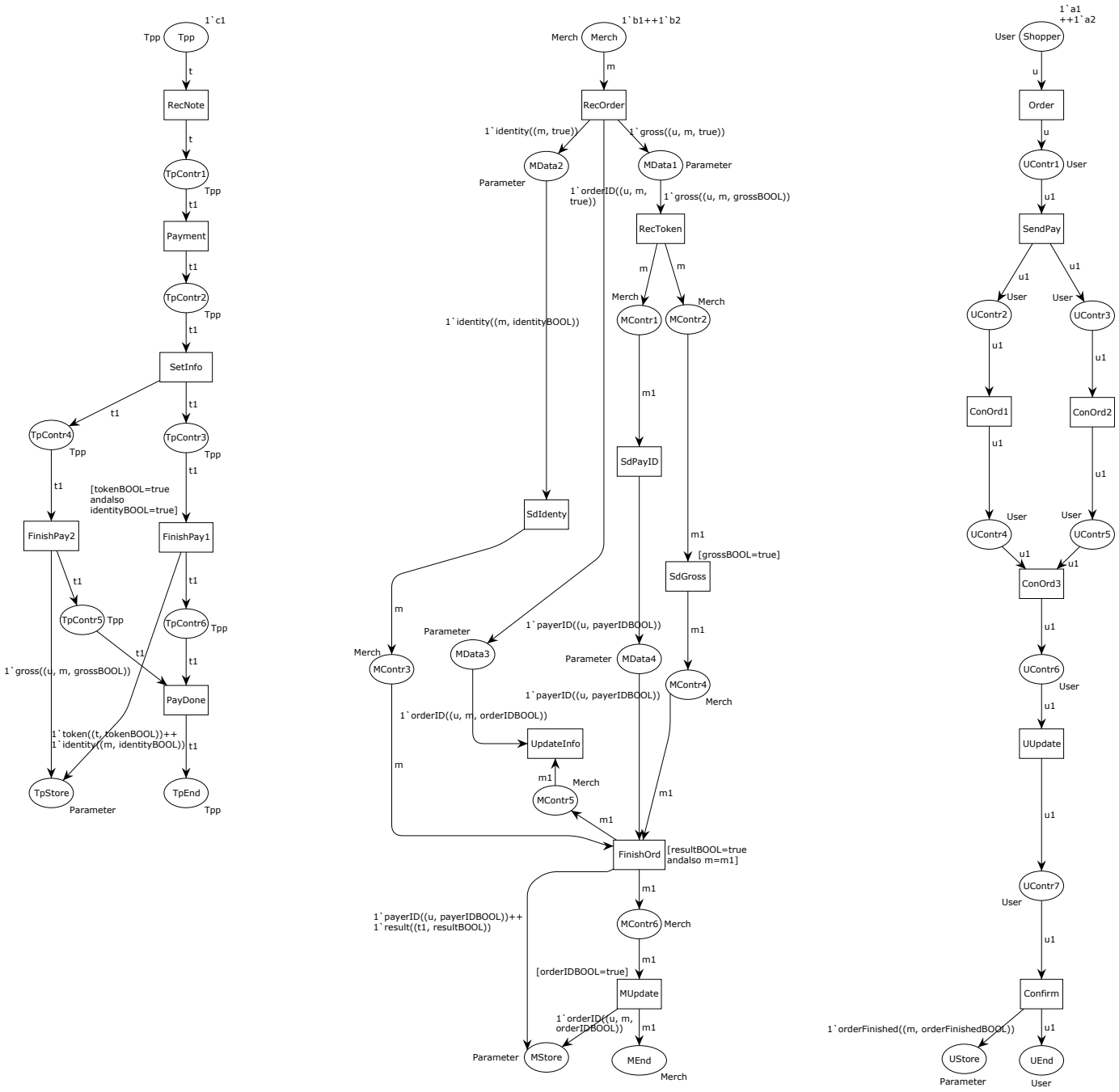


Fig. 2. The *LBPs* of participants.

parameters exchanged between the clients and servers in order to modify the application data, such as user credentials and permissions, quantity of products and price, etc. Especially, any HTML form controls can be tampered with, and the form submissions can be intercepted. Thus, one can bypass the client-side validations [17], [30]. A malicious user however can interact with a vulnerable web application using a browser of which he has full control to manipulate any client-side code as well as HTTP request parameters [17]. For example, an attacker can tamper with URL parameters directly. Thus, in this paper, we adopt the following underlying assumptions [1]:

1) Requests from shoppers are untrusted.

2) Unsigned cashier requests that are sent via insecure channels are untrusted.

3) Cashier responses that are transmitted by shoppers to merchants via HTTP redirection are untrusted.

Therefore, order total, order ID, currency and merchant ID can be tainted during the trading process. Then, we use *BOOL* type to depict this scene, and a *BOOL* value is assigned to every trading parameter. The expression "*parameter name*"+"BOOL" is used to name them, e.g., *orderID* and *orderIDBOOL*.

B. Formal Definitions

A Colored Petri Net is a nine-tuple $CPN = (P, T, A, \Sigma, V, C, G, E, I)$ whose basic definition and more related concepts, such as multiset, color set, and inscription, can be found in [19]. The CPN modelling language is a general-purpose one, i.e., it is not aimed at modelling a specific class of systems [19]. The formal representation of CPN forms the foundation for defining various behavioral properties and their analysis methods. According to the characteristics of e-commerce business interaction processes, we impose some restrictions on the structures and data types of CPN for accurately depicting the business logic of multi-participants e-commerce systems. Thus, we propose the following definitions.

Definition 1 A *Logic Business Process (LBP)* is a CPN $LBP = (P, T, A, \Sigma, V, C, G, E, I)$, where

1) LBP has three special places α, β and γ, where $\alpha \in P$ is the initial place, $\beta \in P$ is the terminal place, $\gamma \in P$ is the memory place, and ${}^\bullet\alpha = \beta^\bullet = \gamma^\bullet = \emptyset$.

2) Let LBP_E=(P, $T \cup \{\tau\}$, $A \cup \{(\tau,\alpha),(\beta,\tau),(\gamma,\tau)\}$, Σ , V, C, G', E', I) be the trivial extension of LBP, in which $E' : \{(\tau,\alpha),(\beta,\tau),(\gamma,\tau)\} \rightarrow EXPR_V$, and $a{\in}A{\rightarrow}E'(a){=}E(a)$; $t{\in}T{\rightarrow}G'(t){=}G(t)$, and τ has no guard. Then, LBP_E is strongly connected.

An *LBP* is a special CPN, and corresponds to the internal business process of a participant in the entire e-commerce interaction process. Fig. 2 shows the *LBPs* with initial markings of the shopper, merchant and TPP as of the case discussed in Section 2. An *LBP* starts from an initial place α and ends at terminal place β and memory place γ, where α depicts the beginning state of a participant, β represents the end of a executing process, and γ is used to store the internal data after the transaction. For example, in Fig. 2, the place *Merch* is the initial place of the merchant. *MEnd* and *MStore* are the terminal place and memory place respectively. If we connect *Merch* with *MEnd* and *MStore* using a transition and three arcs among them, the *LBP* of the merchant would be strongly connected, and it is used to guarantee the intercommunication feature of *LBP*.

Definition 2 Let $LBP = (P, T, A, \Sigma, V, C, G, E, I)$ be a *LBP*. Then, a *Logic Business Process with Channels (LBPC)* is $LBPC = (P{\cup}P_{IN}{\cup}P_O, T, A{\cup}A_{IN}{\cup}A_O, \Sigma, V, C', G, E', I')$ such that the following holds.

1) P_{IN} is a set of input channel places, P_O is a set of output channel places, $P_{IN} \neq \emptyset$, $P_O \neq \emptyset$, ${}^\bullet P_{IN} = P_O^\bullet = \emptyset$, $P_{IN} \cap P_O \cap P = \emptyset$.

2) $\{\alpha, \beta, \gamma\} \not\subset (P_{IN} \cup P_O)$.

3) $A_{IN} \subseteq P_{IN} \times T$, $A_O \subseteq T \times P_O$, $A_{IN} \cap A_O \cap A = \emptyset$.

4) $C' : (P_{IN} \cup P_O) \rightarrow \Sigma$, and $p{\in}P{\rightarrow}C'(p){=}C(p)$.

5) $E' : (A_{IN} \cup A_O) \rightarrow EXPR_V$, and $a{\in}A{\rightarrow}E'(a){=}E(a)$.

6) $I' : (P_{IN} \cup P_O) \rightarrow EXPR_\emptyset$, and $p{\in}P{\rightarrow}I'(p){=}I(p)$.

An *LBPC* is an extension of *LBP* integrated with some interaction places (channels) which are used to depict the communication channels of trading parameters among different participants. For the channel places, we use the convention "*prefix*"+"Chan$k$" for naming, $k \in \mathbb{N}+$. For example, Fig. 3 is the *LBPC* of the merchant, which is extended from the *LBP* shown in Fig. 2 by using five input channel places, *UChan1*, *TppChan1*, *UChan3*, *TppChan3*, *UChan4*; and five output channel places, *MerChan1*, *MerChan2*, *MerChan3*, *MerChan4*, *MerChan5*. *UChan1* represents a channel place launched from the user; *MerChan2* represents a channel place launched from the merchant; and *TppChan3* represents a channel place launched from TPP.

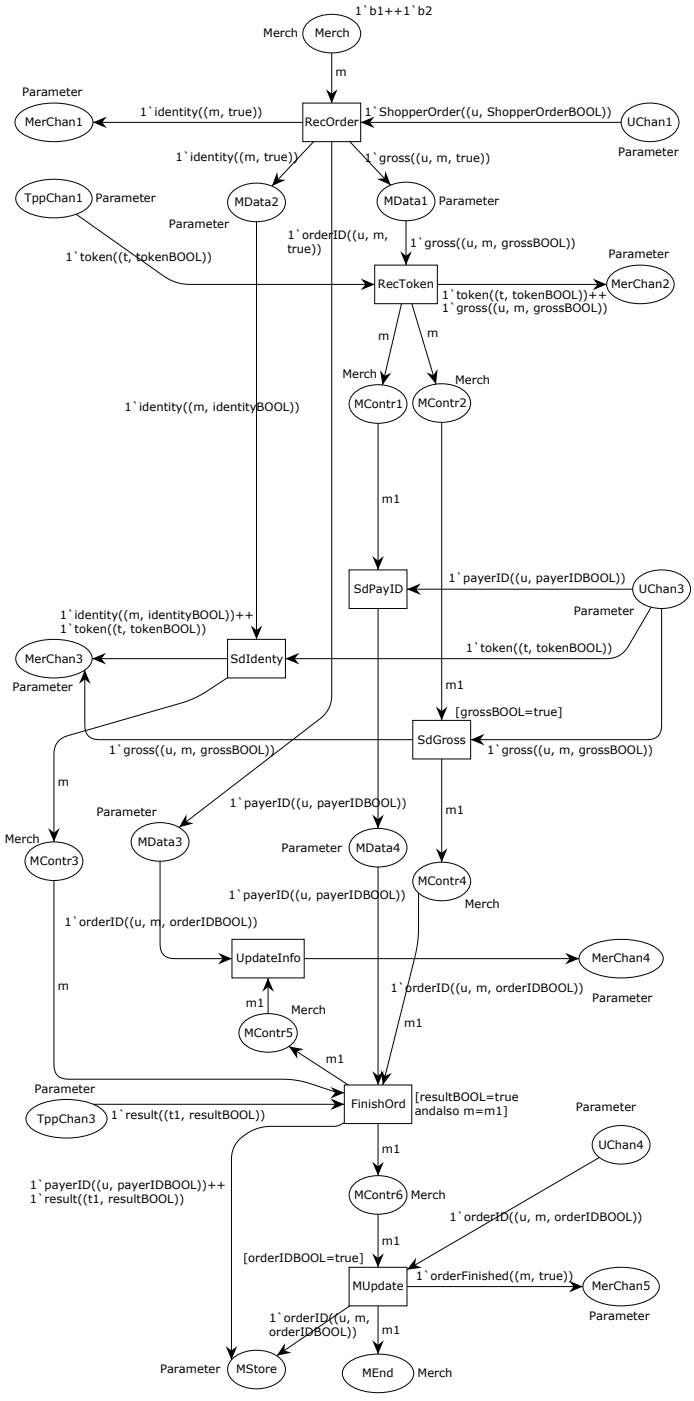


Fig. 3. The *LBPC* of the merchant.

Definition 3 Let $LBPC_n$=($P_n{\cup}P_{INn} \cup P_{On}$, T_n, A_n, Σ_n, V_n, C_n, G_n, E_n, I_n), $n{\in}\mathbb{N}_2$, be two $LBPCs$ which are disjoint, and have common places, such that $P_1 \cap P_2 = \emptyset$, $P_{IN1} \cap P_{IN2} = \emptyset$, $P_{O1} \cap P_{O2} = \emptyset$, $P_{IN1} \cap P_{O2} \neq$

$\emptyset$, $P_{O1} \cap P_{IN2} \neq \emptyset$, $T_1 \cap T_2 = \emptyset$, $A_1 \cap A_2 = \emptyset$, $\Sigma_1 \cap \Sigma_2 \neq \emptyset$, $V_1 \cap V_2 \neq \emptyset$; $C_1(p) = C_2(p)$, $I_1(p) = I_2(p)$, $p \in (P_{IN1} \cap P_{O2}) \cup (P_{O1} \cap P_{IN2})$. Then, the $Fusion\ Operation$ Φ of the two nets is $LBPC_1 \Phi LBPC_2$ = $LBPC = (P \cup P_{IN} \cup P_O, T, A, \Sigma, V, C, G, E, I)$, where $P = P_1 \cup P_2$, $T = T_1 \cup T_2$, $A = A_1 \cup A_2$, $\Sigma = \Sigma_1 \cup \Sigma_2$, $V = V_1 \cup V_2$, $(p \in P_n) \rightarrow (C(p) = C_n(p)) \wedge (I(p) = I_n(p))$, $(a \in A_n) \rightarrow (A(a) = A_n(a))$, $(t \in T_n) \rightarrow (G(t) = G_n(t))$.

Definition 3 is a rigor formal definition of $fusion\ operation$ of two *LBPCs*. It is just like the synchronous synthesis method in the original Petri net. It specifies a synthesis method based on places of CPN, by which an integrated e-commerce business process can be built in a step-by-step manner. The benefit is to help designers understand and analyze a business process with different views by composite operations. Note that, for two *LBPCs*, only their input and output channel places with the arc inscriptions have intersection and their corresponding *LBPs* have no intersection.

Definition 4 Let $LBPC_n$=$(P_n \cup P_{INn} \cup P_{On}, T_n, A_n, \Sigma_n, V_n, C_n, G_n, E_n, I_n)$, $n, m \in \mathbb{N}+$, $n \leq m$, $m \geq 2$, be m $LBPCs$ satisfying Definition 2, and $\bigcup_{n=1}^{m} P_{INn} = \bigcup_{n=1}^{m} P_{On}$. Then, $IBPF = (P \cup P_{IN} \cup P_O, T, A, \Sigma, V, C, G, E, I) = LBPC_1 \Phi LBPC_2 \Phi ... \Phi LBPC_m$ is called a *Interactive Business Process Fusion (IBPF)* net.

Definition 4 is based on Definition 3. A set of *LBPCs* can construct an *IBPF*. Definition 4 defines the *Fusion Operation* between two *LBPCs*, and an *IBPF* is the union of m *LBPCs* $LBPC_1 - LBPC_m$ based on the *Fusion Operation*, via a set of common places, i.e., $\bigcup_{n=1}^{m} P_{INn}$ and $\bigcup_{n=1}^{m} P_{On}$. Not that input channel places can only be combined with output ones. Based on Definitions 1-3 and the examples in Figs. 3-5, as shown in Fig. 6, a complete business interaction process-*IBPF* is constructed by composing the *LBPCs* presented in Fig. 3 through to Fig. 5, in accordance with the shopper, merchant, and TPP respectively. The three *LBPCs* synthesised by 12 channel places such as *UChan1*, *MerChan2*, *TppChan1*, *TppChan1*. Then, the properties of *IBPFs* are described as follows.

Property 1 A *Interactive Business Process Fusion (IBPF)* net $IBPF = (P \cup P_{IN} \cup P_O, T, A, \Sigma, V, C, G, E, I) = LBPC_1 \Phi LBPC_2 \Phi ... \Phi LBPC_m$ holds the following properties.

1) $IBPF$ has m initial places α_1, α_2, ..., α_m, m terminal places β_1, β_2, ..., β_m, and m memory places γ_1, γ_2, ..., γ_m, where ${}^\bullet \bigcup_{n=1}^{m} \{\alpha_n\} = \bigcup_{n=1}^{m} \{\beta_n\}^\bullet = \bigcup_{n=1}^{m} \{\gamma_n\}^\bullet = \emptyset$.

2) If $p \in P$, and $p \notin (\bigcup_{n=1}^{m} \{\alpha_n\} \cup \bigcup_{n=1}^{m} \{\beta_n\} \cup \bigcup_{n=1}^{m} \{\gamma_n\})$, then ${}^\bullet p \neq \emptyset$, and $p^\bullet \neq \emptyset$.

3) Let $IBPF_E = (P,\ T \cup \{\tau\},\ A \cup \bigcup_{n=1}^{m} \{(\tau, \alpha_n)\} \cup \bigcup_{n=1}^{m} \{(\beta_n, \tau)\} \cup \bigcup_{n=1}^{m} \{(\gamma_n, \tau)\},\ \Sigma,\ V,\ C,\ G',\ E',\ I)$ be the trivial extension of $IBPF$, in which $E' : \bigcup_{n=1}^{m} \{(\tau, \alpha_n)\} \cup \bigcup_{n=1}^{m} \{(\beta_n, \tau)\} \cup \bigcup_{n=1}^{m} \{(\gamma_n, \tau)\} \rightarrow EXPR_V$, and $a \in A \rightarrow E'(a) = E(a)$; $t \in T \rightarrow G'(t) = G(t)$, and τ has no guard. Then, $IBPF_E$ is strongly connected.

Proof. Every LBP has three special places α, β and γ respectively. In $LBPC$, $\{\alpha, \beta, \gamma\} \not\subset (P_{IN} \cup P_O)$. According to Definition 3, the $fusion\ operation$ Φ of two $LBPCs$ just refers to the combination of input and output channel places and their arcs; $P = \bigcup_{n=1}^{m} P_n$, $T = \bigcup_{n=1}^{m} T_n$, $A = \bigcup_{n=1}^{m} A_n$. Thus, $IBPF$ has m initial places α_1, α_2, ..., α_m, m terminal places β_1, β_2, ..., β_m, and m memory places γ_1, γ_2, ..., γ_m. $\forall t \in T_n \rightarrow (t, \alpha_n) \notin A_n$; $\forall t \in (T \backslash T_n) \rightarrow (t, \alpha_n) \notin (A \backslash A_n)$. Then, $\forall \alpha_n \in P_n \rightarrow (\alpha_n \in P \wedge {}^\bullet \alpha_n = \emptyset)$, i.e., ${}^\bullet \bigcup_{n=1}^{m} \{\alpha_n\} = \emptyset$. Likewise, $\bigcup_{n=1}^{m} \{\beta_n\}^\bullet = \bigcup_{n=1}^{m} \{\gamma_n\}^\bullet = \emptyset$.

According to Definition 4, as $\bigcup_{n=1}^{m} P_{INn} = \bigcup_{n=1}^{m} P_{On}$, $\exists i, j \in \mathbb{N}+$, $i, j \leq m$, making that $(\forall p \in P_{INi}) \rightarrow (p \in P_{Oj})$, $(\exists t_i \in T_i) \wedge (\exists t_j \in T_j) \rightarrow ((p, t_i) \in A_i) \wedge ((t_j, p) \in A_j)$. Thus, $(p, t_i), (t_j, p) \in A$, i.e., ${}^\bullet p \neq \emptyset$, and $p^\bullet \neq \emptyset$. Similarly, $\exists i, j \in \mathbb{N}+$, $i, j \leq m$, making that $(\forall p \in P_{Oi}) \rightarrow (p \in P_{INj})$, $(\exists t_i \in T_i) \wedge (\exists t_j \in T_j) \rightarrow ((t_i, p) \in A_i) \wedge ((p, t_j) \in A_j)$. Thus, $(t_i, p), (p, t_j) \in A$, i.e., ${}^\bullet p \neq \emptyset$, and $p^\bullet \neq \emptyset$.

Let IBP_i, IBP_j are any two *logic business processes* of $IBPC_i$ and $IBPC_j$ respectively in an $IBPF$, $\varepsilon_{i1}, \varepsilon_{i2} \in (T_i \cup P_i)$. According to Definition 1, the subgraph of $IBPF_E$

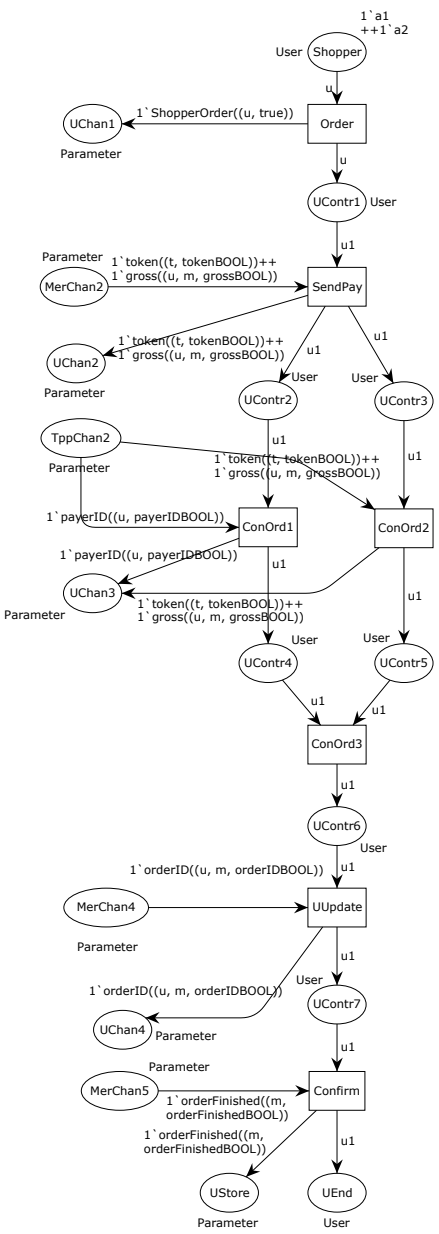


Fig. 4. The *LBPC* of the shopper.

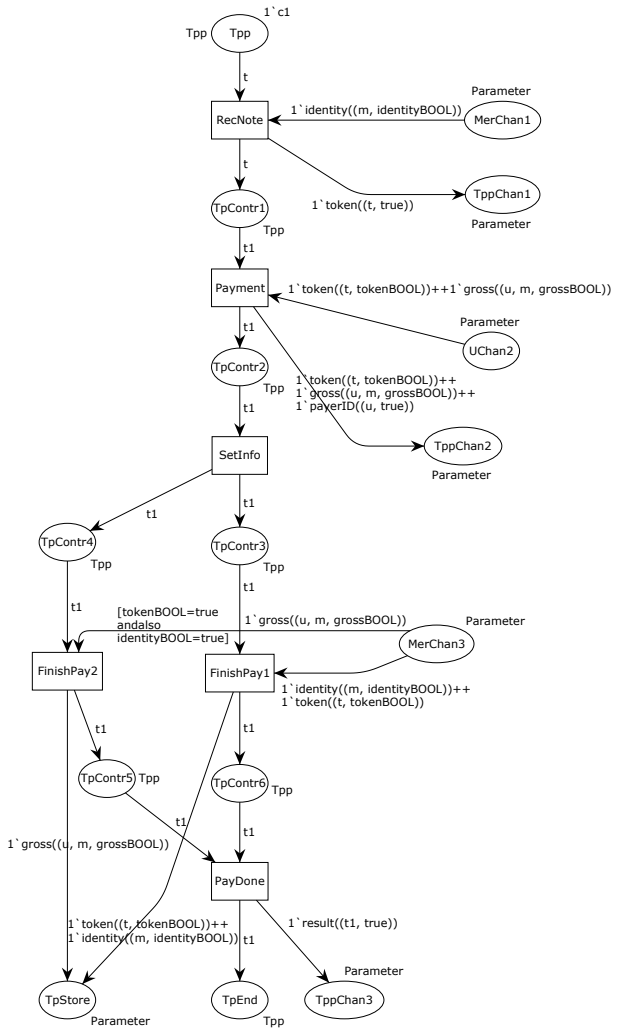


Fig. 5. The *LBPC* of TPP.

LBP_{iE}=(P_i, $T_i \cup \{\tau\}$, $A_i \cup \{(\tau, \alpha_i), (\beta_i, \tau), (\gamma_i, \tau)\}$, Σ_i , V_i, C_i, $G_i{}'$, $E_i{}'$, I_i) is strongly connected. Then, there is a directed path from ε_{i1} to ε_{i2}, and vice versa. Likewise, LBP_{jE} is strongly connected. From any $\varepsilon_i \in (T_i \cup P_i)$, there is a directed path to τ. As (τ, α_j) is an directed arc in $IBPF_E$, there is a directed path from ε_i to any $\varepsilon_j \in (T_j \cup P_j)$. For any $p \in (\bigcup_{n=1}^{m} P_{INn} \cup \bigcup_{n=1}^{m} P_{On})$, $\exists t_k \in T_k$ and $\exists t_l \in T_l$, $k, l \in \mathbb{N}+$, $k, l \leq m$, making that $(t_k, p), (p, t_l) \in A$. Then, there is a directed path from p to any $\varepsilon \in (P \cup T \cup \{\tau\})$ in $IBPF_E$. Thus, $IBPF_E$ is strongly connected. □

Condition 1 represents that an *IBPF* has an initial place set, a terminal place set and a memory place set; condition 2 means that every input or output channel place would be a message interaction bridge between two *IBPs*, and had predecessor and successor nodes; condition 3 guarantees the connectivity of *IBPF*. Definitions 1-4 and Property 1 construct a complete formal specification system based on CPN. These definitions are related and progressive.

C. Basic Structures

Here, we illustrate the basic structures used for the modeling process, including routing, control and data structures. Basic routing structures are used for constructing the control and data ones in internal processes of participants, and business interaction processes among participants.

1) Routing Structures: Petri nets are suitable to describe distributed and concurrent properties. True concurrency is allowed instead of the interleaving-based semantics based on some special structures. In this paper, we define several basic structures for constructing an $IBPF$ according to WFMC [31]. As shown in Fig. 7, sequential structure depicts the scene where several events or APIs fire one after another. Concurrent structure illustrates the way of constructing the concurrent scene. When there exists selective routing in a business process, it would be depicted by the conditional selection structure.

2) Control Structures: In this work, control structures mean the places and transitions that are used to coordinate the activities among the participants. In this work, we present three kinds of control structures:

a) Internal control structures.

Internal control structures are used to construct the internal business process of a participant based on routing structures. The flowing tokens in the control structures belong to the colsets that represent the participants, and the *vars* belong to these colsets, we can call them *participant vars*. For the LBP of TPP shown in Fig. 2, the *participant vars* t and $t1$ belong to the colset *Tpp* which represent the TPP. The arcs which have *participant vars*, together with their associated places and transitions belong to the internal control structures, e.g., the path from *Tpp* to *TpEnd*, and *SetInfo* to *PayDone*. This means that the internal business process, which is composed of APIs, functions and operation events, is controlled and conducted by a participant.

b) Multi-session and multi-participants.

As we know, although Petri nets are good at constructing models of concurrent systems and analyzing their properties, their structures are relatively fixed. Petri nets are effective in modeling and analyzing flexible manufacture systems [32], [33]. However, e-commerce systems based on the Internet are often open and dynamic, characterizing multi-sessions, multi-participants and multi-tasks. A user can open many Internet browsers and sessions, and simultaneously place different orders during the same business process. Then, this scenario can invite security attacks [15].

CPN has the concepts of colsets and colors. Thus, in the modeling scheme, we can make the business structure of *IBPF* as the relatively fixed business process of the e-commerce system, and use different colsets to express multi-participants, and different colors to depict multi-session. In Fig. 2, colsets *User*, *Merch* and *Tpp* represent the user (shopper), merchant and TPP respectively. The colors *a1* and *a2* depict two users, and they can start two sessions according to the same business process. *b1* and *b2* depict two merchant sessions that are ready to be started. In the business process, the flowing tokens express the trading parameters of some session. The complex color types can also help constructing the parameters, e.g., *product* and *union* types. For example, in Fig. 2, *gross((u, m, grossBOOL))* represents that the trading parameter *gross* belongs to a session started by u and m. Therefore, these

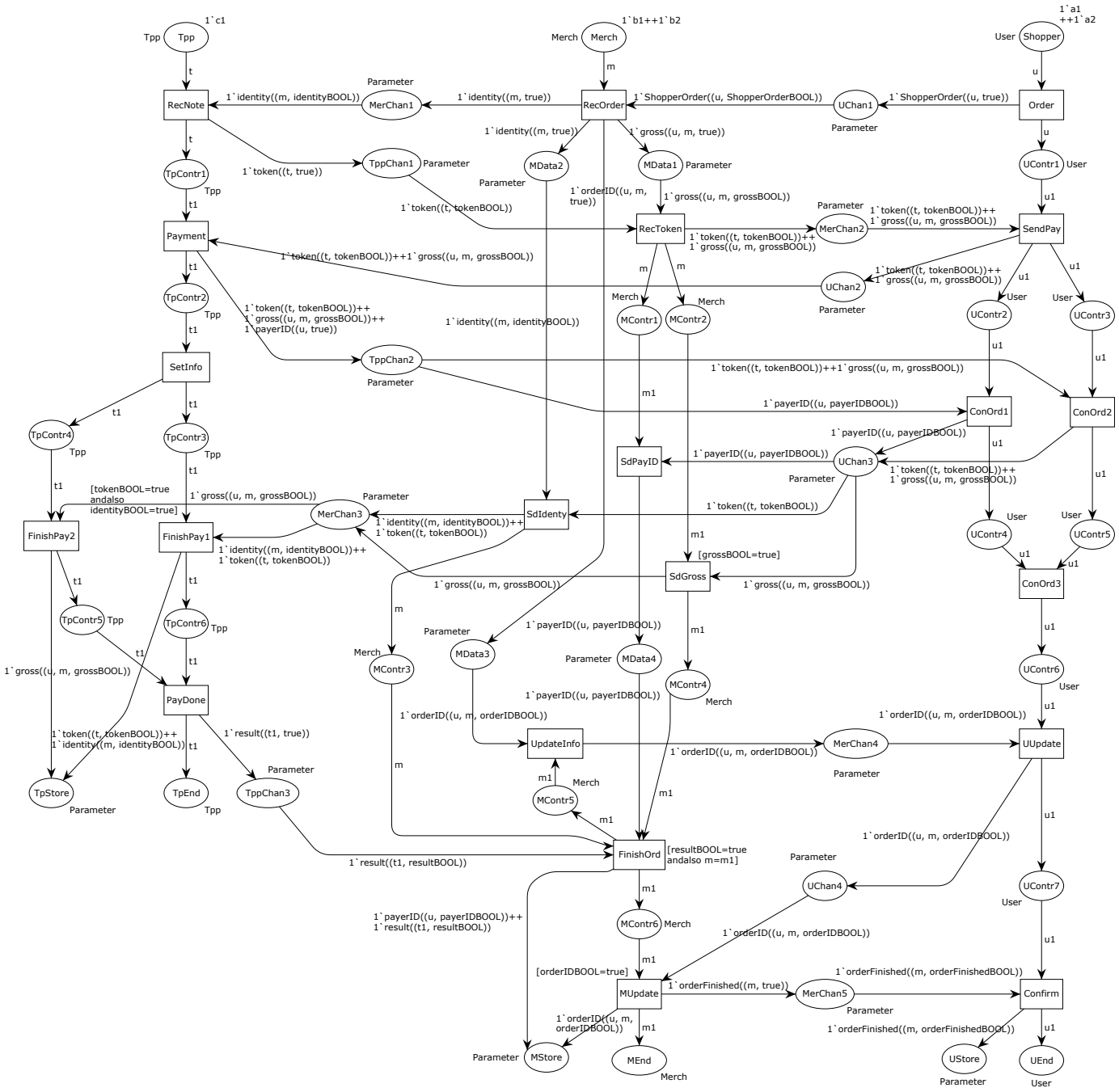


Fig. 6. The complete business process-*IBPF*.

design rules can control and implement multi-sessions and multi-participants in the e-commerce business processes.

c) Validation functions.

In distributed e-commence systems, the designed interaction mechanism is used to guarantee the funds security [6], [7], [12], [34], in which validation functions play important roles. The trading parameters flowing among the distributed participants are checked by the Validation functions in order to guarantee the legality. In *LBPF*, we use guards [19] to depict these validation functions. For instance, as in the case discussed in Section 2, there are some validation functions, such as $ValidatePara$, $FinishOrder$, and $UpdateStatus$. Such functions are depicted by the guards of some key transitions, e.g., $SdGross$, $FinishOrd$, $MUpdate$ in Fig. 6.

3) Data Structures: Data structures depict the data flow (trading parameters flow) in the business processes, including channels of Definition 2, memory places of Definition 1 and internal flowing paths of data. Channels depict the transmission paths of the trading parameters among different participants. Memory places are used to store the internal data after the transaction process, and the data can reflect the trading result of a participant. Internal flowing paths represent the data flow inside the business process of a participant. For example, the flowing path of parameter $orderID$ in the

LBP of the merchant in Fig. 2. According to the case in Section 2, $orderID$ is a trading parameter produced after the merchant accepted the order from the shopper, and this event is represented by the transition $RecOrder$. This corresponds to the event of *Receive and send token* in the case of Fig. 1. Then, the parameter flows through other events and forms a data flow path until it is transmitted to the memory place $MStore$. So as other parameters.

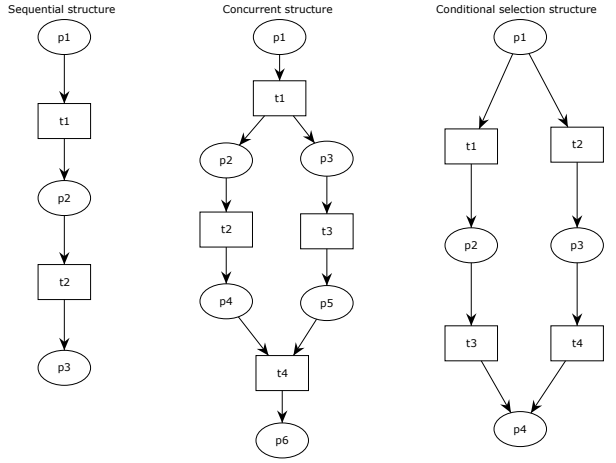


Fig. 7. Routing structures

In CPNtools, for easy modeling, avoiding the two complex parameters holding the same variable is a basic rule. For example, Fig. 8(a) presents a data structure including two parameters $va1((u, va1BOOL))$, $va2((u, va2BOOL))$ belonging to the $union$ type of $Parameter$, in which $var\ u : User$, $var\ va1BOOL, va2BOOL : BOOL$, and every parameter includes three elements: one $constructor$, and two variables belonging to the basic types [19]. Although this looks correct, there would be a binding error at $t2$. As for the input of $t2$, there are two parameters belonging to complex data types holding the same variable u. Through our experiments, we found that splitting can effectively avoid this binding error, which ensures that complex parameters have distinct values. For example, Fig. 8(b) represents an improved data structure. As for every transition, there is just one complex parameter.

D. Modeling Procedure

To construct an ***LBPF***, we elicit the intended functions in terms of the design specifications derived from the requirement analysis during the system design phase, and construct the model based on the following steps:

1) Data Definitions of $LBPF$.

Acquire the trading parameter set from the design specifications, and declare Σ and V.

2) Construct $LBPs$ of Participants.

Identify the order of operation events and APIs in business interaction processes of participants in accordance with the key functionalities and design specifications. Then, construct the control and data structures to depict the internal business functionalities of different participants aided by Definition 1.

3) Construct $LBPCs$ of Participants.

Confirm the message delivering mechanism, and identify the input and output channel places according to Definition 2. Then, connect them to their corresponding $LBPs$ using arcs with E.

4) Fusion Operation of $LBPCs$.

According to Definitions 3 and 4, synthesize the obtained $LBPCs$ and get an $IBPF$. It is worthy of note that an initial marking should be added at last.

IV. Analyzing Logic Vulnerability

In this section, based on $LBPF$, we present the methods of analyzing the logic vulnerabilities. Firstly, we define the *transaction-logical consistency* as the basic property of trustworthy e-commerce business systems. Then, we illustrate the verification procedure.

A. Transaction-Logical Consistency

In e-commerce protocols, atomicity is the basic property [35], which usually applies to the network level, but not to the application level. The validations of protocols in the network level cannot reveal such logic errors and defects in business processes as lack of calibration and inconsistency among data [28]. Therefore, based on the existing properties, we propose the following transaction conditions to guarantee the security goal of an e-commerce business interaction process.

Definition 5 Let $IBPF = (P \cup P_{IN} \cup P_O, T, A, \Sigma, V, C, G, E, I) = LBPC_1\ \Phi\ LBPC_2\ \Phi\ ...\ \Phi\ LBPC_m$ be an *interactive business process Fusion (IBPF)* net under the initial making M_0, $m \in \mathbb{N}+$, and σ is a executable transition sequence of $IBPF$. A reachability marking M ($M_0 \xrightarrow{\sigma} M$) is called *Transaction-Finished State*, if the following conditions hold:

1) $M(\beta_m) = \left|\bigcup^{++}_{t \in {}^\bullet\beta_m} E(t, \beta_m)\right|$.

2) $M(\gamma_m) = \left|\bigcup^{++}_{t \in {}^\bullet\gamma_m} E(t, \gamma_m)\right|$.

Transaction-Finished State depicts a completed state of a transaction. Condition 1 illustrates that a participant had finished a transaction. If a shopper has paid and TPP is at the paid state, then merchant has to reach the state of a finished transaction. On the other hand, if a shopper has not paid and TPP is not at the paid state, then the merchant cannot be at the shipping state of a to-be-finished transaction. For any terminal place β_m, its token number is equal to that of input variables on the input arcs. This ensures the completion of a single transaction. Condition 2 represents that any memory place γ_m should have obtained the necessary trading parameters. Likewise, the token number in γ_m is equal to that of the input variables on the input arcs.

Here, The notation $\bigcup$ means the union operation. However, in CPN, the set is based on Multisets [19]. The symbols ++ is the operator used to construct a multiset consisting of the tokens, for example, in Fig. 6, the expression 1‘b1++1‘b2 represents that there are two tokens which are b1 and b2 respectively. "||" depicts that the token values binding with the arc inscriptions. As E is the set of arc expressions, and we see the expressions as the multiset of tokens that are transmitted to the next transition. For example, in Fig. 9, $\left|\bigcup^{++}_{t \in {}^\bullet MStore} E(t, MStore)\right| = 1'orderID((a2, b2, true))$ ++ $1'payerID((a1, true))$ ++ $1'result((c1, true))$.

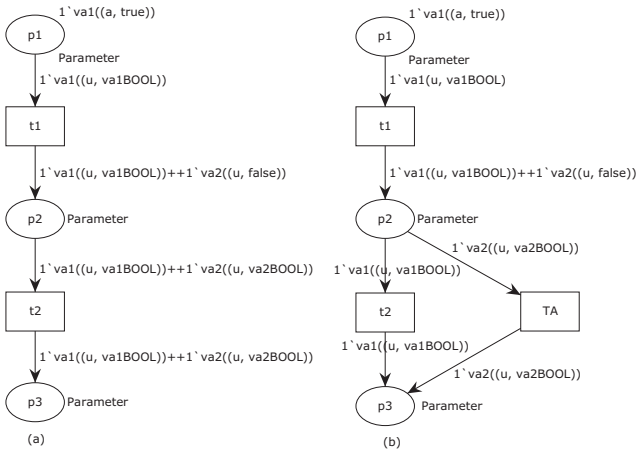


Fig. 8. Data structures

Definition 6 Let $IBPF = (P \cup P_{IN} \cup P_O, T, A, \Sigma, V, C, G, E, I) = LBPC_1\ \Phi\ LBPC_2\ \Phi\ ...\ \Phi\ LBPC_m$ be an *interactive business process Fusion (IBPF)* net under the initial making M_0, $m \in \mathbb{N}+$, then $IBPF$ satisfies the *Transaction-Logical Consistency* if every *Transaction-Finished State* M satisfies the following conditions:

1) For every parameter ν in $\forall\gamma_m$, $\nu BOOL$=true.

2) For every variable ρ in $\forall\beta_m$ and $\forall\gamma_m$ that depicts a session or participant, it has only one value.

Definition 6 depicts the *Transaction-Logical Consistency* by a formal description according to the standard CPN semantic. It is a definition illustrating the conditions that the *Transaction-Finished States* should satisfy. The conditions illustrate that each participant must reach the correct conclusion. Condition 1 shows that there is no invalid data and parameter tampering existing in the trading process. Condition 2 represents that all the trading parameters belong to the same transaction. Definition 6 protects the interests of distributed participants and guarantee that all the participants are in a fair state, i.e., there is no possibility of paying without receiving goods and vice versa. In other words, any given participant cannot damage the interests of another. For example, Fig. 9 is a *Transaction-Finished State* M:

$M(TpStore)$=$1'gross((a1, b1, true))$++$1'token((c1, true))$
++$1'Identity((b1, true))$;
$M(TpEnd) = 1'c1$;
$M(MStore)$=$1'orderID((a2, b2, true))$++$1'payerID((a1,$
$true))$++$1'result((c1, true))$;
$M(MEnd) = 1'b1$;
$M(UStore)$=$1'orderFinished((b2, true))$;
$M(UEnd) = 1'a1$;
$M(MData3)$=$1'orderID((a1, b1, true))$

M represents that a transaction is completed, however, it does not satisfy *Transaction-Logical Consistency*. Although all BOOL variables are true, there are two values for the shopper and merchant variables u and m, i.e., $a1, b1,\ a2, b2$. This state illustrates that the shopper has paid for $orderID((a1, b1, true))$, and received the goods of $orderID((a2, b2, true))$. It damages the interest of the merchant, and does not meet the *Transaction-Logical Consistency*, which is caused by the interaction process that leads to states inconsistency among distributed participants.

B. Analyzing Procedure

A logic vulnerability in a distributed e-commerce business interaction process exists when the merchant or TPP cannot validate whether a given shopper has paid for the correct order total with the expected gross or not. Based on logic vulnerabilities, it is easy to launch attacks in trading process. Using the browser extensions simply, attackers can withhold HTTP requests, modify requests or forge requests. In addition, attackers can exploit a signed token to mimic as a cashier, reuse payment information from previous orders or even change the return URLs in HTTP forms to intercept cashiers' responses. To sum up, logic vulnerabilities are caused by the tainted trading parameters as shown in Tab. I [1].

TABLE I
TAINT TRADING PARAMETERS

Type	Description
Tainted orderID	To bypass order payments, attackers can replay the payment information of previous orders from the same merchant
Tainted gross	Attackers can pay an arbitrary amount for an order by tampering with the gross sent to TPP
Tainted merchantID	When merchantID is tainted, an attacker can set up his/her own merchant account on the designated cashier's server
Exposed token	An exposed signed token invalidates any security checks against trusted symbolic values, which is because such a signed request may be forged by an attacker

For depicting these malicious behaviors, in Section 3, we know that multi-session or multi-user can be depicted by *IBPF* conveniently. Then, tampered parameters can be depicted by changing the BOOL values corresponding to the parameters. Adhering to the principles of the modeling process in Section 3, together with specific initial marking, we can build a specific application behavior scene and then verify the business process. We present our analyzing procedure with related definitions as follows.

Definition 7 An $IBPF = (P \cup P_{IN} \cup P_O,\ T,\ A,\ \Sigma,\ V, C, G, E, I)$ is *terminable* under an initial marking M_0 if:

1) $\exists\Gamma = \left\{M' | M' \in R\left(M_0\right), \forall t \in T \rightarrow \neg M' \xrightarrow{t}\right\}$.

2) $\forall M \in R(M_0)$, there exists a transition sequence σ, making that $M \xrightarrow{\sigma} M'$, $M' \in \Gamma$.

Definition 7 guarantees that an *IBPF* can terminate after the running process. Condition 1 means an *IBPF* has a terminated state set, in which the terminated state cannot fire anymore. Condition 2 depicts that $\forall M \in R(M_0)$ can reach a terminated state by firing a transition sequence.

Definition 8 An $IBPF = (P \cup P_{IN} \cup P_O,\ T,\ A,\ \Sigma,\ V, C, G, E, I)$ is *rational* under an initial marking M_0 if:

1) $IBPF$ is bounded [19].

2) $IBPF$ is terminable.

Definitions 7 and 8 are used to ensure that an $IBPF$ would not run indefinitely. Instead it must reach some ended states to represent that a transaction process has either been successfully completed or terminated for some reason. Then, the state space is limited. These properties can be validated by CPNtools. If an $IBPF$ is rational, we can search the state space and verify the *Transaction-Logical Consistency*.

Definition 9 Let $IBPF = (P \cup P_{IN} \cup P_O, T, A, \Sigma, V, C, G, E, I)$. Then, a 2-tuple ($IBPF'$,

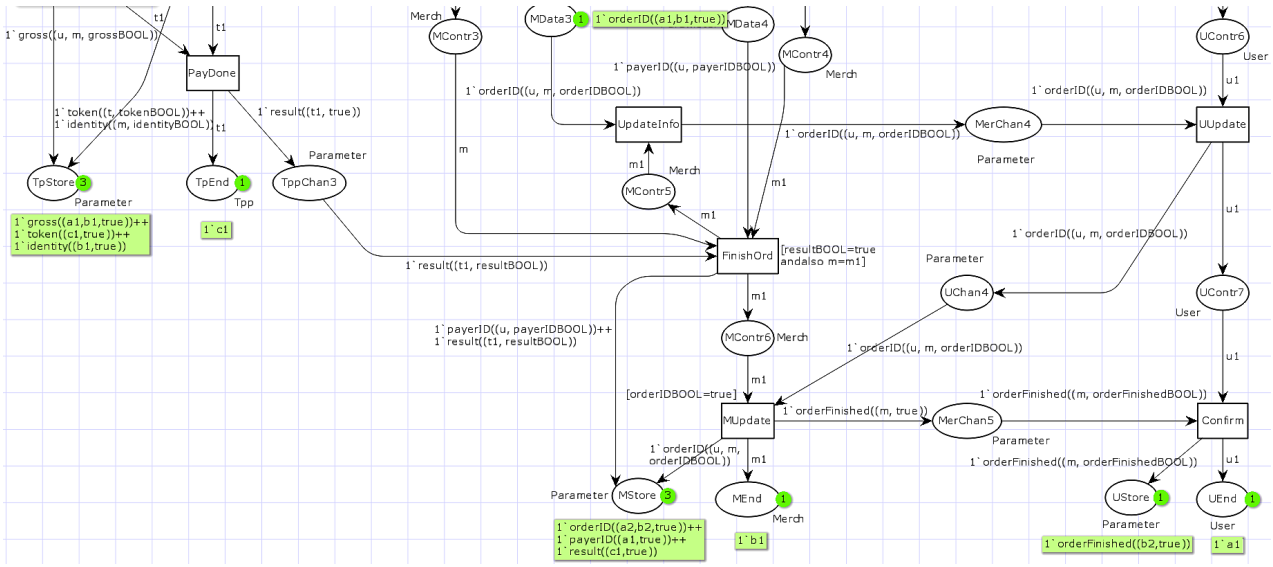


Fig. 9. A Transaction-Finished State.

Algorithm 1: Verify the *Transaction-Logical Consistency* under an *Initial Configuration*

Input: $IBPF' = (P \cup P_{IN} \cup P_O, T, A, \Sigma, V, C, G, E', I)$, initial marking M_0
Output: A marking set Ω
1. $\Omega = \emptyset$, $\Pi = \{M_0\}$;
2. Let M_0 be the root node, and mark it with "Intermediate";
3. **while** *"Intermediate" nodes exist* **do**
   Choose an arbitrary "Intermediate" node as M;
   3.1 **if** *M is a Transaction-Finished State and does not satisfy Transaction-Logical Consistency* **then**
      $\Omega = \Omega \cup \{M\}$;
      Goto 3;
   end
   3.2 **else if** $\forall t \in T \rightarrow \neg M \xrightarrow{t}$ **then**
      $\Pi = \Pi \cup \{M\}$;
      Goto 3;
   end
   3.3 **else**
      for *every enabled $t \in T$ at M* **do**
         $M' = M \xrightarrow{t}$;
         if $M' \notin (\Omega \cup \Pi)$ **then**
            Mark M' as "Intermediate";
            $\Pi = \Pi \cup \{M'\}$;
         end
      end
   end
   3.4 Remove mark "Intermediate" of M;
end

M_0) is called an *Initial Configuration* that is rational, in which M_0 is an initial marking of $IBPF$, $IBPF' = (P \cup P_{IN} \cup P_O, T, A, \Sigma, V, C, G, E', I)$.

The construction of *Initial Configuration* is based on the logical vulnerability which is to be verified. It can be seen as an initial state of an *IBPF*, which represents the behavior conditions before an online transaction begins. E' either shows some variations from E such that changes will be noted on some BOOL values on the corresponding arcs, or E'=E. For example, we can make the $IBPF$ and the initial marking of Fig. 6 as an *Initial Configuration*, which is used to verify the vulnerability of "a user can pay for a cheap order but check out an expensive one", "a user pay for another user's order" and "a user can pay himself/herself for an item he/she buys in an online shop" [7], [13]. In this case, E' of the *Initial Configuration* is the same as E, and of course, $IBPF'$ is the same as $IBPF$. If we want to verify the vulnerability of "a user can pay less" [7], [13], which is also listed as "Tainted gross" in Tab. 1, we can make a new *Initial Configuration* and change the $IBPF$ presented in Fig. 6, with the following changes $M_0(Tpp) = 1'c1$, $M_0(Merch) = 1'b1$, $M_0(Shopper) = 1'a1$, and $E'(SendPay, UChan2) = 1'token((t, tokenBOOL)) + +1'gross((u, m, false))$ where $E(SendPay, UChan2) = 1'token((t, tokenBOOL)) + +1'gross((u, m, grossBOOL))$.

A payment is assumed secure when both the authenticity and integrity of the payment status are verified, and this is guaranteed by the *Transaction-Logical Consistency* in this work. Algorithm 1 is the process of searching the state space and verifying the *Transaction-Logical Consistency* under an *Initial Configuration*. The marking set Ω that do not satisfy *Transaction-Logical Consistency* is the output of Algorithm 1. For example, Fig. 9 is a marking belonging to Ω which is found under the *initial configuration* of Fig. 6.

Theorem 1 Algorithm 1 can be terminated.

Proof. According to Definitions 7 and 8, the input $IBPF'$ of Algorithm 1 has a finite state space under an initial marking M_0. Algorithm 1 searches the markings from M_0, and adds the markings those not satisfying *Transaction-Logical Consistency* to Ω in Step 3.1. Other constructed markings are added to Π in Steps 3.2 and 3.3. In Step 3.3, new produced marking those do not belong to Ω is marked as "Intermediate", and the notation "Intermediate" of current marking is removed in Step 3.4. This represents that the marking set of "Intermediate" is continuously updated until it becomes empty, so it is limited. Above descriptions illustrate that Algorithm 1 can be terminated. Algorithm 1 is a breadth-first traversal method. It only searches a part of the state space. □

We summarize the verification procedure based on the above methods as below.

1) Establish the $IBPF$ according to the design specification.

2) Construct the *Initial Configuration* according to some logic vulnerability.

3) Use Algorithm 1 to obtain Ω and to analyze the corresponding logic vulnerability.

4) Goto Step 2 to verify another logic vulnerability.

TABLE II
VERIFICATION RESULT OF THE CASE

Initial Configurations	Vulnerabilities	Exist?
The $IBPF$ of Fig. 6; $M_0(Tpp)$=$1'c1$, $M_0(Merch)$=$1'b1$++$1'b2$, $M_0(Shopper)$=$1'a1$++$1'a2$	"pay for a cheap order but check out an expensive one"	Yes
The $IBPF$ of Fig. 6; $M_0(Tpp)$=$1'c1$, $M_0(Merch)$=$1'b1$++$1'b2$, $M_0(Shopper)$=$1'a1$++$1'a2$	"pay for another users' order"	Yes
The $IBPF$ of Fig. 6; $M_0(Tpp)$=$1'c1$, $M_0(Merch)$=$1'b1$++$1'b2$, $M_0(Shopper)$=$1'a1$++$1'a2$	"a user can pay himself for an item he buys in an online shop"	Yes
The $IBPF$ of Fig. 6 with $E(SendPay, UChan2) = 1'token((t, tokenBOOL))$ ++$1'gross((u, m, false))$; $M_0(Tpp)$=$1'c1$, $M_0(Merch)$=$1'b1$, $M_0(Shopper)$=$1'a1$	"a user can tamper gross and pay less"	No
The $IBPF$ of Fig. 6 with $E(UUpdate, UChan4) = 1'orderID((u, m, false))$; $M_0(Tpp)$=$1'c1$, $M_0(Merch)$=$1'b1$, $M_0(Shopper)$=$1'a1$	"a user can tamper orderID"	No
The $IBPF$ of Fig. 6 with $E(SendPay, UChan2) = 1'token((t, false))$++$1'gross((u, m, grossBOOL))$; $M_0(Tpp)$=$1'c1$, $M_0(Merch)$=$1'b1$, $M_0(Shopper)$=$1'a1$	"a user can tamper the token of TPP and replay it"	No

Algorithm 1 is a part of the verification process, and the summarized verification procedure is the sketch of the whole verification procedure. Thus, notable logic vulnerabilities of the case discussed in Section 2 can be identified using our proposed methodology. Tab. II illustrates the *Initial Configurations* and their corresponding vulnerabilities, where the third column represents whether the vulnerabilities exist in the e-commerce business interaction process.

Our methods can be potentially implemented to detect various kinds of logic vulnerabilities of various real cases, with suitable modifications to the *Initial Configurations* and arc expressions. If the system development process is strictly consistent with the proposed methods and the obtained system design, the implementation system is accordant with the model, and ultimately more secure than the system without our modeling and analyzing process. Under the *Initial Configurations*, using the proposed modeling and analyzing methods, the known or unknown vulnerabilities can be discovered. Existing related works introduced in Section 1 focus mostly on the testing or detection of implementation systems at the code-level, where our approach is at the the design level and application level. Our approach is a formal methodology that is generic for the relevant types of logic vulnerabilities. The proposed methods can be applied to other design specifications of e-commerce business processes as well. Compared with previous related works, this paper has different research objectives and emphases. The proposed methodology can complement the traditional methods. However, the proposed work in this paper cannot deal with the vulnerabilities of network level, the ones caused by hardware accidents, and the non-logical flaw issues like the account leaking and decryption.

V. Conclusion

This paper proposed a novel methodology to detect the rising logic vulnerabilities in the distributed business interaction processes of e-commerce systems during the design phase. A systematic approach is presented in this work, including formal definitions, modeling scheme and analyzing procedure. The efficiencies of our proposed approach in detecting logic vulnerabilities has been discussed and demonstrated through a case study. The proposed methodology can also be potentially applied to other similar e-commerce business interaction processes by suitably defining the business processes and datasets. This methodology can determine whether a system design is immune to logic vulnerabilities. Then, it can help modify the incorrect design, and eventually guarantee the security of system design against logic vulnerabilities. Despite the effectiveness of our proposed approach, there is scope for further developing the related tools or functions for supporting the full automation of the proposed methodology, or combining third party components with CPNtools.

References

[1] F. Sun, L. Xu, and Z. Su, "Detecting logic vulnerabilities in e-commerce applications." in *NDSS*, 2014.

[2] W. Yu, Y. Wang, L. Liu, Y. An, B. Yuan, and J. Panneerselvam, "A multiperspective fraud detection method for multiparticipant e-commerce transactions," *IEEE Transactions on Computational Social Systems*, pp. 1–13, 2023.

[3] L. Xing, Y. Chen, X. Wang, and S. Chen, "Integuard: Toward automatic protection of third-party web service integrations." in *NDSS*, 2013.

[4] Y. Xie, G. Liu, C. Yan, C. Jiang, M. Zhou, and M. Li, "Learning transactional behavioral representations for credit card fraud detection," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2022.

[5] Y. Xie, G. Liu, C. Yan, C. Jiang, and M. Zhou, "Time-aware attention-based gated network for credit card fraud detection by extracting transactional behaviors," *IEEE Transactions on Computational Social Systems*, pp. 1–13, 2022.

[6] E. Chen, S. Chen, S. Qadeer, and R. Wang, "A practical approach to protocol-agnostic security for multiparty online services," 2014.

[7] R. Wang, S. Chen, X. Wang, and S. Qadeer, "How to shop for free online–security analysis of cashier-as-a-service based web stores," in *Security and Privacy (SP), 2011 IEEE Symposium on*. IEEE, 2011, pp. 465–480.

[8] Y. Wang, W. Yu, P. Teng, G. Liu, and D. Xiang, "A detection method for abnormal transactions in e-commerce based on extended data flow conformance checking," *Wireless Communications and Mobile Computing*, vol. 2022, no. 3, 2022.

[9] E. Y. Chen, S. Chen, S. Qadeer, and R. Wang, "Securing multiparty online services via certification of symbolic transactions," in *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 2015, pp. 833–849.

[10] C. Liu, Q. Zeng, L. Cheng, H. Duan, M. Zhou, and J. Cheng, "Privacy-preserving behavioral correctness verification of cross-organizational workflow with task synchronization patterns," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1037–1048, 2021.

[11] C. W. Enumeration, "Cwe-840 business logic errors," 2014.

[12] C. Cwe, "472: External control of assumed-immutable web parameter."

[13] G. Pellegrino and D. Balzarotti, "Toward black-box detection of logic flaws in web applications." in *NDSS*, 2014.

[14] A. Sudhodanan, A. Armando, R. Carbone, L. Compagna *et al.*, "Attack patterns for black-box security testing of multi-party web applications." in *NDSS*, 2016.

[15] D. Hirschberger, D.-I. V. Mladenov, M. S. C. Mainka, and J. Schwenk, "Bachelor thesis cashier-as-a-service based webshops overview and steps towards security testing," 2016.

[16] D. Balzarotti, M. Cova, V. V. Felmetsger, and G. Vigna, "Multi-module vulnerability analysis of web-based applications," in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 25–35.

[17] A. P. Fung, K. Cheung, and T. Wong, "Faith: Scanning of rich web applications for parameter tampering vulnerabilities," Tech. Rep., 2012.

[18] G. McGraw and J. Viega, "Building secure software," in *RTO/NATO Real-Time Intrusion Detection Symp*, 2002.

[19] K. Jensen and L. M. Kristensen, *Coloured Petri nets: modelling and validation of concurrent systems*. Springer Science & Business Media, 2009.

[20] D. C. Latham, "Department of defense trusted computer system evaluation criteria," *Department of Defense*, 1986.

[21] H. Liu, Y. Feng, J. Li, and J. Luo, "Robust petri net controllers for flexible manufacturing systems with multitype and multiunit unreliable resources," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 3, pp. 1431–1444, 2023.

[22] W. M. van der Aalst, N. Lohmann, and M. La Rosa, "Ensuring correctness during process configuration via partner synthesis," *Information Systems*, vol. 37, no. 6, pp. 574–592, 2012.

[23] Q. Guo, W. Yu, and L. Qi, "Multi-factor balanced feedback and reliability analysis of adaptive cruise control system based on petri nets," in *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2022, pp. 1084–1089.

[24] S. Wang, X. Guo, O. Karoui, M. Zhou, D. You, and A. Abusorrah, "A refined siphon-based deadlock prevention policy for a class of petri nets," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 1, pp. 191–203, 2023.

[25] W. Yu, Z. Ding, L. Liu, X. Wang, and R. D. Crossley, "Petri net-based methods for analyzing structural security in e-commerce business processes," *Future Generation Computer Systems*, vol. 109, pp. 611 – 620, 2020.

[26] C. Liu, H. Duan, Q. Zeng, M. Zhou, F. Lu, and J. Cheng, "Towards comprehensive support for privacy preservation cross-organization business process mining," *IEEE Transactions on Services Computing*, vol. 12, no. 4, pp. 639–653, 2019.

[27] M. Wang, Z. Ding, P. Zhao, W. Yu, and C. Jiang, "A dynamic data slice approach to the vulnerability analysis of e-commerce systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 10, pp. 3598–3612, 2020.

[28] W. Yu, C. Yan, Z. Ding, C. Jiang, and M. Zhou, "Analyzing e-commerce business process nets via incidence matrix and reduction," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 1, pp. 130–141, 2018.

[29] M. Wang, G. Liu, C. Yan, and C. Jiang, "Modeling and vulnerable points analysis for e-commerce transaction system with a known attack," in *9th International Conference on Security, Privacy, and Anonymity in Computation, Communication, and Storage*, 2016, pp. 422–436.

[30] D. Wichers, "Owasp top-10 2013," *OWASP Foundation, February*, 2013.

[31] W. P. D. I.-X. WfMC, "Process definition language," *Document Status-1.0 Final Draft., Document Number WFMC-TC-1025 Workflow Management Coalition, Lighthouse Point, FL*, 2002.

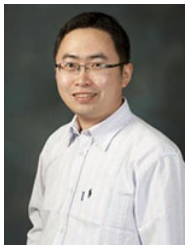
[32] B. Yang and H. Hu, "Maximally permissive robustness analysis of automated manufacturing systems with multiple unreliable resources," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–13, 2022.

[33] Y. Dong, Z. Li, and N. Wu, "Symbolic verification of current-state opacity of discrete event systems using petri nets," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 12, pp. 7628–7641, 2022.

[34] A. L. Doupé, "Advanced automated web application vulnerability analysis," Ph.D. dissertation, University of California, Santa Barbara, 2014.

[35] M. Panti, L. Spalazzi, and S. Tacconi, "Verification of security properties in electronic payment protocols," 2002.

Wangyang Yu received the M.S. degree from Shandong University of Science and Technology, Qingdao, China, in 2009, and Ph.D. degree from Tongji University, Shanghai, China, in 2014. He is currently an Associated Professor with the College of Computer Science, Shaanxi Normal University, Xi'an, China. He was also a visiting scholar from 2016 to 2017 at University of Derby, UK. His research interests include the theory of Petri nets, formal methods in software engineering and trustworthy software.



Lu Liu is the Head of School of Informatics at the University of Leicester, UK. Prof. Liu worked as a Research Fellow of the WRG e-Science Centre at Leeds University on the EPSRC/BAE funded NECTISE Project and the CoLaB Project which was jointly funded by the EPSRC and the China-863 Program. Prof. Liu has over 120 scientific publications in reputable journals (e.g. IEEE Transactions on Computers, IEEE Transactions on Service Computing, ACM Transactions on Embedded Computing Systems). Prof. Liu has secured many research projects which are supported by UK research councils, BIS and leading UK industries. Prof. Liu serves as an Editorial Board member of 6 international journals and the Guest Editor for 5 international journals. He has chaired over 20 international conference workshops and presently or formerly serves as the program committee member for over 50 international conferences and workshops. He is a Fellow of BCS (British Computer Society).

Xiaoming Wang received his Ph.D. degree in computer theory and software from Northwest University, Xi'an, China, in 2005. He is currently a professor and Ph. D. supervisor in Shaanxi Normal University, Xi'an, China. Prof. Wang is head of School of Computer Science, and he was also a research professor(visiting scholar) from 2007 to 2008 at Georgia State University, USA. His current research interests include network security, access control, pervasive computing, wireless sensor network, opportunistic networks, workflow management system, and system dynamics. His research has been supported by the National Science Foundation of China (NSFC), Key Research Project of Ministry of Education of China. Prof. Wang has authored and coauthored more than 40 publications in journal, books and international conference proceedings.

Ovidiu Bagdasar is a Associate Professor in Mathematics at the University of Derby. He received a PhD in Applied Mathematics from University of Nottingham, UK in 2011, and a PhD in Mathematics from UBB Cluj-Napoca, Romania in 2015. Ovidiu's research in Applied Mathematics, Optimization, Traffic Modelling, Discrete Mathematics and Recurrent Sequences produced more than 50 journal articles, book chapters, and presentations at prestigious international conferences. Ovidiu has been involved in the optimization of production lines and the automatic design of ceramic colours and predictive analytics. Ovidiu was also organiser of conferences and was Editor or Guest Editor for international journals.

John Panneerselvam is a Lecturer in Computing at the University of Leicester, UK. His current research is focused on energy efficient cloud systems and he has published his recent research works in notable peer reviewed international conferences, journals and as book chapters. He is an active member of IEEE and his research interests include Cloud Computing, Big Data Analytics, Opportunistic Networking and P2P Computing.