Enhancing Security in E-Business Processes: Utilizing Dynamic Slicing of Colored Petri Nets for Logical Vulnerability Detection

Enhancing Security in E-Business Processes: Utilizing Dynamic Slicing of Colored Petri Nets for Logical Vulnerability Detection

Wangyang Yu[a,b], Jie Feng[a,*], Lu Liu[c,*], Xiaojun Zhai[d], Yumeng Cheng[a]

[a]School of Computer Science, Shaanxi Normal University, No. 620, West Chang'an Street, Chang'an District, Xi'an, 710119, China
[b]Anhui Province Engineering Laboratory for Big Data Analysis and Early Warning Technology of Coal Mine Safety, Anhui University of Science and Technology, An'hui, 232001, China
[c]School of Informatics,University of Leicester, LE1 7RH Leicester, UK
[d]School of Computer Science and Electronic Engineering, University of Essex, Colchester, CO4 3SQ, UK

Abstract

The field of e-business covers multiple aspects and has undergone rapid development, profoundly changing our transaction methods and shopping experiences. However, with the increasing complexity of its business processes, logical vulnerabilities have become an inevitable issue. These logical vulnerabilities can lead to a range of security problems, seriously threatening business stability and consumer trust. To address the challenge of logical vulnerabilities in e-business, we developed a model based on Colored Petri Nets (CPN), the Interactive Business Process Fusion (IBPF) net, which is adept at identifying such vulnerabilities during the design phase. However, the analysis methods for IBPF net still urgently need innovation. In addressing this issue, we use dynamic slicing techniques to analyze IBPF net, serving as a method for revealing logical vulnerabilities. We obtain backward slice, partial forward slice, and bidirectional slice through the slicing algorithms. Eventually, these three types of slices are merged to form the final dynamic slice. This technique, which involves a more targeted analysis than examining the entire IBPF net, simplifies analysis process and prevents state space explosion, thereby providing a distinct advantage. The results of this research are of great value in enhancing system reliability, reducing maintenance costs, and providing analysis techniques in the field of e-business security.

Keywords:
E-business security, Logic vulnerability, Colored Petri Nets, Dynamic slicing

1. Introduction

To clearly describe e-business processes and detect logical vulnerabilities at the design stage, it is essential to model and analyze these processes using formal methods. In previous studies, we proposed the Interactive Business Process Fusion (IBPF) net – a business interaction process model based on Colored Petri Nets (CPN) [1]. It integrates the data flow and control flow features of Colored Petri Nets, meeting the demands for logical complexity and data diversity in business processes. This model correctly reflects the interactions among multiple business entities and the flow of business data, depicting not only the sequence and conditions of business activities but also effectively handling concurrency and dependencies in data flows. Furthermore, IBPF net provides rigorous definition and verification mechanisms for e-business processes, ensuring the accuracy and integrity of transaction processing. By analyzing IBPF net, potential logical vulnerabilities in business processes can be identified, assisting developers in timely detection and correction of these vulnerabilities.

IBPF offers an effective method for enhancing the integrity and reliability of e-business processes. However, analyzing the entire IBPF net directly presents challenges in today's complex e-business process environments with large system scales, mainly due to lengthy processing times and the state space explosion problem. To overcome these challenges and enrich the analysis methods of IBPF net, this paper introduces the dynamic slicing techniques. Through this approach, smaller subnets can be defined and extracted from complex IBPF net for detailed analysis. This method not only reduces the overall size of the model and the generated state space but also ensures the integrity and accuracy of subnets in functional and security analysis.

The structure of this paper is as follows: Section 2 introduces related work. Section 3 presents the definitions related to dynamic slicing techniques, and Section 4 introduces the principles of the algorithms for dynamic slicing techniques. Section 5 demonstrates the application process of this technique through a case study. Section 6 summarizes the content of the research.

2. Related work

E-business maximizes value through innovative approaches, profoundly transforming our transaction modes and shopping experiences [2, 3]. With continuous technological advancements and evolving consumer habits, the e-business sector is poised to play an increasingly significant role in the global economy [4].

The composition of e-business has become more diversified and complex, offering numerous conveniences. However, this

*Corresponding author
Email addresses: fjie@snnu.edu.cn (Jie Feng), l.liu@leicester.ac.uk (Lu Liu)

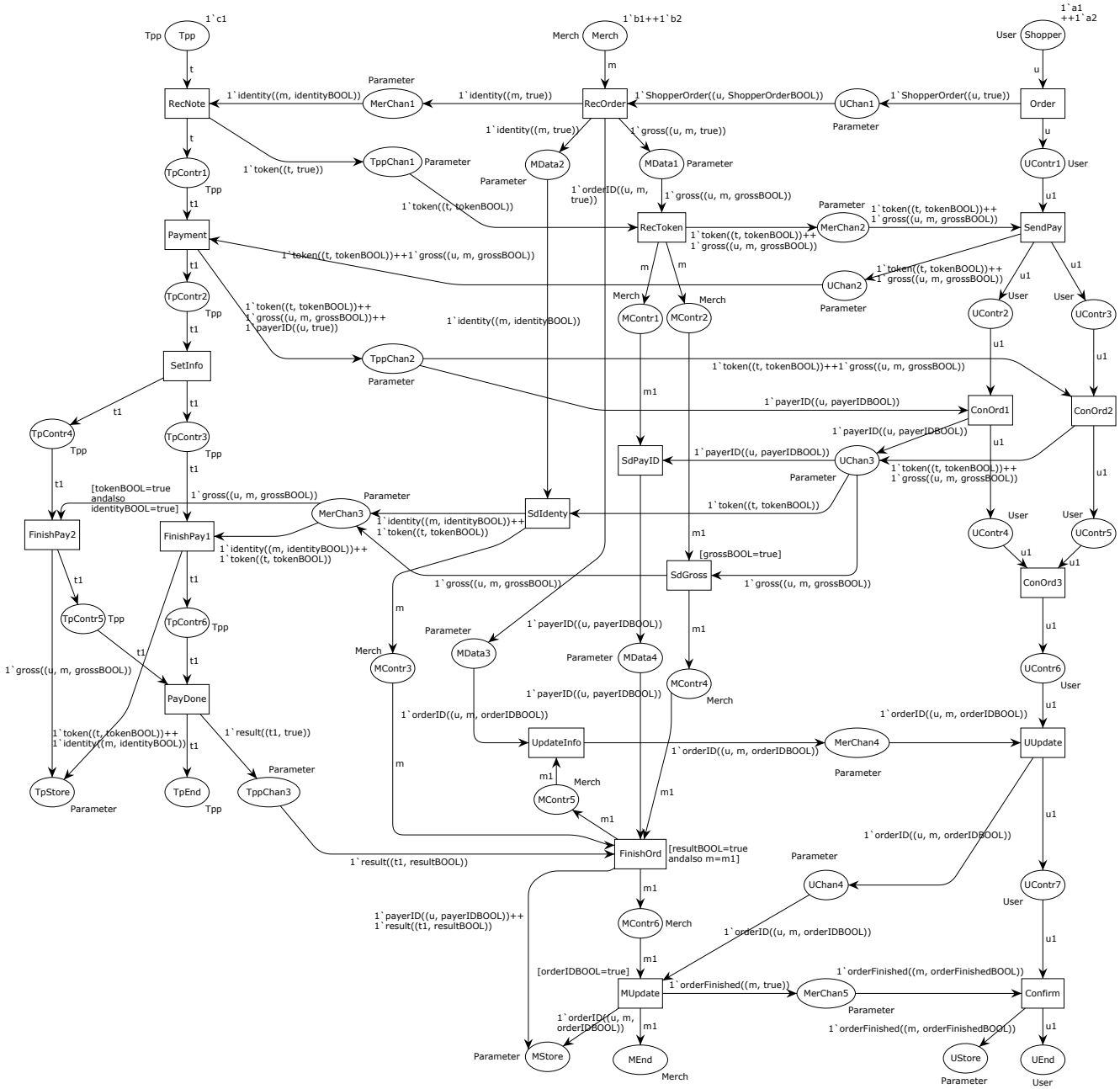


Figure 1: An IBPF net of an e-commerce process

complexity can lead to logical vulnerabilities in the design of e-business systems [5–8]. For instance, the lack of comprehensive data integrity checks during transactions can expose sensitive user data to unauthorized access and misuse [9, 10]. Additionally, malicious entities might forge or alter data to gain undue advantages [6, 11]. These logical vulnerabilities not only degrade the consumer shopping experience but also pose severe threats to corporate reputation and financial security, making it imperative to devise effective solutions [12].

To address the aforementioned issues, we proposed the IBPF net. It is a novel model based on CPN for describing business interaction processes that involve distributed multi-participants and multi-sessions. It effectively captures key attributes of business interactions, including the logical structure of business flows, data definitions, interaction behaviors, and fundamental properties. It also enables the analysis of logical vulnerabilities in business interaction processes during the design phase, ensuring the security and consistency of e-business interactions [1]. Our previous research proposed an IBPF net that describes an e-commerce interaction process involving users, merchants, and third-party payment platform, as shown in Fig. 1 [1].

However, there are currently no adequate analysis methods of IBPF net. Therefore, this search focuses on the analysis methods of IBPF net. Definitions 1–3 related to IBPF net are derived from [1], where detailed illustrations can be found.

Definition 1 (*LBP*)**.** 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 $LBP_E = (P,\ T \cup \{\tau\},\ A \cup \{(\tau,\alpha),(\beta,\tau),(\gamma,\tau)\},\ \Sigma,\ V, C, G', E', I)$ be the trivial extension of LBP, in which $E' : \{(\tau,\alpha),(\beta,\tau),(\gamma,\tau)\} \to EXPR_V$, and $a \in A \to E'(a) = E(a)$; $t \in T \to G'(t) = G(t)$, and τ has no guard. Then, LBP_E is strongly connected.

Logical Business Process (LBP) is used to describe the internal business processes of each participant to clearly understand the role and behavior of each entity in the transaction process [1]. Fig. 1 illustrates an e-commerce process model, focusing on the user's journey from placing an order to its completion, the process by which merchants receive and fulfill orders, and the workflow of third-party payment services in managing the payment process.

Definition 2 (*LBPC*)**.** Let $LBP = (P, T, A, \Sigma, V, C, G, E, I)$ be a LBP. Then, a LBP 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) $C' : (P_{IN} \cup P_O) \to \Sigma$, and $p \in P \to C'(p) = C(p)$.

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

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

Logical Business Process with Channels (LBPC) expands on LBP by integrating communication channels, adding input and output channels to map out transactional interactions among different parties [1]. This approach enables data flow interaction within complex, multi-participant systems. In Fig. 1, the information of merchants, users, and third-party payment platform is transmitted through related input channels and output channels [1].

Definition 3 (*IBPF*)**.** Let $LBPC_n = (P \cup P_{IN_n} \cup P_{ON_n}, T_n, A_n, \Sigma_n, V_n, C_n, G_n, E_n, I_n)$, $n, m \in \mathbb{N}_+$, $n \leq m$, $m \geq 2$, be m LBPCs satisfying Definition 2, and $\bigcup_{n=1}^{m} P_{IN_n} = \bigcup_{n=1}^{m} P_{ON_n}$. Then, $IBPF = (P \cup P_{IN} \cup P_O, T, A, \Sigma, V, C, G, E, I) = LBPC_1\ \Phi LBPC_2 \Phi \ldots \Phi LBPC_m$ is called an IBPF net.

We construct the IBPF net by integrating LBPC with input and output channels. IBPF net not only provides a clear and intuitive description of the interaction processes among various business entities but also allows for the identification of logical vulnerabilities in the system during the design phase by altering the initial marking of the IBPF net [1]. This lays a solid foundation for building secure and reliable e-business processes.

Program slicing techniques aim to streamline program code by analyzing data and control flows within a program, retaining only those parts relevant to specific functionalities or behaviors [13]. This technique is widely applied in software engineering and system analysis, facilitating the understanding, debugging, and analysis of complex programs [14–17]. As a formal modeling tool, Petri nets are extensively used to describe and analyze the behaviors of concurrent and distributed systems [18]. Applying slicing techniques to Petri nets enables the extraction of net elements directly related to specific processes or events, effectively simplifying the model and improving resource utilization and analytical efficiency [19]. This has been reflected in several papers such as [20–22].

In Petri nets slicing techniques, there are mainly two categories: static slicing and dynamic slicing [23]. Static algorithms operate on static Petri nets, thus not considering the token game [24]. Dynamic slicing, on the other hand, takes into account the runtime state and behavior of the system, analyzing the system based on specific sequences of transition occurrences. Compared to the former, for systems processing large-scale data, dynamic slicing techniques exhibit superior analytical efficiency.

Static and dynamic slicing are further differentiated into forward and backward slicing [25]. These methods, based on specific slicing criteria and algorithms, commence from key elements and trace either forward or backward to identify all elements involved in particular sequences of changes or related path flows.

Utilizing dynamic slicing techniques to analyze the IBPF net enables us to concentrate on those aspects critical for detecting systemic logical vulnerabilities or conducting performance analysis, thereby eliminating the need for a comprehensive examination of the entire model. Particularly in the context of complex, large-scale e-business systems, this approach enhances analytical efficiency, circumvents the issue of state space explosion, and elevates the accuracy of logical vulnerabilities identification. Upon detecting relevant logical vulnerabilities, we can promptly optimize and adjust the model to mitigate the complexity and costs associated with future system maintenance, reduce long-term maintenance expenses, and bolster the overall reliability and security of e-business systems.

3. Related definitions of dynamic slicing techniques

To obtain the desired dynamic slice, it is first necessary to establish a slicing criterion [26], which may include:

(1) A set of variables that we are interested in. These variables are crucial to the stability and performance of the entire system. If tampered with, they could introduce serious logical vulnerabilities to the system.

(2) A set of places that we are interested in. They are able to store the final state of key variables.

(3) Key transitions. Within IBPF net, key transitions are those that allow for bidirectional tracing to places or transitions where key variables pass through. By mapping out the routes of these key transitions, we can realize the interactions among various variables.

(4) It is also crucial to consider the initial state of the net system, which can help us extract smaller slices.

For different types of slices, the slicing criterion we use vary. First, we need to obtain a backward slice. The slicing criterion for a backward slice is as follows:

Definition 4 (*Slicing criterion for a backward slice*)**.** Given $IBPF = (P, T, A, \Sigma, V, C, G, E, I)$, we say $\langle Q_1, W\rangle$ is a slicing criterion of a backward slice if the following conditions hold:

(1) W is a set of variables that we are interested in, and $W \subset V$.

(2) Q_1 is a set of places that we are interested in, the final state of the key variables stored in the place inside Q_1, and $Q_1 \subset P$.

Definition 5 (*Backward slice*)**.** Given $IBPF = (P, T, A, \Sigma, V, C, G, E, I)$, let $\langle Q_1, W\rangle$ be a slicing criterion. For $IBPF_1 = (P_1, T_1, A_1, \Sigma_1, V_1, C_1, G_1, E_1, I_1)$, we say that $IBPF_1$ is a backward slice of $IBPF$ under the slicing criterion $\langle Q_1, W\rangle$ if the following conditions hold:

(1) $IBPF_1$ is a subnet of $IBPF$.

(2) $\langle Q_1, W\rangle$ conforms to Definition 4.

(3) In $IBPF_1$, the set of variables associated with any arc intersects with W, and the intersection is non-empty.

Upon obtaining the backward slice, we consider the model's initial marking, focusing on resource availability in the business process, like active merchants or users and third-party services. We then start from the transitions that are enabled in the initial state and employ a dynamic forward tracing method to obtain a partial forward slice. The slicing criterion for a partial forward slice is as follows:

Definition 6 (*Slicing criterion for a partial forward slice*)**.** Given $IBPF = (P, T, A, \Sigma, V, C, G, E, I)$, we say $\langle Q_2, M_0\rangle$ is a slicing criterion of a partial forward slice if the following conditions hold:

(1) Q_2 is the set of places in $IBPF$ where resources are not empty under M_0, i.e., for any $p \in Q_2, I(p) \neq \emptyset$.

(2) M_0 is the initial marking of $IBPF$.

Definition 7 (*Partial forward slice*)**.** Given $IBPF = (P, T, A, \Sigma, V, C, G, E, I)$, $IBPF_1 = (P_1, T_1, A_1, \Sigma_1, V_1, C_1, G_1, E_1, I_1)$, let $\langle Q_2, M_0\rangle$ be a slicing criterion. For $IBPF_2 = (P_2, T_2, A_2, \Sigma_2, V_2, C_2, G_2, E_2, I_2)$, we say that $IBPF_2$ is a partial forward slice of $IBPF$ with respect to $IBPF_1$ under the slicing criterion $\langle Q_2, M_0\rangle$ if the following conditions hold:

(1) $IBPF_1$ is a backward slice of $IBPF$ conforming to Definition 5.

(2) $\langle Q_2, M_0\rangle$ conforms to Definition 6.

(3) $IBPF_2$ is a subnet of $IBPF$.

(4) $IBPF_2$ must include all the places in Q_2, and the new net obtained by taking the union of $IBPF_1$ and $IBPF_2$ is a subnet of $IBPF$.

Considering that backward slicing techniques typically only trace the flow paths of specific related variables and fail to fully capture the interaction information of variables related to participants such as merchants, users, and third-party services, this study proposes an innovative bidirectional slicing method. By correlating these relatively independent execution paths, we are able to construct a more comprehensive dynamic slice. The slicing criterion for a bidirectional slice is as follows:

Definition 8 (*Slicing criterion for a bidirectional slice*)**.** Given $IBPF = (P, T, A, \Sigma, V, C, G, E, I)$ and $IBPF_1 = (P_1, T_1, A_1, \Sigma_1, V_1, C_1, G_1, E_1, I_1)$, we say $\langle transition, W\rangle$ is a slicing criterion for a bidirectional slice of $IBPF$ with respect to $IBPF_1$ if the following conditions hold:

(1) $IBPF_1$ is a backward slice of $IBPF$ conforming to Definition 5.

(2) W is the set of variables that we are interested in, consistent with the W in the slicing criterion of $IBPF_1$.

(3) *transition* is a key transition that facilitates the interaction of key variables with participants in business transactions, such as merchants, users, and third-party services, within the business process.

Definition 9 (*Bidirectional slice*)**.** Given $IBPF=(P, T, A, \Sigma, V, C, G, E, I)$ and $IBPF_1 = (P_1, T_1, A_1, \Sigma_1, V_1, C_1, G_1, E_1, I_1)$, let $\langle transition, W\rangle$ be a slicing criterion. For $IBPF_3 = (P_3, T_3, A_3, \Sigma_3, V_3, C_3, G_3, E_3, I_3)$, we say that $IBPF_3$ is a bidirectional slice of $IBPF$ with respect to $IBPF_1$ under the slicing criterion $\langle transition, W\rangle$ if the following conditions hold:

(1) $IBPF_1$ is a backward slice of $IBPF$ conforming to Definition 5.

(2) $\langle transition, W\rangle$ conforms to Definition 8.

(3) $IBPF_3$ stores the shortest paths from *transition* connection to $IBPF_1$, and the new net obtained by taking the union of $IBPF_1$ and $IBPF_3$ is a subnet of $IBPF$.

Following the incorporation of bidirectional slicing, the dynamic slice obtained no longer feature independent paths related to key variables. Instead, the elements within the variables on each path are mutually constrained, facilitating a synergistic change in states and values. This ensures that the variables' value and state changes during execution more closely align with the original model's operational effects. Consequently, the dynamic slicing can capture and reflect the dynamic interactions and comprehensive influences of the variables within the model with greater precision. Thus, it enhances the accuracy and reliability of analyzing the IBPF net using dynamic slicing.

By merging $IBPF_1$, $IBPF_2$ and $IBPF_3$ from Definition 5, Definition 7 and Definition 9, we can effectively isolate the parts most relevant to system vulnerabilities detection, obtaining the final dynamic slice. The dynamic slicing algorithms will be discussed in detail in the next section.

4. Algorithms for dynamic slicing

In the previous section, we introduced several definitions of slices relevant to this study. In this section, we will focus on analyzing the algorithms associated with these slices.

The backward slicing of IBPF takes into account the dependencies among places, transitions, and key variables. The algorithm starts with the set of places Q_1 and traces back the token flow paths related to the variables in set W. The specific steps of the analysis are as shown in Algorithm 1.

Algorithm 1: Backward slice of IBPF

Input: $IBPF = (P, T, A, \Sigma, V, C, G, E, I), \langle Q_1, W \rangle$
Output: $IBPF_1 = (P_1, T_1, A_1, \Sigma_1, V_1, C_1, G_1, E_1, I_1)$

$P_1 = Q_1, V_1 = W, T_1 = \emptyset, A_1 = \emptyset, \Sigma_1 = \emptyset, C_1 = \emptyset, G_1 = \emptyset, E_1 = \emptyset, I_1 = \emptyset$.
while $(Q_1 \neq \emptyset)$ **do**
  for *each* $p_i \in Q_1$ **do**
    Add related attributes of p_i to $IBPF_1$
    for *each* $t_i \in {}^\bullet p_i$ **do**
      if *there exists* $v_m \in W$ *affected by* $E(t_i, p_i)$ **then**
        All attributes of t_i and the elements related to (t_i, p_i) are added to $IBPF_1$
        for *each* $p_j \in {}^\bullet t_i$ **do**
          if $p_j \notin Q_1$ ***and*** *there exists* $v_m \in W$ *affected by* $E(p_j, t_i)$ **then**
            $Q_1 = Q_1 \cup \{p_j\}$
            Add the elements related to (p_j, t_i) to $IBPF_1$
    $Q_1 = Q_1 \setminus \{p_i\}$
return $IBPF_1 = (P_1, T_1, A_1, \Sigma_1, V_1, C_1, G_1, E_1, I_1)$

Algorithm 1 iterates over each place p_i in the set of places Q_1, tracing back the data flow paths related to key variables in reverse. When it is determined that a key variable in the set W is transmitted by the occurrence of a preceding transaction t_i of p_i, both p_i and t_i, along with their arc relationship, are added to the backward slice $IBPF_1$. Besides, Q_1 is expanded by adding the input place p_j of t_i for further analysis. This ensures the accurate capture of the flow paths of key variables. Once the set Q_1 has been fully traversed, the backward slicing algorithm for $IBPF$ concludes, at which point the algorithm returns the constructed backward slice $IBPF_1$.

We understand that a backward slice only represents the flow paths of key variables during system runtime. However, we also need to consider the system's initial state, such as online users, open shops, and other factors. We must trace forward from the events that can occur in the system's initial marking, all the way until all influencing factors are included in a partial forward slice and need to connect with the backward slice $IBPF_1$. The specific analysis steps are outlined in Algorithm 2.

In Algorithm 2, our slicing criterion is $< M_0, Q_2 >$. Here, M_0 represents the initial marking of the system, and Q_2 is the set of

Algorithm 2: Partial forward slice of IBPF

Input: $IBPF = (P, T, A, \Sigma, V, C, G, E, I), IBPF_1 = (P_1, T_1, A_1, \Sigma_1, V_1, C_1, G_1, E_1, I_1), < M_0, Q_2 >$
Output: $IBPF_2 = (P_2, T_2, A_2, \Sigma_2, V_2, C_2, G_2, E_2, I_2)$

$Q_2 = \{p_m | M_0(p_m) \neq \emptyset\}, IBPF_2 = \emptyset$.
Compute all enabled binding elements $\langle t_i, p_j \rangle$ in initial marking M_0, then get all enabled transitions t_i in initial marking M_0.
while $t_i \neq \emptyset$ **do**
  for *each* $p_j \in {}^\bullet t_i$ **do**
    Add p_j and arc-related elements of (p_j, t_i) to $IBPF_2$
  if t_i *doesn't exist in* $IBPF_1$ **then**
    for *each* $p_j \in t_i{}^\bullet$ **do**
      Add t_i, p_j, and arc-related elements of (t_i, p_j) to $IBPF_2$
  else
    if $Q_2 \subset P_2$ **then**
      break
    for *each* $p_j \in t_i{}^\bullet$ **do**
      Add p_j and arc-related elements of (t_i, p_j) to $IBPF_2$
  Get the new marking M, recalculate enabled binding elements $\langle t_i, p_j \rangle$ under the marking.
return $IBPF_2 = (P_2, T_2, A_2, \Sigma_2, V_2, C_2, G_2, E_2, I_2)$

non-empty places under M_0. We start by tracing forward from the initial marking M_0 and examining all enabled transitions. If a certain enabled transition t_i does not belong to the backward slice $IBPF_1$, we incorporate the transition t_i and its associated output places p_j, along with their attributes and the elements related to arcs between them, into $IBPF_2$. If the transition t_i is already present in $IBPF_1$ and all elements of Q_2 are included in P_2, the loop terminates and Algorithm 2 concludes. Otherwise, we add the associated attributes of the output places p_j of transition t_i, along with the elements related to arcs between t_i and these places, into $IBPF_2$. After the occurrence of this transition, the system reaches a reachable marking M. We then repeat the above operation under the marking M until all elements from Q_2 have been added to $IBPF_2$. Finally, Algorithm 2 returns $IBPF_2$.

The objective of employing dynamic slicing techniques is to extract key components that reflect the system's behavior and logical characteristics. However, when performing backward slice on Petri nets to retrieve parts associated with specific critical variables and places, we may encounter several paths. Although these paths structurally preserve the activity trajectory of critical elements, they overlook the potential interactions between variables.

In Colored Petri Nets, the interactions among variables are not merely reflected in single places or transitions but are manifested through a series of transitions and places interacting with each other, collectively forming the dynamic characteristics of the entire system. To overcome the limitations presented by

Algorithm 3: Bidirectional slice of IBPF

Input: $IBPF = (P, T, A, \Sigma, V, C, G, E, I), IBPF_1 = (P_1, T_1, A_1, \Sigma_1, V_1, C_1, G_1, E_1, I_1)$, ⟨transition, W⟩
Output: $IBPF_3 = (P_3, T_3, A_3, \Sigma_3, V_3, C_3, G_3, E_3, I_3)$

$Q_3 = \emptyset, P_3 = \emptyset, T_3 = \{transition\}, A_3 = \emptyset, \Sigma_3 = \emptyset, V_3 = \emptyset, C_3 = \emptyset, G_3 = \{G(transition)\}, E_3 = \emptyset, I_3 = \emptyset$
Function `traceFromTransition()`
  $depth = 1$
  while $W \neq \emptyset$ **do**
    Set $IBPF_3$ to the initial value
    if $ForwardCPN = depthLimitedSearch(IBPF, IBPF_1, transition, W, forward, depth)$ **then**
      Merge $ForwardCPN$ into $IBPF_3$
    if $BackwardCPN = depthLimitedSearch(IBPF, IBPF_1, transition, W, backward, depth)$ **then**
      Merge $BackwardCPN$ into $IBPF_3$
    $depth = depth + 1$
  return $IBPF_3$
Function `depthLimitedSearch`$(IBPF, IBPF_1, transition, W, direction, depth)$
  if $transition = \emptyset$ **then**
    return $\emptyset$
  if $depth = 0$ **then**
    return $\emptyset$
  if $direction = forward$ **then**
    for *each* p_i *in* $transition^\bullet$ **do**
      Add p_i and arc-related attributes of $(transition, p_i)$ to $path$
      if p_i *exists in* $IBPF_1$ **then**
        Merge $path$ into $resultCPN$ and delete the variables in W that flow through p_i
        continue
      for *each* t_i *in* $p_i^\bullet$ **do**
        Add t_i and arc-related attributes of (p_i, t_i) to $path$
        if t_i *in* $IBPF_1$ **then**
          Merge $path$ into $resultCPN$ and delete the variables in W that flow through t_i
          continue
        $path_1 =$ depthLimitedSearch$(IBPF, IBPF_1, t_i, W, forward, depth - 1)$
        if $path_1 \neq \emptyset$ **then**
          Merge $path_1$ and $path$ into $resultCPN$
    return $resultCPN$
  if $direction = backward$ **then**
    for *each* p_i *in* $^\bullet transition$ **do**
      Add p_i and arc-related attributes of $(p_i, transition)$ to $path$
      if p_i *exists in* $IBPF_1$ **then**
        Merge $path$ into $resultCPN$ and delete the variables in W that flow through p_i
        continue
      for *each* t_i *in* $^\bullet p_i$ **do**
        Add t_i and arc-related attributes of (t_i, p_i) to $path$
        if t_i *in* $IBPF_1$ **then**
          Merge $path$ into $resultCPN$ and delete the variables in W that flow through t_i
          continue
        $path_1 =$ depthLimitedSearch$(IBPF, IBPF_1, t_i, W, backward, depth - 1)$
        if $path_1 \neq \emptyset$ **then**
          Merge $path_1$ and $path$ into $resultCPN$
    return $resultCPN$

independent paths, we introduce key transitions to interconnect the variables within the slices. Key transitions refer to those that can influence or control the interactions among multiple critical variables. They typically involve key processes of transactions, such as the real-time progress of orders and the integrity of financial settlements.

In identifying key transitions, we adopted the following criteria:

(1) *These transitions can be successfully traced forward and backward, linking to the key variables in the related backward slice, thereby facilitating the interaction among various relatively independent paths within the backward slice.*

(2) *Key transitions are capable of constraining the values of certain elements within some variables, which prevent any interference with the original model's verification of order-related information.*

Through this approach, not only do we preserve the key behaviors of the system, but we also enhance the integrity and expressive power of the slice. Based on this, we propose a bidirectional slicing method, aimed at finding the shortest paths from key transition to variables related to the backward slice. The specific analysis steps are outlined in Algorithm 3.

In Algorithm 3, we utilize the Iterative Deepening Depth-First Search (IDDFS) algorithm, which combines the benefits of Depth-First Search (DFS) and Breadth-First Search (BFS), by continuously increasing the search depth with space efficiency, making it suitable for path finding and constraint satisfaction challenges [27]. For our research, this algorithm is particularly suitable as it assists us in finding the shortest paths from key transition to the target transition or place, whether it is forward or backward tracing.

In Algorithm 3, we take a complete $IBPF$ net along with a backward slice $IBPF_1$ as input parameters. Additionally, a slicing criterion composed of a key transition *transition* and a set of variables W must be specified. Here, the definition of W is the same as in Algorithm 1. The output of Algorithm 3 covers the shortest paths tracing forwards or backwards from the specified *transition* to all variables in W.

In the first step of the Algorithm 3, we initialize the depth to 1 and define a function named *traceFromTransition*. This function employs an iterative approach using a depth-first search function called *depthLimitedSearch*, which conducts both forward and backward searches in each iteration, progressively delving deeper. Whenever a path is found in either direction, the algorithm merges this path into the resultant path, *resultCPN*, until the search for elements at this depth is completed. The search depth is then incremented by one to further expand the scope of the search in the subsequent iteration, continuing until the set W is empty.

The *depthLimitedSearch* function is the core of the algorithm. It conducts a search based on the current depth and the direction of the search (forward or backward). The recursion ends when the depth reaches zero. If the *transition* is not in the backward slice $IBPF_1$, it examines the input or output places of the *transition* depending on the direction of recursion. For forward tracing, iterate through each output place p_i of the transition, if p_i exists in $IBPF_1$, then merge the current path into *resultCPN* and remove variables flowing through p_i from W. If p_i does not exist in $IBPF_1$, iterate through the output transitions t_i of p_i, if t_i exists in $IBPF_1$, also merge the path into *resultCPN* and update W. If t_i does not exist in $IBPF_1$, recursively call the *depthLimitedSearch* function on t_i, decreasing the *depth* by 1, if the path returned by the recursive call is not empty, merge this path into *resultCPN*. The function ultimately returns *resultCPN*. For backward tracing, the method is similar, just in the opposite tracing direction.

Employing this bidirectional iterative deepening approach markedly enhances the efficiency of slice generation. Particularly when dealing with large-scale IBPF net, this method can effectively minimize unnecessary search steps, thereby accelerating the discovery of critical paths.

Upon successfully implementing Algorithms 1 – 3 and obtaining $IBPF_1$, $IBPF_2$, and $IBPF_3$ respectively, we integrate them to derive the final dynamic slice. By modifying the initial marking of this dynamic slice and executing the run, we can effectively detect the associated logical vulnerabilities.

Theorem 1. *For an $IBFP = (P, T, A, \Sigma, V, C, G, E, I)$, let a backward slicing criterion be defined as $< Q_1, W >$. Based on Algorithm 1, the corresponding backward slice, denoted as $IBPF_1 = (P_1, T_1, A_1, \Sigma_1, V_1, C_1, G_1, E_1, I_1)$, can be obtained. Similarly, by defining a partial forward slicing criterion as $< M_0, Q_2 >$ and applying Algorithm 2, a partial forward slice, denoted as $IBPF_2 = (P_2, T_2, A_2, \Sigma_2, V_2, C_2, G_2, E_2, I_2)$, is derived. Furthermore, by setting a bidirectional slicing criterion as $<$ transition, $W >$ and utilizing Algorithm 3, a bidirectional slice, denoted as $IBPF_3 = (P_3, T_3, A_3, \Sigma_3, V_3, C_3, G_3, E_3, I_3)$, is produced. By merging $IBPF_1$, $IBPF_2$, and $IBPF_3$, we are able to construct $IBPF_4 = (P_4, T_4, A_4, \Sigma_4, V_4, C_4, G_4, E_4, I_4)$, which is assuredly the final dynamic slice of $IBPF$.*

Proof. Algorithm 1 traverses the places in Q_1 to trace backwards, $\forall p \in Q_1, \exists t \in {}^\bullet p : E(t, p) \cap W \neq \emptyset \Rightarrow t \in T_1$. After this, the tracing continues backward from t, $\forall p \in {}^\bullet t : E(p, t) \cap W \neq \emptyset$ and $p \notin Q_1 \Rightarrow p \in P_1$. This ensures that every path related to the variables in W is added to $IBPF_1$, and $\forall a \in A_1 \Rightarrow E(a) \cap W \neq \emptyset$. Therefore, the backward slice obtained through Algorithm 1, namely $IBPF_1$, conforms to Definition 5.

Algorithm 2 obtains all enabled transitions t_i and $Q_2 = \{p_m | M_0(p_m) \neq \emptyset\}$ under the initial marking M_0. For each enabled transition t_i, if $t_i \notin IBPF_1$, then $t_i \in IBPF_2$, $\forall p \in {}^\bullet t_i, p \in P_2$ and $\forall p \in t_i^\bullet, p \in P_2$. Assume there is a sequence of binding elements $(t_1, b_1)(t_2, b_2) \ldots (t_n, b_n)$ such that $M_0[(t_1, b_1) > M_1[(t_2, b_2) > \ldots > M_{n-1}[(t_n, b_n) > M_n$, where $n \in \mathbb{N}^+$. If under marking M_k implies $t_i \in IBPF_1$ and $Q_2 \subset P_2$, where $k < n$, the obtained $IBPF_2$ is a subnet of $IBPF$ that takes the initial state into account and successfully connects to the backward slice $IBPF_1$. Therefore, the net obtained by taking the union of $IBPF_1$ and $IBPF_2$ is a subnet of $IBPF$. In summary, the partial forward slice obtained through Algorithm 2, i.e., $IBPF_2$, conforms to Definition 7.

Algorithm 3 starts from the transition *transition* and traces in both forward and backward directions, setting the tracing

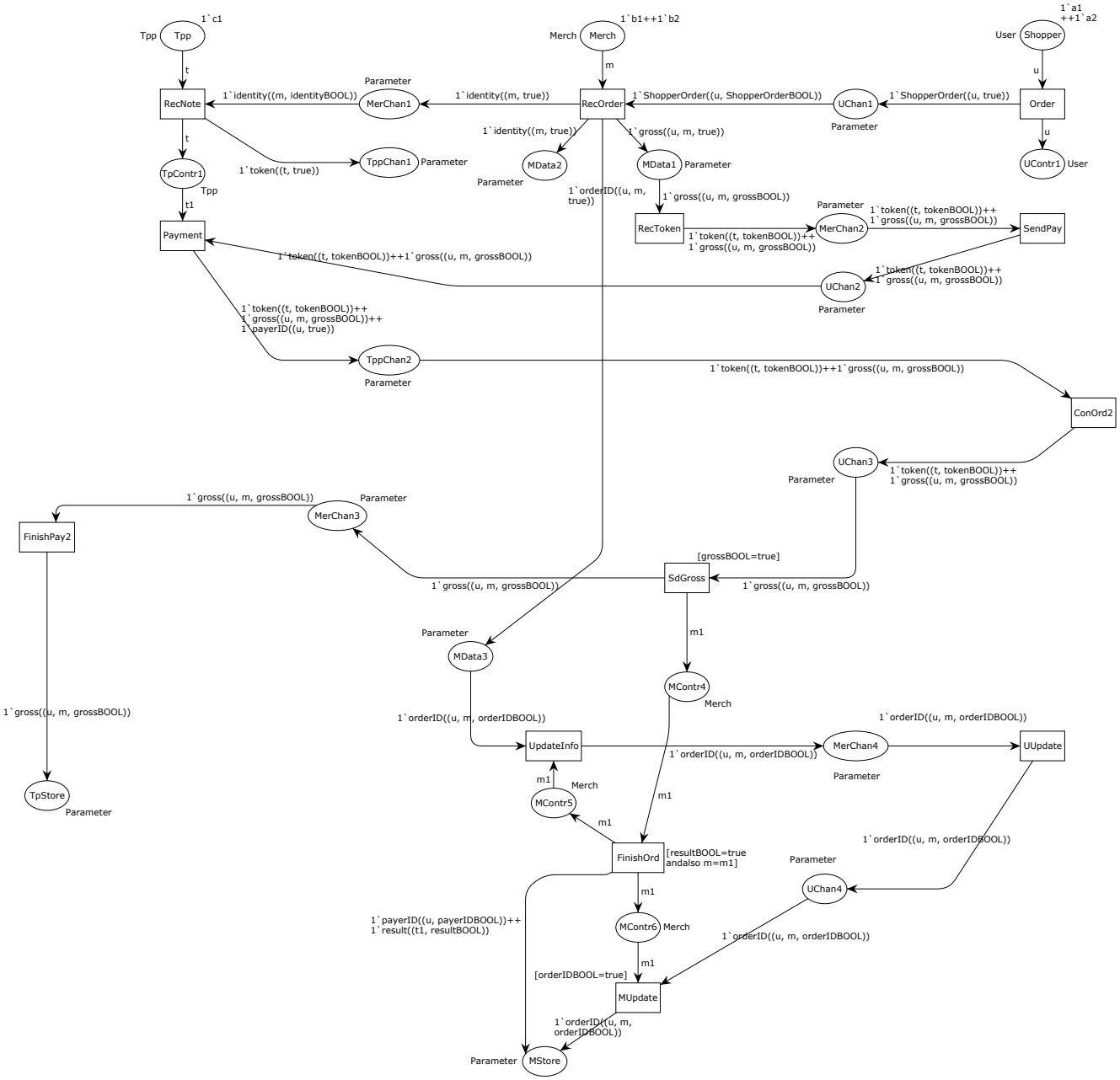


Figure 2: The final dynamic slice

depth $depth = 1$ at the beginning of the algorithm. For forward tracing, $\forall p \in transition^{\bullet}$: there exists at least one variable v flowing through p, where $v \in W \Rightarrow p \in P_3$, a shortest path has been found, and related variables are removed from W. If $p \notin IBPF_1$, $\forall t \in p^{\bullet} : t \in IBPF_1 \Rightarrow t \in T_3$, a shortest path is found, and related variables are removed. If $t \notin IBPF_1 \Rightarrow depthLimitedSearch(IBPF, IBPF_1, t, W, forward, depth-1)$ is called recursively. If $W \neq \emptyset$ under the condition $depth = 1$, it indicates that a greater depth is required, it works similarly for backward tracing, and the tracing continues as before until $W = \emptyset$. In summary, the bidirectional slice obtained through Algorithm 3, namely $IBPF_3$, ensures that every shortest path to the variables in W is found, conforming to Definition 9. □

For the dynamic slicing algorithm of IBPF, we define $n = |P| + |T| + |A| + |\Sigma|$, where $|P|$, $|T|$, $|A|$, and $|\Sigma|$ represent the number of places, transitions, arcs, and elements in the color set in IBPF net, respectively. The complexity of Algorithm 1 and Algorithm 2, in the worst case, is $O(n)$. For Algorithm 3, assuming the depth of traceability is d, the worst-case time complexity is $O(n^d)$. Therefore, the total time complexity is: $O(2n + n^d)$. This indicates that, even in the worst-case scenario, the time complexity of the dynamic slicing algorithm is very close to the scale of the original IBPF net.

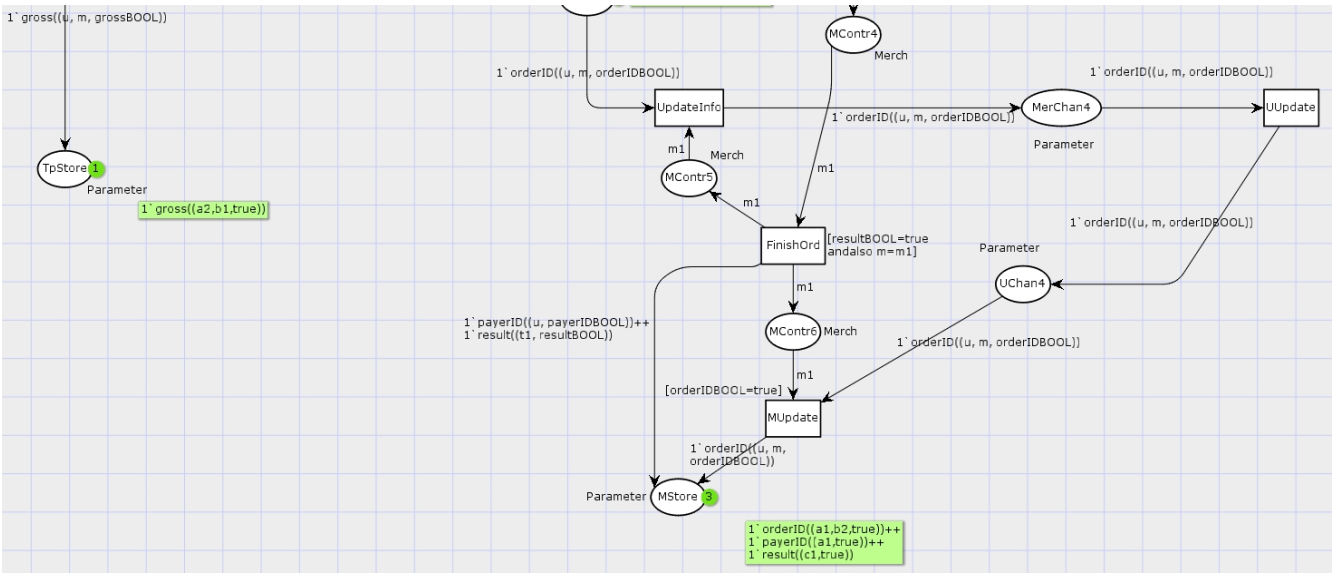


Figure 3: Detection of system operation result that do not conform to transaction consistency

5. Detecting logic vulnerabilities

In the previous section, we discussed algorithms related to dynamic slicing. This section will further introduce how to apply the method we proposed to detect logical vulnerabilities in the IBPF net of the e-commerce process illustrated in Fig. 1.

In e-commerce systems, the tampering of critical variables such as user authentication, transaction, and payment data may lead to vulnerabilities in related logic, threatening system stability [12]. Therefore, it is essential to rigorously protect and monitor these variables to ensure security. Particularly noteworthy in this context are the two variables: *orderID* and *gross*. They are involved in multiple logical processes and may trigger a variety of potential security vulnerabilities. In light of this, we designate these two variables as our focus variables, and the places $MStore$ and $TpStore$, which hold their final states, as our focus places. That is, we set $Q_1 = \{MStore, TpStore\}$ and $W = \{orderID, gross\}$. In the initial state M_0, according to Definition 7, we set $Q_2 = \{TPP, Merch, Shopper\}$. Based on Definition 9, we designate the key transition as $FinishOrd$.

We applied Algorithms 1 – 3 to the IBPF net shown in Fig. 1, with the slicing criteria set as $\langle Q_1, W\rangle$, $\langle M_0, Q_2\rangle$, and $\langle FinishOrd, W\rangle$ respectively. The resulting dynamic slice model is presented in Fig. 2. We observed that the final dynamic slicing model is significantly superior to the original IBPF net in terms of size and state space. This has made us more efficient in detecting logical vulnerabilities in our business processes.

Further analysis of the dynamic slice revealed a vulnerability, as shown in Fig. 3. Despite the completion of the process, there was a deviation in transaction logic consistency. Specifically, the variables for the shopper and the merchant, u and m, exhibited dual values, namely a_1, a_2, b_1, and b_2. This indicates that although the shopper completed the payment for the order $orderID(a_2, b_1, true)$, they received the goods for the order $orderID(a_1, b_2, true)$, violating the consistency requirement of transactions and pointing to a logical vulnerability in the system that needs rectification.

6. Conclusion

IBPF is a formal method suited for the modeling and verification of e-business processes involving multiple participants. It detects logical vulnerabilities in business processes during the system design phase. However, current analytical methods for IBPF are not sufficiently comprehensive. Given the large scale of e-business systems and the complexity of their business processes, direct analysis of IBPF net is challenging. To address this, we introduce dynamic slicing techniques to analyze IBPF net. Compared to the original IBPF net, our dynamic slice has a smaller state space, which reduces the need for processing during analysis and enhances the operability and efficiency of detecting vulnerabilities. This new strategy offers innovation and effectiveness in the timely identification, prevention, and correction of potential logical vulnerabilities in e-business processes. Nevertheless, our method still requires optimization. In future work, we intend to delve into how auxiliary means can be employed to automatically identify key transitions within the model. Furthermore, we aspire to extend this methodology to other domains, thereby enriching the application scenarios of the approach proposed in this study.

Acknowledgments

This work was in part supported by the Natural Science Foundation of Shaanxi Province, China under Grant 2021JM-205, the Fundamental Research Funds for the Central Universities, China under Grant GK202205039, and the Open Research Fund of Anhui Province Engineering Laboratory for Big Data Analysis and Early Warning Technology of Coal Mine Safety, China under Grant CSBD2022-ZD05.

References

[1] W. Yu, L. Liu, X. Wang, O. Bagdasar, J. Panneerselvam, Modeling and analyzing logic vulnerabilities of e-commerce systems at the design phase, IEEE Transactions on Systems, Man, and Cybernetics: Systems (2023).

[2] R. Amit, C. Zott, Value creation in e-business, Strategic management journal 22 (6-7) (2001) 493–520.

[3] M. Asbari, et al., Scope of e-business & e-commerce to business and modern life, Journal of Information Systems and Management (JISMA) 2 (1) (2023) 33–38.

[4] M. Sadeeq, A. I. Abdulla, A. S. Abdulraheem, Z. S. Ageed, Impact of electronic commerce on enterprise business, Technol. Rep. Kansai Univ 62 (5) (2020) 2365–2378.

[5] H. Taherdoost, E-business security and control, in: E-business essentials: Building a successful online enterprise, Springer, 2023, pp. 105–135.

[6] F. Nabi, J. Yong, X. Tao, M. S. Malhi, M. Farhan, U. Mahmood, Process of security assurance technique for application functional logic in e-commerce systems, Journal of Information Security 12 (3) (2021) 189–211.

[7] F. Nabi, Secure business application logic for e-commerce systems, Computers & Security 24 (3) (2005) 208–217.

[8] N. L. Bhatia, V. K. Shukla, R. Punhani, S. K. Dubey, Growing aspects of cyber security in e-commerce, in: 2021 International Conference on Communication information and Computing Technology (ICCICT), IEEE, 2021, pp. 1–6.

[9] A. Belghith, Investigation on e-commerce platforms for tackling e-business security challenge., International Journal on Engineering Applications 10 (6) (2022).

[10] M. Zhou, R. Zhang, W. Xie, W. Qian, A. Zhou, Security and privacy in cloud computing: A survey, in: 2010 Sixth International Conference on Semantics, Knowledge and Grids, IEEE, 2010, pp. 105–112.

[11] R. Wang, S. Chen, X. Wang, S. Qadeer, How to shop for free online–security analysis of cashier-as-a-service based web stores, in: 2011 IEEE symposium on security and privacy, IEEE, 2011, pp. 465–480.

[12] F. Sun, L. Xu, Z. Su, Detecting logic vulnerabilities in e-commerce applications., in: NDSS, 2014, pp. 1–16.

[13] M. Weiser, Program slicing, IEEE Transactions on Software Engineering SE-10 (4) (1984) 352–357.

[14] B. Xu, J. Qian, X. Zhang, Z. Wu, L. Chen, A brief survey of program slicing, ACM SIGSOFT Software Engineering Notes 30 (2) (2005) 1–36.

[15] M. Harman, S. Danicic, Using program slicing to simplify testing, Software Testing, Verification and Reliability 5 (3) (1995) 143–162.

[16] B. Korel, J. Laski, Dynamic program slicing, Information processing letters 29 (3) (1988) 155–163.

[17] H. Agrawal, J. R. Horgan, Dynamic program slicing, ACM SIGPlan Notices 25 (6) (1990) 246–256.

[18] J. Desel, W. Reisig, The concepts of petri nets, Software & Systems Modeling 14 (2015) 669–683.

[19] A. Rakow, Slicing petri nets, in: Proceedings of the Workshop on FABPWS, Vol. 7, 2007, pp. 56–70.

[20] W. Yu, Z. Ding, X. Fang, Dynamic slicing of petri nets based on structural dependency graph and its application in system analysis, Asian Journal of Control 17 (4) (2015) 1403–1414.

[21] W. Yu, J. Kong, Z. Ding, X. Zhai, Z. Li, Q. Guo, Modeling and analysis of etc control system with colored petri net and dynamic slicing, ACM Transactions on Embedded Computing Systems (2023).

[22] P. Chariyathitipong, W. Vatanawood, Dynamic slicing of time petri net based on mtl property, IEEE Access 10 (2022) 45207–45218.

[23] Y. I. Khan, A. Konios, N. Guelfi, A survey of petri nets slicing, ACM Computing Surveys (CSUR) 51 (5) (2018) 1–32.

[24] R. Davidrajuh, A. Roci, Performance of static slicing algorithms for petri nets, Int. J. Simul. Syst. Sci. Technol 20 (2018) 15.1–15.7.

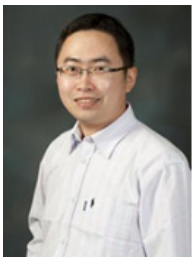
[25] Y. Wangyang, Y. Chungang, D. Zhijun, F. Xianwen, Extended and improved slicing technologies for petri nets, High Technology Letters 19 (1) (2013).

[26] M. Llorens, J. Oliver, J. Silva, S. Tamarit, G. Vidal, Dynamic slicing techniques for petri nets, Electronic Notes in Theoretical Computer Science 223 (2008) 153–165.

[27] R. E. Korf, Depth-first iterative-deepening: An optimal admissible tree search, Artificial intelligence 27 (1) (1985) 97–109.

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

Jie Feng obtained her Bachelor's degree in Information Security from Chengdu University of Information Technology, China, in 2023. Currently, she is pursuing a Master's degree at the School of Computer Science, Shaanxi Normal University, Xi'an, China. Her research interests include the theory of Petri nets and formal methods in software engineering. As she continues her academic journey, Jie Feng is committed to enhancing her knowledge and proficiency in these areas.



Lu Liu received his PhD degree from Surrey Space Centre at the University of Surrey. Professor Liu had worked as a Research Fellow at the WRG e-Science Centre at the University of Leeds. He served as the Head of the School of Computing and Mathematical Sciences at the University of Leicester, UK, from 2019-2023. His research interests are in the areas of data analytics, service computing, sustainable computing and the Internet of Things. He has over 250 scientific publications in reputable journals, academic books and international conferences. He has been the recipient of 7 Best Paper Awards from international conferences and was invited to deliver 8 keynote speeches at international conferences. Professor Liu is a Fellow of BCS (British Computer Society). He is currently serving as an Associate Editor for Peer-to-Peer Networking and Application (PPNA) and Big Data Mining and Analytics (BDMA).

Xiaojun Zhai received the PhD degree from University of Hertfordshire, UK, in 2013. He is currently a senior lecturer in the School of Computer Science and Electronic Engineering, University of Essex. He has authored/co-authored over 100 scientific papers in international journals and conference proceedings. His research interests mainly include the design and implementation of the digital image and signal processing algorithms, custom computing using FPGAs, embedded systems, and hardware/software co-design. He is a BCS, IEEE member, and HEA Fellow.

Yumeng Cheng obtained her Bachelor's degree in Computer Science and Technology from Beijing University of Civil Engineering and Architecture, China, in 2022. Currently, she is pursuing a Master's degree at the School of Computer Science, Shaanxi Normal University, Xi'an, China. Her research interests include the theory of Petri nets and formal methods in intelligent industrial manufacturing. As she continues her academic journey, Yumeng Cheng is committed to enhancing her knowledge and proficiency in these areas.